MuCIGREF: Multiple Computer-Interpretable Guideline Representation and Execution Framework for Managing Multimobidity Care

by

Eda Bilici Özyiğit

Thesis

Submitted to the University of Warwick for the degree of

Doctor of Philosophy in Engineering

Supervisors

Prof. T.N. Arvanitis, Assoc. Prof. G. Despotou

University of Warwick

July 2020

Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

r_i	A semantic relation, $i = 1, \ldots, n$
α_i	A variable, $i = 1, \ldots, n$
Λ	Finite set of variables
$\mathcal{ACS}$	Set of activity concurrency status
$\mathcal{AE}$	Set of activity execution status
$\mathcal{ALS}$	Set of activity lifecycle status
$\mathcal{ANS}$	Set of activity synchronisation status
$\mathcal{ATS}$	Set of activity time status
$\mathcal{A}$	Set of axioms
$\mathcal{C}$	Set of classes
$\mathcal{I}$	Set of named and anonymous individuals
$\mathcal{P}$	A patient's personal care plan
$\mathcal{R}$	Set of semantic relations
$\mathcal{TA}$	Set of assigned transitions
$\mathcal{TC}$	Set of transition conditions
Ω	Set of conditions
Π	Set of CIG models (domain)
$\Pi(\alpha_i)$	A variable i which ranges over the domain, $i = 1, \ldots, n$
π_i	A CIG model, $i = 1, \ldots, n$

Θ	An ontology
φ_i	A condition, $i = 1, \ldots, n$
c_i	A class, $i = 1, \ldots, n$
e_i	End time of a clinical activity, $i = 1, \ldots, n$
ln_{ik}	An activity synchronisation status i of activity k, $i = 1, \ldots, n$, $j = 1, \ldots, m$, $k = 1, \ldots, l$
ls_{ij}	An activity lifecycle status i of activity j, $i = 1, \ldots, n$, $j = 1, \ldots, m$
lt_{ik}	An activity time status i of activity k, $i = 1, \ldots, n$, $j = 1, \ldots, m$, $k = 1, \ldots, l$
s_i	Start time of a clinical activity, $i = 1, \ldots, n$
t_i	Duration of a clinical activity, $i = 1, \ldots, n$
tc_{ijk}	A transition condition i whose condition type j of activity k, $i = 1, \ldots, n$, $j = 1, \ldots, m$, $k = 1, \ldots, l$
ACEi	Angiotensin-Converting Enzyme Inhibitor
ADE	Adverse Drug Event
AF	Atrial Fibrillation
ASP	Answer Set Programming
BP	Blood Pressure
CDSS	Clinical Decision Support System
CHF	Chronic Heart Failure
CIG	Computer-interpretable Guideline
CKD	Chronic Kidney Disease
CPG	Clinical Practice Guideline
CPT	Current Procedural Terminology
DB	Diabetes
DL	Description Logic

DP	Depression
DPP-4	Dipeptidylpeptidase-4
eGFR	Estimated Glomerular Filtration Rate
EHR	Electronic Health Record
EMF	Eclipse Modelling Framework
EOL	Epsilon Object Language
EVL	Epsilon Validation Language
FOL	First-order Logic
GIN	Guidelines International Network
GSD	Guideline Service Deployment
HbA1c	Hemoglobin A1c
HCP	Healthcare Professional
HES	Health Care Service
HIS	Health Information System
HL7	Health Level 7
HTN	Hypertension
ICD	International Classification of Disease
ICSI	Institute for Clinical Systems Improvement
IRI	Internationalised Resource Identifier
LOINC	Logical Observation Identifiers, Names and Codes
MAN	Multi-Activity Management
MuCEE	MuCIGREF's CIG Execution Engine
MuCIGREF	Multiple Computer-interpretable Guideline Representation and Execution Framework
MuCRL	MuCIGREF's CIG Representation Language

NICE	National Institute for Health and Care Excellence
OCL	Object Constraint Language
OWL	Web Ontology Language
PCP	Patient Care Personalisation
PECAP	Personal Care Plan
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RIM	Reference Information Model
SNOMED	Standard Nomenclature of Medicine
STC	SNOMED-CT Codes
SWRL	Semantic Web Rule Language
TNM	Task-Network Model
UMLS	Unified Medical Language System
URI	Unique Resource Identifier
W3C	World Wide Web Consortium
XMI	XML Metadata Interchange
XML	Extendible Markup Language

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisors, Professor Theodoros Arvanitis and Associate Professor George Despotou for their encouragement, and guidance throughout my thesis. Their knowledge were very valuable for me and much appreciated.

I would like to thank my thesis committee, Professor Carsten Maple and Professor Yiannis Papadopoulos, for their feedback and advice for this work, and WMG (University of Warwick) for supplying me financial support during my PhD studies.

I am also very thankful to my family members, my parents and my brother for their support and encouragement with their best wishes throughout my research. Lastly, I am most grateful to my husband for his love, support and understanding throughout this challenging journey. Without his encouragement, I would never have been able to finish the degree. Thank you.

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. I hereby declare that, except where acknowledged, the work in this thesis has been composed by myself and has not been submitted in any previous application for the purpose of obtaining any academic degree. Parts of this thesis have been published by the author within the period of her study.

Eda Özyiğit

Publications

Peer Reviewed Journal Papers

- E. Bilici, G. Despotou, T.N. Arvanitis, “Ontology-driven representation and management of clinical practice guidelines and their associated knowledge constructs: A case study for multimorbidity,” *Journal of Biomedical Semantics*, 2019 (under review).
- E. Bilici, G. Despotou, T.N. Arvanitis, “The use of computer-interpretable clinical guidelines to manage care complexities of patients with multimorbid conditions: A review,” *Digital Health*, vol. 4, pp. 1–21, 2018.

Peer Reviewed Conference Papers

- E. Bilici, G. Despotou, T.N. Arvanitis, “Concurrent execution of multiple computer-interpretable clinical practice guidelines and their interrelations,” in *17th Int. Conf. on Informatics, Management and Technology in Healthcare*, Athens, Greece, Jul. 5–7, 2019. Published in: Studies in Health Technology and Informatics ISSN 0926-9630.
- E. Bilici, S.N. Lim Choi Keung, G. Despotou, T.N. Arvanitis, “Computer-interpretable guidelines driven clinical decision support systems: An approach to the treatment personalisation routes of patients with multi-diseases,” in *The 3rd West Midlands Health Informatics Network Conf.*, Coventry, UK, 2017.

Conference Posters and Abstracts

- E. Bilici, G. Despotou, T.N. Arvanitis, “An ontologically represented generic computer-interpretable guideline framework to manage patients with multi-morbidity – An artificial intelligence approach”, in *The Digital Healthcare & Safety of Connected Health: Improvement & Applications Conf.*, UK, 2018.

- E. Bilici, G. Despotou, T.N. Arvanitis, “Understanding benefits of incorporating computerised care protocols into clinical decision-making for the management of multimorbidities," in *WMG Doctoral Research & Innovation Conf.*, Coventry, UK, 2017.
- E. Bilici, G. Despotou, T.N. Arvanitis, “Using ontologies towards developing a generic computer-interpretable guideline framework,” in *Conf. in Medical Informatics Europe*, Gothenburg, Sweden, 2018.

Abstract

Clinical Practice Guidelines (CPGs) supply evidence-based recommendations to healthcare professionals (HCPs) for the care of patients. Their use in clinical practice has many benefits for patients, HCPs and treating medical centres, such as enhancing the quality of care, and reducing unwanted care variations. However, there are many challenges limiting their implementations. Initially, CPGs predominantly consider a specific disease, and only few of them refer to multimorbidity (i.e. the presence of two or more health conditions in an individual) and they are not able to adapt to dynamic changes in patient health conditions. The manual management of guideline recommendations are also challenging since recommendations may adversely interact with each other due to their competing targets and/or they can be duplicated when multiple of them are concurrently applied to a multimorbid patient. These may result in undesired outcomes such as severe disability, increased hospitalisation costs and many others. Formalisation of CPGs into a Computer Interpretable Guideline (CIG) format, allows the guidelines to be interpreted and processed by computer applications, such as Clinical Decision Support Systems (CDSS). This enables provision of automated support to manage the limitations of guidelines.

This thesis introduces a new approach for the problem of combining multiple concurrently implemented CIGs and their interrelations to manage multimorbidity care. MuCIGREF (Multiple Computer-Interpretable Guideline Representation and Execution Framework), is proposed whose specific objectives are to present (1) a novel **multiple CIG representation language**, MuCRL, where a generic ontology is developed to represent knowledge elements of CPGs and their interrelations, and to create the multimorbidity related associations between them. A systematic literature review is conducted to discover CPG representation requirements and gaps in multimorbidity care management. The ontology is built based on the synthesis of well-known ontology building lifecycle methodologies. Afterwards, the ontology is transformed to a metamodel to support the CIG execution phase; and (2) a novel **real-time multiple CIG execution engine**, MuCEE, where CIG models are dynamically combined to generate consistent and personalised care plans for multimorbid patients. MuCEE involves three modules as (i) *CIG acquisition module*, transfers CIGs to the personal care plan based on the patient's health conditions and to supply CIG version control; (ii) *parallel CIG execution module*, combines concurrently implemented multiple CIGs by performing concurrency management, time-based synchronisation (e.g., multi-activity merging), modification, and time-based optimisation of clinical activities; and (iii) *CIG verification module*, checks missing information, and inconsistencies to support CIG execution phases. Rule-based execution algorithms are presented for each module. Afterwards, a set of verification and validation analyses are performed involving real-world multimorbidity cases studies and comparative analyses with existing works. The results show that the proposed framework can combine multiple CIGs and dynamically merge, optimise and modify multiple clinical activities of them involving patient data.

This framework can be used to support HCPs in a CDSS setting to generate unified and personalised care recommendations for multimorbid patients while merging multiple guideline actions and eliminating care duplications to maintain their safety and supplying optimised health resource management, which may improve operational and cost efficiency in real world-cases, as well.

Chapter 1

Introduction

1.1 Research Motivation

Clinical practice guidelines (CPGs) [1] document evidence-based care instructions for clinicians, ranging from diagnosis to long-term management. The use of them, in clinical practice, supplies many benefits such as supporting clinical decision-making, improving quality of care, guiding health resource use, reducing healthcare costs and unwanted variations [2, 3, 4, 5]. CPGs mainly concentrate to a particular health condition (e.g., diabetes, hypertension, chronic heart failure) [6]. The Guidelines International Network (GIN), the Institute for Clinical Systems Improvement (ICSI) and the UK National Institute for Health and Care Excellence (NICE) are examples of sources of such guidelines.

Since early 2000s, many types of guideline-driven computerised platforms, also known as clinical decision support systems (CDSSs), have been developed to manage CPGs and support clinicians in the delivery of care [7, 8]. To manage CPGs, they initially need to be authored to a computer-interpretable guideline (CIG) [9, 10] format, which enables their representation and execution by computers, and can provide automated connections between clinical knowledge, patient data, and other applicable guidelines for supplying error-free and consistent care recommendations, in order to maintain patient safety. This can be achieved, by formalising the concepts included in CPGs and their interrelations, through the use of unambiguous and computer interpretable representation.

The existing CIG formalisms (also called CIG languages), such as Arden Syntax [11], Asbru [12], EON [13], GLARE [14], GLIF3 [15], PRO*forma* [16] or SAGE [17], adopt different approaches that supply computer interpretable representations. These include, ontologies [18] for knowledge acquisition, clinical task management and decision-making activities, along with their execution engines and tools, such as GLEE, Arezzo or DeGeL, that can run the CIGs on computers (see [19]).

The prevalence of multimorbidity [20] (i.e. patients with multiple diseases) increases with age. More than 95% of multimorbid patients have an age of 65 years and over [21]. Of them, 60% have at least two health conditions [22], and 58% generate 78% of all GP patient visits [23, 24]. Multimorbid conditions affect each other, and are closely associated with mortality, severe disability, care variations, increased health resource use and costs [25]. The management of multimorbid patients is complex, because the number of risk factors increases with the number of clinical conditions [26, 27]. To handle these patients, care plans need to be customised for each individual, considering their needs and conditions (e.g., allergies, syndromes, signs) incorporating a number of multidisciplinary stakeholders, such as nurses, doctors, therapists and clinical technicians [28]. Furthermore, patients have varied care requirements, including aspects such as allergies, drug intolerances and personal preferences.

For the treatment of a multimorbid patient, in most cases, more than two CPGs need to be concurrently implemented, alongside with the use of associated clinical knowledge and patient data, during the patient consultation. Thus, generating personal care plans for each multimorbid patient, through the reconciliation knowledge elements encapsulated in multiple guidelines related with patient disorders, while managing their conflicting recommendations and inconsistencies and tailoring them based on the patient-specific data, is crucial to maintain patients' safety. Nevertheless, multimorbid patient management requires integrated efforts of primary, secondary, acute and community-based care systems with human service systems like diagnosis, treatment, and long-term management for enhancing outcomes (e.g., quality of care, patient satisfaction and clinical efficiency) [29].

1.2 Research Challenges

Evaluating, individually, a patient health status and developing a personal care plan, as a result, is not a trivial task. This raises two main challenges, which have not yet been sufficiently addressed [30, 31], in providing integrated care approach to multimorbid patients:

- **Currently available clinical practice guidelines have several restrictions in their application in practice, while they supply mainly single-disease centred care recommendations, involve general care state-**

ments and do not cover the context of managing patients with multimorbid conditions.

Besides their benefits, CPGs have limitations for developing personalised therapy plans, especially for multimorbid patients [32, 33]. This is due to (i) CPGs being often in the form of text/schemas, which cause difficulties in the interpretation of guideline contents, by healthcare professionals (HCPs), during patient–carer encounters, and subsequently their implementations in care [34, 35]; and (ii) few existing CPGs refer to multimorbidity. Instead, they consider conditions in isolation [32]. However, in multimorbidity care, many CPGs need to be followed in parallel [36] and, there is little guidance on CPGs regarding how to merge strategies and recommendations to cope with multimorbid conditions, and the complex needs of patients with such conditions [37]. Thus, following multiple guidelines, which have been originally developed in isolation, may result in conflicting or inconsistent advice for care actions, since they do not sufficiently support dynamic changes in patient health conditions, when being translated to care plans without personalised adjustments.

In addition, evaluating patient data, as well as the vast number of clinical knowledge elements, manually, is a process susceptible to medical errors [38, 39]. When handling a patient with multimorbidity, HCPs may prefer a personalised version of a guideline [40]. The chosen guideline version, and the associated personalised version, have to be mutually consistent; this is important, in order to evaluate whether the guideline is being properly applied by HCPs [41].

- **There exists complexity in handling and merging actions of multiple, concurrently implemented, single-disease CPGs and their interrelations, in order to generate personalised care plans for a multimorbid patient.**

The automatic management of CPGs is complex, due to the several reasons. For instance, the issue of polypharmacy [42] (i.e. the use of multiple medications by an individual) has become one of the main concerns in caring for elder patients, who are fragile and have multiple health conditions [43]. CPGs do not adequately address the polypharmacy issues, which can be induced by multimorbid conditions [32]. Some of the main consequences of polypharmacy-related issues are inappropriate medication prescribing, poor adherence to care and adverse drug events (ADEs) [44] (i.e. an injury arising from medical intervention related to a drug) [45]. These are significant contributors of increased health risk, hospitalisation and subsequent increased health resource use and costs [46, 47]. For instance, ADEs constitute more than 6% of unplanned patient hospital attendances and are responsible for 4% of hospital bed occupancy in the UK [48]. ADEs are mainly arising from inadequate drug management but can be preventable [49].

Therapeutic actions can be applied at a single time event or spread over time. Correct timing of guideline actions plays a significant role upon achieving safe therapy implementation [50]. Thus, carers need to perform proper time management and correct chronological ordering of clinical activities (e.g., laboratory tests, or drug recommendations), accordingly. Since the complexity of managing care plans grows with the number of patients' health conditions [51], HCPs need to sequence all treatment steps, arrange parallel processes, and consider time constraints such as start, end and duration of treatments/signs/symptoms, and frequency of interventions to be appropriate [52]. When a healthcare professional is inexperienced with multimorbidity management or needs to manage complicated care plans, there is an obvious need for support in handling these issues, which should be included in the guidelines.

There exists a further difficulty in integrating CPGs within the care personalisation process for multimorbid patients. To personalise care for each patient, HCPs need to interpret clinical guidelines and patient's input, individually. However, this is not straightforward in the case of multi-morbidity, as this involves multiple guideline interactions and integration of numerous clinical knowledge elements (e.g., lab tests). These bring about the need of arranging concurrency and synchronisation relations between clinical activities, merging of clinical activities to eliminate care duplications (e.g., inefficient use of resources) or the need of making adjustments on clinical activities, and their interrelations (e.g., to avoid conflicts such as adverse drug interactions).

1.3 Research Aim and Objectives

The aim of this thesis is to devise a way to represent and reconcile multiple CPGs in a computer environment. To achieve this, the thesis presents a Multiple Computer-Interpretable Guideline Representation and Execution Framework (MuCIGREF) for the representation and real-time execution of multiple clinical guidelines and their associated knowledge elements, in order to generate personal care plans for multimorbid patients. The specific objectives of the research are:

- **To develop a CIG Representation Language that involves required semantics to represent knowledge constructs of existing CPGs and their interrelations, and to create the multimorbidity related associations between them.**

When a knowledge representation model is designed just for representing a specific CPG, then this may be prone to errors, when different CPGs (in case of multimorbidity) need to be represented, simultaneously. Bringing together and merging different CPGs may result in semantic heterogeneity [53] (i.e. differentiations in the level of concept coverage, details, and objectives of the intended users), which can be induced by using different terminologies to represent the same or similar domains, and lack of semantic interoperability across different healthcare organisations.

Existing research (e.g., [54, 55]) is mainly single disease centred and/or provided substantially detailed knowledge models which are mainly based on ontologies that are not easily adaptable, generalisable or re-usable to manage multi-activities recommended by different CPGs. Some of the published work provided general models for multimorbid disease management, yet they did not supply enough instances that can reflect real-life applications (e.g., [36, 56]) and/or face difficulties in merging more than two, concurrently applied clinical actions together [30], offered by multiple CIGs.

To resolve the semantic heterogeneity issue, this thesis initially presents a generic ontology to provide a standardised form for representing all CPGs and their associated knowledge constructs that can be easily adaptable and reusable in many clinical domains (e.g., obesity, hypertension, chronic heart failure). This ontology also supports performing required mappings between instances of formalised CPGs, namely CIGs, which are developed for each disease using the same vocabulary (ontology entities), in order to create the multimorbidity relations between them. Moreover, merging multiple CIGs and their actions may cause many complexities in care, such as adverse interactions (e.g., drug-drug, drug-disease and drug-patient interactions), which may threaten patient safety, or inefficient health resource use due to the care duplications, which may result in increased healthcare costs [31]. The ontology also needs to involve the required knowledge elements to handle such complexities.

Thus, there is a significant need for the development of a CIG representation language, which is built on a generic, and expressive (i.e. able to model real-world cases) ontology. Such ontology should represent knowledge constructs of all CPGs and their interrelations. It should support establishing required mappings between CPGs, in order to create multimorbid relations, such as merging of multiple clinical actions, which need advanced management of synchronisation and concurrency relations. Finally, such ontology should support users to make care modifications and optimisations, in order to handle multimorbidity related care complexities and lastly, support users to generalise, adapt and share the ontology over different health organisations, with the aim of enhancing semantic interoperability.

- **To develop an execution engine for performing real-time acquisition, parallel execution and verification of multiple, concurrently employed CIGs, in order to generate a unified personal care plan for a multimorbid patient.**

The execution engine should execute CIGs and their interactions to meet an intended objective (e.g., recommendation generation, etc.). The identified major requirements of an execution engine are to handle multiple CIGs, involve processing concurrently employed CIGs and creating automated mappings (e.g., finding similar or common care elements) between them, handling complexities induced by CIG interactions, and verifying CIG executions and, as a result, generating consolidated care recommendations.

Existing published research has major limitations in real-time executions of more than two concurrently implemented CIGs and performing multi-merging of their CIG actions [9, 30], which requires managing concurrency and synchronisation relations of multiple actions, while handling associated knowledge interactions (e.g., drug, disease, time, etc.). In addition, verification of CIGs, in the instantiation phase, is crucial to initialise care, with a consistent plan, as well as in execution phase, where the care plan is dynamically updated and needs to be verified to maintain patient-safety. Thus, a novel, real-time execution engine is needed for handling all these issues.

1.4 Summary of the Research Methodology

The research methodology, used within this thesis, has two stages. The first stage is about CIG language development (see Section 3.3) and the second stage is about the real-time CIG execution engine development (see Section 4.3) which are as follows:

Multiple CPG Representation Language (MuCRL). This stage is about the development of a novel, generic, and domain independent ontology to represent knowledge elements of all CPGs and their interrelations to manage multimorbidity care.

Initially, an extensive systematic literature review [31] is performed, where related knowledge representation approaches, namely CIG Languages (see Section 2.4.1), and then organisational workflow patterns [57] required to construct a care flow, existing ontology libraries (e.g., BioPortal [58]), clinical standards (e.g., [59]) and terminologies (see Section 2.4.2) and many CPGs from different sources (e.g., NICE) are analysed. Complexities in multimorbidity care (see Section 2.5) and gaps in managing multimorbid patients are discovered and discussed (see Section 2.6).

The method covers the principles of, largely existing, ontology building lifecycle methodologies [60, 61] to develop the knowledge representation structure. The ontology design stage (see Section 3.4) involves the definition of purpose and scope of the ontology, while includes the knowledge acquisition and specification of ontology requirements. The ontology development stage (see Section 3.5) focuses on the definition of the required concepts, properties and relations for representing and managing guidelines and their interrelationships. The ontology implementation stage involves encoding (see Section 3.6), where the ontology is encoded in Web Ontology Language (OWL) [62, 63] format from the World Wide Web Consortium (W3C). Afterwards, transformation from OWL ontology to the Ecore metamodel of EMF (Eclipse Modeling Framework) [64] which provides a modelling and code generation architecture in Eclipse, is performed (see Section 3.7). This supports the development of the MuCIGREF's CIG execution engine to dynamically handle multiple CIGs. Subsequently, different CPGs are instantiated using MuCRL, in order to generate CIGs (models). Finally, the evaluation stage looks at verification and validation of the MuCRL (see Section 6.3).

Real-time Multiple CIG Execution Engine (MuCEE). This stage is about the development of a novel execution engine (see Section 4.3) for real-time management of multiple CIGs and their interrelations. Initially, challenges in dynamic management of comorbid and/or multimorbid patients and supplying guideline-driven care (see Section 2.5) and their management approaches with their limitations (see Sections 2.6) are discovered. Accordingly, MuCEE is built, which involves three modules with different objectives as: (i) CIG model acquisition; (ii) parallel CIG execution; and (iii) CIG verification.

MuCEE uses CIG models, which are the instantiations of the EMF metamodel, and developed for each CPG to represent their knowledge constructs and their interrelations. Once CIG models are created, then they are acquired based on patients' diseases and satisfaction of a set of conditions through CIG acquisition module (see Section 4.3.1), based on the Epsilon Object Language (EOL)[65]. EOL is an imperative programming language to create, query and modify EMF models. Then, the acquired CIG models are unified under the personal care plan (model). In this thesis, managing multimorbidity care is defined as a multiple CIG combination problem and we supply a novel solution approach for this problem, where personal care plan (model) is used by the parallel CIG execution module (see Section

4.3.2). This module is designed to handle multiple concurrently implemented CIGs and therefore to generate personal care recommendations step-by-step based using a specialised execution algorithm. Afterwards, CIG verification module (see Section 4.3.3) is introduced. This module discovers inconsistencies, errors and missing values in guideline model and resulting personal care plan; and generates messages for users to amend them to maintain consistent care recommendations. The Epsilon Validation Language (EVL) [65] is built on EOL, as a variant of Object Constraint Language (OCL) [66] is used to build this module. Algorithms of CIG acquisition, parallel execution and verification are presented in Section 4.4.

For testing, a set of CPGs is selected from NICE and multiple CPG combinations are made based on the suggestions of the H2020 C3-cloud project (*www.c3-cloud.eu*) [67, 68] (as these have been validated by a clinical reference group), the suggestions of the research papers and patient scenarios available in these works (see Section 5.2.1). Implementations on multimorbidity case studies in generating personal care plans are presented in Section 5.3.5. Lastly, CIG execution verification and validation results are presented in Section 6.4.

1.5 Thesis Contributions

The contributions of this thesis can be summarised as follows:

1. *Multiple CIG Representation Language* – The development of a novel generic, and expressive CIG language, which involves required semantics to:

    (a) Represent knowledge constructs of all CPGs and their interrelations where knowledge is categorised under four distinct groups, whose elements are extendable and designed to manage multimorbidity care;

    (b) Represent required mappings (i.e. directed alignments) between knowledge constructs of multiple CIGs to create multimorbidity relations;

2. *Real-time Multiple CIG Execution Engine* – The development of a new real-time CIG execution mechanism to:

    (a) Manage multiple CIGs (models) under a unified personal care plan;

    (b) Support merging and concurrency relation management of synchronously implemented multiple activities recommended by multiple CIGs;

    (c) Support inconsistency, missing value and error management both on pre- and real-time CIG execution phases. A new CIG verification approach is proposed for this purpose involving error messages to support users.

To summarise, this thesis identifies the limitations of the existing CIG representation languages and their execution tools within the context of multimorbidity care. Accordingly, the research devises and provides a novel framework, which involves a multiple CIG representation language that is built upon a generic and expressive ontology and a multiple CIG execution engine to perform real-time execution, where the model is developed using this ontology, is at the core of this system. As far to contributions to the current literature, this thesis achieved a set of publications. These involve (i) systematic literature reviews [31, 69] on the challenges of multimorbidity care management, and CIG-driven CDSS applications to handle multimorbidity care involving gap analysis; and on the development of (ii) CIG representation language [70] and (iii) multiple CIG execution engine [71].

1.6 Thesis Outline

A brief overview, on the thesis chapters, is described below:

- Chapter 2 presents an extensive systematic literature review involving key concepts and background information for the entire thesis. This involves (i) knowledge engineering primitives and concepts; (ii) knowledge modelling and management languages; (iii) components of CIG-driven CDSSs; (iv) challenges and barriers in multimorbidity care; and lastly (v) existing approaches for managing multimorbidity care;

- Chapter 3 presents the MuCIGREF's multiple CIG representation language, MuCRL, which is designed to represent knowledge constructs of CPGs and their interrelations required to manage a multimorbid patient care pathway. Then, an ontology-driven CIG language building methodology, which has design, development, implementation and evaluation stages, is introduced. Lastly, design-time implementations of a CIG is also presented in this chapter. This chapter also presents the transformation process, which is based on creating mappings from OWL ontology to EMF model. These mappings are required to develop CIGs (models) in the Eclipse environment which supports code generation and therefore supply more functionality in real-time CIG executions;

- Chapter 4 presents the MuCIGREF's multiple CIG execution engine, MuCEE, for the real-time management of multiple concurrently implemented computerised guidelines. This engine has three modules, including: (i) CIG model acquisition, (ii) parallel CIG execution, and (iii) CIG verification. These modules are developed based on the CIG execution requirements as well as gaps in managing multimorbidity care. The CIG execution methodology and specifications of execution algorithms are presented in this chapter;

- Chapter 5 presents the implementations of MuCEE in real-life multimorbidity case studies. These implementation results demonstrate how challenges that are caused by the combination of multiple concurrently implemented CIGs can be handled, and a personalised care plan can be generated for a multimorbid patient based on combined CIGs;

- Chapter 6 presents the evaluation results of MuCIGREF. Evaluation involves verification and validation analysis, for both MuCRL and MuCEE. Firstly, the evaluation of MuCRL involves ontology evaluation (e.g., correctness, consistency, etc.) guideline representation with different patient scenarios to evaluate its expressiveness, and comparison with well-known CIG languages to maintain CPG representation requirements are satisfied for validation. Secondly, the evaluation of MuCEE involves the evaluation of CIG verification module's results as part of the CIG verification. The evaluation of CIG acquisition module follows this. Then, evaluation of parallel CIG execution module in personal care plan generation is evaluated. Lastly, MuCEE is compared with existing CIG execution engines for the validation purpose;

- Chapter 7 provides the main conclusions of the research. It discusses the results, main contributions, limitations and future research directions.

Chapter 2

Literature Review

2.1 Introduction

Ontologies [18] supply a shared and common meaning on a domain of interest, which can be used to link users and technologies. Ontologies play significant role upon knowledge representation, sharing and reuse. Knowledge is an interpretation of a domain of interest by satisfying a set of properties as a declarative statement. Thus, a knowledge base involves a set of these statements, where one can make inferences from them. While problem solving approaches are used to define the reasoning (e.g., inferring conclusions from the knowledge) mechanism, ontologies are used to represent the domain knowledge of a knowledge-based system.

This chapter reviews the core concepts of the thesis. Initially, knowledge engineering primitives and concepts are reviewed involving knowledge representation and management approaches. Since the objectives of this thesis are to develop a framework to represent and execute multiple CIGs to manage multimorbidity care, the major components of CIG-driven CDSSs are then reviewed. These involve existing CIG representation languages, their execution engines and tools; CIG integration with workflow control patterns, clinical information systems and terminologies; and lastly, CIG verification and validation approaches. The review then discusses challenges in multimorbidity care, which involve shortcomings of CPGs to treat multimorbidity, adverse interactions affecting care, and the issue of care personalisation. Subsequently, existing approaches to manage multimorbidity care in terms of guideline merging, adverse interaction handling and personalised care plan generation aspects are reviewed, and their limitations are addressed.

Some of the elements discussed in Sections 2.4 – 2.6 are the parts of our previously published systematic literature review; see Bilici et al. [31] for the publication.

2.2 Knowledge Engineering: Primitives and Concepts

Knowledge engineering is an area of Artificial Intelligence that uses knowledge to represent a specific process, system and/or its environment. This section presents knowledge primitives, their relations with ontologies and how knowledge can be used to construct machine-readable models for automated knowledge representation, reasoning, and validation purposes.

2.2.1 Models, Meta-models and Modelling

A model is a collection of statements, which can be defined as the simplification of an application domain (system), and a *metamodel* is a model of language that defines the model structure [72]. For the purposes of this thesis, *modelling* is defined as the process of developing a machine-readable model of a system and/or its processes.

2.2.2 Ontology as Knowledge Models

Ontologies are models of knowledge, which can be defined as formal descriptions of concepts and their relationships in a domain, such that knowledge can be represented in a computerised system.

There are different types of ontologies [18, 73]: (i) *generic* (also called top level) ontologies, which have top-level concepts (e.g., OpenGALEN [74]) and provide basic notions, such as objects, relations, properties, and processes used to define varied domains (e.g., obesity, diabetes, hypertension); (ii) *domain* ontologies, which represent a particular domain that involves concepts and vocabulary of a domain such as stroke, or hypertension guidelines; ; (iii) *local* ontologies, which are the parts of domain ontologies, and supply definitions of terms for a specific application in a specific domain; and, lastly, (iv) *task* ontologies, which involve knowledge to accomplish a task (e.g., diagnosis). The difference between domain and task ontologies is that domain ontologies describe the knowledge with which a task is implemented. Generic ontologies are useful for knowledge reuse, integration and sharing, due to their domain independence nature. Domain independence enables users to place concepts more accurately [75] and to handle semantic heterogeneity and/or inconsistency problems [76], which can be caused by the different definitions made for the same or similar domain.

To develop ontologies, an ontology building lifecycle methodology should be adopted. The prominent ones are METHONTOLOGY [60], Noy [61], TOVE [77], On-To-Knowledge [78], CommonKADS [79], and NeON [80]. The sequences of the ontology building steps of these methodologies includes: (i) determination of purpose and scope of the ontology; (ii) considering reusing of existing ontologies; (iii) enumerating important terms in the ontology; (iv) defining required concepts and their topological orders; (v) defining properties and restrictions of concepts; (vi) creating instances; (vii) implementing the ontology; and (viii) evaluating the ontology. The following section presents how ontologies can be encoded and embedded in a computerised environment, within the context of knowledge management.

2.3 Knowledge Modelling and Management Systems

Modelling languages represent specific aspects of the system, which has different expressiveness on statements of a model. Ecore [64], semantic web technologies, such as Web Ontology Language (OWL) [62], or the Unified Modelling Language (UML) [81] are some of the modelling languages that can be used in software developments and also in knowledge engineering. In this section, the widely used modelling languages and frameworks (relevant to this thesis) for knowledge management (e.g., representation, reasoning, and validation) are reviewed.

2.3.1 Unified Modelling Language

The Unified Modelling Language (UML) [81] is a modelling language that uses graphical notations and a metamodeling approach for developing, visualising, and documenting knowledge constructs of a system. UML involves three model element classifications, including classifiers (i.e. represent a collection of objects), events (i.e. possible occurrences of actions) and behaviours (i.e. possible executions of actions).

UML involves a set of *structural* diagrams such as class, deployment, object, component, deployment, or object diagrams; and *behaviour* diagrams such as activity, sequence, state, communication, interaction, or use-case diagrams. Among these diagrams, class diagrams have been widely used as a modelling approach, where the object types (classes) are depicted with boxes involving name, attribute and operation type information.These objects can have different relation types [82] as generalisation (or namely inheritance) (i.e. representing a subclass has the definition of its super class), association (i.e. representing relations between instances) or aggregation (i.e. representing object composition), which are represented with arrows and lines that link objects. In the context of this thesis, activity diagrams are used with the objective of defining algorithms (see Section 4.4) and guideline care flow (see Section 5.3.5) with the symbols presented in Figure 2.1.

Symbol	Name
●	Shows the starting node
◉	Shows the end node
▭	Represents an action/process
◇	Represents decisions
⟶	Represents control flow

Figure 2.1: Basic symbols and names of an activity diagram

2.3.2 Semantic Web Technologies

Semantic Web (i.e. web of linked data) [84] driven languages which are based on ontologies, mainly use the extensible Markup Language (XML) [85] to make an ontology as computer-interpretable [86]. These involve the Resource Description Framework (RDF) [87] and RDF Schema (RDF(S)) [88], OWL version 1 [62], and its latest version OWL2 [63, 89], and the DAML + OIL [90].

Description Logic (DL) [91] is a knowledge representation language, which involves concepts, roles, instances and their relationships. In a DL knowledge base, an ontology is defined as the combination of TBox (also called terminology box) and Abox (also called assertional box). TBox formally defines classes and their relations (e.g., isA). For example, the relationship "patient is a human" is formalised in DL, by considering Patient and Human as two different concepts and a statement in TBox is represented as $\mathsf{Patient} \sqsubseteq \mathsf{Human}$. ABox involves assertions about instances using the ontology terms. In ABox, for example, the assertion of "Mary is a patient" is `Mary` : $\mathsf{Patient}$. Roles can be used as binary predicates to represent a relationship between instances in DL. For example, *hasCarer* (Robert, Mary) represents the knowledge of "Robert is the carer of Mary".

OWL is a language to define ontology entities, such as classes, properties, instances and their relations and is based on DLs. Figure 2.2 presents OWL entities and their relations [83] using UML class diagram.

OWL has three sub-categories, i.e. OWL-Lite, OWL-DL and OWL-full, and it is built on RDF and DAML + OIL. The formal definition (controlled vocabulary) of the OWL ontology, can be defined as follows:

Definition 2.3.1. Ontology is defined as $\Theta :\ = \langle \mathcal{C},\ \mathcal{R},\ \mathcal{I},\ \mathcal{V},\ \mathcal{A} \rangle$ where $\mathcal{C} = \{c_1, c_2, ..., c_n\}$ is the set of classes (concepts) in the ontology that defines the domain. $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ represents the set of semantic relations (also called roles or properties) between classes including object and data relationships, where $r_i \in$

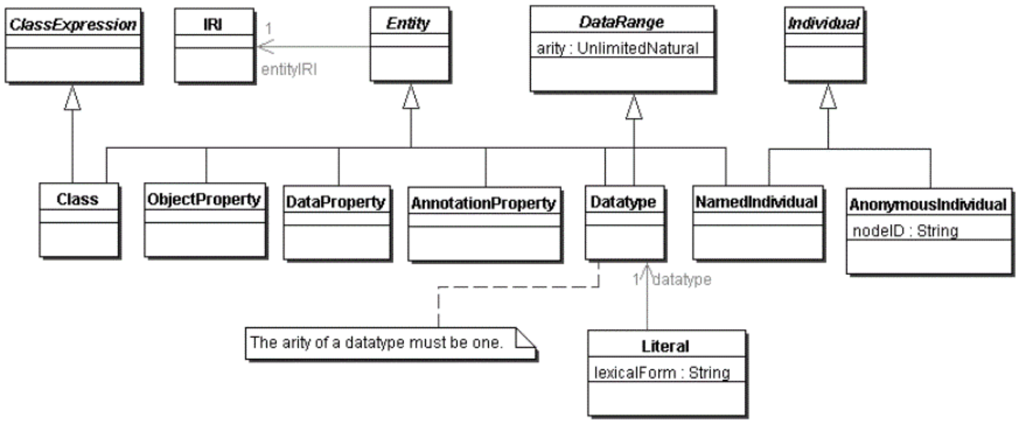


Figure 2.2: OWL ontology entities in UML class diagram representation [83]

$\mathcal{R}$ denotes a binary relation (e.g., isA, partOf) between classes. $\mathcal{V}$ is the set of data values. $\mathcal{I}$ is the set of named and anonymous individuals (instances) of classes and properties that represents elements or instances of an ontology. $\mathcal{A}$ represents the set of axioms defining dependencies (logical assertions) between elements of $\mathcal{C}$, $\mathcal{R}$ and $\mathcal{I}$.

Classes can have descendants and need to be organised into a superclass-subclass hierarchy. $c_i \sqsubseteq c_j$ represents the class c_i as a subclass of c_j, which is also known as taxonomy, where c_i, $c_j \in \mathcal{C}$. Properties can have varied facets that define or limit the knowledge, such as the value type (e.g., string, boolean, number, etc.), cardinality (e.g., allowed number of values), domain and range of properties. Value information for properties and their facets need to be supplied for named individuals. To illustrate for the context of this research; $x : c_i$, e.g., refers to patient x that is an instance of class Patient c_i, where $c_i \in \mathcal{C}$. "x: *patientAge* 40 xsd: integer", denotes the age of patient x where *patientAge* is the data property representing the age whose data value is 40 and data type is integer. There are several types of ontological relations, such as equivalence ($\equiv$), subsumption ($\sqsubseteq$), disjointness ($\perp$), union ($\sqcup$), membership ($\in$), and overlapping ($\sqcap$). To illustrate, $\mathcal{C} \equiv \{c_i\} \sqcup \ldots \sqcup \{c_n\}$ represents the topologically ordered classes; or $c_i \sqcap c_j \sqsubseteq \perp$ represents the disjoint (mismatched) classes. Ontology entities can be identified by an IRI (Internationalised Resource Identifier).

In the work of Munir and Anjum [86], RDF(S), OWL1 and OWL2 were extensively compared, in terms of their knowledge representation concepts to write ontology-driven relational database queries. Authors deduced that OWL2 includes more vocabulary, (e.g., properties and cardinality restrictions) and has a stronger syntax and computer-interpretability, compared to RDF(S).

To use ontologies, for knowledge inference from a knowledge base through query writing, an expressive ontology language, involving a reasoning support, should be selected. Reasoning can be performed to infer conclusions from a set of actions, namely assertions that are encoded in a model. Reasoning in DL (e.g., with its reasoners such as HermiT [92], FaCT++ [93]) can be classified as [94]: (i) TBox reasoning, which involves satisfiability checking and subsumption checking; (ii) ABox reasoning, which involves consistency checking, instance checking, instance retrieval checking and property filling checking, and (iii) query answering (e.g., SPARQL [95]).

In this thesis, knowledge representation is performed using OWL2 in Protégé (see Section 3.6), and FaCT++ and HermiT reasoners are used as part of ontology evaluations (see Section 6.3).

2.3.3 Ecore

The Eclipse Modelling Framework (EMF) [64] is a modelling environment, which supports model-driven software development by enabling users to describe models and supplying them execution-time support for the models. In EMF, models are represented in the XML metadata interchange (XMI) format. EMF has its own metamodel, called Ecore, based on textual syntax called Emfatic [96], which is used to define models and support customisation of Java code generation. EMF models involve a containment hierarchy of model elements (`EObject`), where objects can be defined as a set of operations over instance variables (i.e. a set of attributes, namely, primitive data values).

EMF involves Ecore metamodels (see Figure 2.3) to define the abstract syntax of a modelling language. The main components of Ecore are; `EClass` representing classes and includes `EStructuralFeature` (which is a `ETypedElement`), such as `EAttribute` or `EReference`; `EAttribute` representing attributes of classes; `EDataType` representing the type of attributes (e.g., string, java.lang.Float); `EEnum` representing enumerations of class instances; `EReference` representing the type of relations between classes; and `EPackage` representing the metamodel structure.

EMF is used to generate guideline models and then support their real-time executions (see Section 3.7).

2.3.4 Object Constraint Language

The Object Constraint Language (OCL) [66] is based on first-order logic and, widely used, declarative language to write constraints for model querying and validation.

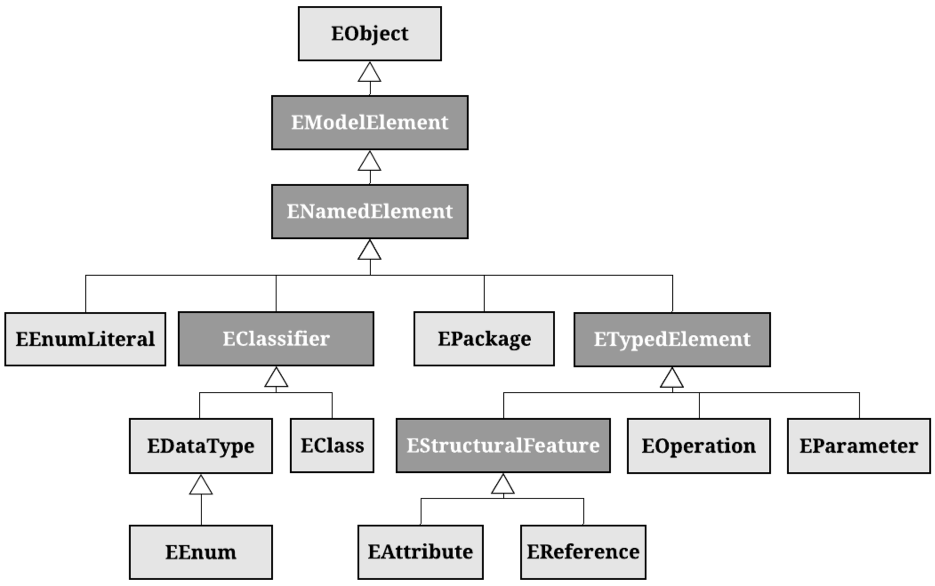


Figure 2.3: Ecore metamodel

A constraint, is defined by Warmer and Kleppe [97], as *a restriction on at least one value of an object-oriented model or system.*

OCL supports the following constraints: `Invariants` (associated with a classifier), which are used to define necessary conditions that must be satisfied in all the instances of the class or type, and must be true; `Pre-` and `Post-` conditions (associated with operations), which are used to define restrictions, and must be true before or after the current state of an operation; and `guard` which must be true before a state transition is invoked.

OCL uses expressions to denote an instance, condition, parameter value or define a value for an attribute or an operation. These expressions are interpreted in the context of an instance, namely, `Self`, which represents an object on which the operation is invoked. Other expressions include, `Collect`, which represents collection of statements; `In`, represents a function; `allInstances`, which represents a type operation that invokes instances of a type; `Context`, which represents type and operation expressions.

OCL has several benefits for users such as improving the model precision, presenting information on classes and their properties, supporting unambiguous communication between users and handling the limitations of UML by defining invariants on classes, and types in the class models [98]. In this thesis, variants of OCL are used for real-time CIG execution and verification which are Epsilon Object Language (EOL) and Epsilon Validation Language (EVL), respectively.

2.3.5 Epsilon Object Language

Epsilon [99] is an Eclipse Modelling project which provides tools and domain-specific languages for a set of model management tasks such as validation, transformation, comparison, migration and refactoring. All these task-specific languages build upon model-oriented language called the Epsilon Object Language (EOL) [65, 100].

EOL supports imperative programming for automated management of tasks which are objects of types defined in metamodels. These involve (i) many operation types such as primitive types (e.g., `min`, `toUpperCase`) that can be called on instances; collection types that are used to represent (non) unique or (un) ordered collections (e.g., `add`, `includes`, `clone`); first-order logic operations (e.g., `select`, `collect`, `reject`, `exists`, `forAll`) to return items of types that satisfy the condition(s); native types, such as an implementation property for generating new objects; and model element types (e.g., `all`, `allInstances`) to classify elements of (multiple) models; (ii) expressions such as literal values (e.g., integer); property navigations where ‘ . ’ syntax is used to invoke a property of an object, ‘ ; ’ syntax is used for sequencing and ‘ { } ’ syntax for grouping statements; arithmetic operators (e.g., $+$, $-$) and comparison operators (e.g., $<>$, $<=$); logical operators (e.g., `and`, `or`, `not`, `implies`); and enumerations of literals; and (iii) a set of statements (e.g., `If`, `Switch`).

EOL can be used to create new models, update, read, delete or query models which built on top of OCL, but can also handle OCL's model management limitations, such as not supporting statement sequencing, handling only one model at a time, and inability to modify models [101]. EOL is used for real-time CIG execution engine (see Section 4.3).

2.3.6 Epsilon Validation Language

The Epsilon Validation Language (EVL) [65, 102, 103] is a model validation language, which is built on EOL, and can be used to define and evaluate validation constraints on models of metamodels.

The syntax of EVL is presented in Figure 2.4, and its associated concepts are presented as follows [65, 102, 103]:

`Context` is used to define the instance types that involved invariants will be checked (e.g., class Patient); `Self` denotes the `context`'s instance (e.g., Patient0001:

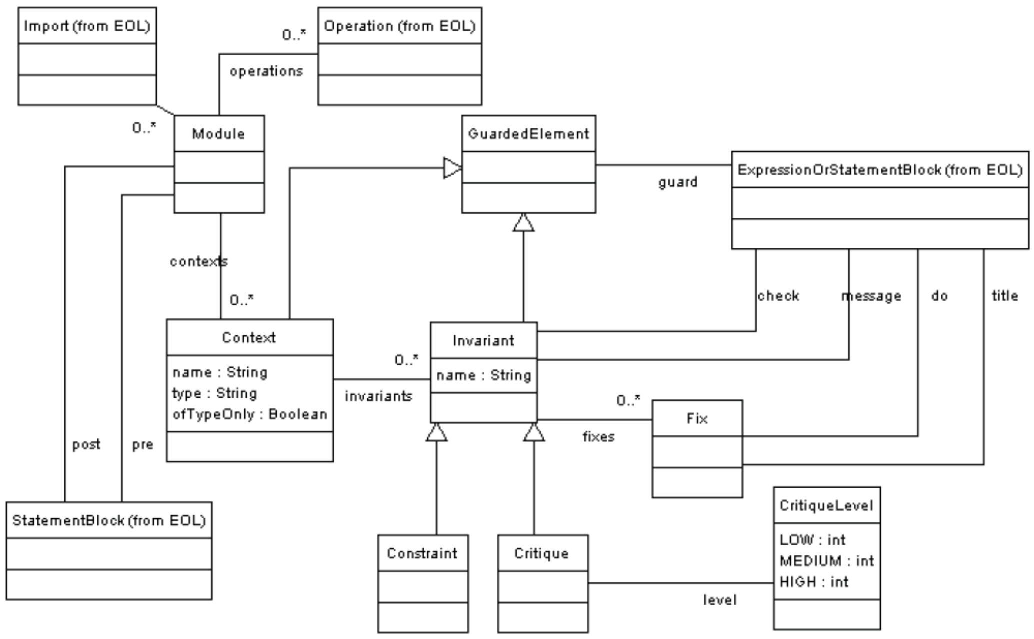


Figure 2.4: EVL syntax [103]

Mary); `Invariant` which is defined as an abstract class, is used to represent necessary conditions and can be used to describe message (as an `ExpressionOrStatementBlock`); `Guard` is used to restrict the applicability of invariants; `Fix` is used to resolve the detected inconsistency (e.g., renaming of a class); `Constraint` which is a subclass of `Invariant`, is used to detect critical errors which violate model restrictions; `Critique` is used to detect non-critical issues, which do not lead to violation of model (yet these issues need to be handled to improve the model quality); and `Pre- and Post-` conditions (blocks), which need to be executed before or after the invariant evaluations.

Customised error messages for users to fix the detected inconsistencies; and checking constraints, between multiple models, to handle incompleteness and contradiction if they occur are supported in EVL [102]. Thus, EVL handles the limitations of OCL's model validation features, such as supplying limited support for users in error correction (e.g., message generation to indicate unsatisfied invariants), no support for warning on non–critical issues, and inconsistency fixing. The execution mechanism of EVL is based on top-down depth-first search scheme. For further information on its execution semantics and comparison between OCL and EVL, please see Kolovos et al. [65, 103]. EVL is used in CIG verification module of this thesis (see Section 4.3.1).

This section reviewed knowledge representation and management tools and platforms. The following section discusses the components of CIG-driven CDSSs, in managing care of patients and how they can be used to handle the complexities of multimorbidity care management.

2.4 Computer-interpretable Guideline-driven CDSSs

Decision support systems unify large amount of knowledge in one platform, to help users in their decision-making processes. There are various types of decision support systems (e.g., document, communication, or knowledge oriented) that differ based on their capabilities and scope. For example, knowledge-based systems are one of the widely used systems in clinical settings to provide clinical decision support, see for example, [104] and [105].

In the clinical context, the general purpose of clinical decision support systems (CDSSs) [32] is *"providing clinicians or patients with computer-generated clinical knowledge and patient related information, intelligently filtered or presented at appropriate times, to enhance patient care"*. They are designed to help HCPs for a variety of clinical issues such as data access, disease diagnosis and prognosis, treatment, monitoring and prevention where CIGs are the core of these systems. Thus, HCPs can create care plans that are planned with a set of activities to handle patient's health condition in a time frame, and therefore, supply related care recommendations to patients.

Based on the requirements for automatic application of CPGs to support HCPs in their clinical actions, many formalisms and supporting tools have been developed to make guidelines computer-interpretable where ontologies have been used for CIG modelling, while coping with their complexities and associated clinical knowledge constructs. The following sections review the components of CDSSs which involve CIG representation approaches and associated execution engines and tools; their integrations with workflow-control patterns, clinical information systems and terminologies; and lastly, CIG verification and validation approaches, respectively.

2.4.1 CIG Representation Languages and Execution Approaches

Formalisation can be defined as the process of translating knowledge constructs, which can be available in textual form, tables or diagrams, into a machine-readable format. Guideline formalisation entails that knowledge, involved in CPGs, include elements that are well structured (such as concepts and care recommendations), and decision criteria that are explicitly described [106]. CIG formalisms (also called CIG languages) are a collection of knowledge elements (i.e. different types of concepts and relations), which are instantiated when the formalism is used to acquire a CIG based on the given values (i.e. attributes). Ontologies can be used as a standard form to represent these knowledge elements (e.g., guideline recommendations, and care workflow constructs required to establish the care pathway), in order to enable the development of computerised techniques to discover, and coordinate many types of interactions between them and to facilitate knowledge sharing and dissemination across professionals and institutions. CIG languages are used to represent the declarative knowledge of clinical care pathways, which can be defined as follows:

Definition 2.4.1. (Care Pathway) Care pathway represents a care plan(s) involving a set of clinical tasks (e.g., decision-making, medication recommendation, diagnosis, etc.) which are required to be performed in a defined period of time to meet the health needs of a patient.

There are different types of formalisms and some of the well-known approaches to represent and structure information. These include: (i) guideline document models (e.g., the Guideline Elements Model (GEM) [107]; (ii) frame-based models (e.g., GASTON [108]); (iii) rule-based models, which consider algorithms to establish information flows in guidelines (e.g., the Arden Syntax [11, 109, 110] and task-network models (TNMs) [9], which represent guidelines as graphical networks of tasks (usually defined as hierarchical graphs), in which nodes denote the actions to be executed and arcs denote the discerned relationships between them. In particular, TNM-based approaches may involve several task models like plan, action and decision. Plan denotes the collection of tasks that aims to achieve a certain objective. Action denotes the collection of tasks (such as medication prescription, or tests) that need to be performed during the execution of a guideline. Decision denotes the rules associated with conditions that are shaped with the patient's health states. A significant portion of formalisms use TNMs and include patients' states, execution states, eligibility criteria, classification schemes, goals, decisions and actions [111]. Asbru [12], EON [13], GLARE [14], GLIF3 [15], PRO*forma* [16], SAGE [17], and GUIDE [112] are TNM based formalisms. Each formalism has its own CIG execution approach. CIG execution can be defined as follows:

Definition 2.4.2. (CIG Execution) Inferencing from CPG instantiations that are encoded in the chosen formalism by the users (HCPs) with patient data using an execution mechanism (e.g., engine or tool). This mechanism shows users appropriate patient-specific care plans.

There are different types of CIG execution approaches, including graph traversing (e.g., [15, 108, 113]), and rule-based methods (e.g., [114]), where a rule refers to a query or encoded knowledge that can be used to make inferences from the knowledge base (semantically defined data). Inferences can be recommendations to supply care or to find the care point, conflicting care action or any care element to make a decision. Further below, in this section, CIG execution engines and tools, which are used to supply decision support, acquisition of guidelines (i.e. any activity during which a CDSS-driven execution engine will load or fetch a CIG from the knowledge base), verification, or making inferences from the knowledge base, are also reviewed along with the selected guideline formalisms.

Arden Syntax

Arden Syntax [11, 109, 110] was initially introduced by a group at the Arden Homestead in Harriman and then its development continued by the Arden Syntax Special Interest Group of the Clinical Decision Support Technical Committee of Health Level 7 (HL7). This formalism was developed as a rule-based language and uses medical logic modules (MLMs) (i.e. data, event, logic and action slots) maintained by HL7 [59], which is based on XML [85], for clinical knowledge representation and execution.

MLM has three categories: the *maintenance category*, to represent slots involving information like date, author, version; the *library category*, to represent slots regarding textual description of the logic; and the *knowledge category*, to represent executable statements required to supply decision support. The Arden Syntax IDE is the development and test environment of MLM. One of its applications has been the prediction of metastatic events in patients with melanoma [115].

However, MLMs have limited capability in identifying the complicated interacting recommendations of guidelines and coping with the representation of temporal constraints, such as repetitions, starting and end time of clinical actions.

GLARE

GLARE [14, 116, 117, 118] was developed by the collaboration of Dipartimento di Informatica, Universita del Piemonte Orientale "Amedeo Avogadro" Alessandria and Azienda Ospedalieta S. Giovanni Battista in Turin [119].

GLARE uses two types of actions [14, 120]: (i) atomic actions, such as *work actions*, which represent actions that must be executed (e.g., pharmacological actions); *decision actions*, which represent actions involving a set of conditional options, such as diagnostic or therapeutic decisions; *query actions*, which represent actions requesting input from users; and *conclusions*, which represent actions regarding the decision output), and (ii) composite actions, which represent control relations as *sequence*, *controlled*, *alternative* and *repetition*, used to create action execution orders.

GLARE represents a wide set of temporal constraints, treatment repetitions and periodicities in CIGs [14]. In the work of [121], ontology model of GLARE was extended to handle temporal interactions between actions and their effects occurring in time. GLARE's acquisition tool supplies a graphical interface, the ability to search guidelines and automatic consistency checking of temporal constraints. Its execution engine can execute acquired CPGs involving patient data and store the execution status of actions. XML [85] for data interchange, and the International Classification of Diseases version 9 (ICD-9) [122], as a clinical terminology, are used in this formalism [118]. Bottrighi and Terenziani [118] introduced a recent extension of GLARE, called META-GLARE, to acquire, represent and execute CIGs based on meta-programming. Execution of CIGs has been achieved using the execution of the graphs, which represents a network of guideline tasks, composing the network along with the execution algorithm. The execution states are [123]: (i) go_on, to represent continuation of the execution process; (ii) repeat, to represent the reputation of the execution; (iii) suspend, to represent suspension of the execution; (iv) abort, to represent termination of the execution; (v) goto, to represent restarting execution from the designated node; and (vi) fail, to represent execution failure and termination of the execution.

Parallel and concurrent execution of the tasks are also supported. The major limitations of GLARE are the lack of model-driven verification approaches and considering patients with co- or multi-morbid diseases, which requires coordination of diverse knowledge sources [118].

PRO*forma*

PRO*forma* [16, 124, 125, 126] was developed by the Advanced Computation Laboratory of Cancer Research, UK as a first-order logic formalism to represent and execute medical knowledge in guidelines, as a set of tasks such as decision, action, enquiry and plan.

Decisions represent tasks involving a set of options to the decision problem such as treatment options, and logical statements, which associated with satisfaction of each options; *actions* represent procedures, which need to be performed in an external source (e.g., HCP); and *enquiries* represent input requested from an external source, such as questions to patients. These are atomic tasks whereas *plans* are collections of tasks, denoting the objective of the treatment. Each guideline task emanated from a root task, namely, a common task. Thus, a root task involves many guidelines that are represented as plans (i.e. task sets). A clinical task is linked with an objective, defined in the red representation language (R_2L), which is a time-oriented, control-flow representation language. Then, R_2L is translated into a language based on predicate logic, called logic of R_2L (L_{R2L}). A plan can describe logical and temporal constraints, which are defined as time durations between tasks, repetitions and cycles as well. Each task may involve preconditions that must be satisfied for the enactment of the task, and trigger conditions for initiating a task, abort and termination conditions to end tasks. PRO*forma*'s arguments, which represent decisions, are based on rules.

Its execution platforms are Arezzo (commercial platform) and Tallis (experimental development platform) [16, 126] that involve a composer to support creation, editing and graphical visualisation of CIGs; a *tester* to debug the implementation and a *performer*, namely, enactment engine to execute guidelines, prompt user and to access patient data through the integration of local data base (e.g., electronic health records (EHRs)). Tallis also involves a publishing suit to make implementations online. The execution states of this formalism are [16]: (i) dormant if task has not been initialised; (ii) in_progress if task has been under execution; (iii) completed if task execution has been done; and (iv) discarded if task has been cancelled.

PRO*forma* has been used in many healthcare projects. For instance, the CAPSULE application to support HCPs in prescribing medications and the CREDO project [126] for the care of patient at risk or with breast cancer. The PRO*forma* representation was used in Health Care Services (HeCaSe2) [51, 127], which suggested an agent-based healthcare system for modelling CPGs and their interactions between agents (e.g., nurse, cardiologist, physician).

The major limitations of PRO*forma* are the lack of constraint support (e.g., when one action is activated, then the other activities cannot be activated until its completion) [124]; lack of duplication and syntactical error handlings; and inadequate expression language, due to the fact that it does not supply expression functions for simple or multi-merging [128], which are required to handle concurrently implemented guidelines and their multiple actions.

Asbru

Asbru [12, 129] was developed by the Vienna University of Technology and the Stanford Medical Informatics Department. It is a task-specific, time-oriented and intention-based language. This formalism was developed to represent CPGs and their inter-relationships as a group of skeletal plans (i.e. possible steps in a CPG) in XML [85], involving knowledge roles, such as preferences, intentions, conditions and effects.

Like PRO*forma* [16, 124], a plan is used in this formalism to represent a set of knowledge elements which are intensions to represent patient states and actions of HCPs. A set of conditions are also considered. These include, filter-preconditions to represent condition that must be satisfied, setup conditions to represent the applicability of a plan and suspend/abort conditions to represent suspension and abortion conditions regarding execution of a plan. The execution of a plan is represented with effects, which involve argument-dependency (i.e. effects between functional relations between arguments of plan and parameters), and plan-effect (i.e. expected effect on interrelationships of plans). In Asbru, the following execution states are considered [129]: (i) activated; (ii) completed; (iii) aborted; and (iv) suspended.

Asbru's execution platform is DeGeL [130]. It supports design and run-time actions in CIG-driven care. It involves a meta-ontology that has a documentation to define knowledge roles for all guideline ontologies, and a specification meta ontology to define a new ontology to include the DeGeL knowledge base. DeGeL involves tools that communicate with clinical databases for abstraction and clinical terminologies, including the Standard Nomenclature of Medicine (SNOMED) [131], the International Classification of Diseases version 9 (ICD-9) [122], the Logical Observation Identifiers, Names and Codes (LOINC) [132], and the Current Procedural Terminology (CPT) to retrieve diagnosis, observation, procedure and lab test codes. DeGeL has several modules to perform the activities such as search, extract, visualise and browse guideline sources/documents; view guideline indices and classify guideline documents; view, edit, search within documents; guideline retrieval, testing and execution via XML databases; supporting group management; and system administration. The Spock [133, 134] is a hybrid run-time execution system for Hybrid-Asbru, and can be used to extract information from the selected guidelines that are supplied by the DeGeL library; communicating with clinical databases and querying; extracting information from vocabulary servers; and evaluating Hybrid-Asbru guideline content using the Spock Engine. Asbru has been used in many projects (e.g., [135, 136]). For instance, OncoCure DSS project [135] is one of that helps oncologists in their various phases of decision-making for the treatment of breast cancer patients.

The major limitations of the Asbru formalism is its language complexity and its execution engine, Spock, does not support generation of user messages and alternative care options, while there is no evidence about the execution of multiple concurrently implemented CIGs.

GLIF

GLIF [15, 137] was developed by the InterMed Collaboratory and its latest version is GLIF3. This formalism has a layered mechanism, which involves the clinical definition of the concepts using codes based on clinical terminologies; the structure of the patient data elements; and an interface to link the guideline encoding with clinical information and knowledge sources (e.g., EHRs) [138].

GLIF uses the object-oriented programming paradigm for guideline modelling. In this formalism, the Guideline class represents a guideline and related attributes. The Algorithm class represents guideline recommendations as a set of steps, namely, the Guideline_Steps class, which involves following sub-classes [137]: (i) the Action_Step, represents recommended actions/tasks; (ii) the Decision_Step, represents recommended decisions; (iii) the Branch_Step, represents parallel actions/tasks; (iv) the Patient_State_Step, represents patient health state and eligibility for the guideline; and lastly, (v) the Synchronisation_Step, represents synchronisation of parallel branches. GLIF includes temporal constraints on guideline steps (e.g., actions and decisions) and patient data elements. The support language of GLIF is RDF [139] and the medical data model is the HL7 Reference information Model (RIM) [59]. Unlike Arden Syntax, GLIF can manage complex guidelines with many care steps [140].

The execution engine GLEE [15] was designed to execute CPGs that encoded in GLIF3 formalism. GLEE involves a set of layers, including a representation model that involves advices for specific clinical tasks, and supplies an execution environment for GLIF3, where CIG actions are traversed and executed with patient data. GLEE was used to implement guidelines and to assist conceptual modelling, encoding and validation of them. GLEE involves four execution states [15]: (i) prepared; (ii) started; (iii) stopped; and (iv) finished. While the active state is associated with prepared and started states, the inactive state is associated with finished and stopped states.

In Peleg et al. [141], web-based interactive clinical algorithms were developed, based on this formalism for the sequencing of tasks to evaluate patient with thyroid nodules. The major limitations of this formalism is its authoring tool, which is tied to the Protégé tool [142] and lack of mechanism for combining multiple concurrently implemented CIGs [137].

EON

EON [13] was developed by the Stanford Medical Informatics Department as a component-based modelling approach that involves a set of models. These include, a patient-data model, where patient data classes and attributes are defined; a medical specialty model, where abstract entities are modelled; and a guideline model, which is Dharma and represents knowledge constructs of guidelines as criteria, abstractions, guideline algorithms (a collection of scenarios that represent a patient's state), decisions, and recommended activities. EON uses temporal properties to perform scheduling of guideline activities, and handles duration constraints about them.

HL7 RIM [59] is also used in EON. Protégé-2000 is the authoring tool of this formalism. Padda is its execution system to produce recommendations using computerised guidelines and patient data, Tzolkin is its temporal data mediator to handle temporal relationships and WOZ is its explanation server that communicate with other tools. The execution states of the EON are [143]: (i) active; (ii) aborted; (iii) suspended; and (iv) completed. EON was used in the ATHENA project to manage hypertension.

The major limitations of this formalism are the coupling to the Protégé tool [142], for guideline implementations like GLIF, and the lack of expressions for managing concurrently implemented CIGs.

SAGE

SAGE [17, 144] was developed by SAGE project consortium as the evaluation of GLIF3 and EON formalisms. SAGE uses activity graphs that represent the relationships among several clinical and computational actions, and decision maps that represent a set of alerts and reminders. SAGE involves a guideline model, an authoring tool and an execution platform. The guideline ontology is represented in RDF [139]. SNOMED-CT [131] and LOINC [132] are adapted, in this formalism, for their use as terminologies. Like GLIF3, SAGE als uses HL7 RIM [59].

Protégé [142] is the authoring tool of this formalism and its testing tool is SAGE Desktop [145] that can test one CIG implementation at a time. Since SAGE supplies guideline-specific implementations, this cause a barrier to have a sharable representation. In addition, there is no evidence that multiple concurrently implemented guidelines can be handled and their activities merged with this formalism. SAGE has not been used in practice, and has some prototype applications for immunisation, diabetes and community-based pneumonia [146].

2.4.2 CIG Integrations with Workflow Patterns, Clinical Systems and Terminologies

This section reviews the necessary integrations of CIGs with workflow control patterns, clinical information systems and standards.

Workflow-Control Patterns

Workflow is a business process involving a collection of business activities, which need to be performed to achieve a business aim and a collection of conditions, which determine the sequence of these activities [147]. Likewise, a CIG formalism is a computerised version of a clinical care process involving a set of clinical activities and conditions to achieve a specific clinical aim [30]. The workflow control patterns [57] considered in CIG languages are as follows:

- **Sequencing**. Ordering activities with sequence numbers;
- **Splitting**. Splitting of activities based on a set of conditions (e.g., preconditions) and wait execution until these are satisfied;
- **Synchronisation**. Synchronisation (join) of parallel activities when all the sub-activities are completed. Otherwise, wait the completion of them or perform exclusive wait where the completion of the sub-activity will end all the sub-activities in the exclusive wait group;
- **Conditional Routing**. The enablement of activity control is passed to the following activity based on the results of condition expression (e.g., OR/XOR) of a preceding activity;
- **Parallel Activity**. Activities can also never be synchronised and may continue their respective paths in parallel;
- **Merging**. Multiple activities are merged into a single activity without forcing synchronisation of these activities;
- **Triggers, pre- and post-conditions**. Conditions involve a set of criteria that need to be satisfied to initialise or end an activity. These can be represented with Boolean expressions such as AND, OR, XOR or TRUE, FALSE. While trigger conditions represent event-based execution, post-conditions represent intentionality;
- **Iteration**. Activities can be repetitive and causes cycles until exit condition is triggered;

- **Cancellation**. An activity can be cancelled or removed even if the execution of this activity is started;
- **Termination**. An activity can be terminated if (i) there are no remaining work elements – activity is completed; or (ii) the remaining clinical process of the activity is cancelled as it reaches a designated state.

The limitations of the existing works (see, [30]) are mainly identified in supporting advanced workflow patterns like advanced merging, and synchronisation (which as discussed in Section 3.5.5 demonstrates how, in this area, this thesis fills the gaps of the existing literature).

Clinical Information Systems and Terminologies

To embed CIGs into a clinical practice, they need to be integrated with the care flow where patient-specific information, HCPs (e.g., doctors, nurses, lab technicians, etc.) and clinical actions (e.g., decisions, plans, examinations) are managed.

CIGs supply recommendations through combining CIG concepts with patient data and CIG execution engines can support automatic dynamic patient data handling [148]. Different care institutions can use different terminologies to define the same knowledge element of a guideline. This causes challenges in CIG sharing and maintaining semantic interoperability and standard clinical concept expressions. The major problem of CIG sharing is the use of different codes that a guideline denotes to patient information [149]. Clinical standards are important elements of CIG sharing and maintaining standardisation in clinical practice. For instance, clinical terminology standards like ICD-9 [122], SNOMED-CT [131] and LOINC [132], as well as concepts and attributes defined by HL7-RIM [59], play a significant role on CIG sharing, by bringing standardisation in the representation of guideline knowledge and related patient data.

In the literature, many studies [148, 150] proposed mapping ontologies and tools to perform temporal abstractions, integrating CIGs with the clinical terminologies (e.g., SNOMED-CT) and EHRs. A query translation approach based on mappings has been widely used for this purpose [138]. However, the major limitation of the existing works is presenting an ontology that can map knowledge elements of concurrently implemented, multiple CIGs, and their associations, with the clinical information systems and terminologies.

In the following section, the CIG verification and validation approaches are reviewed, which are necessary in order to maintain error-free and appropriate CIG implementations.

2.4.3 CIG Verification and Validation

Due to the high volume of the knowledge elements and their interactions involved in guidelines, CIG implementations are prone to errors. These may cause unwanted outcomes regarding care management and patient safety. For this reason, CIG verification and validation analysis is an integral part of the CIG-driven CDSSs. Preece [151] defines the verification as a process to check whether the software system satisfies the requirements defined by the users, and the validation as a process to check whether the software system satisfies the users' actual requirements.

In this context, verification process aims to supply consistent and error-free CIG specification, ensuring that CIG involves the required properties (e.g., medical practice, guideline goal, patient-specific clinical condition, time, patient groups) to satisfy the intended use and meaning, ensuring consistent guideline implementations [10, 152, 153]. Consistency checking as part of the verification process is required for acquiring consistent knowledge from guidelines, when they include new concepts and relations or exclude, or modify existing ones. This can be performed, for instance, to check names, the new term, relation or value defined by the HCP such as not allowing cycles in the model or for meeting design requirements such as ensuring that decisions have alternative options [120], or to check conflicting recommendations of guidelines. For instance, Karadimas et al. [11] used the CUP parsing tool to verify CIG actions represented in Arden Syntax formalism by checking each newly entered rules whether they cause conflicts. On the other hand, validation process can be performed by confirming the correctness of the ontology, in terms of the defined concepts and their properties in representing the domain. Domain experts can review the ontology [154] and evaluate the ontology correctness manually or by inferring knowledge (e.g., specific care recommendations) from knowledge base with patient data, using CIG execution engines, or by comparison with existing works.

There are many CIG verification techniques such as model checking [153, 155, 156], theorem proving [152, 157, 158], knowledge-based verification [50, 159] and hybrid approaches [27].

The model checking verification approach can be used to verify specifications of a given system. Specifications can be defined as temporal or tree logic formulas or algorithms that traverse the model expressed by the system and check whether the specification is satisfied or not [160]. For instance, Giordano et al. [156] proposed a model checking method, where GLARE encoded guidelines were translated to the Promela agent-based language of the SPIN model checker [161], and the linear time temporal logic formulas were used to check the properties (e.g., path, and executable guideline actions). A similar approach was adopted by Pérez and Porres

[153]. Authors proposed a model checking approach for CIG verification against semantic errors and inconsistencies in guideline definitions, and a model-driven development approach to automatically process manually created CIG specifications against specific verification requirements (e.g., correctness, structural) as well as to verify temporal-logic statements.

Theorem proving is a verification approach that can be used to perform reasoning on logically represented knowledge, which requires a syntax, a collection of axioms and knowledge inference rules. Hommersom et al. [152] formalised guidelines using Asbru and proposed an interactive theorem proving approach, which was implemented using the KIV theorem prover [162] for verification (e.g., checking model consistency, and treatment order, success and its optimality). In their later work [158], the authors proposed an automated reasoning approach to check the quality of guidelines by applying a meta-level quality check (i.e. formalising general requirements that a CPG should satisfy and then discover feasibility of them) over the type 2 diabetes guideline.

The knowledge-based verification approach can be used to check logical anomalies (e.g., syntax errors, redundant, conflicting or missing elements) in a knowledge base [163]. For instance, Duftschmid and Miksch [159] proposed a knowledge-based CIG verification method, where guidelines were formalised using the Asbru formalism. The proposed verification algorithm automatically checks anomalies (i.e. specification violations) such as unsatisfiable conditions occurring during execution, conditions with redundant parameter values, or incompatible conditions that may exist in CIGs.

Clinical actions defined in CIGs have to be performed according to a set of temporal constraints [50, 164]. These constraints can be qualitative constraints (e.g., simultaneously, after, or before) or quantitative constraints (e.g., days, delays, or durations such as '3 consecutive days', and clinical task T_1 starts '1 hour' after clinical task T_2) between clinical actions, periodic/repeated actions or the temporal constraints which can be the part of the relations between these actions [27]. To avoid the occurrence of any duplications and/or conflicts in care steps, temporal statements need to be checked for their validity when implementing guidelines [52, 165]. Temporal constraints are particularly important to apply correct prognosis [166], and multiple medication administrations in a certain time window. Anselma et al. [121] presented an instance on how temporal knowledge about the medication Anticoagulant has an impact upon the medication action of Warfarin administration. In the later work of Duftschmid et al. [50], authors proposed a verification approach which is based on calculating the minimal network of temporal constraints on the execution of CIG activities. Temporal scheduling constraints which are quantitative, and their implications from the contol-flow and hierarchical structuring of guideline actions are qualitative, were mapped to a Simple Temporal Problem (STP) [167]. Therefore, constraint propagation was implemented to find out inconsistencies and supply the minimal constraints between guideline actions. The limitations of this work involve inability to check unordered sequential activities of guidelines, and dealing with non-binary constraints.

Lastly, a hybrid CIG verification approach which was proposed by Anselma et al. [27] involves: (i) a constraint-based temporal reasoning method, STP framework, to check whether temporal constraints (which can be defined as twice a day, periodically, or after 2 days) in a CIG are consistent; (ii) model checking approach, where CIGs were modelled in Promela, and a SPIN model checker [161] was used, to check clinical properties of a CIG through formalising them as linear time temporal logic formulas. This was also used by Giordano et al. [156], for determining the best actions to be executed on the patient based on his/her current health conditions and symptoms as well as specific actions to be necessary for this patient; and (iii) probabilistic modelling, where the causal probabilistic decision time logic was proposed, to check probabilistic properties of a CIG for identifying best treatment options for a patient at hand.

In the literature, a few interactive guideline verification approaches were proposed which were mainly on temporal verification (e.g., [27, 50, 168]). However, there is a significant need for further verification tools to support dynamic execution of CIG languages [158]. This thesis addresses this gap (see Section 4.3.3) by introducing a CIG verification module which verifies multiple CIGs that are concurrently implemented; dynamically support users with customised messages to resolve the detected inconsistencies and/or errors; and involve mechanisms to perform automatic updates on CIGs which helps reducing allocation of users' time.

The following section reviews the major challenges and barriers in multimorbidity care. Multimorbidity forms the basis of the challenge of the objectives of the work of this thesis.

2.5 Challenges in Multimorbidity Care

Co-existence of multiple health conditions in an individual is steadily increasing, and aging is one of the main factors of the occurrence of these conditions [22, 23, 24]. The combinations of genetic functions, lifestyle choices, environmental issues, multiple drug usage, complications of past diseases and aging, generate patients with various combinations of multimorbidity. This creates several challenges and barriers to implement care for patients with multimorbid needs. In this section, the most prominent ones are reviewed. These involve shortcomings of CPGs to treat multimorbidity; and conflicting actions affecting care and personal care plan generation. In Section 2.6, we review proposed approaches, used to handle such challenges.

2.5.1 Shortcomings of CPGs to Treat Multimorbidity

CPGs can offer substantial benefits for the healthcare system and patients, such as helping to reduce health costs, improving consistency and quality of care [2, 4]. However, they are not sufficient for developing personalised therapy plans, especially for multimorbid patients [32, 33]. For the treatment of a patient with multiple diseases, more than two CPGs need to be implemented along with associated clinical knowledge and patient data during the patient consultation. Evaluating the patient health status individually and developing a patient-tailored care plan as a result, is not a trivial task due to several reasons.

CPGs are often in the form of texts/schemas that cause difficulties to HCPs in the interpretation of the guideline contents during patient–HCP encounters, and subsequently their implementation in care [34, 35]. This also causes dissemination and maintenance difficulties (e.g., updates and versioning) across healthcare organisations [40, 41]. When handling a patient with multimorbidity, HCPs may prefer a personalised version of a guideline [40]. CIGs facilitate this treatment personalisation process, as there can be instances of a CPG tailored to the specific information of a patient (e.g., preferences, allergies, drug intolerance and social needs), which due to their computer executable nature can also be adapted to dynamic changes of the patient's status. Versioning of guidelines plays a significant role for managing them, as defined by Grandi et al. [40] with several dimensions such as valid time (i.e. the time of a guideline belongs to the state-of -the art) and transaction time (i.e. the time the guideline is applied in a computerised platform). The chosen guideline version, and its personalised version, have to be mutually temporally consistent. This is important to evaluate whether the guideline is being properly applied by HCPs [41]. Evaluating patient data, as well as the vast number of clinical knowledge elements manually, is a process susceptible to medical errors [38, 39].

In multimorbidity care, many CPGs need to be followed in parallel [36]. However, CPGs are mostly designed for the treatment of a single disease, and there is little guidance on a CPG regarding how to merge strategies and recommendations to cope with multimorbid conditions, and the needs of these complex patients [37]. The simultaneous combination of multiple guidelines is also prone to adverse interactions. We review approaches for guideline merging are reviewed in Section 2.6.1 and the subsequent sections.

2.5.2 Adverse Interactions Affecting Care

Treatments of multimorbid conditions involve both pharmacological (e.g., drugs) and non-pharmacological (e.g., patient-education, surgery, rehabilitation, psychotherapy, etc.) activities. Pharmacological activities offered by each guideline are susceptible to adverse interactions with recommendations offered by other guidelines in varied forms such as drug–drug interactions, drug–disease interactions and drug–patient interactions that can reduce the efficacy of the care or affect the life expectancy of a patient [169]. The two main classifications of interactions are single-action interactions and multi-action interactions. A single action interaction appears as two guidelines having different recommendations for the same therapy. For instance, drug–dose variation may occur where two guidelines recommend different dose levels for the same drug. Multi-action interactions appear when medications, recommended by different guidelines interact with each other. Drug–drug, drug–disease, drug–patient, as well as drug–food interactions, can be considered as multi-action interactions, See the GuideLine INteraction Detection Architecture (GLINDA) [170] project, for further information on guideline interactions. Some of the widely occurring conflicts in guideline implementations are described in the paragraphs, below.

Drug–drug interactions occur in the case of two (or more) drugs, which (in the case of multimorbidity) may be recommended by two different guidelines. There are two main groups of drug–drug interactions [171]: (a) pharmacodynamic interactions that may occur when two drugs are taken together, and their concurrent usage causes serious health outcomes, and (b) pharmacokinetic interactions that may occur when one drug affects the other drug's efficacy. Adverse drug reactions [39] are, in general, linked with pharmacokinetic drug interactions [172]. Overdose of medication, if it results from multiple guideline medication recommendations, can cause serious adverse reactions with other drugs, as well as with the physiology of the patient. Drug–disease interactions occur when the intake of a drug interacts with a disease. For instance, a patient with asthma should not use non-selective beta-blocking drugs [173]. Drug–patient interactions occur if a patient has allergies or intolerances for (a) prescribed drug(s), and intake(s) of this drug(s) may have an adverse effect upon the patient.

To illustrate the above, let us consider a patient with the following chronic diseases: diabetes mellitus and hypertension. These two diseases involve different sets of clinical information (e.g., drugs to be taken, or side effects) and associated care flows. In the study of Kovalov and Bowles [37], guideline interactions of these diseases are considered. Here, the medication Nadolol offered for the care of hypertension conflicts with diabetes, as it causes a major drug–disease conflict; the medication Sitagliptin conflicts with a patient characteristic that causes a moderate level of patient allergy; and the use of Metformin and Acarbose medications cause a minor drug–drug conflict. Consequently, conflicting activities need to be detected and resolved before any treatment provision, in order to maintain safe care. Interactions can exist between drugs and foods that occur when drugs interact with foods or beverages (such as coffee, alcohol, orange juice, grapefruit juice, etc.), that annul or worsen the effect of drugs on the body. Lastly, CPGs mostly supply information about specific time elements such as the consecutive implementation of two certain drugs. These can temporally interact in time (i.e. two drugs interact with each other if they are administered within a specific time window).

Therapies can be given at one time or spread over time. Correct timing of guideline actions plays a significant role upon achieving safe therapy implementation [50]. Thus, HCPs need to perform proper time management and chronological ordering of clinical activities (e.g., laboratory tests, or drug recommendations). Piovesan and Terenziani [165] supply the following instance on the potential temporal sequence related interactions in guidelines. Calcium carbonate intake leads to alkalinisation of the urine, interacting with the nalidixic acid absorption. Thus, nalidixic acid should be given after calcium carbonate intake to avoid any conflicts. Arranging drug administration sequences can make both of them beneficial for the patient, and thus perhaps not causing any health risk, please see Anselma et al. [121] for further information on temporal interactions between guidelines. Approaches on how to handle adverse interactions are reviewed in Section 2.6.2.

2.5.3 Issue of Care Personalisation

Maintaining patient adherence to recommended interventions is a crucial factor in decreasing the risk of hospitalisation and in improving patient outcomes [174, 175]. Patient adherence, as defined by Christensen [176], is *"the extent to which a person's actions or behaviour coincides with advice or instruction from a health care provider intended to prevent, monitor, or ameliorate a disorder"*.

However, CPGs face integration difficulties in a care personalisation process, since they alone, are not best suited for showing adaptation to shared decision-making between patient and HCP, patient preferences or requests while providing a care plan. To customise care for each patient, HCPs need to interpret clinical guidelines and patient's input individually. However, this is not straightforward in the case of multi-morbidity, which involves multiple guideline interactions and integration of numerous clinical knowledge elements. These approaches are reviewed in Section 2.6.3.

2.6 Multimorbidity Care Management Approaches

This section reviews the approaches used in the management of multimorbid patients through CIG-based decision support technologies. Initially, we review methods that use CIGs to create a combined care plan. This follows with reviewing the methods to deal with the issue of polypharmacy and management of adverse interactions in guideline actions and associated knowledge elements and generating personalised care plans.

2.6.1 Guideline Combination Approaches

Over the past decades, many methodologies have been developed to cope with the unification (combination) and execution of multiple CIGs [28, 36, 114, 177, 178, 179, 180, 181, 182, 188, 189, 190, 191].

To generate a unified care plan for a multimorbid patient, guidelines need to be merged. Computerised CPG merging is referred in the literature as being performed in in two levels: manual and electronic. In the manual merging level, (i) domain experts first provide the required merging points between CPGs, and then these guidelines are merged manually based on this information and then computerised using the selected guideline formalism; or (ii) guidelines are formalised first, then they can be merged in a software system based on the domain experts' recommendations (e.g., [183]) which is a semi-automatic process. In the full electronic guideline merging level, guidelines are merged automatically based on the execution logic of the system.

The Semantic Web based formalism is one of the broadly applied approaches that merge CPGs, by initially formalising them as CIGs. Semantic Web technologies, such as the OWL [89], are characterised by formal semantics that have been used to represent clinical knowledge in CPGs.

For instance, Abidi et al. [179] proposed an OWL-based CDSS to represent multiple guidelines and generated a unified knowledge model for handling comorbid patients. In the study of Jafarpour and Abidi [180], multiple guidelines were merged using the merge criteria (i.e. a set of workflow, medical and institutional constraints) which was used to discover merging points between CPGs. To do so, a merging representation ontology was developed to detect these merge points. OWL-driven execution engine and SWRL [184] rules were used to achieve guideline merging according to the merge criteria. The major common limitations of these works were representing and merging more than two tasks of concurrently implemented guidelines. In their later work [181], the authors extended their guideline execution approach. Initially, they used OWL1 DL-based [185] execution engine to represent clinical task transitions between executional states and rules for managing the clinical task satisfaction criterion. Afterwards, OWL2 DL-based execution engine was used that provide more functionalities (e.g., cardinality restrictions and data type expressivity) than OWL1 DL. Here, OWL2 DL supports automatic comparisons of patient values with predefined values. Lastly, an OWL2 DL + SWRL-based guideline execution engine was used that also supports mathematical calculations and iterative clinical actions. Authors emphasized that combined guideline execution approaches supply more executional performance for reasoning on complicated guideline workflow patterns like iterative clinical actions than OWL1 DL. Lack of representation and execution of temporal constraints in guidelines were the main limitations of this work. In their recent work [182], authors extended their CIG combination (integration) approach by proposing semantics with the CIG-IntO ontology in OWL2 format using FOL rules in order to handle temporal constraints, resolve conflicts (e.g., adverse drug interactions) and maintain operational efficiency (e.g., elimination of redundant clinical tasks). Afterwards, ontology was executed with the CIG Integration Engine which was based on Apache Jena [186] reasoner. Lack of automatic detection of real-time multiple guideline merging points which were performed by HCPs for the selected guidelines, inability to merge more than two clinical tasks at a time and lack of mechanism (e.g., DrugBank) for performing automatic detection of the type of conflicts and their conflict degrees (see, [187]) were the limitations of this work.

There are also other approaches for merging guidelines, as the one proposed by Riaño and Collado [28] that adopted a divide and conquer approach (i.e. divides the problem into sub-problems until it can be solved) to merge many treatment plans of multiple guidelines considering the severity of the patient disease. Here, three main knowledge elements were considered: decision elements, which are related to the acuteness of a patient condition; action blocks denoting a set of actions such as tests to discover the severity of a condition; and table blocks, denoting the treatment matrices involving treatment, patient symptom and the recommended treatment. However, concurrency relations of multiple CIG actions and how to handle parallel tasks were not discussed.

Logic-based methods (e.g., Wilk et al. [36, 188, 189] and Michalowski et al. [190, 191]), which are formal approaches to representing and reasoning the knowledge involved in CPGs, are also widely used in the literature for combining care plans of multiple guidelines. Since merged guidelines may involve duplicated clinical actions (e.g., laboratory tests, examinations, medications) and possible contradictory or inconsistent actions, these should be discovered and eliminated before initialising any care to ensure patient safety. In the subsequent sections, these issues and associated works are reviewed.

2.6.2 Adverse Interaction Handling

Adverse interactions (conflicts) occur in presence of mutually inconsistent declarative sentences, and therefore, the system may lead to produce invalid outputs [192]. Adverse interactions can be induced by contradicting targets of the guideline actions, the effects of CIG actions, the medication conflicts offered by different guidelines or inappropriate timing of medical processes [116]. Zamborlini et al. [193] also demonstrated that interactions may not only occur in CIG actions which were mainly considered as drug–drug interactions in the existing works [188, 189], but also in CIG-independent interactions. Authors classified interactions as internal interactions (e.g., repetition interaction because of the same action, contradiction interaction) and external interactions (e.g., incompatible drugs and alternative drug interactions).

Discovery and resolution of adverse interaction are imperative to generate reliable and safe combined therapies. Studies have demonstrated that CIG-driven computerised systems facilitate the elimination of medication administration errors and ADEs, by recommending safe drug dose levels, arranging drug frequencies and associated durations of medications. For instance, Koutkias et al. [111, 194] proposed a CIG driven clinical decision support model based on GASTON [108], to help identification of drug safety risks and produce alerts and recommendations for HCPs to prevent ADEs.

Besides the use of logic-based models (e.g., [36, 188, 189, 190, 191]) for representing and merging knowledge elements of guidelines, such models have also been used for discovering adverse interactions caused by the synchronous implementation of multiple guidelines. For instance, Wilk et al. [36] proposed a constraint logic programming (CLP [195])-based model to represent guidelines and mitigate conflicting clinical actions which may occur between pairs of concurrently applied CPGs in order to provide guidance to HCPs for revising therapies in managing multimorbid patients. In this work, guidelines were represented as actionable graphs that are directed acyclic graphs used for representing guidelines and involve incomplete information. Yet, there were several assumptions made regarding the model (e.g., temporal constructs of CPGs were not considered) and mitigation algorithm (e.g., only binary variables were used). In their latter work [191], assumptions related to the mitigation algorithm that can handle cycles and numerical measurements were relaxed, while reconciling guidelines. The issues of temporal and related precedence relationships between guideline actions were addressed in Michalowski et al. [190]. To handle them, authors extended CLP to first-order logic (FOL) theories for developing a generalised mitigation framework. The major shortcomings of this work were the need to automate the maintenance of the precedence relationships between guideline actions, and the lack of parallel tasks and temporal characteristics. In the recent work of Wilk et al. [189], authors addressed limitations of their previous works [36, 188, 191] such as handling parallel tasks of multiple guidelines while generating a reconciled treatment. Besides parallel clinical tasks, temporal actions like time offset (i.e. lag between care steps) and duration in care steps were also considered. The major limitations of this study were the lack of complex decision nodes (i.e. more than two options) that represent real-world cases, with many decision options and handling more than two concurrently implemented actions of multiple CIGs.

While some works (e.g., Wilk et al. [36, 188, 189] and Michalowski et al. [191]) use actionable graphs or pharmaceutical graphs [37], some [196, 197, 198] prefer other methods, such as the Petri Nets-based models, to represent guidelines. For instance, Tan [198] presented the situation calculus ontology of the Petri Nets framework to generate combined therapy plans without adverse interactions. The major limitations of this work were the manual mitigation of two guidelines and not being able to adapt to execution-time modifications. Furthermore, there was no evidence on the applicability and efficiency of their method, when more than two guidelines are concurrently implemented.

2.6.3 Personalised Care Plan Generation

Several CIG-based CDSSs have emerged to personalise guideline knowledge to supply patient-tailored recommendations, considering patients' multiple health conditions, clinical history and preferences [199].

Isern et al. [51] suggested an agent-based K4Care platform, which supplies personalised homecare services for patients with multiple conditions. To personalise care, each patient's health conditions and their social context were considered. A state-decision-action (SDA)-based [200, 201] formalism was adopted, which represented CPGs as diagrams with a set of variables to determine the health condition of a patient; to choose a clinical or administrative task among a set clinical or management options, to represent the clinical or administrative tasks. As the part of the K4Care project, Riaño et al. [177] proposed methodologies for personalisation of patients' conditions (e.g., clinical and social information about the patient), and intervention plans to discover clinical and social inconsistencies in the patient data. Authors represented CIGs as SDA diagrams and presented a visualisation tool to edit and unify the diagrams of all intervention plans recommended for a multimorbid patient. The major limitation of these two works can be identified on the generation of patient-specific intervention plans, where the combination of therapy plans, and personalisation processes were manually performed. This may also limit the generation of alternative interventions when the number of diseases grows. In their later works [28, 114], therapy plans were combined, considering the patient's health conditions and adverse drug interactions. However, interactions can occur in many different levels of CIG actions (see, for example, Piovesan et al. [202]).

In many studies, the shared-decision making process was considered as an integral part of a CDSS, where guideline-driven advice can be addressed when needed and alternative care recommendations can be obtained involving patient's preferences and personal context (e.g., wedding, holidays, etc.). For instance, Peleg et al. [203] introduced two types of patient preferences, as local and global preferences. Local preferences denoted the personalisation of a certain CIG action, such as arranging the blood glucose measurement alert after a specific mealtime, whereas global preferences denoted choosing a CIG branch among alternatives such as preferring Warfarin medication instead of Asprin. In this work, guidelines were represented using Asbru and a graphical framework, called, GESHER [204]. Lack of clinical implementation, parallel paths and methodologies for detection of interactions occurred between multiple CIGs and resolution of them were the major limitations of this work. In their recent work [136], the authors extended their approach by proposing methods for acquiring and specifying information of parallel paths in care workflows based on CIG recommendations and making CIGs patient-centred, by customising them with patient's personal preferences and psychosocial context. However, this work falls short in conflict handling, creating multi-versions of CIGs [40, 41], and including extensive personalisation processes (see Riaño et al. [177]) for patients with varied multimorbid conditions.

2.7 Summary

This chapter begins with core concepts of knowledge engineering - which supplies information about ontology and knowledge modelling with associated knowledge management platforms. These constitute the basis of this thesis where ontologies are used for knowledge representation, and subsequently in the development of a model-driven software. Then, review is continued with the presentation of the components of CIG-driven CDSSs. These involve existing CIG languages which are used to formalise CPGs and associated knowledge constructs, and their execution engines to generate automatic care recommendations for the patients. Following this, the roles of CIG integrations with workflow patterns, clinical information systems and terminologies for creating a care pathway for a patient are reviewed. Lastly, CIG verification and validation approaches are reviewed to demonstrate how CIGs can be tested and their applicability in clinical practice can be measured.

This thesis focuses on developing a new CIG language and its associated execution engine, which can be used as a CIG-driven CDSS for patient management. For this reason, related existing works are reviewed. Afterwards, the specific challenges and barriers in multimorbidity care are reviewed. These include the limitations of CPGs to handle multimorbidity care, namely the adverse interactions affecting care and integration difficulties of CPGs in care personalisation process, which may result in limited or impaired patient adherence to care. Lastly, existing approaches for managing multimorbidity care, with their limitations are reviewed. This review has been conducted as this thesis targets the development of a novel CIG language and its execution mechanism to handle multimorbid patients. Accordingly, this part of the review involves methods for guideline merging (combination), dealing with adverse interactions and personal care plan generation. The core concepts of managing multimorbidity care through using CIG-based CDSSs are: (i) unification of multiple guidelines; (ii) handling of contradictory and inconsistent activities within guidelines and of their associated knowledge elements; and (iii) generating patient-tailored care plans. This chapter concludes that the adoption of CIG-driven CDSSs can enhance the benefits of CPGs for HCPs who can use standard reference to improve the quality of care and support their decision-making processes, if such CIGs are adequately formalised and executed.

Chapter 3

Representation of Multiple CIGs and Their Interrelations

3.1 Introduction

Clinical Practice Guidelines (CPGs) are unstructured, mostly free text, clinical statements that supply evidence-based recommendations from diagnosis to long-term management [1]. Use of CPGs is seen as substantially beneficial [2, 34], yet they fall short in two aspects, necessary for managing multimorbidity: (i) they do not sufficiently support dynamic changes in patient health conditions, when being translated to care plans, without personalised adjustments [136, 190]; and (ii) they do not provide the means to reconcile potential conflicts, when applied in parallel to other guidelines necessary for multimorbid patients [20]. Formalisation of CPGs, as Computer-interpretable Guidelines (CIGs) [10], enables their execution by computers, integrating with patient data, and guidelines, into a patient-centred care plan. This requires capturing (in a CIG format) information such as clinical actions, decisions and a set of conditions and/or constraints (e.g., temporal or executional) that must be satisfied.

In the existing literature, various guideline modelling approaches (e.g., rule-based, document-based) based on ontologies have been proposed to represent knowledge constructs of guidelines and execute them with their associated execution engines and tools for supplying decision support to HCPs (see Section 2.4.1). However, there is still an obvious need for the decision support-based application of multiple CPGs which raises two main challenges: (1) combining multiple concurrently implemented clinical activities; and (2) methods to manage (e.g., merge, modify or optimise) these activities and their interactions (see Section 2.5). These challenges have not yet been sufficiently addressed [30, 31] (also see Section 2.6).

The methodology introduced in this chapter is about the development of a domain-independent (e.g., applicable to health conditions ranging from chronic heart failure to cancer) language, namely, MuCIGREF's Multiple CIG Representation Language (MuCRL), which is built upon the generic ontology (see Bilici et al. [70] for the related publication). This thesis intends to fill the gaps in representing and managing multiple concurrently implemented CPGs and their interrelations, which are crucial to create a personalised care plan for a multimorbid patient to maintain patient safety and their adherences to care (see Section 2.5.3). The major aims of MuCRL are to:

- supply generic and expressive vocabulary for CIG applications, where different diseases (e.g., obesity, chronic heart failure) with different scopes (e.g., treatment, prevention, management) can be represented;
- supply mapping semantics to establish multimorbidity relations between multiple CIGs;
- supply an extendable formalism, enabling users to introduce new concepts and properties if needed;
- support care complexity handling elements such as care duplications, conflicts and inefficient health resource uses.

In the following sections, CPG representation requirements, the method of CIG representation language development and related results are presented, respectively.

3.2 CIG Representation Requirements

The common CPG representation requirements [19, 118, 127] of existing guideline formalisms [11, 12, 13, 14, 15, 16, 17, 109, 110, 116, 117, 118, 124, 125, 126, 129, 137, 144] are discovered to ensure that MuCRL has adequate knowledge representation power.

These are supporting; (i) the main workflow control patterns (e.g., iterations, activity synchronisation and sequencing) required to create a network of guideline activities; (ii) common collection of guideline activities such as action, decision, and conditions to be satisfied; (iii) guideline activity nesting - activities can have smaller units of activity which can be made into sub-activities nested underneath that parent activity; (iv) guideline representation, acquisition and execution; (v) storing the guideline execution status (e.g., active, in progress, pending, completed); (vi) data definitions; and (vii) description of clinical actions.

Following section presents MuCRL's development methodology where CPG representation requirements are considered in this stage.

3.3 MuCIGREF's Multiple CIG Representation Language (MuCRL) Building Method

MuCRL provides three main aspects, which are to supply; (i) a knowledge representation structure for multimorbidity, whose role is to represent knowledge elements of all CPGs and their interrelations; (ii) a knowledge reconciliation, which involves required semantics to perform alignments, mapping and merging in/between knowledge elements of multiple CIGs; and (iii) a knowledge management, which involves required semantics to manage multiple CIGs such as handling concurrency and synchronisation relations, care modifications and optimisation issues.

The method is based on best practice principles of largely used ontology building life-cycle methodologies, which are METHONTOLOGY [60] - supplies a well-structured and comprehensive methodology for step-by-step ontology building from scratch and Noy and McGuinness [61] - covers the fundamentals of ontology design and development stages with clear definitions. Thus, the resulting method involves four main stages including seven steps, as shown in Figure 3.1. Each stage of the ontology development lifecycle is documented.

- **Step 1: Specification of the purpose, scope & ontology requirements**

In this step, the ontology specification involving the domain, scope, purpose and requirements of the ontology, is determined. The purpose of the ontology involves the intended users and end-users, the scope (that involves a set of terms to be represented), its characteristics and granularity of a domain to be represented. This can be determined by a set of competency questions [205]. These competency questions, and associated answers are discussed in the design stage part of the result section (see Section 3.4). Requirement analysis is also performed to clarify the knowledge to be represented and the (competency) questions to be answered.

- **Step 2: Knowledge acquisition**

Knowledge acquisition, which is part of the ontology design stage, is performed simultaneously with the ontology specification requirement step, resulting in identification of the ontology classes, and their associated knowledge artefacts. While

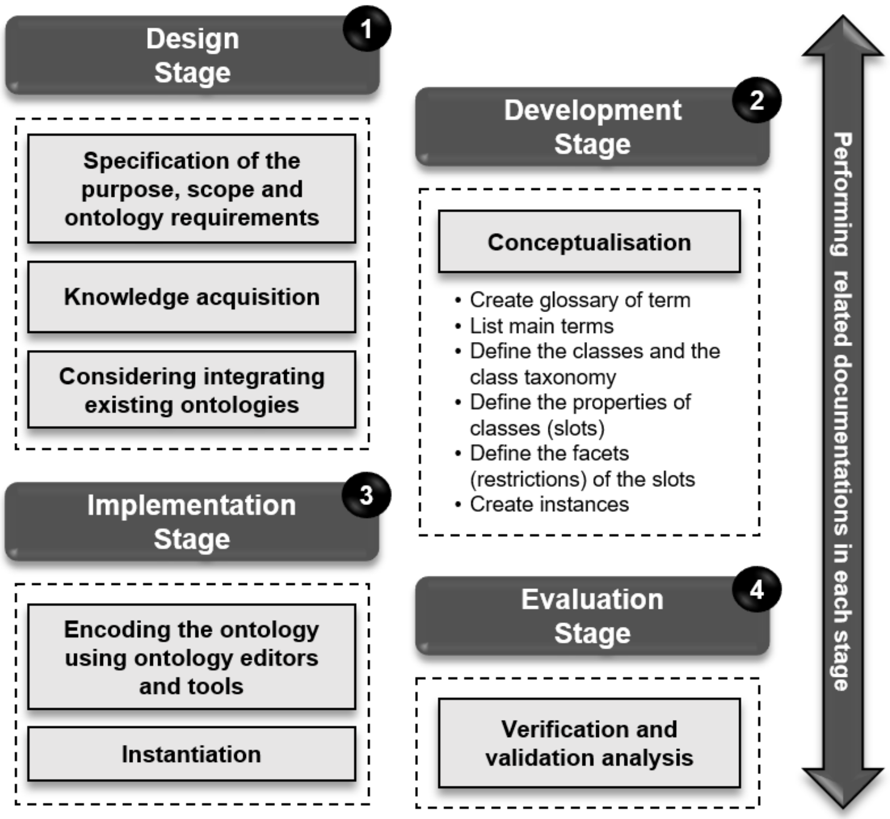

Design Stage
1
Specification of the purpose, scope and ontology requirements
Knowledge acquisition
Considering integrating existing ontologies
Development Stage
2
Conceptualisation
• Create glossary of term
• List main terms
• Define the classes and the class taxonomy
• Define the properties of classes (slots)
• Define the facets (restrictions) of the slots
• Create instances
Implementation Stage
3
Encoding the ontology using ontology editors and tools
Instantiation
Evaluation Stage
4
Verification and validation analysis
Performing related documentations in each stage

Figure 3.1: Stages of the ontology development lifecycle

designing the MuCRL, a set of design criteria [206] (e.g., clarity, completeness, etc.) is also considered, with the aim of knowledge sharing and maintaining interoperation among software systems.

- **Step 3: Considering integrating existing ontologies**

Existing ontologies and clinical information systems are reviewed for concepts, already defined and widely accepted in practice. The review included: (i) terminology classifications, such as the UMLS (Unified Medical Language System) [207] which involves, for example, the UMLS Semantic Network [208], which represents semantic concepts and their relationship and the SPECIALIST Lexicon – a lexical resource for biomedical words; (ii) Metathesaurus schemata involving clinical vocabularies (e.g., SNOMED–CT [131], LOINC [132], ICD-10 [209], and RxNorm [210]), where clinical terms and associated codes are defined; and (iii) existing ontology libraries (e.g., BioPortal [211], The Open Biological and Biomedical Ontology Foundry [212]). Moreover, existing literature (see Section 2.4) representing and managing guideline actions and approaches for handling multiple formalised guidelines, in order to manage care pathway of a multimorbid patient, are also examined; and lastly, several web sites such as DrugBank [213] and Drugs.com [214] to support representation of drug interactions, and the workflow control patterns [215] to support representation of control-flow and hierarchical structuring of guideline activities are also examined.

- **Step 4: Conceptualisation**

The ontology involves classes, their properties, constraints on properties, and instances to represent the intended knowledge in the domain. Initially, the glossary of terms is created where terms are enumerated and grouped to create the ontology. Then, classes and the class topology are defined. To develop MuCRL's classes, the middle-out approach is adopted that combines top-down approach (i.e. enabling one to control the details of the ontology) and bottom-up approaches (i.e. supporting to develop high level of details); and supplies balance between them [216]. Then, the properties of classes, relations, axioms and restrictions (facets) on them, are defined. Every term is checked for relevance, duplication, appropriate level of granularity and semantic alignment with the intended objectives of the thesis.

- **Step 5: Ontology encoding**

The objective of this step is to implement the developed ontology using a software system or tool. Firstly, guidelines are formalised based on MuCRL using the Protégé editor [142], which supports OWL2 [63], a widely-used Semantic Web ontology language. Afterwards, OWL2 metamodel is mapped to Ecore metamodel of EMF [64] (see Section 2.3.3). This mapping is performed to support the reusability and sharing of the ontology, which means users can import and export the entire or part of the ontology among different software platforms as well as to support the dynamic execution and verification using the power of imperative programming.

- **Step 6: Instantiation**

Instantiation refers to defining knowledge elements of guidelines and/or their interrelations in a class. In this thesis, instantiation is manually performed. Each health condition is associated with a particular guideline which has its own CIG which represents guideline knowledge constructs and its associations to construct the care pathway of a patient. If multimorbidity exists, instances of CIGs, which represent different diseases, need to be aligned (i.e. creating semantic relations and correspondences between knowledge elements of guidelines through finding similarities between them) and mapped [217] (i.e. finding directed alignments-semantic overlaps between similar elements in different CPGs and their associations). Ontology mapping [217] can be defined as finding equivalences and semantic similarities between knowledge constructs.

Multiple CIGs and their associated clinical actions may need to be merged one or multiple times in a care pathway, in order to obtain consolidated guideline recommendations (e.g., aligning with the intended care objective, eliminating care duplications and conflicting advices). CIG merging can be defined as follows:

Definition 3.3.1. (CIG Merging) Merging is the ability of a clinical decision support enabled system to be able to execute CIGs in parallel for creating automated mappings and adjustments between them when needed based on a patient's health context.

To perform CIG merging, required semantics need to be supplied. In the context of this thesis, this can be achieved by matching and mapping many guideline instances over classes, properties and instances which are supported by the multi-activity management (MAN) ontology group (see Section 3.5.4).

- **Step 7: Ontology evaluation**

In this stage, a set of ontology verification and validation analyses is performed. These analyses are required to check whether the developed ontology coherent and free from inconsistencies and working correctly for its intended purpose. While verification is about checking the system application (e.g., syntactic errors, missing information, logical consistency), validation checks the correctness of the developed ontology (e.g., clinical validity), see Section 2.4.3. Ontology reasoners [92, 93] are used to identify inconsistency and incoherency occurring while developing the ontology. Ontology evaluation metrics [218, 219, 220, 221, 222] are used to assess the semantic quality of the ontology. Ontology requirements specification and end user requirements also need to be met, as part of the ontology validation. The ontology is also tested with different guidelines and patient scenarios to evaluate its representational efficacy. Thus, the evaluation step enables users to compare the goal and completeness of several meta-ontologies, their shareability and reusability with the proposed ontology; and determine whether or not it is useful and applicable to its intended purpose (i.e. representing knowledge elements of guidelines and their interrelations and required semantics to manage them) [60]. The evaluation phase is discussed extensively in Section 6.3. The following sections discuss the results of the ontology development stages.

3.4 MuCRL Design Results

As part of the CIG language design phase, the ontology specification requirements involve the following information: (i) the purpose of the ontology (determined by a set of competency questions and answers see Table 3.1), is to develop multiple computerised guideline representation language to represent knowledge constructs of guidelines and their interrelations to manage multimorbid patients in a CDSS setting; (ii) the intended users of the ontology can be knowledge engineers, HCPs, health organisations, researchers, and academicians. When a CDSS that has been modelled using the language, then knowledge engineers and HCPs can work together to create the instances of the given ontology. Accordingly, the end-users can be HCPs, who supply care, based on the given recommendations, and patients who receive care; and (iii) the conceptualisation and implementation details (which are discussed extensively in the following sections).

Table 3.1: Competency questions and their answers

Competency Questions	**Answers**
"Why a new ontology is needed to represent and manage knowledge elements of guidelines and their interactions?"	MuCRL will be used as a reference model and a tool for representing and managing multiple concurrently implemented guidelines and their interrelations. Consequently, combined and consistent personal care plan for a multimorbid patient can be generated which is still an open area in the published literature.
"How to represent different guidelines with different scope and granularities through a formal and clear representation?"	CIGs (formalised guidelines) use the semantics defined in MuCRL (i.e. involves four groups). Accordingly, Guideline Service Deployment and Healthcare Service groups are introduced for representing guideline specific information and knowledge constructs required for establishing a patient care pathway, respectively. Patient Care Personalisation group is introduced for representing patient-specific information. Multi-Activity Management group is introduced for multi-activity coordination, care modification and optimisation issues.
"How formalised knowledge elements of guideline and their interrelations can be managed in a personal care plan?"	In the context of this thesis, multimorbid patients can be managed by supplying a personal care plan for each patient based on their health needs. Once the health needs of a multimorbid patient is determined, CIGs are transferred to the personal care plan where dynamic updates (e.g., modification, merging, etc.) on their actions will be performed over this plan.

The knowledge used for the development of MuCRL are determined with the following steps:

- Performing a systematic literature review [31] where gaps are identified in terms of the expressive power of current approaches in CIG applications, with respect to multi- or co-morbidity care;
- International guidelines, such as the GIN, the ICSI, the National Institutes of Health are analysed, but mainly publicly available NICE guidelines in the UK are used, where these exist in the form of texts, figures and tables;
- The time ontology from W3C, namely, OWL-Time [223] and HL7 RIM v3 [59] which are meta-ontologies from BioPortal [211], have influenced the identification of temporal constraints and related timing information. HL7 RIM v3 has also influenced the activity execution states. In addition, SNOMED–CT [131] has influenced the patients' observable entities and the instances of the defined concepts, mainly the Guideline Service Deployment group of the ontology. The SNOMED-CT codes are used as clinical IDs in the instantiation phase of the ontology with the aim of knowledge standardisation. The Drug-Drug Interactions Ontology [224] also supported the design of the concepts related with the representation of adverse interactions;
- Official web sites as DrugBank [213] (*www.drugbank.ca*) and Drugs.com [214] (*www.drugs.com*) representing drug information and adverse drug interactions such as drug-disease, and drug-disease interactions are used for obtaining medication details (e.g., active ingredient) and their conflict degree (e.g., minor, moderate, severe) with other medications;
- Guideline design patterns [225] and workflow control patterns [30, 57] and its official website as Workflow Pattern [215] (*www.workflowpatterns.com*) (see Section 2.4.2) are used to develop the care workflow of a patient;
- Existing well-known CIG languages (see Sections 2.4 – 2.6) such as PRO*forma*, GLARE, GLIF3, SAGE, EON, Asbru, and Arden Syntax involving case studies in representing and managing guidelines are analysed to check whether common elements exist for reuse or missing element exists to contribute to the literature.

The objective of this thesis is to fill the gap of representing and managing concurrently implemented multiple guidelines by supplying a comprehensive ontology which can be used for modelling multiple CIGs, to perform their concurrent executions and verifications. Especially, multi-activity management ontology group and their associations are the major contributions of this thesis (see Section 3.5.4). The following section presents the ontology development results.

3.5 MuCRL Development Results

This section presents the ontology conceptualisation results. MuCRL consists of four conceptual ontology, Θ, groups, representing distinct facets of care. These conceptual groups are as follows:

- GSD: Guideline Service Deployment
- PCP: Patient Care Personalisation
- HES: HEalthcare Service
- MAN: Multi-Activity maNagement

Development started with the definitions of guideline and HCP metadata. Thus, GSD ontology group is initially introduced. Afterwards, required elements (e.g., temporal constraints, clinical activities and their associations such as examinations, procedures, etc.) for establishing patients' care pathway are developed. HES ontology group is introduced for this purpose. The development continued with multi activity management elements required for managing multiple concurrently implemented care pathways, involving elements required to handle conflicts and/or optimise care. Thus, the MAN ontology group is introduced. Lastly, the development of metadata for representing patient-specific information is performed. The PCP ontology group is introduced accordingly.

While developing the MuCRL ontology, the class taxonomy in the form of class-subsumption (subClassOf) relation (e.g., isA), is created. Afterwards, the properties of classes, axioms and restrictions (facets) on them, are defined. In the following sections, all the classes belonging to all ontology groups and associated properties are presented. In Figure 3.2, all the classes ($\mathcal{C}$) of the ontology are presented. The lists of object and data properties of MuCRL are also presented in Appendix A. In the text, selected axioms and facets (e.g., exactly, some, min, max) can only be shown due to the space limitation. However, the full lists of classes, and their properties ($\mathcal{R}$), the instances ($\mathcal{I}$), the axioms ($\mathcal{A}$) and associated restrictions are available in the ontology file presented in Section 7.5.

To illustrate, a portion of data and object properties with associated domain and range information is presented in Table 3.2. Class names, in the ontology, are defined with a sans serif font (e.g., Decision), object and data property names are defined with lower italic case letters (e.g., *hasIncomingTransition*), quotation marks are used to represent the data property values (e.g., "2"), $< \cdots >$ is used to represent class instances (e.g., <CIGDBDA1> which denotes the first instance of the

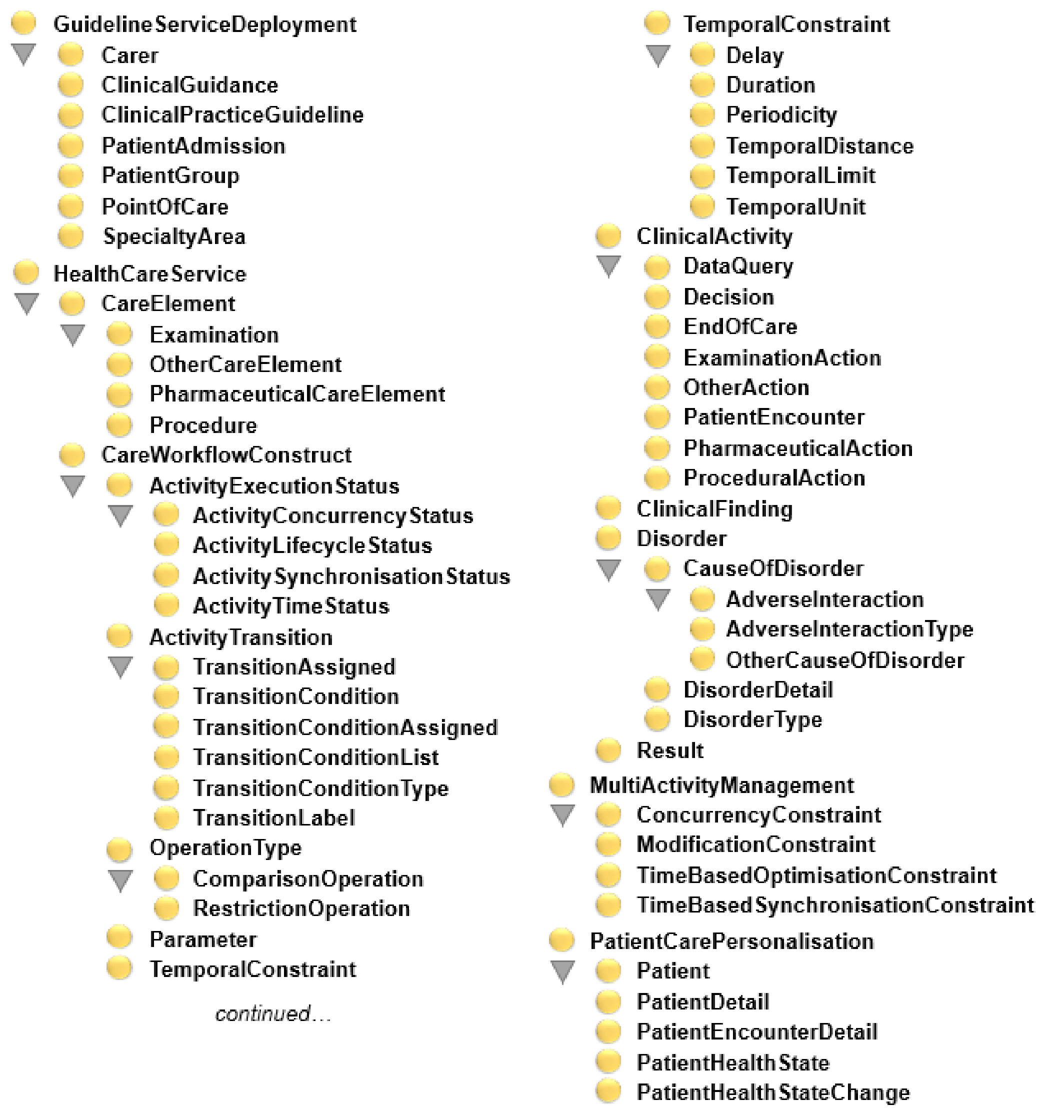


Figure 3.2: Class hierarchy of MuCRL ontology

Decision class of diabetes guideline and <PECAPP0001..> represents an instance of a combined personal care plan (PECAP) of a patient whose patient ID is P0001).

Instances can be knowledge elements of individual guidelines and their interrelations defined in a class or they can be members of enumerated classes such as the PatientAdmission whose instances are inpatient, and outpatient. In MuCRL, a meta-ontology is shaped as a graph, where nodes represent instances of classes, and edges represent the relationships between them. Ontology involves a set of data types as string, integer, float, double, long, and data properties define the relationship between an instance and data values such as the *definition* data property whose data type is string and its data value is a statement defining the instance. In the following sections, four ontology groups of MuCRL are presented.

Table 3.2: Excerpt of data and object properties of MuCRL with associated domain and range values

Property	Type	Domain	Range
examinationName	Data	Examination	xsd:string
modificationTime	Data	ModificationConstraint	xsd:dateTime
reuseCareElementOf	Object	TimeBasedOptimisationConst.	ClinicalActivity
hasTemporalDistance	Object	TimeBasedSynchronisationConst.	TemporalDistance
hasConcurrency	Object	ClinicalActivity	ConcurrencyConst.

3.5.1 Patient Care Personalisation

The Patient Care Personalisation (PCP) group is introduced with the PatientCarePersonalisation, represents information about the patient. PCP has five sub-classes, and these are used to represent patient-specific information such as demographics, health state and patient encounter details. PCP is required for patient care personalisation. To illustrate, the following example is presented:

Example 3.5.1. Patient whose patient ID is <P0001> diagnosed as hypertension (HTN) six months ago. Today, he presented to the hospital with high blood pressure again with the signs of chronic heart failure (CHF). His health condition is recorded as moderate. Then, patient details were updated.

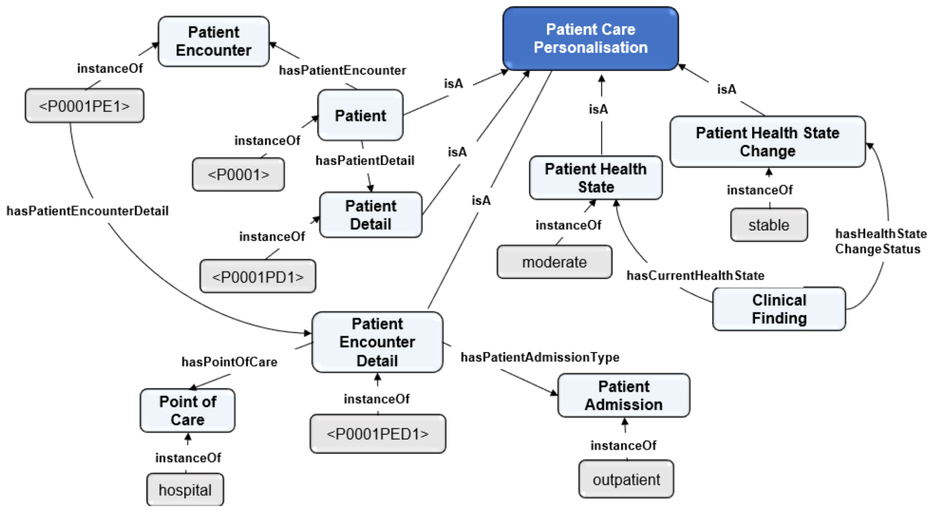


Figure 3.3: Illustration of PCP ontology group

Figure 3.3 illustrates how PCP ontology can be used for representing patient-specific information based on Example 3.5.1.

Patient-Specific Information

Initially, the Patient is introduced to represent the initial registration details (e.g., patient forename, age, birth date, ID, gender, etc.) - patients (e.g., <P0001>) are the instances of this class. The *hasPatientEncounter* object property links the instance of the Patient with the instances of the PatientEncounter (e.g., <P0001PE1>) where patient encounter details (PatientEncounterDetail ,e.g., <P0001PED1>) are defined. The *hasPatientDetail* object property links the instances of the Patient with the instances of the PatientDetail (e.g., <P0001PD1>) where more information about the patient can be defined such as weight, occupation, country of residence, lifestyle information, last update on patient details and many others.

Patient health state must be recorded, because care may need to be managed based on this information. Accordingly, the PatientHealthState is introduced which represents current health condition of the patient as an enumerated class. The instances of this class involve good, moderate, severe and unknownSeverity. Since the patient health state is dynamic, any health change on the patient must be recorded. Thus, the PatientHealthStateChange can be used to represent the changes in patient health state as an enumerated class. The instances of this class involve decreasing, improving, increasing, stable, worsening and otherChangesInHealthState.

Patient Care Encounter Information

Representation of patient encounter information which can be with a medical centre or any care point (e.g., home, hospital) is important because this supports recording patients' visits to medical centres or other care points.

The PatientEncounterDetail represents the details of the patient encounter. Thus, the *hasPatientAdmissionType* object property links the instance of this class with exactly one instance of the PatientAdmission class where patient admission type (e.g., inpatient, outpatient) is defined. The *hasPointOfCare* object property also links the instance of the PatientEncounterDetail with instances of the PointOfCare to define the place of care (e.g., hospital, general practice, home). Lastly, the *admissionDate* data property represents the date of admission to the medical centre or start of care in any other care points.

3.5.2 Guideline Service Deployment

The Guideline Service Deployment (GSD) group which, is introduced with the GuidelineServiceDeployment, represents individual CPGs and associated care information in a structured way. The following sub-sections presents the sub-classes, object and data properties of GSD, see Figure 3.4.

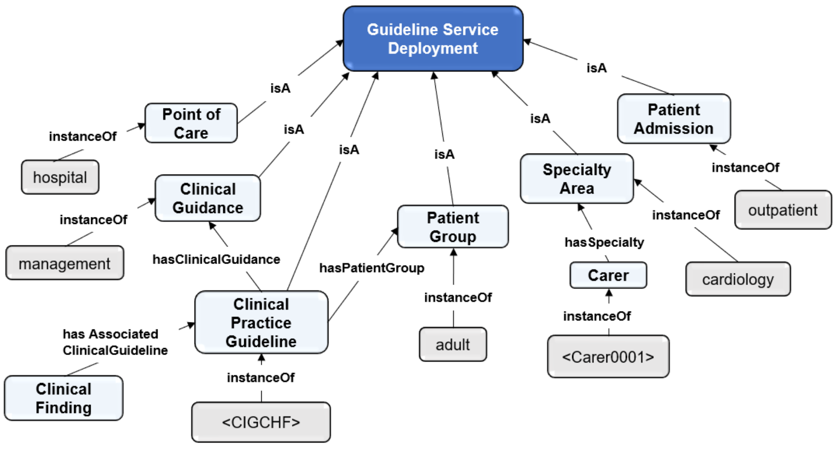


Figure 3.4: Illustration of GSD ontology group

Care Information

GSD can be used to represent care information regarding carer (also called as HCP) details who supply care, involving his/her clinical specialty as well as patient admission details to the primary and/or secondary care. Accordingly, the Carer represents information about the responsible HCP of a care step involving carers' phone number, forename, surname, ID and email. The main aim of this class is to create a repository for HCPs, which are then assigned as a performer to a related clinical activity(s). Each clinical activity must have minimum one carer.

The *hasSpecialty* object property links the instances of the Carer with the instances of the SpecialtyArea to represent the clinical specialty relating to the disease in question as an enumerated class. To illustrate, cardiology, diabetologyAndEndocrinology, and dentistry are some of the instances of this enumerated class. Based on the Example 3.5.1, the carer (e.g., <Carer0001>) of the considered patient is cardiologist and his area of specialty is cardiology.

Care points where patients receive care are also represented. To do so, the PointOfCare is introduced to represent care points as an enumerated class with the following instances: home, hospital, generalPractice, otherCarePoint. To supply care for a patient, his/her type of admission to the medical centre or any other care point must be recorded. Thus, the PatientAdmission, represents the type of patient admission as an enumerated class. The instances of this class involve inpatient (i.e. patient who is admitted by a medical centre and stays overnight for the treatment) and outpatient (i.e. patient who receives a treatment by a medical centre without overnight stay).

As a result, information regarding who will perform the care and where it is performed need to be represented through using the above-mentioned classes before supplying any care to a patient.

Clinical Practice Guidelines and Their Associations

Clinical practice guidelines are used to support HCPs in their clinical decision-making phases and supplying care recommendations to a patient based on his/her health condition(s). Thus, the guideline information must be represented while developing a care pathway for a patient. Thus, the ClinicalPracticeGuideline is introduced which represents the metadata of guidelines. Each clinical guideline is associated with a type of clinical guidance and a target patient group. The *hasClinicalGuidance* object property links the instances of this class with the instances of the ClinicalGuidance to define related clinical guidance information of guidelines as an enumerated class. Some of its instances are assessment, diagnosis, prevention, treatment, and management. The *hasPatientGroup* object property links instances of the ClinicalPracticeGuideline with the instances of PatientGroup to define the patient group where the type of patient group is defined as an enumerated class. The instances involve adult, child, elderly and infant. For example, an instance of the ClinicalPracticeGuideline can be <CIGCHF> which represents Chronic Heart Failure (CHF) in adults: management guideline [226].

3.5.3 Health Care Service

Healthcare Service (HES) group, which is introduced with the HealthCareService, represents the common guideline knowledge and its associations, and workflow control mechanisms [57] required for their implementations. The following sub-sections present the sub-classes, object and data properties of HES.

Clinical Activities and Their Associations

The ClinicalActivity denotes a set of clinical activities that will be performed in a patient care pathway.

This class has a set of sub-classes (see Figure 3.5), which are as follows:

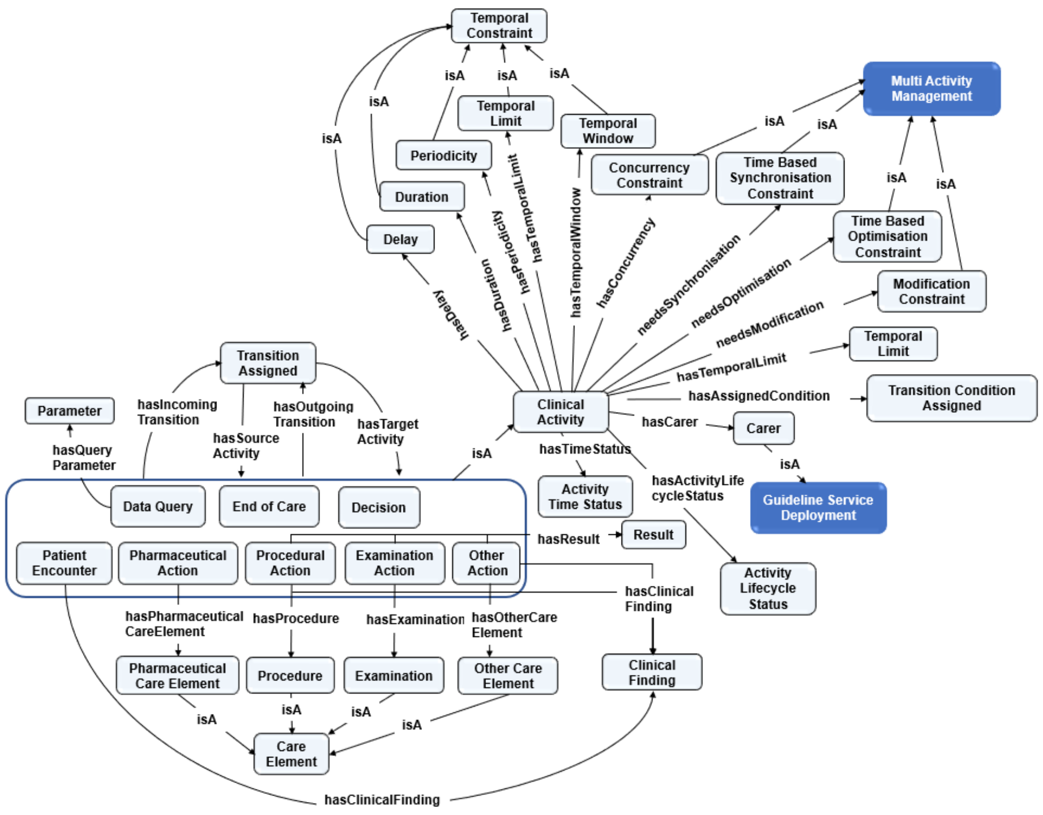


Figure 3.5: Illustration of clinical activity class relations of HES ontology group

- The PatientEncounter represents the entry point of care where initial patient-HCP encounter is realised. HCP defines the initial clinical findings. The *hasPatientEncounterDetail* object property links the instances of the PatientEncounter class with the PatientEncounterDetail to define patient-specific information such as patient admission type and admission date, and *hasClinicalFinding* object property links the PatientEncounter with the ClinicalFinding to define patient-specific consultation notes such as clinical findings about the disorder of the patient, current health state and associated CPG with the disease if the disorder is diagnosed;

- The Decision represents minimum two care options based on a set of conditions. For instance, "existence of fluid overload?" has two conditional options as yes/no, but some decisions can involve alternative conditional options where more than two conditions may involve;

- The DataQuery represents data query actions such as whether or not the lab test ready or patient information (e.g., existing drug use). The *hasQueryParameter* object property links the instances of the DataQuery with the instances of the Parameter to define the query parameters. For example, "is the medication x in use?" or "lab test result is ready?";

- The PharmaceuticalAction represents pharmaceutical recommendation actions. Each pharmaceutical action must have one pharmaceutical care element (Pharm aceuticalCareElement) where the details of the medication are defined (e.g., active ingredient of medication, its dose level, side effects, etc.). For example, Diuretic therapy is a pharmaceutical action and its pharmaceutical recommendation is the Diuretic product. The carer prescribes this medication to Patient <P0001> once a day for 3 months;

- The ExaminationAction represents examination actions such as lab test requests, or blood pressure measurement. Each examination action must have one examination care element (Examination) to define the details of the examination (e.g., blood pressure measurement). The *hasResult* object property links instances of the ExaminationAction with instances of the Result to represent the values of the requested examination results;

- The ProceduralAction represents procedural actions in a care pathway such as diagnostic procedure, laboratory procedure (e.g., analysis of lab results), or follow-up;

- The OtherAction represents clinical actions and associated details that cannot be expressed with the other clinical activities. This class is designed to capture knowledge which was not covered by other classes of the clinical activities;

- The EndOfCare represents the end of care/conclusion.

These activities have common data and object properties which are as follows:

definition. Each clinical activity must have an appropriate definition such as "Medication recommendation is made to manage high blood pressure level of the patient". The definition data property is a common property that each element of the ontology has a definition.

activity transitions. Each clinical activity needs a transition (namely, an arc) to follow the next clinical activity. Thus, the *hasIncomingTransition* object property to represent incoming activity transitions and the *hasOutgoingTransition* object property to represent outgoing activity transitions are used. These object properties link the instances of the clinical activities with instances of the TransitionAssigned to represent the transition information (e.g., transition label, source and target activities) between clinical activities.

carer. Each clinical activity must have minimum one HCP who performs the clinical activity.

time. Each activity must have two associated time statuses (TimeStatus)(e.g., actual/expected start or end time), and related activity start and end time information which can be defined as the *activityStartTime* and the *activityEndTime* data properties.

temporal constraints. Each clinical activity is associated with a set of temporal constraints such as delay, periodicity, and duration. These are extensively defined under the TemporalConstraint.

identification codes. Identification codes of the clinical activities and interrelations are defined for each clinical activity. To do so, the clinicalID data property, which represents the codes of clinical terminology standards (e.g., SNOMED-CT, ICD, or LOINC), is used to improve semantic interoperability and maintaining consistency in the instantiation phase. This is crucial for eliminating care duplications, which can be possible by detecting identical care recommendations with their clinical IDs. For instance, the data value "717854002" represents the Aspirin therapy associated with a pharmaceutical action. The sources of codes can also be defined using the *sourceOfClinicalID* data property as "SNOMED-CT (STC)" or other sources.

execution status. Each clinical activity has exactly one activity lifecycle status, e.g., done, pending, etc., at a time. *hasActivityLifecycleStatus* object property links each clinical activity with the ActivityLifecycleStatus.

multi-activity management. When multiple clinical activities are concurrently implemented, they may cause care conflicts or duplications. Thus, clinical activities may need to be modified by the HCP to avoid them. Accordingly, activities may need to be synchronised for merging at the following common clinical activity or may need to be optimised to avoid care duplications. Accordingly, the following object properties of clinical activities are presented to handle these issues. These properties are linked with the MultiActivityManagegement ontology group which is extensively discussed in Section 3.5.4:

- *needsSynchronisation* object property links the instances of clinical activities with the instances of TimeBasedSynchronisationConstraint to handle synchronisation needed clinical activities to avoid care duplications;
- *needsOptimisation* object property links the instances of clinical activities with the instances of the TimeBasedOptimisationConstraint to perform required care optimisations such as reusing a clinical activity instead of performing a new one if it is performed in a reasonable time period;
- *needsModification* object property links the instances of clinical activities with instances of the ModificationConstraint to perform required care modifications to replace existing care element (e.g., drug) of a clinical activity with its safe alternative or modify its time element (e.g., starting time) to avoid potential conflicts and/or duplications;
- *hasConcurrency* object property links the instances of clinical activities with the instances of the ConcurrencyConstraint to define concurrency relations between clinical activity pairs.

conditions. Each clinical activity is associated with a set of conditions (e.g., activating, precondition, outcome or end) which must be satisfied to follow the next clinical activity. The *hasAssignedCondition* object property links the instances of clinical activities with the instances of the TransitionConditionAssigned to represent related transition condition details and associated restrictions.

Care Elements of Clinical Activities

The CareElement represents care elements of examination, pharmaceutical, procedural and other actions. These involve; (i) the PharmaceuticalCareElement represents details of the pharmaceutical actions and related information such as pharmaceutical element name, its definition, active ingredient, dose, application dose, and cost information. The type of drug therapy can also be defined with the *firstLineTherapy* (i.e. initial treatment given for a disease) whose value can be "true" or "false" or other type of therapies (e.g., secondary) can be defined with the *otherTypeOfTherapy* data property whose data type is string. The *possibleSideEffect* data property can also be used to state the possible side effects of a medication; (ii) the Examination represents examination details such as examination name (e.g., CT-scan, blood pressure measurement, etc.), examination definition and associated cost information; (iii) the Procedure represents the details of the procedural actions (e.g., diagnostic, follow-up, etc.); and lastly (iv) the OtherCareElement represents care elements which cannot be expressed with the aforementioned care elements. For instance, this class can be used to represent the instances of non-pharmaceutical care recommendation elements (e.g., lifestyle or diet advices).

Clinical Findings and Associations

The ClinicalFinding is introduced to represent patient-specific consultation notes (e.g., the current health state, health state changes). The *hasDisorderDetail* object property links the instances of this class with the instances of the DisorderDetail to define the disorder details (e.g., diagnosis of cancer). The *hasCurrentHealthState* and the *hasHealthStateChangeStatus* object properties link the ClinicalFinding with the PatientHealthState and the PatientHealthStateChange to define current health state of the patient and changes in his/her health condition, respectively. Lastly, the *hasAssociatedClinicalGuideline* object property links the instances of the ClinicalFinding with the instances of the ClinicalPracticeGuideline where guideline (e.g., author, version, name of the guideline, etc.) and clinical guidance (e.g., management , diagnosis, etc.) information are defined.

Disorders and Their Causations

The Disorder represents patient disorder information based on the findings on his/her health condition(s) involving disorder types (e.g., disease, sign, symptom, allergic condition, injury, pathologic function and other disorder types) and (potential) disorder causes. This class has three sub-classes as:

- The DisorderDetail represents information related with the possible and/or existing disorders. The *hasDisorderType* object property links the instances of this class with the instances of the DisorderType enumerated class (e.g., sign, allergicCondition, symptom).

- The CauseOfDisorder represents definitions of causes of disorders involving AdverseInteraction, AdverseInteractionType and OtherCauseOfDisorder classes. The cause of disorder, induced by adverse interactions can be defined, as follows: *isACauseOfDisorder* object property links the instances of the DisorderDetail to instances of the AdverseInteraction. The AdverseInteraction defines the cause of disorder if it is induced by adverse interactions of care actions or patient allergy. The AdverseInteractionType enumerated class represents the type of adverse interactions as an enumerated class whose instances are timing, drugDisease, drugDrug, drugFood, drugPatient, and otherInteraction. The other causes of disorder can be defined as follows: the *hasOtherCause* object property links the instances of the DisorderDetail to the instances of the OtherCauseOfDisorder to define the causes of disorder which are different from adverse interactions where the details of the cause of disorder are defined.

- The DisorderType represents the type of disorders. Some of the instances of this class involve allergicCondition, disease, adverseInteraction, sign, symptom, and otherDisorder. Disorder can have some causes, which can be associated with adverse interactions (also called conflicts) or some other causes.

Handling Results of Clinical Activities

The Result represents the results (e.g., blood pressure level) of a clinical activity, such as results of examinations, measurements and/or any other clinical observations. The *hasResultParameter* object property links this class to the instances of the Parameter class to define result parameters (e.g., a systolic blood pressure measurement) and associated values (i.e. results). The *booleanResult* (data type: boolean), the *qualitativeResult* (data type: string) and the *quantitiativeResult* (data type: string) are data properties of this class to represent result types. To illustrate, a value of the *quantitiativeResult* can be "systolic blood pressure result" and its associated value (e.g., "120.0") that needs to be defined under the Parameter using the *parameterNumericValue* data property whose data type is float.

Care Workflow Constructs

The CareWorkflowConstruct represents the elements required to construct a care workflow. This involves the representation of;

- **Parameters**

The Parameter represents the data query, or result parameters (e.g., lab test results). The data properties of this class involve the *parameterName* and the *parameterDefinition* to define parameters (e.g., blood pressure level). The *parameterNumericValue*, the *parameterOrdinalValue*, and the *parameterTextValue* represent the parameter value based on its parameter type. For instance, "140.0" can be a value of the *parameterNumericValue* data property. The *scaleTypeNumeric* data property represents a boolean parameter type as "true" if the parameter value is numeric, otherwise, "false". Lastly, unit data property represents the unit of the parameter as string such as "mmHg".

- **Temporal Constraints**

The TemporalConstraint represents a set of temporal constraints involving exact time, and interval time periods [min, max] [50, 121] in float data type regarding the application of a clinical activity. Accordingly, (i) the Duration represents the duration of a clinical activity (e.g., 3 months); (ii) the Periodicity represents the periodicity of an activity with its associated frequency (cycle) (e.g., every day 2 times); (iii) the TemporalLimit represents allowed time limit -deadline- of performing an activity. In this class, exact, minimum and maximum temporal limits as well as deadline of a clinical activity as a dateTime data type (e.g., "20181005T12:00:00") can be defined; (iv) the TemporalDistance represents time required to reach a time point which can be required to synchronise multiple clinical activities or reuse the result of a clinical activity; and (v) the TemporalWindow represents the allowed time limit that a clinical activity's care element can be reused by other clinical activities or to use the associated information for clinical decision-making. The *hasTemporalUnit* object property links instances of the aforementioned temporal constraints with (vi) the TemporalUnit which represents the time units as an enumerated class. The instances of this class involve day, hour, minute, month, second, week and year.

- **Activity Transitions and Operations**

The ActivityTransition represents a set of transition types, transition labels (i.e. name of the transition) and transition conditions that need to be satisfied in order to activate the transitions between clinical activities. There are six sub-classes of this class (see Figure 3.6), as discussed below.

The TransitionAssigned represents the assigned transitions between guideline activities involving a transition label. This class is required to sequence clinical activities. The *hasSourceActivity* and the *hasTargetActivity* object properties link the *TransitionAssigned* with the ClinicalActivity to represent source and target activities of transitions, respectively. The *hasTransitionLabel* object property links the TransitionAssigned with the TransitionLabel enumerated class to assign a label.

The OperationType represents a set of operations for transition condition handling under three classes as (i) the ComparisonOperation, represents a set of comparison operations as an enumerated class (e.g., differentFrom, equalTo, greaterOrEqualThan, lessThan, notEqualTo, etc.); and (ii) the RestrictionOperation, represents

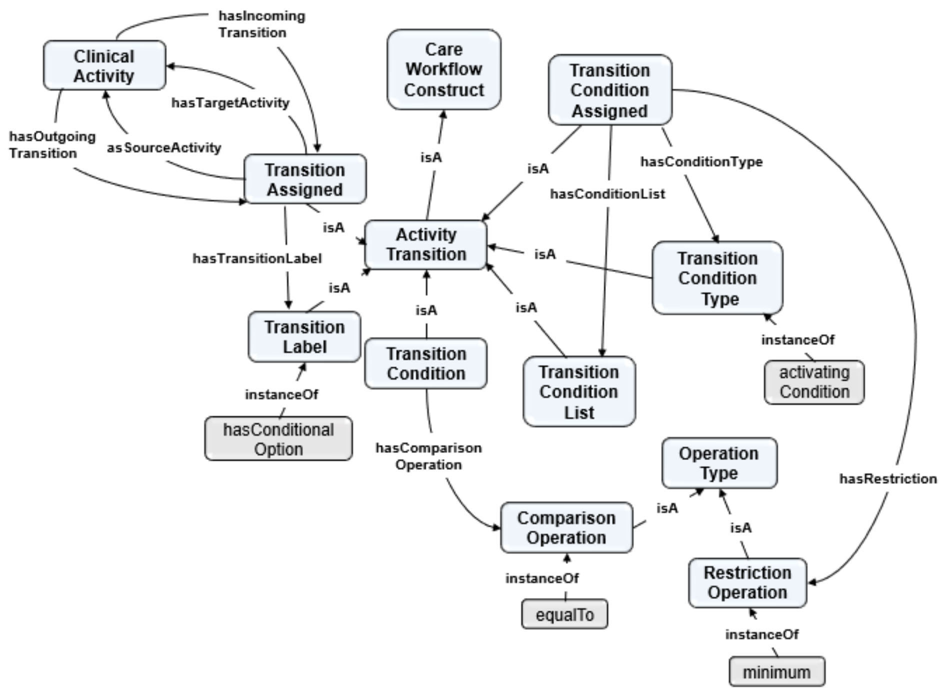


Figure 3.6: Illustration of clinical activity transitions of HES ontology group

a set of restriction operations as an enumerated class. The instances of this class involve all, any, maximum and minimum. This class is designed to define restrictions on conditions of transitions.

The TransitionCondition represents required conditions which need to be satisfied to initialise a clinical activity. The data properties *qualititativeValue* (e.g., "140.0", data type: float) and *quantitativeValue* (e.g., "true", data type: string) represents the value of the condition parameter, which represented with *conditionParameter* (e.g., "the presence of depression", data type: string) data property and unit data property represents the unit of the condition parameter value (e.g., "mmHg", data type: string). The *hasComparisonOperation* object property links the TransitionCondition with the ComparisonOperation to define associated comparison operation (e.g., equalTo) for condition parameter. This can be interpreted as, e.g., "the presence of depression" equalTo "true".

The TransitionConditionAssigned represents conditions which must be satisfied to perform the transitions between clinical activities. This class involves a list of transition conditions. Conditions are represented as a list because the transition of a clinical activity can be associated with multiple conditions. For this reason, the *hasConditionList* object property is introduced to link the TransitionConditionAssigned with the TransitionConditionList for associating these list of conditions. Condition list can involve one condition or multiple conditions. Yet, all the conditions at the same list must have same condition type. Thus, the *hasConditionType* object property links the TransitionConditionAssigned with the TransitionConditionType enumerated class to represent the type of transition condition. As defined above, a clinical activity can be associated with one condition or multiple conditions, and all of these conditions or some of them must be satisfied for the activation of the transition. For this reason, restrictions on conditions must be defined. Accordingly, the *hasRestriction* object property to links the TransitionConditionAssigned with the RestrictionOperation to define restrictions (e.g., minimum, all, etc.) on condition lists. The *hasRestriction* object property and the *restrictionValue* data property are used in a similar way with the TransitionConditionAssigned. The *restrictionValue* data property represents the number of condition lists (e.g., "4") that must be satisfied. To illustrate, minimum 2 out of 4 transition condition list or all of them must be satisfied.

The TransitionConditionList represents a bag of conditions which needs to be satisfied to initialise the transition. Each instance of the class involves one or more conditions from the TransitionCondition. Thus, *hasCondition* object property links the TransitionConditionList to get these conditions. The *numberOfCondition* data property represents the number of conditions (e.g., "1" or "6") whose data type is integer. The *isConditionMet* data property represents the status of assigned transition conditions with boolean data type. The value of this property can be "true", if conditions are satisfied. Otherwise, "false" that refers to required conditions for the transition are not satisfied.

The TransitionConditionType represents the type of transition conditions which need to be satisfied as an enumerated class. The instance of this class involves activatingCondition, preCondition, outcomeCondition and endCondition. The definitions of these condition types are: (i) activatingCondition is a required condition to activate a clinical activity which associated with conditional options of decision activities. For instance, "(blood pressure level $>$ 12.0) AND (diabetes exists $=$ yes)" if these two activating conditions of a decision activity are satisfied, then, say, a pharmaceutical action can be recommended; (ii) preCondition is a condition associated with data query activities. For instance, *queryParameter* $=$ "lab test result ready?" can be queried. If the (*conditionParameter* $=$ "lab test result ready") AND (*qualitativeValue* $=$ "false") ; then the precondition of the transition is not satisfied, and the next activity cannot be initialised; (iii) outcomeCondition is a condition to represent patient health outcomes, or the progression of the disease, such as (*conditionParam-*

eter = "tumor size") AND (*qualitativeValue* = "increased"); (iv) endCondition is a condition represents end condition of a clinical activity. For instance, if (*condition-Parameter* = "side effects occurred") AND (*qualitativeValue* = "yes"), then stop recommendation of the given drug.

The TransitionLabel represents the label of the transition which connects clinical activities to each other as an enumerated class. The instances of this class involve:

– StartingActivity represents the label of a transition associated with the starting activity;

– FollowingActivity represents the label of a transition to follow the next activity;

– AlternativeActivity represents the label of a transition associated with alternative clinical activities that can be preferred by a HCP;

– ConditionalOption represents the label of a transition from a decision activity to the next clinical activities if they satisfy required conditions. Each decision must have two conditional options;

– AlternativeConditionalOption represents the label of a transition from a decision activity to the next clinical activities. The difference between this type from the previous transition label is that AlternativeConditionalOption represents alternative conditional care options which can be taken into account by HCPs (i.e. n-to-m mappings);

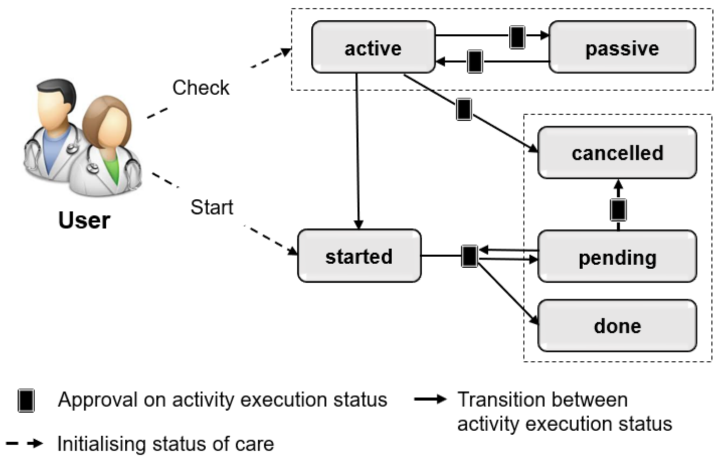
– Split represents the label of a transition regarding divergence from a single activity to multiple activities (i.e. 1-to-n mappings);

– isMergedAt represents the label of a transition that are merged at the subsequent clinical activity (i.e. n-to-1 mapping).

- **Activity Execution Status**

The ActivityExecutionStatus represents a set of activity execution states (also called status) under four group as follows.

The ActivityLifecycleStatus represents the instances of activity lifecycle statuses as an enumerated class. The instances of this class involve (i) active – represents up-to-date clinical activities and they are ready for execution; (ii) passive – represents clinical activities which are no longer in use; (iii) started – represents execution of clinical activities that are in progress; (iv) cancelled – represents aborted clinical activities and execution of these activities are terminated; (v) pending – represents clinical activities that waits information to continue their executions; and (vi) done – represents completion of the clinical activity execution. Figure 3.7 shows the activity execution statuses and their transitions.



active
passive
Check
cancelled
User
Start
started
pending
done
Approval on activity execution status
Transition between activity execution status
Initialising status of care

Figure 3.7: Activity execution statuses and their transitions

The ActivityConcurrencyStatus is an enumerated class which represents the instances of concurrency statuses of multiple clinical activities that are implemented in parallel. The instances of this class involve: (i) concurrencyStarted – represents clinical activities that have concurrent executions; (ii) concurrencyCompleted – represents the concurrency relations of activities as completed – if concurrent execution of the activities completed; and (iii) concurrencyDiscarded – represents cancellation of concurrent execution of activities. This class is used as part of managing concurrency relations of clinical activities of multiple concurrently implemented CIGs to avoid care conflicts. In Figure 3.8, the activity concurrency statuses and their transitions are presented.

The ActivitySynchronisationStatus is an enumerated class that represents instances of synchronisation statuses of multiple clinical activities at the designated care point. When activities reach this point, then they can be either merged or use results of each other and follow their respective care paths. The instances of this class involve: (i) toBeSynchronised – represents synchronisation required activities that need to reach a synchronisation point to merge or use results of other clinical activity, and this state represents the activity has not reached the synchronisation point yet; (ii) synchronisationPointReached – represents activities which reached to the synchronisation care point; and (iii) synchronisationDiscarded – represents activ-

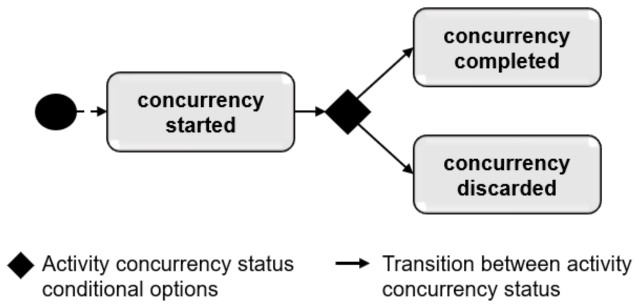

concurrency started
concurrency completed
concurrency discarded
◆ Activity concurrency status conditional options
→ Transition between activity concurrency status

Figure 3.8: Activity concurrency statuses and their transitions

ities whose synchronisations are cancelled. This class is designed as part of handling multiple clinical activities to avoid care duplications. In Figure 3.5.4, the activity synchronisation statuses and their transitions are presented.

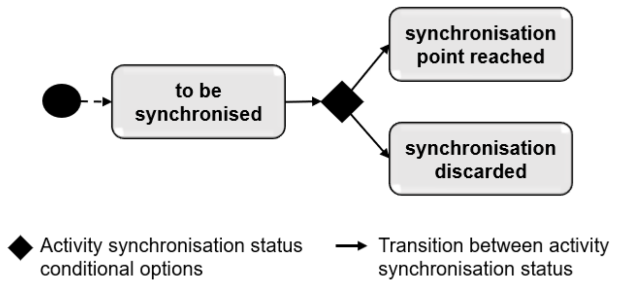

to be synchronised
synchronisation point reached
synchronisation discarded
◆ Activity synchronisation status conditional options
→ Transition between activity synchronisation status

Figure 3.9: Activity synchronisation statuses and their transitions

The ActivityTimeStatus represents time status of a clinical activity as an enumerated class. The instances of this class involve (i) actualStartTime; (ii) actualEndTime; (iii) expectedStartTime; and (iv) expectedEndTime. This class supports activity sequencing by providing actual and estimated temporal knowledge. To handle temporal uncertainty, a similar approach with Duftschmid et al. [50] is adopted. Time intervals as [min, max] are defined in temporal constraints (TemporalConstraints). Activity start/end time of clinical activities are represented as actual if they are performed. Otherwise, expected (estimated) start and end time of clinical activities can be supplied to support activity sequencing.

3.5.4 Multi-Activity Management

The Multi-Activity maNagement (MAN) group is introduced with the MultiActivityManagement ontology group (see Figure 3.10), which has four sub-classes, and designed to: (i) facilitate concurrent implementation of the same or (multiple) different guideline activities; (ii) combine activities of parallel guidelines that may involve delays and need synchronisation; (iii) handle reconciliation of activities (e.g., a recommendation to avoid a detected drug-drug interaction); and (iv) optimise the time and resource use. In the following sections, these are extensively presented.

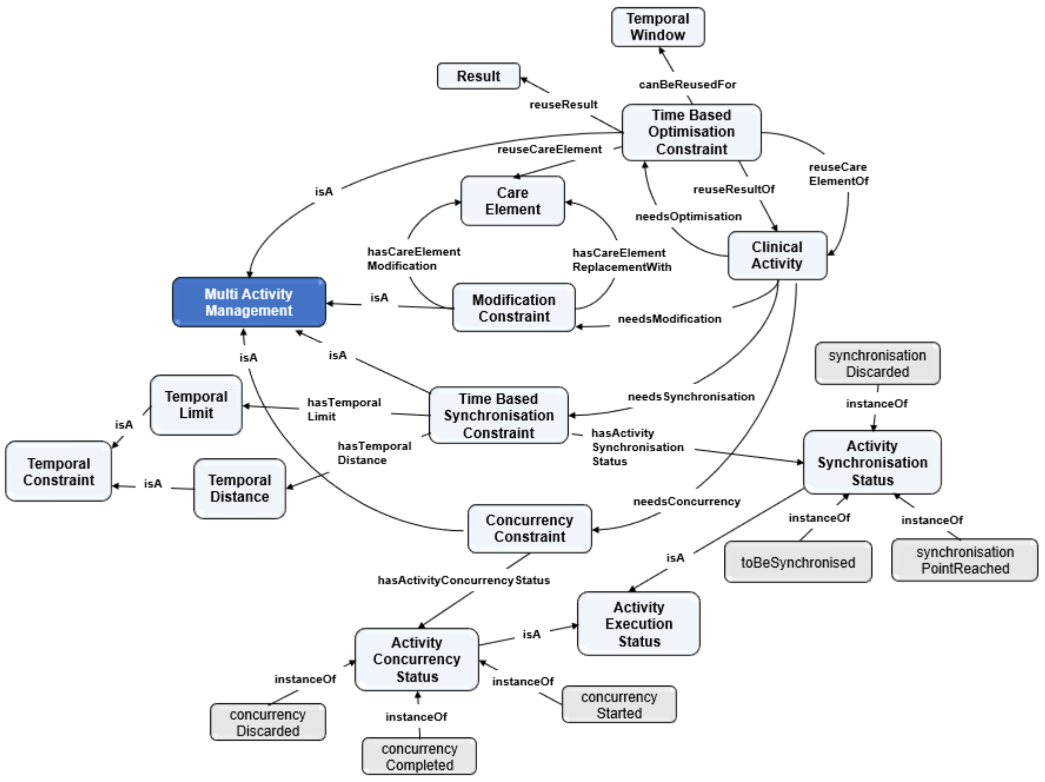


Figure 3.10: Illustration of MAN ontology group and their interrelations

Managing Multiple Clinical Activity Concurrency Relations

Clinical activities can be executed in order, concurrently (in parallel), unordered or periodically. Multiple activities can be merged at the chosen (synchronisation) care point. Figure 3.11 (1–5) presents a number of concurrency relations that result from parallel execution of guidelines.

The ConcurrencyConstraint defines the concurrently implemented two or more activities of (multiple) CPGs that can be merged, when they reach their synchronisation point (i.e. specific time point -deadline- defined for a set of clinical activities) or follow their respective care paths. However, by means of illustration in Figure 3.11, the relations between only two activities are shown, since the number of ordering relations exponentially increases when the number of activities increases. Here,

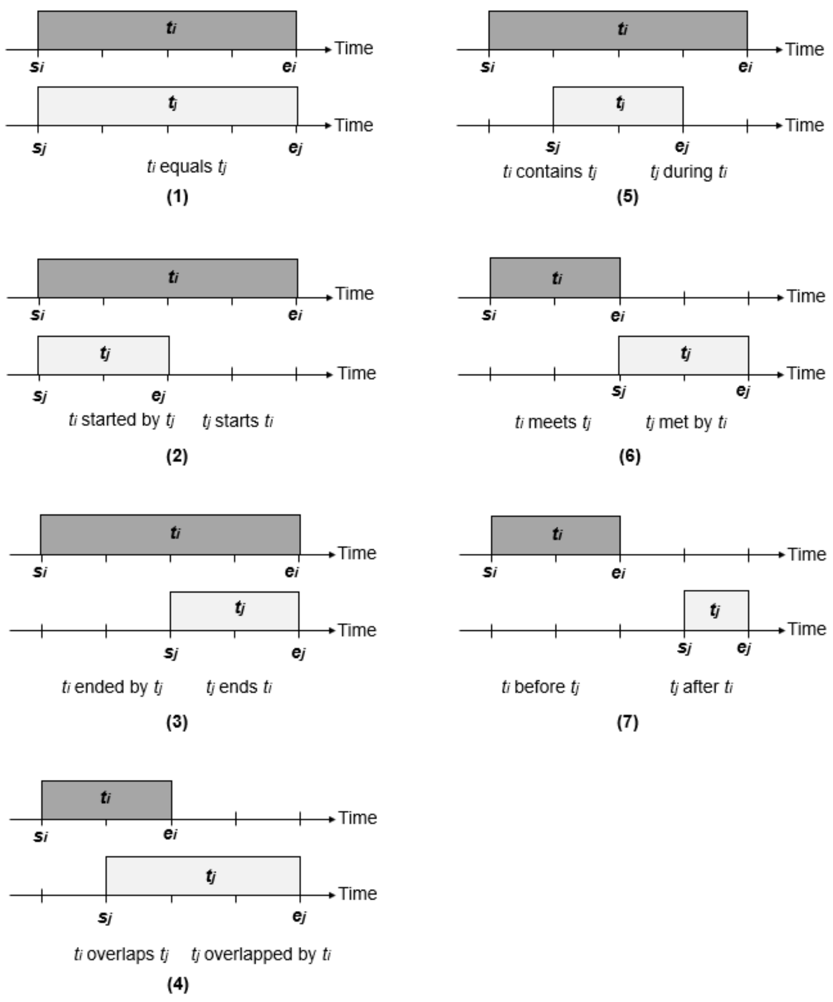


Figure 3.11: Types of temporal relations [227] between guideline activities

$t_i = e_i - s_i$ and $t_j = e_j - s_j$ denote time intervals of implementing the activity i and activity j, respectively, where $i, j = 1, 2, \ldots, N$. $(s_i,\ e_i)$ and $(s_j,\ e_j)$ denote the start and end time of these activities. Concurrency constraint can be defined with the following axioms:

$$
\begin{array}{llll}
\text{ConcurrencyConstraint} & \sqsubseteq & \text{MultiActivityManagement} & \sqcap \\
 & =1 & \text{hasActivityConcurrencyStatus.ActivityConcurrency-} & \sqcap \\
 & & \text{Status \{concurrencyStarted,} & \\
 & & \text{concurrencyCompleted, concurrencyDiscarded\}} & \\
 & =1 & \text{concurrencyStartTime.dateTime} & \sqcap \\
 & =1 & \text{concurrencyEndTime.dateTime} & \sqcap
\end{array}
$$

Concurrency constraints need to be used when concurrency between clinical activities are detected, where time information regarding the concurrency like the *concurren-*

cyStartTime and the *concurrencyEndTime* data properties can be defined; and based on the realisation of the concurrency, associated concurrency status can be defined and updated using the *hasActivityConcurrencyStatus* object property which links the instances of this class with exactly one instance of the ActivityConcurrenyStatus to define the concurrency status (e.g., concurrencyStarted, concurrencyCompleted and concurrecyDiscarded). To illustrate concurrency management, the following example is provided, as follows:

Example 3.5.2. A multimorbid patient whose patient ID is <P0001> has initially diagnosed with Diabetes (DB). Afterwards, he diagnosed with Hypertension (HTN), and Chronic Heart Failure (CHF), respectively.

Based on Example 3.5.2, a patient has both HTN and DB diseases. Thus, some of their clinical actions such as pharmaceutical actions have concurrency relations which need to be recorded (see Figure 3.12).

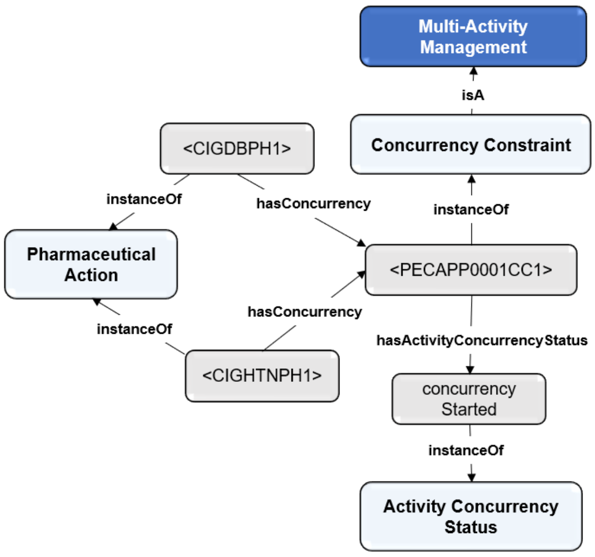


Figure 3.12: Example of concurrency constraint of MAN ontology group

For example, <PECAPP0001CC1> represents the instance of the ConcurrencyConstraint added to the personal care plan to manage concurrency relations between pharmaceutical actions of DB guideline <CIGDBPH1> and HTN guideline <CIGHTNPH1> where the *hasConcurrency* object property links these actions (e.g., initiating Metformin for DB and ACEi for HTN). Under <PECAPP0001CC1>, *concurrencyStartTime* (e.g., "20190612100000") and *concurrencyEndTime* can be recorded as well as activity concurrency status such as concurrencyStarted.

Time-based Synchronisation of Multiple Clinical Activities

The TimeBasedSynchronisationConstraint defines the required expressions to synchronise multiple guideline activities that can have different ordering relations and/or processing times. A time-based synchronisation constraint can be defined with the following axioms:

$$\begin{array}{lcll}
\text{TimeBasedSynchronisation} & \sqsubseteq & \text{MultiActivityManagement} & \sqcap \\
\text{Constraint} & \leq 1 & \text{hasTemporalDistance.TemporalDistance} & \sqcap \\
 & =1 & \text{hasActivitySynchronisationStatus.} & \\
 & & \text{ActivitySynchronisationStatus \{synchronisation-} & \\
 & & \text{Discarded, synchronisationPointReached,} & \\
 & & \text{toBeSynchronised\}} & \\
 & =1 & \text{withinTemporalLimit.boolean} & \sqcap \\
 & =1 & \text{hasTemporalLimit.TemporalLimit} & \sqcap \\
 & =1 & \text{syncLastUpdate.dateTime} & \sqcap
\end{array}$$

This class can be used to represent activities that wait other activities in a given time period for merging or follow their own paths. Temporal distances can be defined to represent clinical activities that have (exact or approximate) time to reach the synchronisation care point where the *hasTemporalDistance* object property links maximum one instance of this class to with the instances of the TemporalDistance.

Temporal limits can be defined to set a deadline for a clinical activity to synchronise where the *hasTemporalLimit* object property links the instances of this class with exactly one instance of the TemporalLimit. If temporal limits are violated, then synchronisation of activities will be discarded. This specification can be made through the *withinTemporalLimit* data property where its realisation can be stated with the following data values "true" or "false". Accordingly, activity synchronisation status needs to be updated (e.g., synchronisationDiscarded) where the *hasActivitySynchronisationStatus* object property links the instances of the TimeBasedSynchronisationConstraint with exactly one instance of the ActivitySynchronisationStatus for this purpose. The *syncLastUpdate* data property represents the last time of recording the temporal information regarding the synchronisation of the selected activity.

To illustrate this constraint (see, Figure 3.13), the following example is presented. Let <PECAPP0001TBS1> "Synchronisation of CHF in diuretic prescription" and <PECAPP0001TBS2> "Synchronisation of HTN in diuretic prescription" are the instances of the TimeBasedSynchronisationConstraint, respectively. The CHF

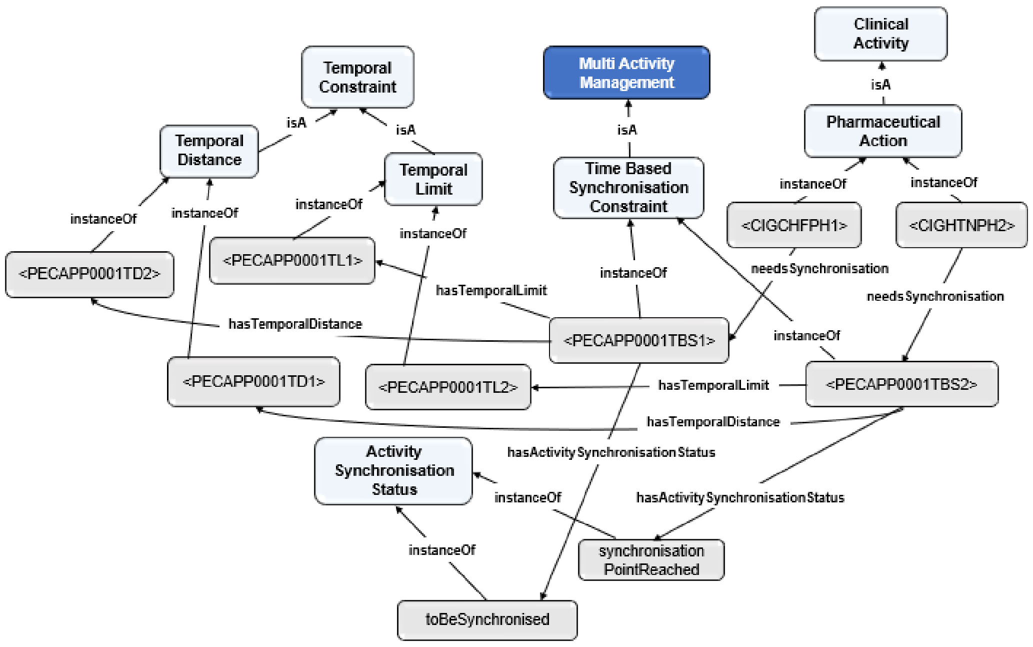


Figure 3.13: Example of time-based synchronisation constraint of MAN ontology group

pharmaceutical action <CIGCHFPH1> "Initiate diuretic" and the HTN pharmaceutical action <CIGHTNPH2> "Add diuretic" are linked with the *needsSynchronisation* object property to these instances to perform synchronisation on a time point.

Under the time-based synchronisation constraint, activity synchronisation status of each activity is recorded, which has link with the *hasActivitySynchronisationStatus* object property to the ActivitySynchronisationStatus to assign one instance (e.g., toBeSynchronised, synchronisationDiscarded or synchronisationCompleted) from this class based on the status of synchronisation. Once activities are synchronised on a time point, then they can be merged in their subsequent actions or follow their own care paths. To do so, synchronisation status of these activities must be synchronisationPointReached. In the given example, the activity synchronisation status of <CIGCHFPH1> is toBeSynchronised which represents synchronisation in progress, and the synchronisation status of <CIGHTNPH2> is synchronisationPointReached which represents this activity ready for merging when the <CIGCHFPH1> reaches its time point. Thus, more than two activities can be synchronised and merged with the defined semantics.

Synchronisation of activities is subject to temporal limits (i.e. maximum waiting time such as 3 hours to initiate the subsequent activity) of each activity. <PECAPP0001TL1> and <PECAPP0001TL2> represent the instances of the TemporalLimit where temporal limits of each activity are defined. Each activity has also associated temporal distances which represent remaining time to reach a time point. Thus, <PECAPP0001TD1> and <PECAPP0001TD2> are the instances of the TemporalDistance where temporal distances of each activity are defined.

Time-based Optimisation of Clinical Activities and/or Their Care Elements

While maintaining patient-safety, it is imperative to supply efficient care and use health resources accordingly. This is called "care optimisation", in this context. Thus, the TimeBasedOptimisationConstraint can be used to manage identical or similar clinical activities and their related care elements (e.g., lab test result), offered by different guidelines, to avoid unnecessary care repetitions. These can be retrieved using clinical IDs, time information, associated life cycle statuses and type of clinical activity. Time-based optimisation constraint can be defined with the following axioms:

$$\begin{array}{lcll}
\text{TimeBasedOptimisationConstraint} & \sqsubseteq & \text{MultiActivityManagement} & \sqcap \\
 & \geq 1 & (((\text{reuseResult.Result}) & \sqcap \\
 & \leq 1 & (\text{reuseResultOf.ClinicalActivity})) & \sqcup \\
 & \leq 1 & ((\text{reuseCareElement.CareElement}) & \sqcap \\
 & \leq 1 & (\text{reuseCareElementOf.ClinicalActivity})) & \sqcap \\
 & \leq 1 & \text{canBeReusedFor.TemporalWindow} & \sqcap \\
 & =1 & \text{needMoreAction.boolean} & \sqcap \\
 & \leq 1 & \text{withinTemporalWindow.boolean} & \sqcap
\end{array}$$

The TimeBasedOptimisationConstraint is designed to eliminate care duplications and related unwanted outcomes. For instance, to avoid care repetitions, results of clinical activities can be reused, if they are performed in a reasonable time window (e.g., valid time period of a test for reuse, 1 week) [50]. Thus, the *reuseCareElement* object property links the instances of the TimeBasedOptimisationConstraint with maximum one instance of the CareElement, in order to define the care element which can be reused. Similarly, the *reuseCareElementOf* object property links the instances of the TimeBasedOptimisationConstraint with maximum one instance of the ClinicalActivity to define the clinical activity whose care element is reused. If more action is needed, then the *needMoreAction* data property must be defined with "true" or "false" data values based on the need of more action. The *withinTemporalWindow* data property can be used to define whether or not reusing of a care element within the given acceptable time window (TemporalWindow) whose data values are "true" or "false". This class aims to resolve internal interactions (e.g., repetitions because of the same action) [187]. When the reuse period is completed due to the reused activity ending for treating a disorder, new actions related with the clinical activity such as duration, periodicity and frequency, can be added. This is achieved with the *needMoreAction* data property with its value set as "true".

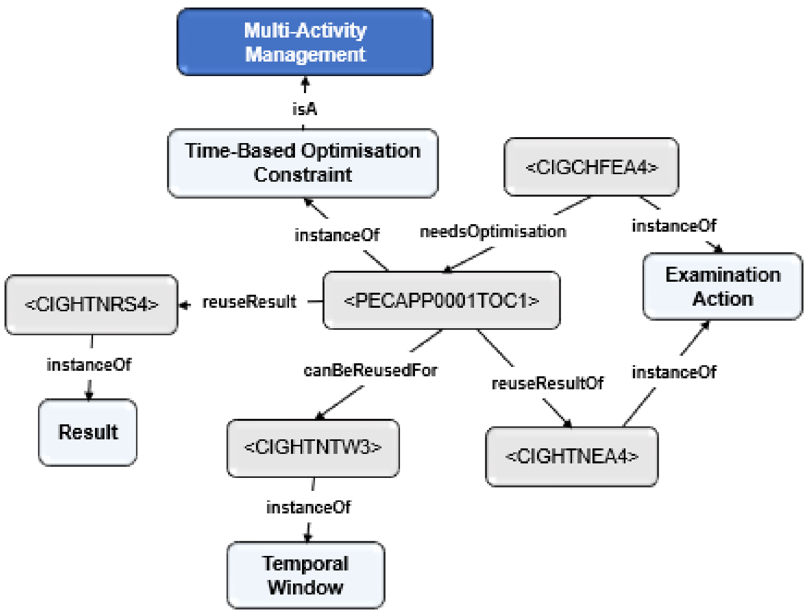


Figure 3.14: Example of time-based optimisation constraint of MAN ontology group

To illustrate this constraint (e.g., <PECAPP0001TOC1>) (see Figure 3.14, lets consider the examination action of a CHF guideline <CIGCHFEA4> "Blood test request" which was also recommended before by the HTN guideline <CIGHT-NEA4>. Because the lab result of blood test (<CIGHTNRS4>) is still within its acceptable time window (<CIGHTNTW3>) of reuse, then this result can be reused by <CIGCHFEA4>. Reuse is not only performed on lab results but also on medications.

Modification of Clinical Activities

The concurrent implementation of multiple CPGs, along with patient data, may lead to conflicts in care. These can be between drug-drug, drug-disease, or drug-patient (e.g., allergic condition) which are crucial to be avoided in order to maintain patient safety. To handle such conflicts, the ModificationConstraint is introduced to modify (i.e. replace or update) clinical activities and their associated care elements such as replacing a pharmaceutical recommendation that may reduce the efficacy of the other medication in use. A modification constraint can be defined with the following axioms:

$$\begin{array}{lll l}
\text{ModificationConstraint} & \sqsubseteq & \text{MultiActivityManagement} & \sqcap \\
 & \leq 1 & \text{descriptionOfModification.string} & \sqcap \\
 & \leq 1 & ((\text{hasCareElementModification.CareElement}) & \sqcup \\
 & & (\text{hasCareElementReplacementWith.CareElement})) & \\
 & =1 & \text{modificationTime.dateTime} & \sqcap
\end{array}$$

With this class, the definitions of care element or clinical activity modifications which represent the reasons why the modification is performed can be defined. The *descriptionOfModification* data property is presented for this purpose.

Care elements such as drug recommendations, examinations or diet recommendations can be replaced or modified. Thus, the *hasCareElementReplacementWith* object property is introduced to link the instances of the ModificationConstraint with maximum one instance of the CareElement to represent the new care element which will be used instead of the existing one. Likewise, the *hasCareElementModification* object property is introduced to link the instances of the ModificationConstraint with the maximum one instance of the CareElement to update (e.g., drug dose level) the modification needed care element. Lastly, modification time must be stated through using the modificationTime data property.

To illustrate this constraint (see Figure 3.15), lets consider the pharmaceutical action of a CKD guideline <CIGCKDPH3> "Initiate Warfarin" which has a conflicting recommendation with a medication recommended by the DB guideline. For this reason, this needs to be replaced with its safe alternative. Thus, pharmaceutical action <CIGCKDPH3> is linked with the object property *needsModification* with the instance <PECAPP0002MC1> of ModificationConstraint for performing a medication replacement. <PECAPP0002PCE2> is the instance of the pharmaceutical care element which represents information of the new medication (e.g., Rivaroxaban) defined for the <CIGCKDPH3>.

In Section 5.3.5, further information on the above-mentioned examples and their implementations in a personal care plan are presented.

3.5.5 Care Workflow Patterns

This section aims to show how MuCRL ontology can support workflow control patterns proposed in the existing works [9, 30, 57].

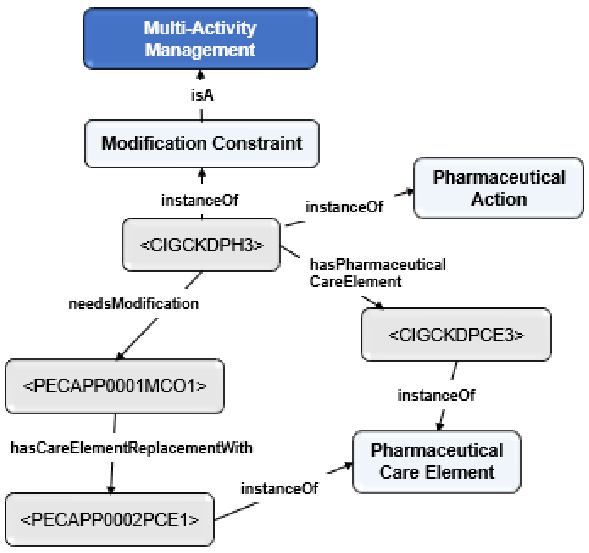


Figure 3.15: Example of modification constraint of MAN ontology group

Sequencing

Like many TNM-based formalisms (see Section 2.4.1), guideline activities need to be hierarchically ordered and the type of connections between them should be well defined. The execution of a guideline starts by triggering the first clinical activity that may have a set of sub activities, and then continues with the subsequent clinical activities.

In Figure 3.16, the illustration of an activity sequencing is shown. The transitions between clinical activities are performed through the instances of the TransitionAssigned.

<CIGDBPH1> and <CIGDBEA1> represent the instances of pharmaceutical action (PharmaceuticalAction) and examination action (ExaminationAction) of a diabetes guideline, respectively. <CIGDBTA1> represents the instance of the TransitionAssigned which creates connection between clinical activities. The *hasTransitionLabel* object property links this instance with an instance of the TransitionLabel. In this example, transition label is FollowingActivity. This means when pharmaceutical action is completed, then examination action can be started if there is no limiting condition.

The source activity of <CIGDBTA3> is <CIGDBPH1> and its target activity is <CIGDBEA1>. <CIGDBTA3> is the outgoing transition of <CIGDBPH1> and the incoming transition of the <CIGDBEA1>. Thus, pharmaceutical action follows examination action when it is completed if a limiting condition does not

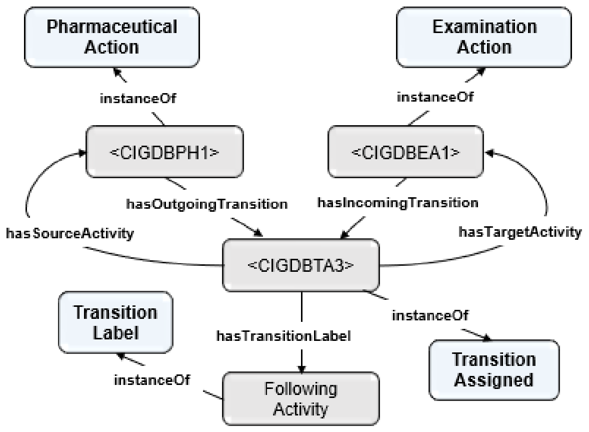


Figure 3.16: Example of an activity sequencing

exist.

Parallel Routing (Splitting)

Parallel routing represents the divergence of a clinical activity into multiple concurrently implemented activities. The *hasTransitionLabel* object property links the instances of the TransitionAssigned with the Split instance of the TransitionLabel.

Synchronisation

Synchronisation of multiple activities into one activity can be possible when all the sub-activities are completed. Otherwise, they need to wait the completion of their sub-activities or perform an exclusive wait where the completion of a sub-activity, which will end all the sub-activities in the exclusive wait group. In this thesis, the TimeBasedSynchronisationConstraint is introduced to support the synchronisation of multiple activities. The *needsSynchronisation* object property links the instances of clinical activities (ClinicalActivity) with the instances of the TimeBasedSynchronisation to handle the synchronisation, see Section 3.5.4. For instance, if the synchronisation of the selected activity is not within the predefined temporal limit (TemporalLimit) in reaching the synchronisation care point, the synchronisation of this activity will be discarded and the value of the *withinTemporalLimit* data property must be set as "false".

Conditional Routing

Conditional routing (also called multi-choice) pattern represents activation of an activity that associated with multiple care options. One or more of these options can be selected based on the satisfaction of given conditions related with them.

This pattern is like a parallel split pattern, but it does not force activation of all activities (e.g., n out of m activities can be activated). The *hasAssignedCondition* object property links the instances of clinical activities with instances of the TransitionConditionAssigned to define the condition details. *hasConditionalOption* is a transition label and instance of the TransitionLabel. This instance is used to represent outgoing transitions of a decision activity (Decision) that has minimum two conditional options. Based on the satisfaction of required conditions, care options can be selected, and therefore, care process can be continued. For example, <CIGDBDC5> "ACE inhibitors tolerated?" decision activity of a diabetes guideline can have two conditional options. If the result of a decision is "no", then continue with the pharmaceutical action <CIGDBPH3> "Initiate an angiotensin II-receptor antagonist for the ACE inhibitor". Otherwise, follow the pharmaceutical activity <CIGDBPH4> "Start insulin based treatment", see Figure 3.17. Conditional options do not necessarily have binary options. There is no restriction on the conditional option representations.

The instances of the TransitionConditionAssigned represent the required condition sets and their related condition satisfaction statuses. Restriction operation (e.g., min, all, any) denotes the restriction value of the number of required condition groups to be satisfied (e.g., all or minimum 2 conditions). These groups can have different condition combinations (e.g., and, or, xor) but share the same condition type (e.g., activatingCondition, precondition) when they defined under the same instance. If conditions are satisfied, then the subsequent activity(s) can be activated (i.e. If-then-else). This is represented with the *isConditionMet* data property whose value is "true". <CIGDBTCA6> and <CIGDBTCA7> represent the instances of the TransitionConditionAssigned. This is associated with the instances <CIGDBPH3> and <CIGDNPH4> of the PharmaceuticalAction. To activate the <CIGDBPH3>, the transition condition <CIGDBTC6> which is in the transition condition list <CIGDBTCL6> must be satisfied. In this example, only one condition (conditionParameter= "tolerated") is defined and its values as "true" or "false" will activate the related pharmaceutical action. The comparison operation (e.g., all, minimum) is defined for each assigned condition instance. However, multiple conditions can also be defined under a single transition condition list.

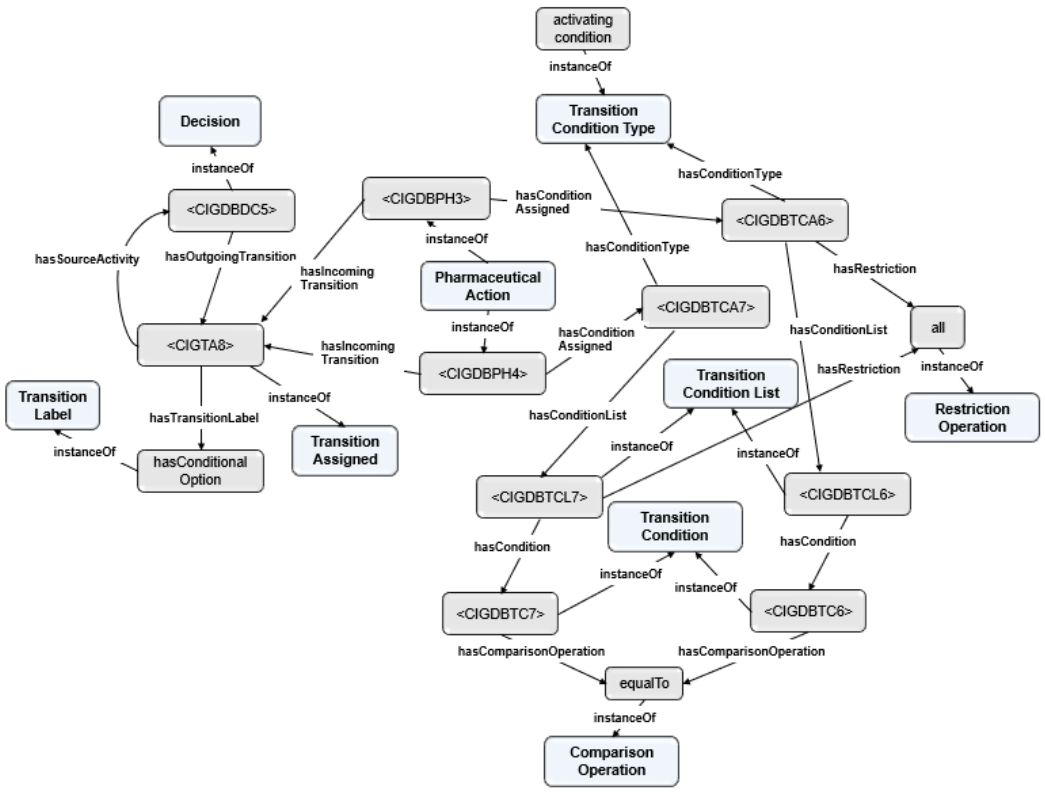


Figure 3.17: Example of conditional routing

Merging

Merging of clinical activities can be categorised as simple merging (see Section 2.4.2) and advanced merging as defined in existing works [9, 30, 57].

- **Simple Merge.** In simple merging, CIG activities are merged into a single activity without forcing the synchronisation of them. The main components of this type of merging are the number of guideline instances (e.g., pharmaceutical action, decision, etc.) which must be defined under one of the sub-classes of the ClinicalActivity and their associated transitions (TransitionAssigned) as illustrated above. The transition label (TransitionLabel) is the isMergedAt. Here, the associated transition conditions (TransitionConditionAssigned) on each activity need to be satisfied. However, these are shown in subsequent steps extensively. Simple merging is supported by existing CIG formalisms but advanced merging patterns are the main limitations found in the existing literature [30, 57].

- **Multi-merge.** This pattern represents merging of concurrently implemented multiple activities into a single activity. In this context, to handle this, activities that are going to be concurrently implemented and have common subsequent clinical activities, are initially discovered using a set of similarity elements such as time period, clinical IDs, type of clinical activities (e.g., pharmaceutical action) and activity execution status (e.g., active, done). If common activities are found, then they can be concurrently executed if they satisfy the required conditions. Handling and recording concurrency relations between clinical activities are crucial to avoid care conflicts and duplications. Hence, concurrency constraints (ConcurrencyConstraint) need to be created to represent concurrency relations between two activities. Once concurrency relations are managed, then these activities can be merged in the subsequent common clinical activity at the end of the care process, see Section 3.5.4 for the associated example.

- **Synchronising Merge.** This pattern involves three subtypes of merging as follows: (i) *structured synchronising merge*, represents merging of multiple activities into a single activity; but all the incoming activities must be activated to merge at the following activity. Activities must wait the completion of the activities to be merged together. The following patterns are like structured synchronising merge, but they have differences as in the (ii) *local synchronising merge* where the decision of synchronisation of the number of activities depends on the local information (e.g., some activities can be cancelled or may exceed their synchronisation time limits, then these activities' synchronisations are defined as discarded. This is a real-time activity implementation information); and in the (iii) *general synchronising merge*, where merging can be realised when the incoming activities have been activated or these activities will be activated in the future when required. A time-based synchronisation approach is adopted to handle synchronisation relations of multiple activities (TimeBasedSynchronisationConstraint). Synchronising merge is achieved by combining the synchronisation and conditional routing patterns.

Iteration

Iteration pattern handles repetitive and cyclic activities until an exit condition (see TransitionConditionAssigned) is triggered. Clinical activities (ClinicalActivity) such as pharmaceutical actions involve periodicity (e.g., every day) and frequency (e.g., 2 times) information (Periodicity). However, the executions of clinical activities are performed in a directed way and do not allow cycles.

Cancellation

Cancellation pattern deals with cancelling activities, cases or multiple instance activities. In the design-time, activities can be removed from the care plan. For instance, activities can be defined as passive (ActivityLifecycleStatus). This represents the selected activity is not available or not appropriate for the execution. In the execution-time, the instance of the ActivityLifeCycleStatus, which is cancelled, represents the cancellation of the clinical activity even if its execution has been started. Cancellation decision can be made in any point of the care activities. Multiple instances of the activities can also be cancelled. For instance, the ActivityConcurrencyStatus represents the concurrency status of the concurrently implemented activities as an enumerated class. Its instance concurrencyDiscarded represents the cancellation of the concurrency relations of the activity with the other concurrently implemented activities due to the cancellation of itself or the other concurrently implemented activity(s). Likewise, the ActivitySynchronisationStatus represents the synchronisation status of the clinical activities as an enumerated class. Its instance synchronisationDiscarded represents the cancellation of the synchronisation of the activity due to the similar reasons of the concurrency relations or the clinical activity is not being able to reach the synchronisation point within a predetermined time period.

Termination

Termination pattern handles conclusions of the activities by categorising them as implicit and explicit terminations. Implicit termination can be realised if no remaining activity exists and all of them are successfully completed. When the last clinical activity is completed (e.g., done), then its subsequent activity can be represented with the instance of the EndOfCare to conclude the care pathway. Each care plan must have one conclusion. However, there can be many conclusions in the care plan that represents the end of a group of care steps. Explicit termination can be realised even if there are any remaining activities in the clinical activity group. This type of termination is represented with endCondition (TransitionConditionType) type involving the associated transition condition (TransitionConditionAssigned) information. The termination condition of a clinical activity can be defined as (*conditionParameter* = "the health status of a patient is optimised", *qualitativeValue* = "true" (outcome condition)) AND (*conditionParameter* = "the hypertension prevention is needed " = "false " (end condition)).

Trigger

A trigger pattern is required to activate an activity based on the inputs received from other activities or sources (e.g., HCPs' inputs). Data query activities can be achieved if and only if knowledge base involves values of the queried data. This can be associated with the completion of an examination activity (ExaminationAction) (e.g., blood glucose measurement) or input from external environment (e.g., patient). For instance, the result of a lab test (e.g., cholesterol level) should be ready for a HCP who will decide the subsequent care activity according to this result (Result). To successfully complete data query, queried data should be ready, which is associated with the satisfaction of pre-condition (TransitionConditionType) (e.g., *parameterName* = "is lab test ready?", *parameterTextValue* = "true") .

Decision activities (Decision) involve a set of conditions which are associated with a set of conditional care options (see Section 3.5.5). These conditions which are namely activating conditions (TransitionConditionType) must be satisfied to initialise the subsequent clinical activity. If a clinical activity is reached to a certain state (e.g., conditionParameter = "patient has a cancer", *qualitativeValue* = "true"), the outcome condition (TransitionConditionType) must be defined. This condition also triggers the activation of the subsequent clinical activity.

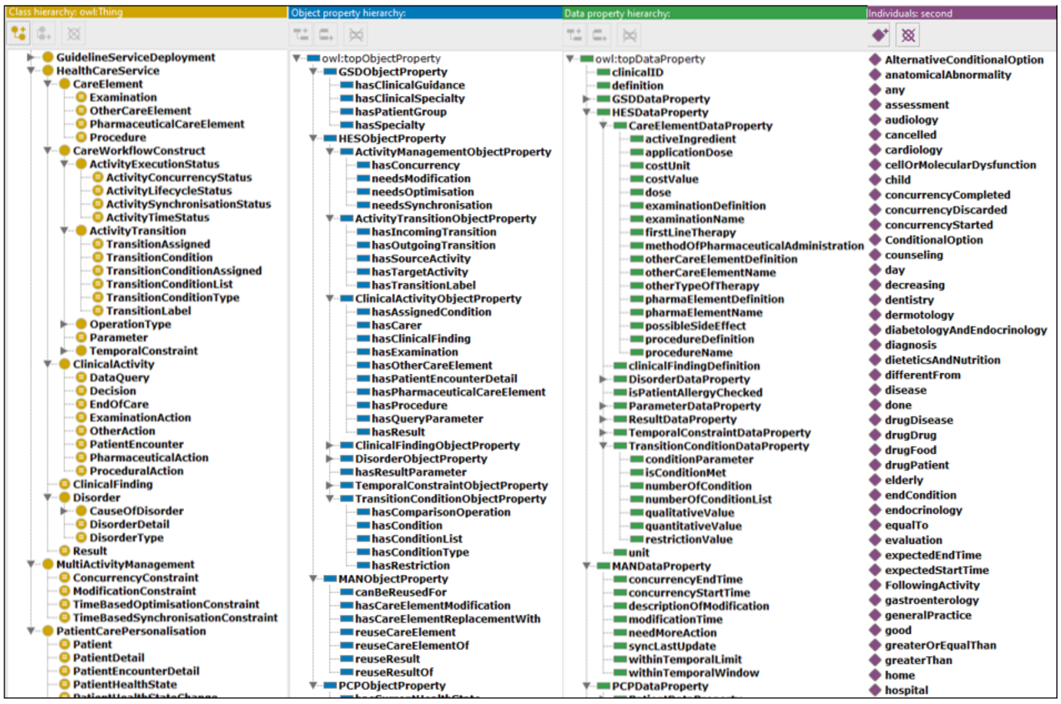
3.6 Design-time CIG Implementation Results

This section presents how the ontology encoding (i.e. developing machine-readable guideline models based on the syntax of a CIG language) is performed, involving technologies such as ontology editors and tools. While design-time tools supporting CIG modelling activities, real-time tools support the dynamic application of them. Here, design-time CIG implementation is presented.

To encode the MuCRL ontology, Protégé 5.2.0 ontology editor [142] that supports OWL2 format [63] is used since this thesis aims to support sharing of ontology and reuse by other users. Protégé is a widely used environment, and easily understandable by users with diverse backgrounds and does not need advance programming skills. Ontologies in Protégé are represented as a hierarchy of classes and each class is represented with a collection of properties (e.g., data and object), instances and their relations.

In Figure 3.18, the portion of concepts, object and data properties and defined instances of MuCRL ontology from the Protégé ontology editor is presented. Figure 3.19 shows the pharmaceutical recommendation implementation from a Type-II diabetes in adults: management guideline [228] in Protégé.

Here, <CIGDBPH1> represents the first instance of the PharmaceuticalAction of a diabetes guideline. The definition of this action is "Initiate Metformin". Its activity lifecycle status is started, therefore, the *actualStartTime* of this activity is recorded. Its *expectedEndTime* (most likely) is also stated for care management. Each clinical activity has minimum one HCP (Carer) (e.g., <Carer0001>). Clinical activities can also be associated with clinical standards for standardisation. In this



```
Class hierarchy: owl:Thing
GuidelineServiceDeployment
HealthCareService
    CareElement
        Examination
        OtherCareElement
        PharmaceuticalCareElement
        Procedure
    CareWorkflowConstruct
        ActivityExecutionStatus
            ActivityConcurrencyStatus
            ActivityLifecycleStatus
            ActivitySynchronisationStatus
            ActivityTimeStatus
        ActivityTransition
            TransitionAssigned
            TransitionCondition
            TransitionConditionAssigned
            TransitionConditionList
            TransitionConditionType
            TransitionLabel
        OperationType
        Parameter
        TemporalConstraint
    ClinicalActivity
        DataQuery
        Decision
        EndOfCare
        ExaminationAction
        OtherAction
        PatientEncounter
        PharmaceuticalAction
        ProceduralAction
    ClinicalFinding
    Disorder
        CauseOfDisorder
        DisorderDetail
        DisorderType
    Result
MultiActivityManagement
    ConcurrencyConstraint
    ModificationConstraint
    TimeBasedOptimisationConstraint
    TimeBasedSynchronisationConstraint
PatientCarePersonalisation
    Patient
    PatientDetail
    PatientEncounterDetail
    PatientHealthState
```

```
Object property hierarchy:
owl:topObjectProperty
    GSDObjectProperty
        hasClinicalGuidance
        hasClinicalSpecialty
        hasPatientGroup
        hasSpecialty
    HESObjectProperty
        ActivityManagementObjectProperty
            hasConcurrency
            needsModification
            needsOptimisation
            needsSynchronisation
        ActivityTransitionObjectProperty
            hasIncomingTransition
            hasOutgoingTransition
            hasSourceActivity
            hasTargetActivity
            hasTransitionLabel
        ClinicalActivityObjectProperty
            hasAssignedCondition
            hasCarer
            hasClinicalFinding
            hasExamination
            hasOtherCareElement
            hasPatientEncounterDetail
            hasPharmaceuticalCareElement
            hasProcedure
            hasQueryParameter
            hasResult
        ClinicalFindingObjectProperty
        DisorderObjectProperty
        hasResultParameter
        TemporalConstraintObjectProperty
        TransitionConditionObjectProperty
            hasComparisonOperation
            hasCondition
            hasConditionList
            hasConditionType
            hasRestriction
    MANObjectProperty
        canBeReusedFor
        hasCareElementModification
        hasCareElementReplacementWith
        reuseCareElement
        reuseCareElementOf
        reuseResult
        reuseResultOf
    PCPObjectProperty
```

```
Data property hierarchy:
owl:topDataProperty
    clinicalID
    definition
    GSDDataProperty
    HESDataProperty
        CareElementDataProperty
            activeIngredient
            applicationDose
            costUnit
            costValue
            dose
            examinationDefinition
            examinationName
            firstLineTherapy
            methodOfPharmaceuticalAdministration
            otherCareElementDefinition
            otherCareElementName
            otherTypeOfTherapy
            pharmaElementDefinition
            pharmaElementName
            possibleSideEffect
            procedureDefinition
            procedureName
        clinicalFindingDefinition
        DisorderDataProperty
        isPatientAllergyChecked
        ParameterDataProperty
        ResultDataProperty
        TemporalConstraintDataProperty
        TransitionConditionDataProperty
            conditionParameter
            isConditionMet
            numberOfCondition
            numberOfConditionList
            qualitativeValue
            quantitativeValue
            restrictionValue
        unit
    MANDataProperty
        concurrencyEndTime
        concurrencyStartTime
        descriptionOfModification
        modificationTime
        needMoreAction
        syncLastUpdate
        withinTemporalLimit
        withinTemporalWindow
    PCPDataProperty
```

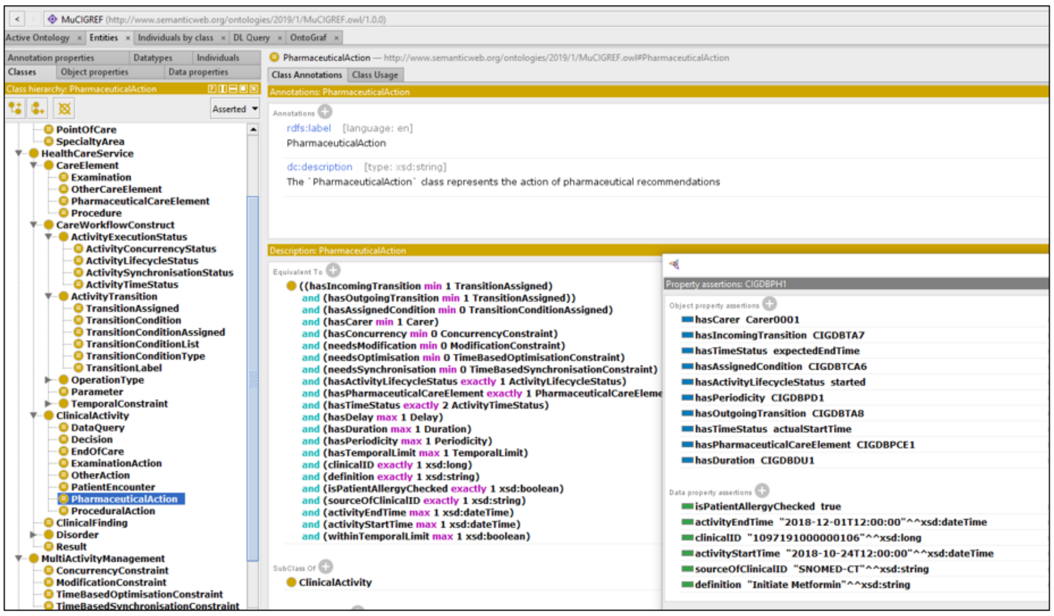
```
Individuals: second
AlternativeConditionalOption
anatomicalAbnormality
any
assessment
audiology
cancelled
cardiology
cellOrMolecularDysfunction
child
concurrencyCompleted
concurrencyDiscarded
concurrencyStarted
ConditionalOption
counseling
day
decreasing
dentistry
dermotology
diabetologyAndEndocrinology
diagnosis
dieteticsAndNutrition
differentFrom
disease
done
drugDisease
drugDrug
drugFood
drugPatient
elderly
endCondition
endocrinology
equalTo
evaluation
expectedEndTime
expectedStartTime
FollowingActivity
gastroenterology
generalPractice
good
greaterOrEqualThan
greaterThan
home
hospital
```

Figure 3.18: A screen shot of MuCRL from the Protégé editor

example, SNOMED-CT is used as the *sourceOfClinicalID* and related *clinicalID* of this action is "1097191000000106". Each pharmaceutical action has one pharmaceutical care element to state medication details (e.g., active ingredient, dose). <CIGDPPHE1> and <CIGDPPCE1> represents the instance of the PharmaceuticalCareElement of this pharmaceutical action. Duration and periodicity information are also represented with the <CIGDBDU1> and <CIGDBPD1> instances, respectively.

Each clinical activity in a guideline is connected through transition definitions. For example, <CIGDBTA7> and <CIGDBTA8> are the instances of the TransitionAssigned and represent incoming (source activity) and outgoing (target activity) transitions, respectively. Clinical activities can have a set of conditions which must be satisfied for their activation. <CIGDBTCA6> is an instance of the TransitionConditionAssigned. The condition(s) associated with this instance is satisfied, therefore, pharmaceutical action can be started.

The following section presents transformation from OWL ontology to EMF Ecore metamodel to support real-time execution of CIGs.


MuCIGREF (http://www.semanticweb.org/ontologies/2019/1/MuCIGREF.owl/1.0.0)
Active Ontology
Entities
Individuals by class
DL Query
OntoGraf
Annotation properties
Datatypes
Individuals
Classes
Object properties
Data properties
Class hierarchy: PharmaceuticalAction
Asserted
PharmaceuticalAction — http://www.semanticweb.org/ontologies/2019/1/MuCIGREF.owl#PharmaceuticalAction
Class Annotations
Class Usage
Annotations: PharmaceuticalAction
Description: PharmaceuticalAction
Property assertions: CIGDBPH1

```
      PointOfCare
      SpecialtyArea
  HealthCareService
    CareElement
      Examination
      OtherCareElement
      PharmaceuticalCareElement
      Procedure
    CareWorkflowConstruct
      ActivityExecutionStatus
        ActivityConcurrencyStatus
        ActivityLifecycleStatus
        ActivitySynchronisationStatus
        ActivityTimeStatus
      ActivityTransition
        TransitionAssigned
        TransitionCondition
        TransitionConditionAssigned
        TransitionConditionList
        TransitionConditionType
        TransitionLabel
      OperationType
      Parameter
      TemporalConstraint
    ClinicalActivity
      DataQuery
      Decision
      EndOfCare
      ExaminationAction
      OtherAction
      PatientEncounter
      PharmaceuticalAction
      ProceduralAction
    ClinicalFinding
    Disorder
    Result
  MultiActivityManagement
    ConcurrencyConstraint
    ModificationConstraint
    TimeBasedOptimisationConstraint
    TimeBasedSynchronisationConstraint
```

Annotations
rdfs:label [language: en]
PharmaceuticalAction

dc:description [type: xsd:string]
The `PharmaceuticalAction` class represents the action of pharmaceutical recommendations

Equivalent To

```
((hasIncomingTransition min 1 TransitionAssigned)
 and (hasOutgoingTransition min 1 TransitionAssigned))
 and (hasAssignedCondition min 0 TransitionConditionAssigned)
 and (hasCarer min 1 Carer)
 and (hasConcurrency min 0 ConcurrencyConstraint)
 and (needsModification min 0 ModificationConstraint)
 and (needsOptimisation min 0 TimeBasedOptimisationConstraint)
 and (needsSynchronisation min 0 TimeBasedSynchronisationConstraint)
 and (hasActivityLifecycleStatus exactly 1 ActivityLifecycleStatus)
 and (hasPharmaceuticalCareElement exactly 1 PharmaceuticalCareEleme
 and (hasTimeStatus exactly 2 ActivityTimeStatus)
 and (hasDelay max 1 Delay)
 and (hasDuration max 1 Duration)
 and (hasPeriodicity max 1 Periodicity)
 and (hasTemporalLimit max 1 TemporalLimit)
 and (clinicalID exactly 1 xsd:long)
 and (definition exactly 1 xsd:string)
 and (isPatientAllergyChecked exactly 1 xsd:boolean)
 and (sourceOfClinicalID exactly 1 xsd:string)
 and (activityEndTime max 1 xsd:dateTime)
 and (activityStartTime max 1 xsd:dateTime)
 and (withinTemporalLimit max 1 xsd:boolean)
```

SubClass Of
ClinicalActivity

Object property assertions

```
hasCarer  Carer0001
hasIncomingTransition  CIGDBTA7
hasTimeStatus  expectedEndTime
hasAssignedCondition  CIGDBTCA6
hasActivityLifecycleStatus  started
hasPeriodicity  CIGDBPD1
hasOutgoingTransition  CIGDBTA8
hasTimeStatus  actualStartTime
hasPharmaceuticalCareElement  CIGDBPCE1
hasDuration  CIGDBDU1
```

Data property assertions

```
isPatientAllergyChecked  true
activityEndTime  "2018-12-01T12:00:00"^^xsd:dateTime
clinicalID  "1097191000000106"^^xsd:long
activityStartTime  "2018-10-24T12:00:00"^^xsd:dateTime
sourceOfClinicalID  "SNOMED-CT"^^xsd:string
definition  "Initiate Metformin"^^xsd:string
```

Figure 3.19: A screen shot of a pharmaceutical action implementation in the Protégé editor

3.7 Towards Dynamic Multiple CIG Execution

Ontologies have been widely used in software development applications [72] such as MobiGuide [136], which was developed as a CDSS to manage patients with chronic health conditions. OWL [63] is one of the widely used ontology language to represent the knowledge constructs of the domain of interest.

Eclipse Modelling Framework (EMF) [64] is a well-known model management platform to generate metamodels and Java code with its metamodel, Ecore and its representation language Emfatic [96]. MuCRL is a CIG language, designed for the representation of knowledge constructs of CPGs and their associations. MuCRL is initially implemented using OWL2 ontology in Protégé and then transformed to an EMF Ecore metamodel in Eclipse. In this section, the need of this transformation and how this can be performed are presented.

3.7.1 Transformation from OWL Ontology to EMF Models

Ontologies support the development of a domain model by supplying concepts, relations and constraints with which can be used for modelling the domain. Thus, the domain model can be created through creating the ontology instances. In the previous section, MuCRL is introduced with a generic ontology-driven guideline representation structure. This implementation is performed using the Protégé environment which supports the OWL2 format.

This section discusses the transformation from OWL ontology to EMF Ecore metamodel (see Section 2.3.3) and, subsequently, to an EMF model. To execute knowledge represented with OWL2, Semantic Web Rule Language (SWRL) [184] which is a rule language for ontology-driven models and has built-in plugin in the Protégé has been widely used in the existing literature (e.g., [180, 181, 182]). However, SWRL has several limitations in practice. For instance, SWRL cannot handle complex rules (which is needed to handle concurrently implemented multiple actions recommended by multiple guidelines), or modify existing knowledge in an ontology (e.g., not able to change an instance value) or generate new instances [229, 230]. For this reason, this transformation is required for the development of the real-time CIG execution engine to define, modify and manage classes in dynamic behaviour [231].

The transformation from ontology to a domain model is a straightforward process. To achieve this, the EMF of Eclipse (i.e. its syntax is Emfatic [96]), is used. This is then transformed into Ecore metamodel and subsequently into an EMF model. EMF is selected because it supplies the framework to use models effectively based on customisable codes, and therefore, supply flexibility to users in model management [81, 231]. EMF has direct communications with Epsilon Object Language (EOL) [101] which is an imperative programming language (see Section 2.3.5). This enables users to generate, query and update EMF models (i.e.instantiations of Ecore metamodel) automatically; and Epsilon Validation Language (EVL) [65] to validate models through customised user messages as well which is a variant of Object Constraint Language (OCL) [66] (see Section 2.3.6).

The methodology of the transformation from OWL ontology to EMF models is as follows. Initially, the consistency and completeness of the OWL ontology are checked by using the ontology reasoners (e.g., HermiT [92] and FaCT++ [93] which are discussed in Section 6.3.1). Afterwards, the OWL ontology entities (e.g., classes, instances, object and data properties) are analysed, and mapped with the elements of EMF Ecore metamodel (based on Emfatic) such as `EClasses`, `EAttributes`, and `EDataTypes`. Subsequently, mappings of object properties to topological relationships to `EReferences` are performed. Enumerated classes in the OWL ontology can be mapped with `EEnumerations` (see Table 3.3). Lastly, EMF models are then created by instantiating the Ecore metamodel. Each class instance is a specific assignment of values to the attributes of $c_i \in \mathcal{C}$. These instances are called as object instances. These object instances can be linked from one to another through links, namely, references (association instances).

The following data types are used in knowledge model. These are represented

Table 3.3: Comparison between OWL ontology and Ecore metamodel elements

OWL ontology	**Ecore metamodel**
Class	`EClass`
Object property	`EReference`
Data property	`EAttribute`
Data types	`EDataType`
Enumerations	`EEnumerations`

as `EDataType` in EMF. These involve Long, Integer, Java.Lang.DateTime (i.e. this is defined in Protégé as dateTime), String, Boolean, and Float. Instead of using the DateTime, the data type Long is used in EMF for the executional issues (e.g., "20181205123000").

After the Ecore metamodel is transformed into TBox (e.g., isA relation) of the OWL ontology, the ABox of the ontology which involves the instances of the Ecore model is created. For example, TBox may involve the Patient class and their associated properties. The ABox may involve the instances and properties of this class (e.g., P0001 represents the instance of the *patientID* data property).

Inheritance (extends) can exist between classes, representing that the inherited class possesses all the attributes of its parent class, and its instances are instances of the predecessor class as well. After the transformation (i.e. performing direct mappings) from the ontology to its Emfatic version, the Ecore metamodel is created. The Ecore model can also be serialised in XMI which is a standard of the Object Management Group to exchange metadata information through XML [85].

Following section presents the definitions and development of CIG models using the EMF Ecore metamodel. These are then used in the generation of personal care plans for multimorbid patients.

3.7.2 CIG Models to Generate Personal Care Plans

Instantiations of the EMF Ecore metamodel are defined as EMF models. Thus, CIG models and Personal Care Plans (PECAP) are EMF models.

CIG model (instance model) is a disease-specific (e.g., diabetes, obesity, atrial fibrillation, etc.) model, yet each of them uses the same vocabulary. In other words, the differences between CIG models are the data used in their instantiation phase. CIG model can be defined as:

Definition 3.7.1. (CIG Model) A CIG model is an instantiation of the metamodel with guideline-specific information and its interrelations required to establish a care pathway of a patient.

The personal care plan can be defined as:

Definition 3.7.2. (Personal Care Plan) Personal care plan (PECAP) is a unified model which is an instantiation of the metamodel, involves a set of individual CIG models, where each of them represents a specific disease; multi-activity management constructs to handle parallel guidelines; patient information and HCP details who performs the care.

The terms CIGs and CIG models are used interchangeably in the following sections of the thesis. The use of CIG models and personal care plans are illustrated extensively in Section 5.3.

3.8 Summary

This chapter initially introduces the design, development and design-time implementation stages of MuCIGREF's multiple CIG Representation Language, namely, MuCRL. MuCRL is designed to support; (i) knowledge representation encapsulated in CPGs as hierarchical skeletal plans; (ii) creation of mappings between individual CIGs with patient-specific information, health resource information (e.g., drug information) and HCP information (e.g., carer ID, contact details and care activity); (iii) creation of mappings between CIGs to establish multimorbidity related associations between them towards developing a unified personal care plan; and (iv) managing concurrently implemented CIG actions (e.g., merging, modification, optimisation, time-based synchronisation) to supply consolidated care recommendations. The evaluation results and contributions of MuCRL are discussed extensively in Section 6.3 and Section 7.1, respectively.

Afterwards, the transformation process of MuCRL's knowledge representation, which is in OWL ontology, to EMF models for real-time CIG executions is presented. The objective of the transformation is to support real-time knowledge execution where EMF is used instead of the Protégé editor in the following sections. Mapping process between these two systems are manually performed. Yet, there are several frameworks, perform these mappings and support transformations between systems automatically [232]. However, this is not the scope of this thesis. As a result, ontology transformation into the EMF model is needed for the real-time CIG execution engine where execution is performed based on a set of algorithms to combine multiple CIG models while handling their complexities and make automatic updates over the personal care plan of a patient.

Chapter 4

Real-Time Multimorbidity Care Management

4.1 Introduction

This chapter introduces MuCIGREF's Real-time Multiple CIG Execution Engine, MuCEE, which is a novel software application built for combining multiple concurrently implemented CIGs (models) to generate personal care plans for a multimorbid patient. MuCEE's execution process of each guideline adheres to the semantics of MuCIGREF's multiple CIG representation language (MuCRL). MuCEE has three modules, which are specialised on (i) multiple CIG acquisition; (ii) parallel CIG execution, and (iii) CIG verification.

Initially, MuCEE's CIG model acquisition module is introduced to acquire multiple CIG models (i.e. instantiation of formalised CPGs related with patient health disorders) at any time point of care from the knowledge base. These are then unified under a single model, called, Personal Care Plan (PECAP) where updates on clinical activities are performed to supply consistent and personalised care recommendations for a multimorbid patient. However, when multiple guidelines are concurrently applied for the patient, this becomes a challenge due to the need of managing a set of constraints relating with; for instance, arranging concurrency and synchronisation relations between clinical activities, recommended by the same or different guidelines, in order to avoid care conflicts (e.g., adverse drug interactions), or the need of multi-merging [9, 30] of clinical activities to eliminate care duplications (e.g., inefficient use of resources). MuCEE's parallel CIG execution module is designed to handle these complexities.

Lastly, MuCEE's CIG verification module is introduced to perform automated verification analysis to detect any missing information, inconsistencies and errors which may lead to inappropriate care flow. Moreover, verification analysis is supported with customised user messages explaining how to fix these anomalies. The major capabilities of MuCEE can be summarised as follows:

- Acquiring multiple CIGs based on the patient health conditions, ensuring that CIG elements are up to date;
- Combining and managing multiple CIGs on a generated personal care plan with the aim of treatment personalisation;
- Performing verification to discover inconsistencies, errors or missing values in CIGs and in personal care plans where multiple CIGs are combined, and support users to handle such issues.

In the following section, the CIG execution requirements are summarised. Afterwards, MuCEE's building method and then its algorithm specification are presented, respectively. MuCEE implementations on multimorbidity exemplar case studies and their verification and validation analysis are addressed in Sections 5.3 and 6.4, respectively.

4.2 CIG Execution Requirements

In this section, a synthesis of existing works [15, 19, 113, 118, 127, 130, 133, 134, 149, 181, 235, 236] which address CIG execution engines (see Section 2.4.1), is performed. Accordingly, the major requirements of CIG execution systems are identified as follows:

1. The system must have required functionalities for data pooling, loading and storing the guidelines in a knowledge base (repository);
2. The system must supply a formal language for CPG encoding;
3. The system should retrieve and handle several CIGs relating to the patient health conditions from a knowledge base;
4. The system can execute the workflow control patterns (see Section 2.4.2) involving the patient data;
5. The system must involve execution states (e.g., started, cancelled, done, etc.) associated with the progress of care;

6. The system should handle different types of conditions such as preconditions, post conditions and trigger conditions (see [237]) to manage the patient care;
7. The user can infer information (e.g., following clinical activities, patient medication use, lab results, etc.) through querying from a knowledge base;
8. Each implementations of CIG activities (e.g., HCP decisions, prescriptions and other clinical actions) must be stored at the knowledge base, in order to record the care steps;

Further to the above, the following execution requirements to manage multimorbidity care are identified, and added to the aforementioned list:

9. The system should discover any mapping relations between multiple CIGs, in order to establish required multimorbidity associations;
10. The system should combine concurrently implemented multiple CIGs and their associated actions, due to the increasing number of patients' multimorbid conditions; and supply interactive support for users;

Based on these execution requirements, MuCEE was designed. The following section presents its execution methodology.

4.3 MuCIGREF's Real-time Multiple CIG Execution Engine (MuCEE) Building Method

This section discusses the development method of the MuCEE. Initially, an extensive literature review is performed, where existing CIG execution engines, and their scopes are analysed. Then, CIG execution requirements and gaps of the existing literature in handling multimorbidity care are discovered (see Section 2.5). Accordingly, the MuCEE's architecture is shaped.

To dynamically execute CIGs, MuCRL is implemented using EMF in Eclipse (see Section 3.7.1). Afterwards, CIG models are created using instances of guidelines and their interrelations. Lastly, the personal care plan, where multiple CIG models are unified, is created. To unify these models under this plan, multiple CIG model acquisition module, is introduced to transfer guideline activities to personal care plan (model) based on patient health condition. Then, care steps are step by step executed over this model. To achieve this, the parallel CIG execution module is introduced. There CIG models for multimorbidity scenarios are concurrently executed. Lastly, the CIG verification module is introduced to check implementation inconsistencies, missing information and errors. Algorithms of the related modules are presented in Section 4.4. The implementation of the algorithms is presented in Section 5.3, while verification and validation results are presented in Section 6.4.3 as part of the evaluation process.

To handle multimorbidity care, mappings between multiple CIGs must be created. Thus, knowledge mapping is performed between guideline instances and their associations using their associated classes, their instance labels (names), and clinical IDs. Finding common care elements (e.g., lab tests, drugs, etc.) between multiple CIGs are needed to eliminate care duplications; and to unify their actions when needed. Peleg et al. [138] presented a mapping approach to map CIGs to EHRs based on query writing, and comparing a set of mapping approaches, based on their mapping capabilities. Similarly, in this thesis, query writing is adopted for creating mapping relations. To illustrate, a clinical ID of a clinical activity (e.g., pharmaceutical action) can be automatically queried and other clinical activities that have same clinical IDs can be inferred (see Section 5.3.4). Accordingly, common (i.e. identical/similar) activities can be detected, and their associated information can be obtained which can be used for activity merging. This mapping discovery is embedded into the execution mechanism. Following sections present the MuCEE's execution approach in detail.

4.3.1 Multiple CIG Model Acquisition Module

Patients can have many health conditions with different combinations. Thus, CIGs can be executed for the same patient and different patients as well. For each patient, MuCEE, firstly, acquires all the CIG models (i.e. the instantiations of Ecore metamodel) related with patient health conditions from the knowledge base. The Ecore metamodel is used to build CIG models, namely, CIGs.

In this thesis, the problem of combined care plan generation from multiple concurrently implemented multiple CIGs is considered. The main components of this problem are a set of variables and conditions (requirements, constraints or restrictions) between them. Solving this problem can be possible if designated restrictions of each variables are satisfied. The categorisation of these conditions is discussed in the following section in detail. Conditions must be satisfied to activate a care step of a guideline. Thus, condition satisfaction over CIG models is performed to acquire guideline activities to the personal care plan of the patient. Only up to date (i.e. recommendations currently used in practice) activities whose activity lifecycle status are active (i.e. ready for execution), can be transferred to the personal care plan. For this reason, the given execution restriction which is activity lifecycle status must be satisfied. This also supports version control of the CIG models. The algorithm is presented in Section 4.4 where specifications of CIG model acquisitions are discussed.

Epsilon Object Language (EOL) [65] is used for the algorithm implementation. The following section further presents the parallel CIG execution module.

4.3.2 Parallel CIG Execution Module

As defined in the previous section, this execution module focuses on the problem of combination of concurrently implemented multiple CIGs to generate a unified care plan. To handle this problem, rule-based execution approach is used which has IF ... THEN structure. While IF represents conditions (constraints and restrictions), THEN represents actions or conclusions. When the condition item of a rule is satisfied, the rule is triggered and the action item is executed to find a solution (e.g., care recommendation).

Parallel CIG execution module of MuCEE uses the personal care plan which involves individual CIG models (i.e. acquired using CIG acquisition module), to perform dynamic execution of knowledge over this model. Execution is performed by developing a specialised execution algorithm which searches the potential solutions (recommendations) in a set of variables under a set of conditions (i.e. constraints, restrictions). To discover these execution restrictions which need to be satisfied for finding valid solutions (i.e. consistent recommendations) in CIG execution phases, existing literature [50, 120, 156, 159, 238, 239] and a set of existing guidelines (e.g., *www.nice.co.uk*) are reviewed.

The main focus of this thesis is to solve the problem of combining multiple concurrently implemented CIGs under a personal care plan of a patient which can be formally defined as follows:

Definition 4.3.1. (Multiple CIG Combination Problem (MCCP)) Let $\mathcal{P}$ be the personal care plan (PECAP) of a patient which involves a set of CIG models (domain), Π, where $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$. Each $\Lambda = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ be the finite set of variables (i.e. guideline activities) with an associated set of CIG models. $\Pi = \{\Pi(\alpha_1), \ldots, \Pi(\alpha_n)\}$, where $\Pi(\alpha_i)$ represents each variable α_i that ranges over the domain, and a set of conditions $\Omega = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ over the variables. A condition $\varphi_i (\in \Omega)$ on the variables $\alpha_1, \alpha_2, ..., \alpha_n$ is a subset of $\Pi(\alpha_1) \times \cdots \times \Pi(\alpha_n)$, which represents the set of possible values of variables over the domain.

The model-level state conditions which can be defined as constraints, need to be satisfied by the execution system at real-time. These are associated with each CIG model variables as follows:

Activity Transition Constraints. $\mathcal{TC}$ is the set of transition conditions (TransitionCondition) where tc_{ijk} represents transition condition i and its condition type j of activity α_k ($\in \Lambda$), $i = 1, \ldots, n$, and $j = \{$activatingCondition, preCondition, outcomeCondition, endCondition$\}$. The assigned transition conditions (TransitionCondition Assigned) for an α_k must be satisfied to follow the next clinical activity; and (ii) $\mathcal{TA}$ is the set of assigned transitions of guideline activities.

Activity State Constraints. $\mathcal{AE} = \{\mathcal{ALS}, \mathcal{ACS}, \mathcal{ANS}, \mathcal{ATS}\}$ is the set of activity execution status. These involve:

– $\mathcal{ALS}$ is the set of activities' lifecycle statuses (ActivityLifecycleStatus) where ls_{ik} represents activity lifecycle status i of activity α_k ($\in \Lambda$), $i = \{$active, cancelled, done, passive, pending, started$\}$. Each α_k must have one activity lifecycle status at a time, and α_k must satisfy the execution condition to perform activity state transitions;

– $\mathcal{ACS}$ is the set of activities' concurrency statuses (ActivityConcurrencyStatus) where lc_{ik} represents activity concurrency status i of activity α_k ($\in \Lambda$), $i = \{$concurrencyCompleted, concurrencyDiscarded, concurrencyStarted$\}$. When the concurrency relation is defined, then its associated status must be satisfied to perform concurrency status transitions. This is required for multiple activity concurrency management;

– $\mathcal{ANS}$ is the set of activities' synchronisation statuses where ln_{ik} represents activity synchronisation status (ActivitySynchronisationStatus) i of activity α_k ($\in \Lambda$), $i = \{$toBeSynchronised, synchronisationPointReached, synchronisation Discarded$\}$. When the synchronisation relation is defined, then its associated status must be satisfied to perform synchronisation status transitions;

– $\mathcal{ATS}$ is the set of activities' time statuses (ActivityTimeStatus) where lt_{ik} represents activity time status i of activity α_k ($\in \Lambda$), $i = \{$actualStartTime, actualEndTime, expectedStartTime, expectedEndTime$\}$. Activity start and end time are dependent on the values of the defined temporal constraints (e.g., duration, periodicity).

Multi-activity Management Constraints. Multi-activity management (MAN) (see Section 3.5.4) constraints involve (i) concurrency constraint; (ii) time-based optimisation constraint; (iii) time-based synchronisation constraint; and lastly, (iv) modification constraint. MAN constraint set enables users to adapt to changes in the model, and can be used to manage multiple activities recommended by concurrently implemented multiple CIG models under a combined care plan, $\mathcal{P}$. These involve handling of concurrency, and synchronisation relations for multi-activity merging, modification and reusing of care options.

Temporal Constraints. These constraints stand for the temporal constraints between guideline activities. For example, temporal limit of synchronisation of activity α_k ($\in \Lambda$) (e.g., examination action) with activities $\alpha_1, \ldots, \alpha_n$ should not be executed maximum three days before the expected start time of α_{k+1}.

To solve the problem of multiple CIG combination, constraints on variables are explored on a bounded number of possible solutions, and labels are propagated through recorded constraints. Variable *label inference* approaches [233], which can be defined as searching of possible values of the variables that are refined via the propagated constraints, have also been widely used in the literature (e.g., [234]). Label propagation starts with searching a collection of variables in the knowledge base, with labels and relationships between them that corresponds to the labels of the classes in the query and their relationships. For label inferencing, a query (answering) language (e.g., Structured Query Language) needs to be adopted to perform refinement on constraints. Here, this is performed using the EOL [65].

In this thesis, querying (e.g., specific time periods, patient information, lab results), is also integrated into the execution mechanism to reduce (filter) variables for the conditions and, therefore, the amount of potential solutions. This step is performed incrementally. When the conditions are satisfied and solutions (recommendations) are found, then users can discover new solutions from the knowledge base and/or perform queries, given the set of discovered solutions. Queries can be:

- "What are the possible times (time period) at which activity i may exist?"
- "What are the started or completed clinical activities in the personal care plan"?
- "Is the requested lab test result (e.g., HDL cholesterol level) ready?"

In the light of the CIG execution engine requirements, and limitations in execution of concurrently implemented CIGs, MuCEE's parallel CIG execution module is shaped. The major functionalities of parallel CIG execution module as discussed, in further detail, are as follows:

- Identifying clinical activities to be initialised considering all the CIG models related with patient health disorders;
- Dynamically adding, removing, or replacing clinical activities and/or their associated care elements;
- Managing concurrency relations between multiple clinical activities to avoid harmful care advices, caused by recommendation duplications (e.g., drug overdose);
- Managing merging of concurrently implemented multiple clinical activities ($\geq$ 2) through using time-based synchronisation;
- Performing time-based care optimisation to avoid unnecessary resource (e.g., carer time, lab test) utilisation and potential care duplications which may lead to extra cost for a treating medical centre.

The EOL is also used in this execution module. Following section presents the CIG verification module.

4.3.3 CIG Verification Module

In the existing literature (see Section 2.4.3), verification of CIGs have attracted a great attention of the researchers such that there are many different methodologies that have been proposed for this purpose. In this thesis, verification is performed in the entire care process, in order to maintain consistent and error-free care plans.

As part of the verification process, a new CIG verification method which supplies dynamic user support and model update, is proposed. Accordingly, a set of CIG verification constraints is developed, which comply with the requirements of CIG executions in terms of correctness and consistency. Instantiations of CPGs to generate CIG models are also checked to identify whether missing values exist or to maintain syntactic consistency. These constraints are applied using Epsilon Validation Language (EVL) [65], which is a variant of Object Constraint Language (OCL) [66], checks dependencies between the constraints, specify inconsistencies and help users on how to repair them. EVL has its own syntax (see Section 2.3.6), is used for CIG verification rule implementations. EVL's execution mechanism uses a top-down depth-first scheme, please see Kolovos et al. [65, 103] for further information.

The major advantages of using EVL, compared to OCL, are to support generation of user feedback and warnings, inconsistency fixing, evaluating constraints in multiple models (not bind to a single model) at a time and supply greater flexibility in context definition [65]. The CIG verification is discussed in the following section as part of the MuCEE execution steps, and its application on real-life case studies are presented in Section 5.4.

4.4 Algorithm Specifications of MuCEE

This section presents the algorithm specifications of MuCEE, see Table 4.1. After the patient-HCP encounter, initially a (blank) personal care plan (PECAP) is created where all the care steps related with patient's care management displayed. Then, execution follows with applications of several sub-components (sub-algorithms).

The sub-algorithms involve the CIG model acquisition algorithm, which is part of the acquisition module of the MuCEE; multi- activity merging, time-based care optimisation, modification, and concurrency management algorithms which are part of the parallel CIG execution module of the MuCEE. Dynamic CIG verification is applied pre- and real-time CIG execution phases in the entire care process.

The algorithms presented in this section are presented using UML activity diagrams [81], please see Section 2.3.1 for the notation used in the algorithm specifications with their definitions. Following sections discuss MuCEE's algorithm specifications.

Care Initialisation

Step 1. Initially, a personal care plan, $\mathcal{P}$ of a patient is created when a patient-HCP encounter is realised. Dynamic CIG executions, using patient data, are subsequently performed on this plan, after the relevant CIG models, Π, where $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ are acquired (see Section 5.4.2) from the knowledge base. Then, instantiations of MuCRL's ontology groups are created on $\mathcal{P}$. Initially, a PatientCarePersonalisation is created to record patient information. Afterwards, a GuidelineServiceDeployment is created to record HCP information. Following this, a HealthCareService is created to record patient-HCP encounter, where the health condition information such as signs, symptoms and diseases of a patient and care management constructs. Lastly, a MultiActivityManagement is created to manage multiple activities recommended by multiple CIG models when the related CIG models are acquired.

Multiple CIG Acquisition

Step 2. This phase begins with the retrieval of CIG models, relating to patient

Table 4.1: The MuCEE's execution algorithm

```
1  Initialisation: Create a personal care plan $\mathcal{P}$, PECAP, for a patient which is updated
   in each execution step;
2  Begin acquiring instantiated CIG models $\pi_1, \pi_2, \ldots \pi_n$ of MuCRL which involve
   a set of activities $\alpha_1, \alpha_2, \ldots, \alpha_n$ from the repository, related with patient health
   conditions based on the "CIG acquisition algorithm";
   – If each activity $\alpha_k$ ($\in \Lambda$) is up to date (i.e. $ls_{ik}$ ($\in \mathcal{ALS}$), $i$={active}), then
   transfer $\alpha_k$ of $\pi_i$ ($\in \Pi$) to plan $\mathcal{P}$;
   – Otherwise, quit.
3  Select $\alpha_k$ ($\in \Lambda$) (whose $ls_{ik}$ ($\in \mathcal{ALS}$), $i$={active}) from $\pi_i$ ($\in \Pi$) in $\mathcal{P}$;
4  Check,
   – If the associated conditions (constraints and restrictions) $\varphi_1, \ldots, \varphi_n$ on $\alpha_k$ ($\in \Lambda$)
   are satisfied, then follow the next step;
   – Otherwise, wait for the satisfaction of conditions over $\alpha_k$ ($\in \Lambda$);
5  Check,
   – If concurrencies exist between $\alpha_k$ ($\in \Lambda$) and activities $\alpha_1, \alpha_2, \ldots, \alpha_n$ that exist
   in $\mathcal{P}$, then follow the "Concurrency management algorithm";
   – Otherwise, follow the next step.
6  Check,
   – If $\alpha_k$ ($\in \Lambda$) and $\alpha_1, \alpha_2, \ldots, \alpha_n$ have commonalities in their subsequent activities,
   then follow the "Multi-activity merging algorithm" for multi-activity merging;
   – Otherwise, follow the next step.
7  Check,
   – If $\alpha_n$ ($\in \Lambda$) is recommended before (whose $ls_{in}$ ($\in \mathcal{ALS}$), $i$={started, completed}),
   then follow the "Time-based care optimisation algorithm" for considering activity
   reusing by $\alpha_k$ ($\in \Lambda$);
   – Otherwise, follow the next step;
8  Check,
   – If $\alpha_k$ ($\in \Lambda$) needs modification, then follow the "Modification algorithm";
   – Otherwise, follow the next step.
9  Update the $ls_{ik}$ ($\in \mathcal{ALS}$) of $\alpha_k$ ($\in \Lambda$) upon its completion, and query its target activities
   and associated conditions $\varphi_1, \varphi_2, \ldots, \varphi_n$ on $\alpha_k$ to follow the next care step;
10 Check,
   – If inconsistencies and/or missing information exist in $\mathcal{P}$, then generate user
   messages regarding how to fix them;
   – Otherwise, wait till next execution and follow the next step;
11 Check,
   – If a new health condition needs to be managed, then Return Step 2,
   – Otherwise, Return Step 3 until there is no remaining activity exists and quit.
```

health conditions from the repository, by the HCP(s) (see Figure 4.1).

Afterwards, CIG models need to be acquired based on the satisfaction of the activity lifecycle requirements, $\mathcal{ALS}$, which is the element of activity execution status, $\mathcal{AE}$, that has been used in existing execution engines (see Section 2.4.1) in a similar way to represent the current status of an activity. In the acquisition phase, if CIG models' activities where $\Lambda = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ represents the activity set, to be

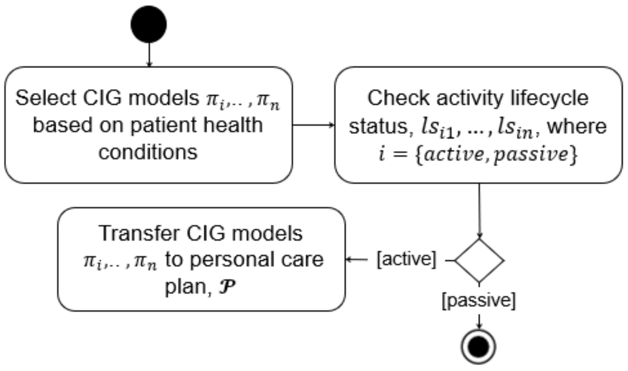

Select CIG models $\pi_i, .., \pi_n$ based on patient health conditions
Check activity lifecycle status, $ls_{i1}, ..., ls_{in}$, where $i = \{active, passive\}$
Transfer CIG models $\pi_i, .., \pi_n$ to personal care plan, $\mathcal{P}$
[active]
[passive]

Figure 4.1: CIG model acquisition algorithm

acquired are up to date, then their activity lifecycle statuses, ls_{in} ($\in \mathcal{ALS}$) where i represents activity lifecycle status of α_n ($\in \Lambda$) should be stated as active. Otherwise, these should be passive which means they are absolute or not able to be implemented and must not take place at the personal care plan.

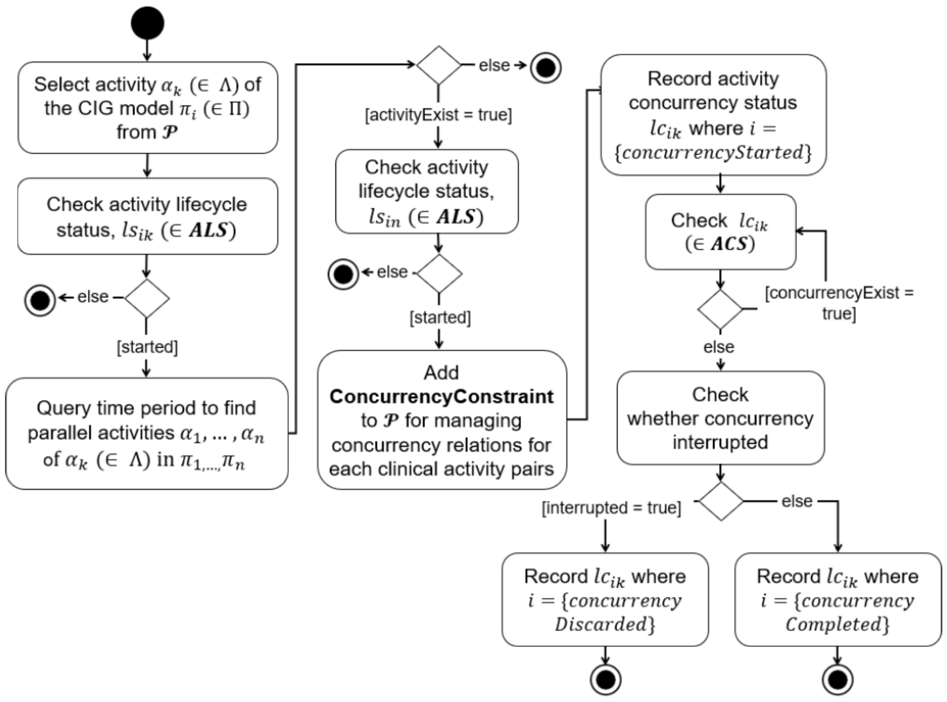
<u>**Parallel CIG Execution**</u>

Step 3. Activity α_k ($\in \Lambda$) (whose ls_{ik} ($\in \mathcal{ALS}$), i={active}) is selected from π_i ($\in \Pi$) in the patient's personal care plan, $\mathcal{P}$, based on the care flow and then follow the next step;

Step 4. To fire the clinical activity α_k ($\in \Lambda$) (e.g., medication recommendation) for a patient, its associated (constraints and restrictions) $\varphi_1, \ldots, \varphi_n$ on α_k ($\in \Lambda$) must be satisfied. For example, patients' BP levels should be below the predefined limits (e.g., 140/80 mmHg) to prescribe an ACE inhibitor. Thus, the the BP level threshold condition needs to be satisfied for this care recommendation action.

Step 5. Concurrency management of multiple activities is required to detect and eliminate care duplications, which can be the sources of adverse interactions (e.g., drug-drug interactions, which cause drug overdose or affects efficacy of each other) (see Section 2.5.2). Figure 4.2 presents the concurrency management algorithm. The algorithm allows the users(s) to manage concurrently implemented multiple activities that can be recommended by multiple different CIGs.

The algorithm begins with the selection of the clinical activity α_k ($\in \Lambda$) from the CIG model π_i ($\in \Pi$) that exists in the personal plan, $\mathcal{P}$. Then, the activity lifecycle status of this activity is checked. If it is started, then it checks its activity time



Select activity α_k ($\in \Lambda$) of the CIG model π_i ($\in \Pi$) from $\mathcal{P}$
Check activity lifecycle status, ls_{ik} ($\in \boldsymbol{ALS}$)
else
[started]
Query time period to find parallel activities $\alpha_1, \ldots, \alpha_n$ of α_k ($\in \Lambda$) in $\pi_{1,\ldots,}\pi_n$
else
[activityExist = true]
Check activity lifecycle status, ls_{in} ($\in \boldsymbol{ALS}$)
else
[started]
Add **ConcurrencyConstraint** to $\mathcal{P}$ for managing concurrency relations for each clinical activity pairs
Record activity concurrency status lc_{ik} where $i = \{concurrencyStarted\}$
Check lc_{ik} ($\in \boldsymbol{ACS}$)
[concurrencyExist = true]
else
Check whether concurrency interrupted
[interrupted = true]
else
Record lc_{ik} where $i = \{concurrency\ Discarded\}$
Record lc_{ik} where $i = \{concurrency\ Completed\}$

Figure 4.2: Concurrency management algorithm

status and any associated time information. The algorithm uses this information to detect concurrency relations in/between clinical activities of CIG models. If concurrency exists, then it allows adding concurrency constraints (ConcurrencyConstraint) to define concurrency relations for each clinical activity pairs. Afterwards, it facilitates the recording of concurrency start and end time. The recording of activity concurrency status (e.g., concurrencyStarted) of the concurrently implemented activities is utilised to state the status of their concurrency. Following this step, the algorithm allows the user to update the personal care plan and follow the next care step.

Step 6. Multi-activity merging is required for handling (i) care duplications which may cause conflicts if the same drug is recommended, say twice, within an overlapping period of time; this may cause drug overdose and may reduce the efficacy of other drugs, something that can result in a life threatening impact for the patient; and (ii) unnecessary health resource uses, and therefore, added health costs. For instance, requesting again the same lab test, within a short period of time for managing different diseases, may cause the existence of extra test results, and therefore, needs extra HCP time for the analysis. This has major implications of additional resource utilisation and delays at medical centres. Figure 4.3 describes the multiple clinical activity merging algorithm.

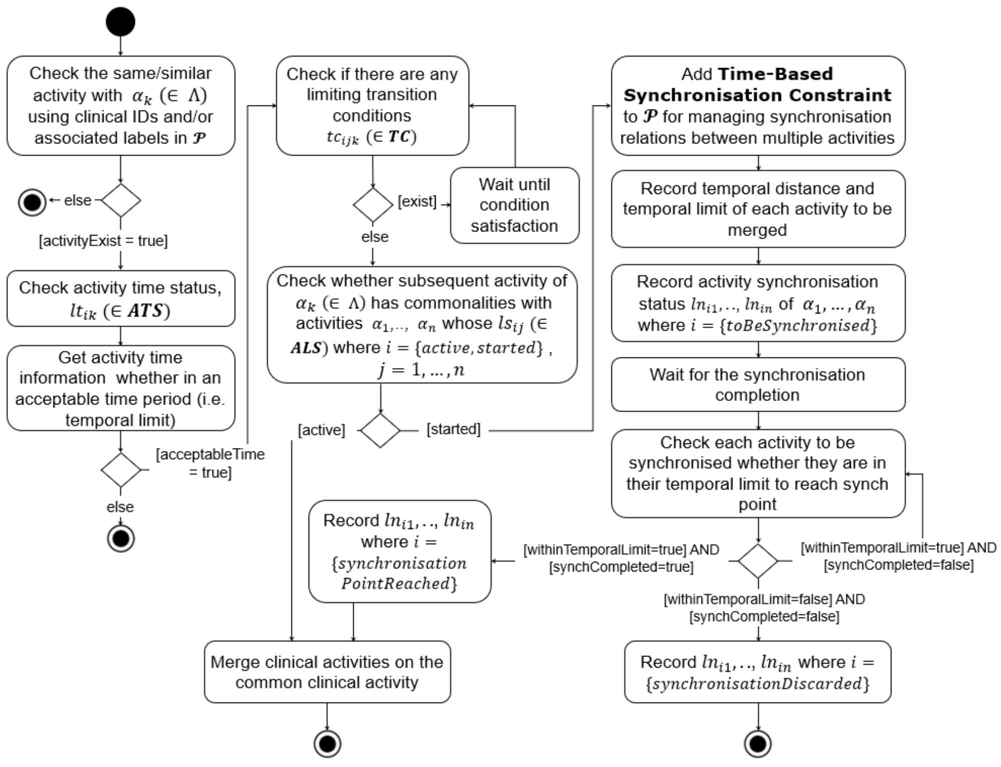

Check the same/similar activity with α_k ($\in \Lambda$) using clinical IDs and/or associated labels in $\mathcal{P}$
else
[activityExist = true]
Check activity time status, lt_{ik} ($\in$ $\boldsymbol{ATS}$)
Get activity time information whether in an acceptable time period (i.e. temporal limit)
[acceptableTime = true]
else
Check if there are any limiting transition conditions tc_{ijk} ($\in$ $\boldsymbol{TC}$)
[exist]
Wait until condition satisfaction
else
Check whether subsequent activity of α_k ($\in \Lambda$) has commonalities with activities $\alpha_1, .., \alpha_n$ whose ls_{ij} ($\in$ $\boldsymbol{ALS}$) where $i = \{active, started\}$, $j = 1, \dots, n$
[active]
[started]
Record $ln_{i1}, .., ln_{in}$ where $i = \{synchronisation PointReached\}$
Merge clinical activities on the common clinical activity
Add **Time-Based Synchronisation Constraint** to $\mathcal{P}$ for managing synchronisation relations between multiple activities
Record temporal distance and temporal limit of each activity to be merged
Record activity synchronisation status $ln_{i1}, .., ln_{in}$ of $\alpha_1, \dots, \alpha_n$ where $i = \{toBeSynchronised\}$
Wait for the synchronisation completion
Check each activity to be synchronised whether they are in their temporal limit to reach synch point
[withinTemporalLimit=true] AND [synchCompleted=true]
[withinTemporalLimit=true] AND [synchCompleted=false]
[withinTemporalLimit=false] AND [synchCompleted=false]
Record $ln_{i1}, .., ln_{in}$ where $i = \{synchronisationDiscarded\}$

Figure 4.3: Merging algorithm to unify multiple clinical activities

This algorithm is used to handle more than two concurrently implemented clinical activities recommended by the same or different CIGs (models) $\pi_1, \pi_2, \dots, \pi_n$ by discovering common activities to be merged using the clinicalID of the subsequent activity(es) (if they exist) and/or their class names for potential merging of clinical activities and then synchronise them under a set of constraints.

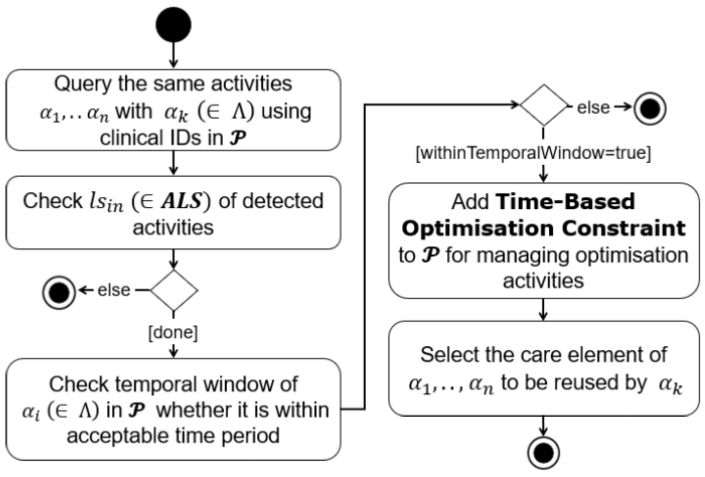
If a common activity exists, then the activity time status of this activity should be recorded, stating the expected or actual time information or a given time period that the subsequent activity must be performed. If the potential subsequent activity where multiple clinical activities are to be merged, is within the acceptable time period for each clinical activity, then the algorithm checks the activity life cycle status for such activities, as well as the associated transition conditions which may limit the transition (e.g., isConditionMet = "false"). If there is no limiting condition but activities need to be synchronised, then the algorithm creates a synchronisation constraint (TimeBasedSynchronisationConstraint) to record the synchronisation status for merging such activities.

Each activity must have its own synchronisation constraint. If the activity is reached to the synchronisation point, within an acceptable time period (where activities are associated with a temporal distance to reach the synchronisation time point), then the activity synchronisation status for each activity should be stated. If a synchronisation point is reached within the temporal limit (i.e. the exact or time interval of the deadline), then this activity is associated with the synchronisationPointReached literal. When synchronisations of multiple activities are completed, their transitions to the subsequent (to be merged) activity, as well as their associated transition labels (e.g., isMergedAt), are created and added to the personal care plan. If activities could not reach to the synchronisation point within the temporal limit, then synchronisation of activities will be discarded (synchronisationDiscarded). These activities are then followed through their respective care paths, as stated in their associated CIG models.

Step 7. This step targets to eliminate care duplications using the time-based care optimisation algorithm, see Figure 4.4. Handling of care duplications is also addressed in multiple clinical activity merging approach, but the major difference of the optimisation approach is in reusing existing activities if they are same or similar with the clinical activity to be recommended and their activity synchronisations are not forced. Reuse can be for the result of a clinical activity such as an examination result, or for the reuse of its care element such as drug information to avoid medication duplications. The reuse time period (i.e. temporal window) of the result of a clinical action should be within the acceptable time range (i.e. withinTemporalWindow = "true"). If the reuse is performed for a care element, the temporal window of the reuse must be stated as well.

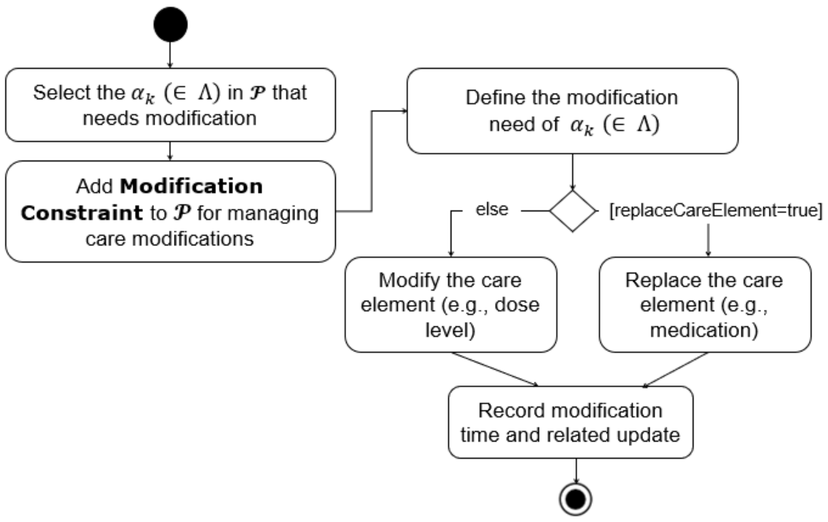
Step 8. Care modification is required for resolving (potential) or detected conflicts which can be prompted by a set of adverse interactions such as drug-drug, drug-patient, drug-disease and other interactions (e.g., timing) and may result in an undesired patient outcome if they are not handled; or updating a clinical activity such as duration or periodicity. Figure 4.5 describes the care modification algorithm which can be used to perform the care modification activities. Modification is performed when the modification needed activity is known (i.e. the specification of conflict degree is not performed).

Step 9. Activity lifecycle status of activity α_k ($\in \Lambda$) is updated and its following activity considering its outgoing transitions ($\mathsf{TransitionAssigned}$) and associated conditions and restrictions ($\mathsf{TransitionConditionAssigned}$) $\varphi_1, \varphi_2, \ldots, \varphi_n$ are queried.



Query the same activities $\alpha_1, .. \alpha_n$ with α_k ($\in \Lambda$) using clinical IDs in $\mathcal{P}$
Check ls_{in} ($\in$ ***ALS***) of detected activities
else
[done]
Check temporal window of α_i ($\in \Lambda$) in $\mathcal{P}$ whether it is within acceptable time period
else
[withinTemporalWindow=true]
Add **Time-Based Optimisation Constraint** to $\mathcal{P}$ for managing optimisation activities
Select the care element of $\alpha_1, .., \alpha_n$ to be reused by α_k

Figure 4.4: Time-based care optimisation algorithm



Select the α_k ($\in \Lambda$) in $\mathcal{P}$ that needs modification
Add **Modification Constraint** to $\mathcal{P}$ for managing care modifications
Define the modification need of α_k ($\in \Lambda$)
else
[replaceCareElement=true]
Modify the care element (e.g., dose level)
Replace the care element (e.g., medication)
Record modification time and related update

Figure 4.5: Modification algorithm

<u>Dynamic CIG Verification</u>

Step 10. Dynamic CIG verification is applied in the entire care process to assess dependencies between constraints and generate customised error messages for users to repair inconsistencies in the model. Thus, inconsistencies, missing information or errors can be detected and fixed. To do so, a set of constraints with related customised error messages are developed. These involve: syntactic inconsistency and missing information checking such as each knowledge instance of a class in the model, which must have a label, to maintain syntactic consistency; and execution logic inconsistency and missing information checking such as whether the defined care workflow has no cycle, each CIG model that represents a health condition defined under the personal care plan, which must have minimum one source and target activities; each guideline entry, must have one starting and minimum one conclusion activity; and each decision activity, must have minimum two conditional options and many others.

End Of Care

Step 11. This step checks whether there is a remaining activity exists in $\mathcal{P}$ and wait user input to conclude the care plan.

Section 5.2 presents the implementations of MuCEE based on the defined execution steps on multimorbidity case studies; and Section 6.4 discusses its evaluation results.

4.5 Summary

This chapter presents MuCEE – the real-time multiple CIG execution engine of MuCIGREF. This engine is designed to resolve the multiple CIG combination problem (MCCP) which requires to dynamically and concurrently handle multiple CIGs (models) involving a set of complexities (e.g., constraints) to generate personal care recommendations for a multimorbid patient. MuCEE involves three modules that are associated with a novel comprehensive execution algorithm. Modules are designed to meet execution requirements of CIGs. As a contribution, this thesis included more executional requirements into this list based on the needs of multimorbidity care management. These requirements are meeting; concurrency and synchronisation relation management needs between multiple clinical activities recommended by concurrently implemented CIGs; optimisation and modification needs of clinical activities to maintain patient safety and supply more flexibility to users over the guideline; and customised and interactive user support needs for error, inconsistency and missing information handling which helps to save time and improve the care model. In the following section, the implementations of MuCEE in real-world cases are presented. Its evaluation results are discussed in Section 6.4.

Chapter 5

MuCIGREF Implementation: Multimorbidity Case Studies

5.1 Introduction

Combination of demographics, diseases, lab results, medication use, genetics, health conditions and care preferences generates a heterogeneous patient population. To make a personalise care recommendation for a patient, a HCP needs to consolidate all these elements. Figure 5.1 presents an overview of how MuCIGREF can be used by HCPs in generating a personal care plan for a multimorbid patient where multiple CPGs are concurrently executed.

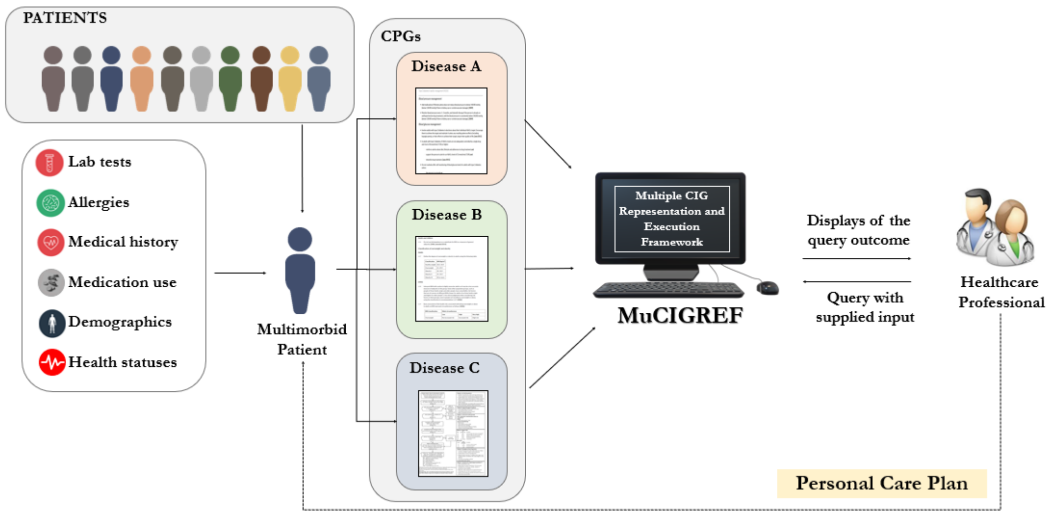


Figure 5.1: An overview of personal care plan generation with MuCIGREF

This chapter presents the implementations of MuCIGREF's guideline representation language, MuCRL, in representing multiple CPGs and their interrelations, and MuCIGREF's real-time execution engine, MuCEE, on multimorbidity case studies to test the applicability of the framework in managing multimorbidity care.

Initially, MuCIGREF's implementation method is presented, involving how guidelines are selected and their combinations representing patient with multiple health conditions are made. Afterwards, multimorbidity case studies are considered. Here, implementation is initiated by creating a personal care plan. This follows the acquisition of CIG models, which are the instantiation of Ecore metamodel (i.e. the implementation of MuCRL) based on the patient health conditions. Lastly, real-time executions of multiple, concurrently applied CIG models are dynamically implemented over the personal care plan. A set of patient scenarios are considered and different challenges of multimorbidity care such as merging of CPGs, and their actions and creation of a personal care plan whose care recommendations must work in an integrity with patient data, are addressed. To deal with these challenges, (i) concurrency management of guideline activities to capture and record overlapping clinical activities; (ii) merging more than two concurrently implemented guideline activities recommended by multiple CPGs to avoid duplications; (iii) guideline activity modification to support patient safety (e.g., medication replacement); (iv) optimisation of guideline activities and their interrelations to optimise health resource uses and improve care efficiency; and lastly (v) interactive CIG verification on individual CIGs and on personal care plan where multiple CIGs are combined. These have not been adequately addressed in the existing literature yet [31].

Implementation results demonstrate that MuCIGREF substantially satisfies the afore-mentioned multimorbidity care management challenges and execution requirements (see Section 4.2).

5.2 MuCIGREF Implementation Method

MuCIGREF's implementation method covers how guidelines and their combinations are selected and subsequently implementation specifications are supplied.

5.2.1 Guideline Selections and Their Combinations

Initially, several guidelines are selected from NICE (www.nice.org.uk) based on the suggestions of the existing literature for the testing the framework. Each guideline has its own clinical guidance as assessment, counselling, diagnosis, evaluation, management or prevention; and patient group as infant, child, adult, or elderly. The considered guidelines for the implementations are as follows:

– Diabetes (DB)	– Chronic Heart Failure (CHF)
– Chronic Kidney Disease (CKD)	– Atrial Fibrillation (AF)
– Hypertension (HTN)	– Depression (DP)

Guideline combinations to represent multimorbidity care are made mainly based the guidelines addressed in the H2020 C3-cloud Project (*www.c3-cloud.eu*) [67, 68], where guidelines are checked by a clinical reference group; case studies and/or suggestions of the existing literature [28, 36, 114, 189, 242], and NICE recommendations (i.e. references of related CPGs are supplied in each case study). The combinations involve:

- Diabetes — Hypertension
- Diabetes — Hypertension — Chronic Heart Failure
- Diabetes — Hypertension — Chronic Kidney Disease — Atrial Fibrillation

In each case study, CPG names, and their sources are provided.

5.2.2 Implementation Specifications

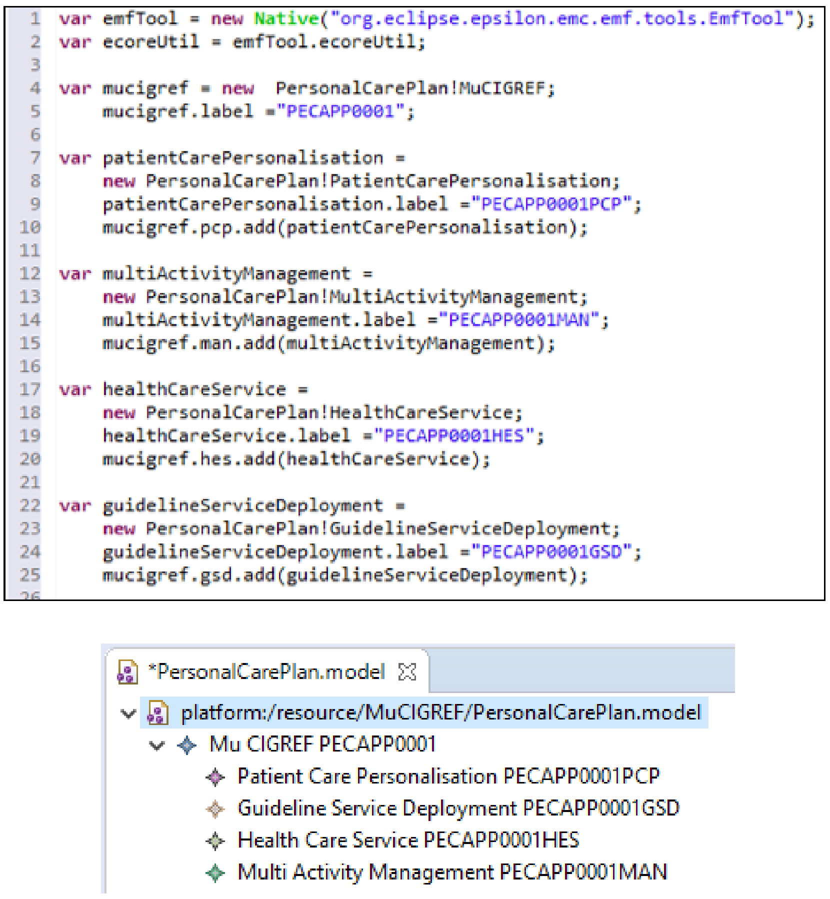
Implementation begins with creation of EMF models (instantiation of Ecore metamodel) which are called CIG models. These models represent instances of a specific guideline such as diabetes, depression, obesity and their associations such as transition conditions and temporal constraints (e.g., duration, periodicity). In this thesis, guideline instances are manually created. Afterwards, CIG executions using MuCEE, the real-time CIG execution engine are performed on multimorbidity case studies based on the algorithms provided in Section 4.4. Thereafter, MuCEE performs execution, using EOL for different patient scenarios with different combinations of multimorbid conditions. Lastly, EVL is used for real-time verification of CIGs and their associations.

5.3 Multimorbidity Care Management: Case Studies

This section discusses care initialisation process, multiple CIG model acquisitions, activity sequencing, and knowledge mapping discovery. Afterwards, combining multiple CIGs by addressing four challenges as concurrency management, multi-activity merging, care modification and optimisation with the case studies is discussed.

5.3.1 Creating A Personal Care Plan

To initialise care, a personal care plan (PECAP), $\mathcal{P}$ needs to be created where dynamic CIG executions, using patient data, are subsequently performed on this plan, after the relevant CIG models are acquired. Figure 5.2 presents an example of creating a personal care plan for a patient whose ID is P0001. Afterwards, the PatientCarePersonalisation is created to record patient data, and the GuidelineServiceDeployment is created to record guideline information and HCP data (Carer). Following this, the HealthCareService is created to record patient-HCP encounter (PatientEncounter), where the health condition information such as signs, symptoms and diseases of a patient. Lastly, the MultiActivityManagement is created to manage multiple activities recommended by multiple CIG models when the related CIG models are acquired. The latter is discussed in the following section.



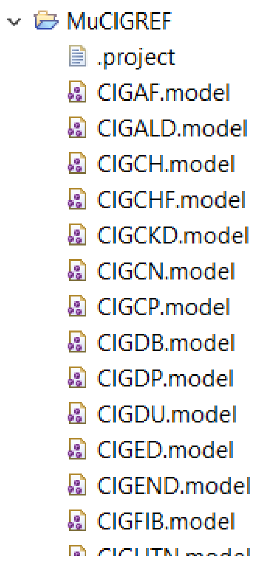
```
var emfTool = new Native("org.eclipse.epsilon.emc.emf.tools.EmfTool");
var ecoreUtil = emfTool.ecoreUtil;

var mucigref = new  PersonalCarePlan!MuCIGREF;
    mucigref.label ="PECAPP0001";

var patientCarePersonalisation =
    new PersonalCarePlan!PatientCarePersonalisation;
    patientCarePersonalisation.label ="PECAPP0001PCP";
    mucigref.pcp.add(patientCarePersonalisation);

var multiActivityManagement =
    new PersonalCarePlan!MultiActivityManagement;
    multiActivityManagement.label ="PECAPP0001MAN";
    mucigref.man.add(multiActivityManagement);

var healthCareService =
    new PersonalCarePlan!HealthCareService;
    healthCareService.label ="PECAPP0001HES";
    mucigref.hes.add(healthCareService);

var guidelineServiceDeployment =
    new PersonalCarePlan!GuidelineServiceDeployment;
    guidelineServiceDeployment.label ="PECAPP0001GSD";
    mucigref.gsd.add(guidelineServiceDeployment);
```

```
*PersonalCarePlan.model
platform:/resource/MuCIGREF/PersonalCarePlan.model
  Mu CIGREF PECAPP0001
    Patient Care Personalisation PECAPP0001PCP
    Guideline Service Deployment PECAPP0001GSD
    Health Care Service PECAPP0001HES
    Multi Activity Management PECAPP0001MAN
```

Figure 5.2: Example of creating a personal care plan

5.3.2 Acquisition of CIG Models

Each CPG is designed as a CIG model based on its scope (e.g., management, diagnosis, recognition, prevention, etc.) and patient group (e.g., child, adult, etc.). CIG models of CPGs can be stored in the repository as shown in Figure 5.3.



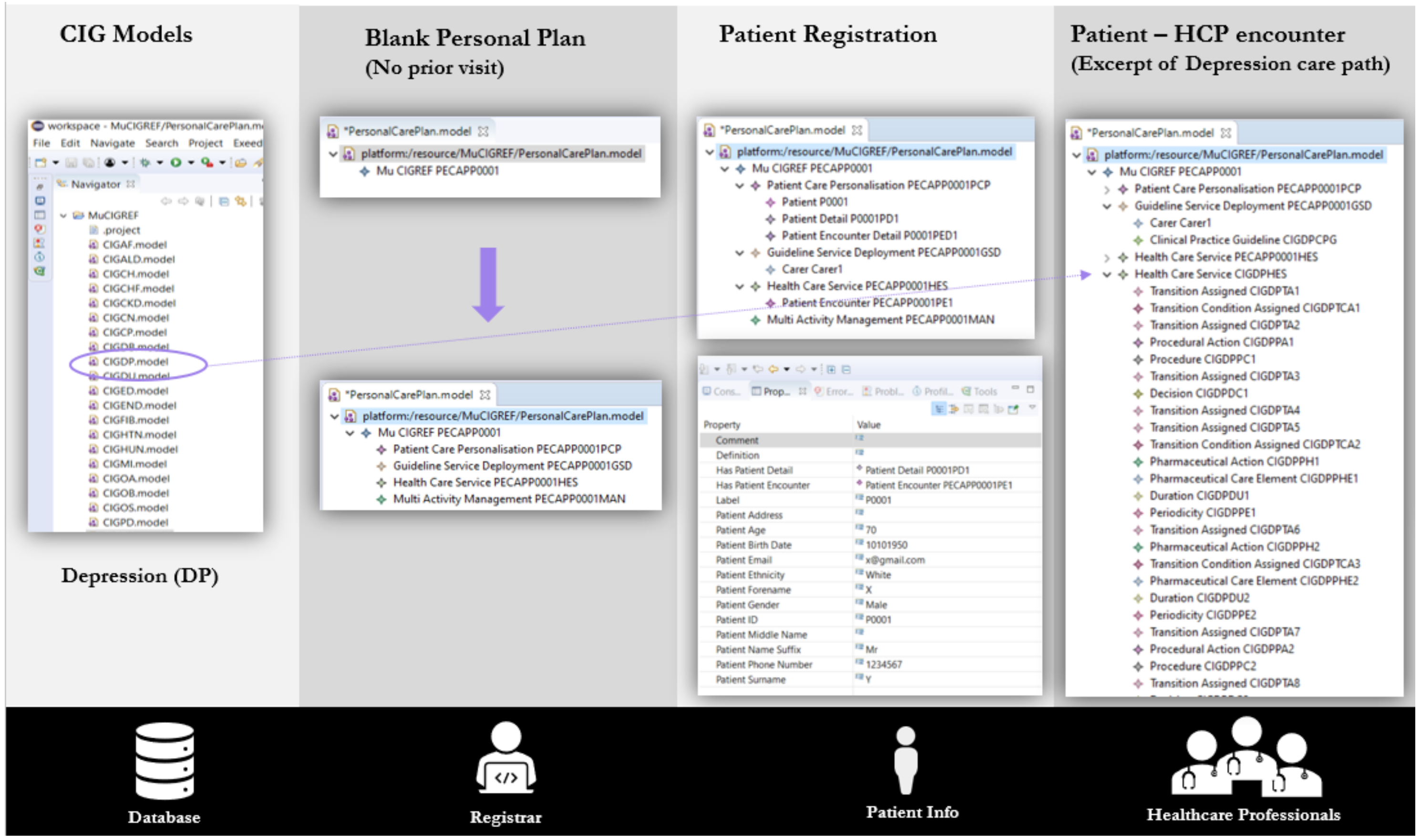
MuCIGREF
.project
CIGAF.model
CIGALD.model
CIGCH.model
CIGCHF.model
CIGCKD.model
CIGCN.model
CIGCP.model
CIGDB.model
CIGDP.model
CIGDU.model
CIGED.model
CIGEND.model
CIGFIB.model

Figure 5.3: A list of CIG models in the repository

Each CIG model has a label starting with <CIG...> and followed with the abbreviation of the guidelines such as diabetes (DB) and hypertension (HTN). Under the CIG model, each class has an instance that associated with a unique label. To illustrate, Figure 5.4 presents a portion of depression in adults: recognition and management guideline [240] that is acquired from the repository (database). It is then transferred to personal care plan of a patient based on the supplied patient information. Then, HCP(s) of patient manage his/her care on this care plan. Each CIG model must involve an instance of a CPG (ClinicalPracticeGuideline) defined under the GuidelineServiceDeployment and the instances of the HealthCareService.

In CIG models, the class names are used as abbreviations such as pharmaceutical action as PH, examination action EA or decision DC along with the implementation sequence number for the instance labelling. For example, <CIGDPPH1> represents the first instance of a pharmaceutical action of a depression management guideline.

For a CIG model acquisition, activity life cycle statuses, of each activity must be checked whether they are eligible for transferring to the personal care plan (see Section 4.4, *CIG acquisition algorithm*). In CIG models, the activity lifecycle status



```
workspace - MuCIGREF/PersonalCarePlan.m
File  Edit  Navigate  Search  Project  Exeed
Navigator
v MuCIGREF
    .project
    CIGAF.model
    CIGALD.model
    CIGCH.model
    CIGCHF.model
    CIGCKD.model
    CIGCN.model
    CIGCP.model
    CIGDB.model
    CIGDP.model
    CIGDU.model
    CIGED.model
    CIGEND.model
    CIGFIB.model
    CIGHTN.model
    CIGHUN.model
    CIGMI.model
    CIGOA.model
    CIGOB.model
    CIGOS.model
    CIGPD.model
```

```
*PersonalCarePlan.model
v platform:/resource/MuCIGREF/PersonalCarePlan.model
    Mu CIGREF PECAPP0001
```

```
*PersonalCarePlan.model
v platform:/resource/MuCIGREF/PersonalCarePlan.model
  v Mu CIGREF PECAPP0001
      Patient Care Personalisation PECAPP0001PCP
      Guideline Service Deployment PECAPP0001GSD
      Health Care Service PECAPP0001HES
      Multi Activity Management PECAPP0001MAN
```

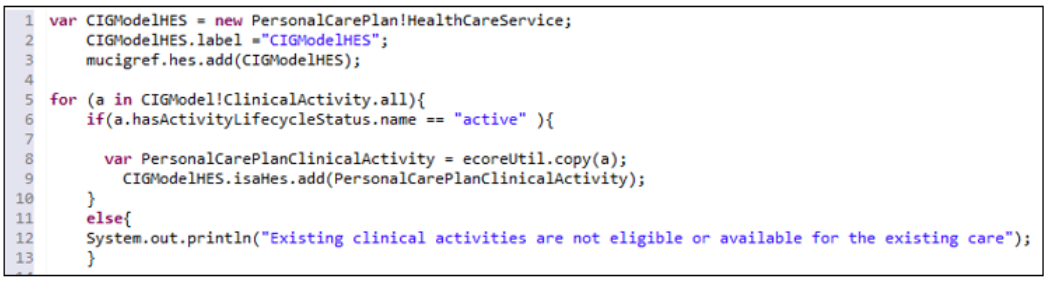
```
*PersonalCarePlan.model
v platform:/resource/MuCIGREF/PersonalCarePlan.model
  v Mu CIGREF PECAPP0001
    v Patient Care Personalisation PECAPP0001PCP
        Patient P0001
        Patient Detail P0001PD1
        Patient Encounter Detail P0001PED1
    v Guideline Service Deployment PECAPP0001GSD
        Carer Carer1
    v Health Care Service PECAPP0001HES
        Patient Encounter PECAPP0001PE1
      Multi Activity Management PECAPP0001MAN
```

```
Cons...  Prop...  Error...  Probl...  Profil...  Tools
Property                  Value
Comment
Definition
Has Patient Detail        Patient Detail P0001PD1
Has Patient Encounter     Patient Encounter PECAPP0001PE1
Label                     P0001
Patient Address
Patient Age               70
Patient Birth Date        10101950
Patient Email             x@gmail.com
Patient Ethnicity         White
Patient Forename          X
Patient Gender            Male
Patient ID                P0001
Patient Middle Name
Patient Name Suffix       Mr
Patient Phone Number      1234567
Patient Surname           Y
```

```
*PersonalCarePlan.model
v platform:/resource/MuCIGREF/PersonalCarePlan.model
  v Mu CIGREF PECAPP0001
    > Patient Care Personalisation PECAPP0001PCP
    v Guideline Service Deployment PECAPP0001GSD
        Carer Carer1
        Clinical Practice Guideline CIGDPCPG
    > Health Care Service PECAPP0001HES
    v Health Care Service CIGDPHES
        Transition Assigned CIGDPTA1
        Transition Condition Assigned CIGDPTCA1
        Transition Assigned CIGDPTA2
        Procedural Action CIGDPPA1
        Procedure CIGDPPC1
        Transition Assigned CIGDPTA3
        Decision CIGDPDC1
        Transition Assigned CIGDPTA4
        Transition Assigned CIGDPTA5
        Transition Condition Assigned CIGDPTCA2
        Pharmaceutical Action CIGDPPH1
        Pharmaceutical Care Element CIGDPPHE1
        Duration CIGDPDU1
        Periodicity CIGDPPE1
        Transition Assigned CIGDPTA6
        Pharmaceutical Action CIGDPPH2
        Transition Condition Assigned CIGDPTCA3
        Pharmaceutical Care Element CIGDPPHE2
        Duration CIGDPDU2
        Periodicity CIGDPPE2
        Transition Assigned CIGDPTA7
        Procedural Action CIGDPPA2
        Procedure CIGDPPC2
        Transition Assigned CIGDPTA8
```

Figure 5.4: Real-time CIG acquisition of depression in adults: recognition and management guideline

of clinical activities must be either active (i.e. these activities are eligible (ready) for execution) or passive (i.e. activities are not currently eligible for execution or absolute) when they stored in the repository. When a HCP and patient encounter and the patient is diagnosed with a disease(s), then activities of the selected CIG model can be acquired (transferred) if and only if activities have active instances to the personal care plan, see Figure 5.5. Clinical activities are part of the HES (HealthCareService).



```
var CIGModelHES = new PersonalCarePlan!HealthCareService;
    CIGModelHES.label ="CIGModelHES";
    mucigref.hes.add(CIGModelHES);

for (a in CIGModel!ClinicalActivity.all){
    if(a.hasActivityLifecycleStatus.name == "active" ){

      var PersonalCarePlanClinicalActivity = ecoreUtil.copy(a);
        CIGModelHES.isaHes.add(PersonalCarePlanClinicalActivity);
    }
    else{
    System.out.println("Existing clinical activities are not eligible or available for the existing care");
    }
```

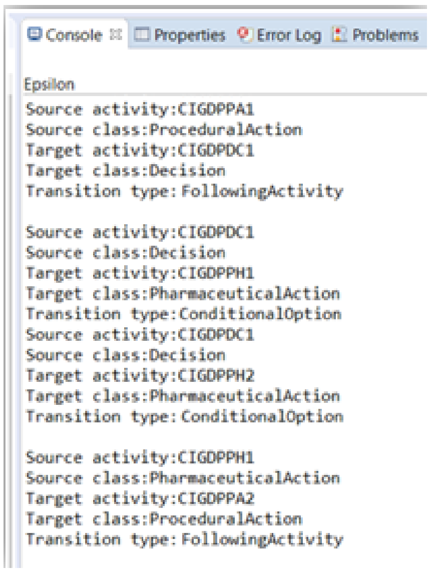
Figure 5.5: Example code of CIG model acquisition

Initially, one CIG model can be acquired based on the patient health condition (disease) and added to the personal care plan. However, a patient may have more diseases in time, and therefore, more CIG models need to be acquired and added to the personal care plan. These are shown in the following section.

5.3.3 Clinical Activity Sequencing in a Personal Care Plan

When the related CIG models are selected based on the patients' health conditions, clinical activities and their hierarchical orders need to be determined. Figure 5.6 presents the portion of activity sequencing view of depression in adults: recognition and management guideline [240]) to supply more insights into readers on how activity sequencing can be performed. However, this is initial activity sequences as presented CIGDP model – conditions are gradually satisfied based on the execution and activity sequences can change accordingly. <CIGDP> represents the label of the depression guideline. <CIGDPDC1> represents the first decision activity of this guideline which has two care options as <CIGDPPH1> and <CIGDPPH2> that represent two pharmaceutical actions of a depression guideline. These two pharmaceutical actions are associated with an activating condition (s) (e.g., "adequate response to medication?").

This can be performed by checking dependencies and constraints between activities (namely classes or vertices) for recommending the ones involving patient data. Clinical activity sequencing can be performed as described below:

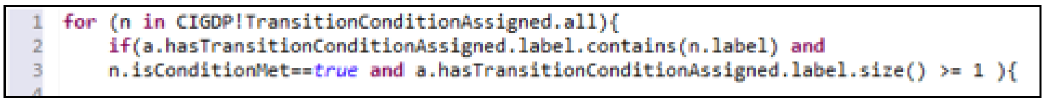


```
Epsilon
Source activity:CIGDPPA1
Source class:ProceduralAction
Target activity:CIGDPDC1
Target class:Decision
Transition type: FollowingActivity

Source activity:CIGDPDC1
Source class:Decision
Target activity:CIGDPPH1
Target class:PharmaceuticalAction
Transition type:ConditionalOption
Source activity:CIGDPDC1
Source class:Decision
Target activity:CIGDPPH2
Target class:PharmaceuticalAction
Transition type: ConditionalOption

Source activity:CIGDPPH1
Source class:PharmaceuticalAction
Target activity:CIGDPPA2
Target class:ProceduralAction
Transition type: FollowingActivity
```

Figure 5.6: Excerpt of depression guideline activity sequence

The clinical activity sequencing begins with finding outgoing transitions (TransitionAssigned) of the initial clinical activity (ClinicalActivity) recommended by a CIG in the personal care plan. Afterwards, whether any associated limiting condition (TransitionConditionAssigned) or activity status (ActivityLifecycleStatus) (e.g., started) exist, these need to be checked in order to realise transitions between activities. If there is no limiting condition, such as a precondition (e.g., the lab test result must be ready before the decision activity), then the algorithm checks the type of transition (TransitionLabel) which can be one of the following instances: FollowingActivity, isMergedAt, Split, ConditionalOption, AlternativeActivity, or AlternativeConditionalOption. Lastly, the algorithm allows the choices of one option and permits following the care steps until the end of care.

The sequence of clinical activities of an acquired CIG model can be automatically listed, if they are satisfied required transition conditions. Figure 5.7 presents the excerpt of transition condition satisfaction codes.



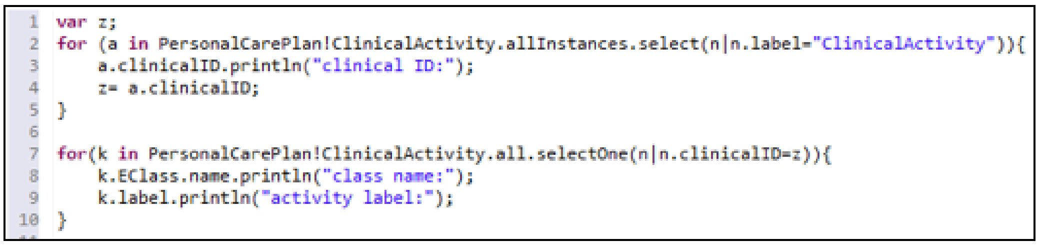
```
for (n in CIGDP!TransitionConditionAssigned.all){
    if(a.hasTransitionConditionAssigned.label.contains(n.label) and
    n.isConditionMet==true and a.hasTransitionConditionAssigned.label.size() >= 1 ){
```

Figure 5.7: Example code of discovering activities that satisfy transition conditions

5.3.4 Mapping Discovery in a Personal Care Plan

Mapping discovery is the knowledge mapping strategy of this thesis which helps users to detect semantic relationships between model entities such as classes, instances, attributes and relations. This is required to create required links between multiple concurrently implemented CIGs to manage multimorbidity. To achieve this, similarities or commonalities between ontology entities such as clinical IDs, concept names, and constraints (e.g., activity lifecycle status, condition realisation statements as Boolean variables as "true" or "false") are used to restrict the solution space, recommendation set. EOL is used to develop query functions based on first-order logic (FOL) OCL operations such as `select`, `selectOne`, `collect`, `exits`, etc. (see [65]) to perform mapping discovery through queries.

In Figure 5.8, the excerpt of mapping discovery between activities in personal care plan (i.e. all the clinical activities of guidelines related with patient diseases) using first-order queries is shown where clinical ID and label of a clinical activity (e.g., <CIGHTNPH1>) are automatically used to restrict the solution set.



```
var z;
for (a in PersonalCarePlan!ClinicalActivity.allInstances.select(n|n.label="ClinicalActivity")){
    a.clinicalID.println("clinical ID:");
    z= a.clinicalID;
}

for(k in PersonalCarePlan!ClinicalActivity.all.selectOne(n|n.clinicalID=z)){
    k.EClass.name.println("class name:");
    k.label.println("activity label:");
}
```

Figure 5.8: Example of mapping discovery in a personal care plan

When activities are found, required mappings between them will be realised. These can be used for activity merging, or activity reuse. The Figure 5.8 just shows a simple illustration of how queries work in MuCEE to establish required mappings between multiple concurrently implemented clinical activities.

Following section presents how multiple concurrently implemented CIG models can be managed and personal care plan for a multimorbid patient can be generated by demonstrating challenges on various patient scenarios.

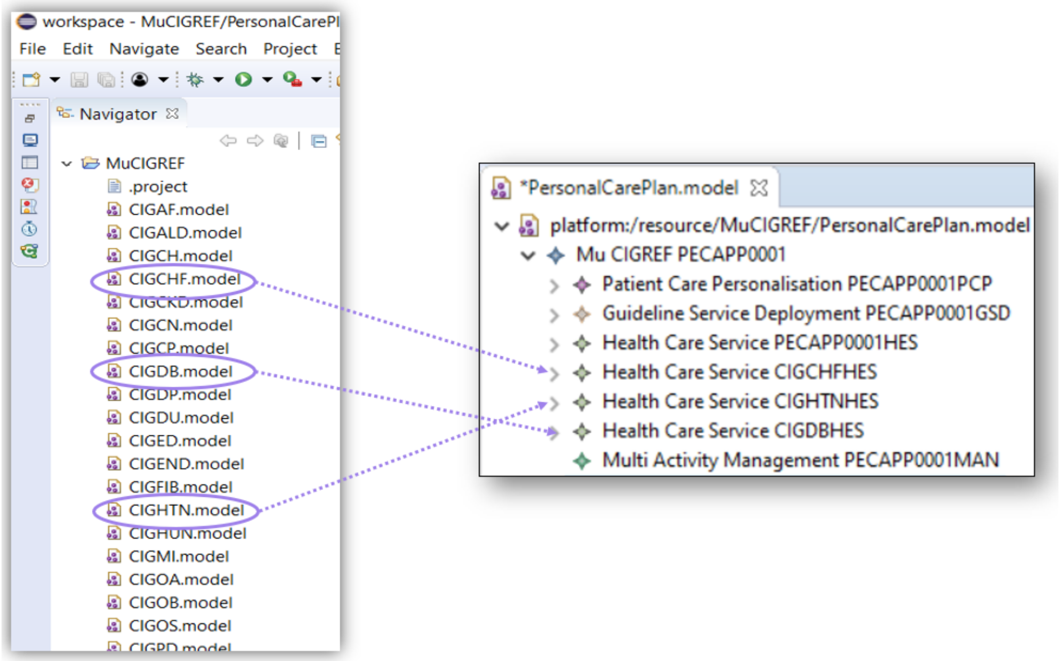
5.3.5 Combining Multiple Concurrently Implemented Guidelines

This section presents a multimorbid patient case study which involves four examples addressing the challenges in combining synchronously implemented multiple CIGs, which are required for managing multimorbidity care, and how MuCEE handles these challenges.

Example 5.3.1. (Diabetes – Hypertension – Chronic Heart Failure) A patient initially got diabetes care. Afterwards, she presented to the hospital where she diagnosed with hypertension. Today, she presented to the hospital with high blood pressure again with the signs of chronic heart failure.

In this case study (see Table 5.1), combined personal care plan of a multimorbid patient involves DB, HTN and CHF guidelines that are implemented in parallel. The scope of this case study is to show how concurrency relations of clinical actions recommended by same/different guidelines can be managed and their multiple common actions are merged and careflow can be optimised using MuCEE based on the algorithm presented in Section 4.4.

To simplify the illustration of multiple CIG management, activity management examples are shown after the associated guidelines have been acquired from the repository based on the patient health condition, and transferred to the personal care plan based on the recommendations of each guideline as shown in Figure 5.9 (please see *CIG acquisition algorithm* in Section 4.4). Then, all the aforementioned multimorbidity care management actions are going to be performed on this combined care plan.



MuCIGREF
.project
CIGAF.model
CIGALD.model
CIGCH.model
CIGCHF.model
CIGCKD.model
CIGCN.model
CIGCP.model
CIGDB.model
CIGDP.model
CIGDU.model
CIGED.model
CIGEND.model
CIGFIB.model
CIGHTN.model
CIGHUN.model
CIGMI.model
CIGOA.model
CIGOB.model
CIGOS.model
CIGPD.model

platform:/resource/MuCIGREF/PersonalCarePlan.model
Mu CIGREF PECAPP0001
Patient Care Personalisation PECAPP0001PCP
Guideline Service Deployment PECAPP0001GSD
Health Care Service PECAPP0001HES
Health Care Service CIGCHFHES
Health Care Service CIGHTNHES
Health Care Service CIGDBHES
Multi Activity Management PECAPP0001MAN

Figure 5.9: Acquisition of multiple CIGs from the repository

Table 5.1: Case Study: Diabetes – Hypertension – Chronic Heart Failure

Patient scenario	DB (primary disease); HTN (secondary disease); CHF (tertiary disease) | HbA1c level $>$ 53 mmol/mol; Fluid overload exists; Microvascular and/or cardiovascular complication = "yes"; BP level $>$ 140/80mmHg; eGFR $>$ 45 ml/min/1.73m2.
CPG sources	DB: Type-II diabetes in adults: management – C3-cloud [67] (Chapter 5.2 , pp. 21–42) and NICE [228]; HTN: Hypertension in adults: diagnosis and management – NICE [241]; CHF: Chronic heart failure in adults: management – C3-cloud [67] (Chapter 5.3, pp. 42–47) and NICE [226]
Care information	DB: Metformin, Education; HTN: ACEi, Diuretic; CHF: ACEi, Diuretic, Education
Merged care	DB + HTN + CHF: Metformin, ACEi, Diuretic
Clinical ID	ACEi product: 41549009 (STC); ACEi therapy: 410682003 (STC); Metformin product: 109081006 (STC); Metformin therapy: 1097191000000106 (STC); Diuretic product: 30492008 (STC); Diuretic therapy: 722048006 (STC)
Abbreviations	DB: Diabetes; HTN: Hypertension; CHF: Chronic Heart Failure; eGFR: Estimated Glomerular Filtration Rate; HbA1c: Hemoglobin A1c; BP: Blood pressure; ACEi: Angiotensin-converting enzyme inhibitor; STC: SNOMED-CT code

Concurrency management

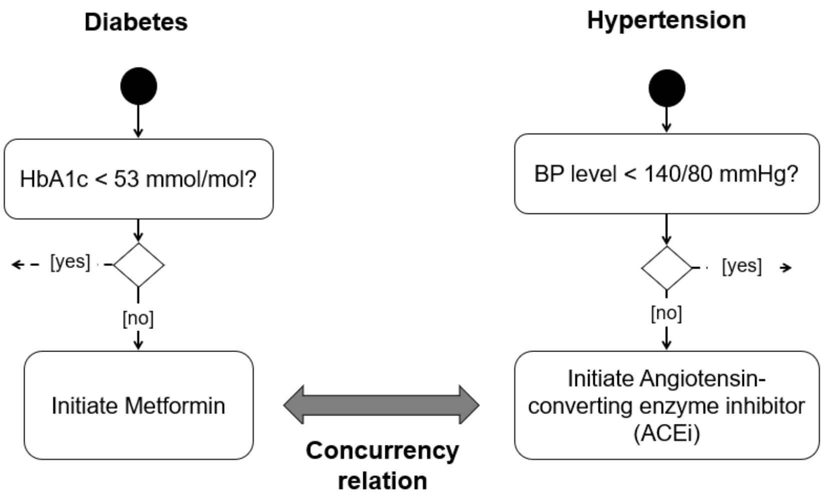
Initially, the multimorbid patient's type-II diabetes is managed as the primary disease. Since the most recent measurement of HbA1c result was higher than 58 mmol/-mol, HP prescribed Metformin (clinicalID "109081006" (STC)) and started the drug therapy (clinicalID "1097191000000106" (STC)). The secondary disease of patient is hypertension. Based on the measured BP levels of the patient, HCP prescribed ACEi (clinicalID "41549009" (STC)) medication and, therefore, started the drug therapy (clinicalID "410682003" (STC)) for managing high blood pressure level of the patient. Because initially DB and HTN guidelines are implemented in parallel, their pharmaceutical actions – Metformin and ACEi drug therapies- have concurrency relations. Capturing these relations in/between CIG models are important because they support avoiding drug overdose, or conflicting drug interactions (e.g., [114]).

In the current patient scenario, the concurrency management is performed as follows: the secondary disease management activity is initialised, considering the activity lifecycle status (ActivityLifecycleStatus) of the existing clinical activity and concurrency relations between already started clinical activities whose activity life cycle statuses are started. If there are existing clinical activities which have started statuses, then check their time periods and time statuses (ActivityTimeStatus) and their associated class names (e.g., PharmaceuticalAction).

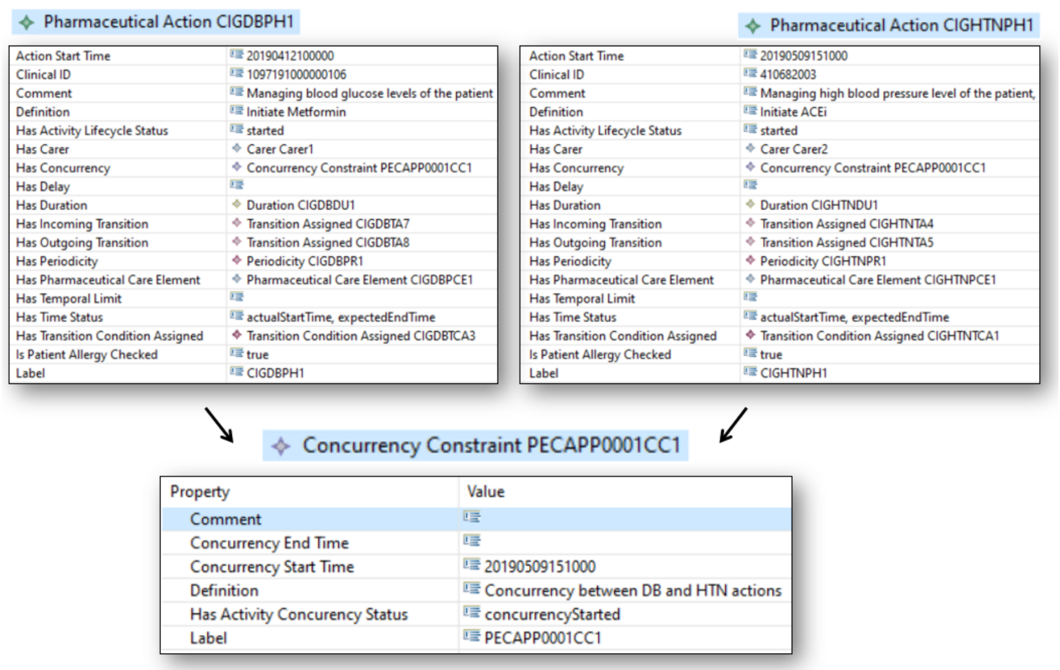
Concurrency constraints (ConcurrencyConstraint) are needed to record concurrency relations between multiple clinical activity pairs. These can be used to detect when and which pairs are performed together. If clinical activities are going to be performed within the same time period, then they are recorded through creating concurrency constraints. These constraints are mainly designed for recording concurrencies between pharmaceutical actions whose concurrencies may cause unwanted patient outcome, or other actions (e.g., education, diet recommendations), and examination actions whose concurrencies may cause unnecessary health resource use. However, further clinical actions can also be recorded if users wish.

Figure 5.10a shows the portion of HTN and DB guideline combinations to illustrate multiple activity concurrency management. Initially, the user must detect whether any concurrency relation exists between the existing drug treatment(s) before initialising a new treatment, see the *Concurrency management algorithm* in Section 4.4, execution step #4). In this case study, the pharmaceutical action (PharmaceuticalAction) of DB, <CIGDBPH1> "Initiate Metformin" (clinicalID "1097191000000106" (STC)) to manage blood glucose levels of the patient. The activity lifecycle status (ActivityLifecycleStatus) of this activity is initially active and then transformed to started. Then a new drug treatment is initialised, which is the pharmaceutical action of HTN, <CIGHTNPH1> "Initiate ACEi" (clinicalID "410682003" (STC)) is activated to manage high blood pressure levels if the patient whose activity lifecycle status is started. Activity time statuses of this action are the *actualStartTime* and *expectedEndTime* and related the *activityStartTime* and *activityEndTime* data values are also recorded based on the duration of this action. Thus, a concurrency constraint is added to the personal care plan, in order to record concurrency relations between these two pharmaceutical actions.

Figure 5.10b illustrates the concurrency between two clinical activities and their associated concurrency constraint. The instantiation of the ConcurrencyConstraint to define the concurrency relations between DB and HTN is added to the personal care plan with the following instance label <PECAPP0001CC1>. The *hasConcurrency* object property defined under the afore-mentioned pharmaceutical actions links them to this concurrency instance. Hence, two different CPGs are mapped through this constraint. Here, data values of *concurrencyStartTime* and *concurrencyEndTime* data properties are filled in. When the activity concurrency



(a) Excerpt of Hypertension and Diabetes guidelines



Pharmaceutical Action CIGDBPH1

Action Start Time	20190412100000
Clinical ID	1097191000000106
Comment	Managing blood glucose levels of the patient
Definition	Initiate Metformin
Has Activity Lifecycle Status	started
Has Carer	Carer Carer1
Has Concurrency	Concurrency Constraint PECAPP0001CC1
Has Delay	
Has Duration	Duration CIGDBDU1
Has Incoming Transition	Transition Assigned CIGDBTA7
Has Outgoing Transition	Transition Assigned CIGDBTA8
Has Periodicity	Periodicity CIGDBPR1
Has Pharmaceutical Care Element	Pharmaceutical Care Element CIGDBPCE1
Has Temporal Limit	
Has Time Status	actualStartTime, expectedEndTime
Has Transition Condition Assigned	Transition Condition Assigned CIGDBTCA3
Is Patient Allergy Checked	true
Label	CIGDBPH1

Pharmaceutical Action CIGHTNPH1

Action Start Time	20190509151000
Clinical ID	410682003
Comment	Managing high blood pressure level of the patient,
Definition	Initiate ACEi
Has Activity Lifecycle Status	started
Has Carer	Carer Carer2
Has Concurrency	Concurrency Constraint PECAPP0001CC1
Has Delay	
Has Duration	Duration CIGHTNDU1
Has Incoming Transition	Transition Assigned CIGHTNTA4
Has Outgoing Transition	Transition Assigned CIGHTNTA5
Has Periodicity	Periodicity CIGHTNPR1
Has Pharmaceutical Care Element	Pharmaceutical Care Element CIGHTNPCE1
Has Temporal Limit	
Has Time Status	actualStartTime, expectedEndTime
Has Transition Condition Assigned	Transition Condition Assigned CIGHTNTCA1
Is Patient Allergy Checked	true
Label	CIGHTNPH1

Concurrency Constraint PECAPP0001CC1

Property	Value
Comment	
Concurrency End Time	
Concurrency Start Time	20190509151000
Definition	Concurrency between DB and HTN actions
Has Activity Concurency Status	concurrencyStarted
Label	PECAPP0001CC1

(b) Concurrency constraint between two clinical activities

Figure 5.10: Illustration of activity concurrency management

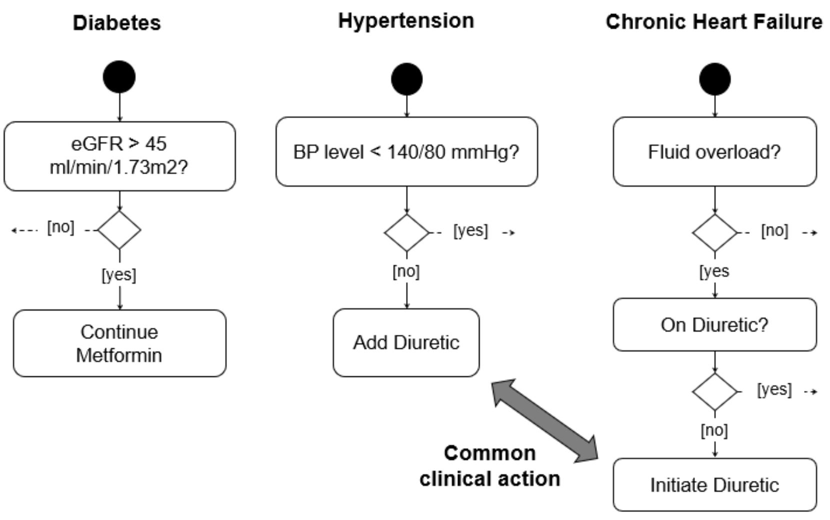
status is transformed from concurrencyStarted status to the concurrencyCompleted status, then concurrency relations between these two drug therapies can be ended.

Multi-activity merging

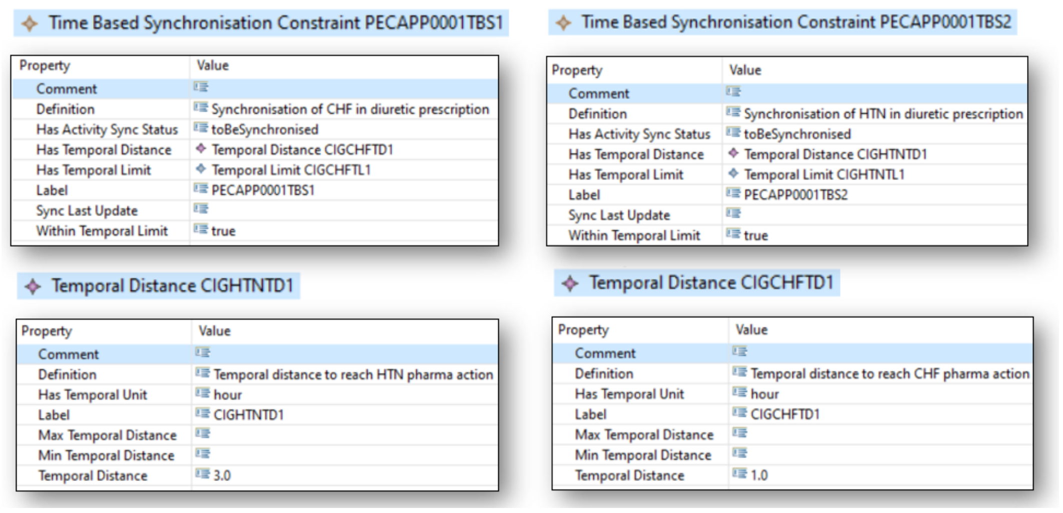
In this plan, two CIG actions, which are pharmaceutical actions of CHF and HTN, demonstrate similarities, therefore, they need to be merged to eliminate potential care duplications that may result in drug overdose. To do so, the prior actions of the considered actions should be completed and then they need to be synchronised for merging. We refer readers to Section 3.5.5 for information about multi-merging and the *Merging algorithm* presented in Section 4.4, execution step #6).

Figure 5.11a shows the HTN, DB and CHF guideline care steps to illustrate where and how merging can be performed. DB is the primary disease, in this example. The decision <CIGDBDC5> "eGFR > 45 ml/min/1.73m2?" based on the patient value in DB is given. Accordingly, care is continued with the pharmaceutical action <CIGDBPH1> "Continue metformin" (clinicalID "1097191000000106" (STC)), after the conditional option [eGFR > 45 ml/min/1.73m2 = "yes"] is satisfied. The patient is still using this drug for the care of her diabetes. The activity lifecycle status (ActivityLifecycleStatus) of this action is started since patient is still in this care step and follows its her own care path.

The secondary disease of the patient is HTN. Pharmaceutical action <CIGHTNPH1> "Initiate ACEi" (clinicalID "410682003" (STC)) is initiated to manage the patient's hypertension. One month later, patient visits the hospital again, blood pressure levels are measured, and the decision action <CIGHTNDC4> "BP level < 140/80?" is given to determine the next care step. Based on the satisfaction of conditional option [BP level < 140/80 mmHg = "no"], the pharmaceutical action <CIGHTNPH2> "Add diuretic" (clinicalID "722048006" (STC)) should be recommended to balance BP levels of the patient. Because patient shows evidence of heart failure (tertiary disease), the pharmaceutical action <CIGCHFPH1> "Initiate diuretic" (clinicalID "722048006" (STC) is recommended whose activity lifecycle status (ActivityLifecycleStatus) is active. However, CHF care of the patient has common actions (which are detected based on the clinical IDs) within the given time period considering time statuses (see ActivityTimeStatus) with the pharmaceutical action <CIGHTNPH2> of the HTN guideline, whose activity lifecycle statuses are active, as well. When similar drugs are recommended by different guidelines within the overlapping time periods, adverse disease – disease interactions may occur. These need to be handled to avoid drug overdoses. In order to do this, time-based synchronisation of care activities needs to be performed. This synchronisation can be realised after the following common activity detection and creating synchronisation constraint through involving, see Figure 5.11b:



(a) Excerpt of Hypertension, Diabetes and Chronic Heart Failure guidelines



Time Based Synchronisation Constraint PECAPP0001TBS1

Property	Value
Comment	
Definition	Synchronisation of CHF in diuretic prescription
Has Activity Sync Status	toBeSynchronised
Has Temporal Distance	Temporal Distance CIGCHFTD1
Has Temporal Limit	Temporal Limit CIGCHFTL1
Label	PECAPP0001TBS1
Sync Last Update	
Within Temporal Limit	true

Time Based Synchronisation Constraint PECAPP0001TBS2

Property	Value
Comment	
Definition	Synchronisation of HTN in diuretic prescription
Has Activity Sync Status	toBeSynchronised
Has Temporal Distance	Temporal Distance CIGHTNTD1
Has Temporal Limit	Temporal Limit CIGHTNTL1
Label	PECAPP0001TBS2
Sync Last Update	
Within Temporal Limit	true

Temporal Distance CIGHTNTD1

Property	Value
Comment	
Definition	Temporal distance to reach HTN pharma action
Has Temporal Unit	hour
Label	CIGHTNTD1
Max Temporal Distance	
Min Temporal Distance	
Temporal Distance	3.0

Temporal Distance CIGCHFTD1

Property	Value
Comment	
Definition	Temporal distance to reach CHF pharma action
Has Temporal Unit	hour
Label	CIGCHFTD1
Max Temporal Distance	
Min Temporal Distance	
Temporal Distance	1.0

(b) Time-based synchronisation constraints and associated temporal distances of synchronisation needed two clinical activities

Figure 5.11: Illustration of activity synchronisation for multi-merging

- synchronisation time limits (TemporalLimit), which represent min and max waiting time or deadline of a clinical activity to wait other clinical activities before proceeding with the next activity;
- temporal distances (TemporalDistance), which represent remaining time to reach the synchronisation time point;
- synchronisation statuses (ActivitySynhronisationStatus), which represent syn-

chronisation realisations (e.g., synhronisationPointReached); and involve synchronisation last update information.

Thus, time-based synchronisation constraints must be added to the personal care plan to define synchronisation restrictions for clinical activities that will be merged in their subsequent care steps. These activities can be linked with the *needsSynchronisation* object property to instances of the TimeBasedSynchronisationConstraint. Under the time-based synchronisation constraint, the activity synchronisation status must be recorded using the *hasActivitySynchronisationStatus* object property that links TimeBasedSynchronisationConstraint to the ActivitySynchronisationStatus. In addition, temporal distance and temporal limit values must be recorded. User input is required for the temporal limit values. The data property *withinTemporalLimit* should be filled in with the data value "true" if the synchronisation is valid and, clinical activity is still in its max temporal limit or "false" if the synchronisation is violated and, if the clinical activity exceeds its synchronisation time. Then, the synchronisation's last update also needs to be recorded to make updates on activity synchronisations. If activities successfully reach to the synchronisation time point, then they can be merged in the subsequent clinical activity. When the activity synchronisation is violated due to exceeding its waiting limit, then its synchronisation with other activities must be discarded and this activity must be activated. The other activities, which have not been started, must reuse this activity. Yet, this case is the issue of time-based care optimisation which addressed in the next example.

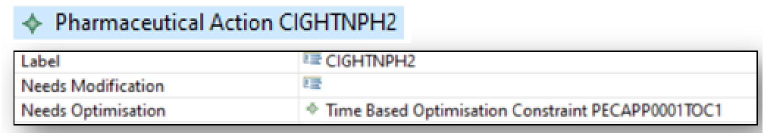
In this case, time-based synchronisation constraints <PECAPP0001TBS1> "Synchronisation of CHF in diuretic prescription" and <PECAPP0001TBS2> "Synchronisation of HTN in diuretic prescription" are added to the personal care plan. The <CIGCHFPH1> CHF pharmaceutical action and the <CIGHTNPH2> HTN pharmaceutical action are linked with the *needsSynchronisation* object property to these instances, respectively. Under the time-based synchronisation constraint, activity synchronisation status is recorded, using the *hasActivitySynchronisationStatus* object property that links the TimeBasedSynchronisationConstraint to the ActivitySynhronisationStatus. Because activities reached to their synchronisation point within the temporal limit (*withinTemporalLimit* = "true"), their concurrency statuses (ActivityConcurrencyStatus) are updated as toBeSynchronised $\longrightarrow$ synchronisationCompleted. Then, they are ready for merging. Thus, the pharmaceutical action <CIGCHFPH1> "Initiate diuretic" (clinicalID "722048006" (STC)) becomes a merging point where <CIGHTNPH2> pharmaceutical action of HTN is merged at this care point whose target activity is <CIGCHFPH1> through using assigned transitions (TransitionAssigned), whose label is isMergedAt (TransitionLabel). In this case study three different CIGs are executed in parallel. Two clinical actions are merged for the considered scenario but there is no limitation on the synchronisation and merging of more than two clinical activities which can be recommended by many different guidelines.

Care Optimisation

The above-mentioned patient scenario can also be as follows: CHF pharmaceutical action the <CIGCHFPH1> "Initiate diuretic" (clinicalID "722048006" (STC)) can be started whose associated pharmaceutical care element is "Diuretic" (clinicalID "30492008" (STC)) and its recommended use is three months, then HTN pharmaceutical action <CIGHTNPH2> "Add diuretic" (clinicalID "722048006" (STC)) can be recommended after one month. Since "Diuretic therapy" recommendation is a common clinical action for both of these guidelines, then this activity can be reused by HTN guideline to avoid care duplications because HTN and CHF are concurrently implemented. Please see the *Time-based optimisation algorithm* in Section 4.4 which addressed in the execution step #7.

To realise the reuse of the pharmaceutical care element, by the pharmaceutical clinical action of <CIGHTNPH2> "Add diuretic" whose activity lifecycle status is active, HTN must be linked with the instance of TimeBasedOptimisationConstraint using the *needsOptimisation* object property, see Figure 5.12a where HTN action is linked to a time-based care optimisation constraint (TimeBasedOptimisationConstraint) for activity reuse. Accordingly, the time-based optimisation constraint <PECAPP00 01TOC1> "Reuse pharmaceutical clinical action of CHF" is added to the patient's personal care plan, see Figure 5.12b. This constraint involves the connection between results and care elements to be reused, a time window of reusing them, and whether more action is needed at the end of the reuse time period of the clinical activity. The *reuseCareElementOf* object property links the instance <PECAPP0001TOC1> with the instance <CIGCHFPH1> "Initiate diuretic" of CHF.

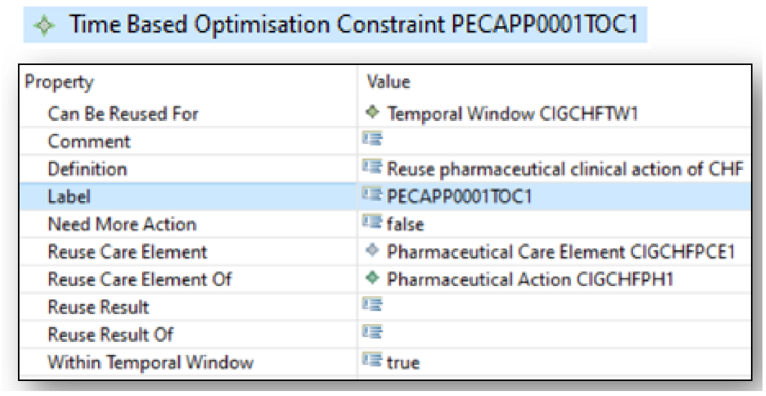
The *canBeReusedFor* object property links the instance <PECAPP0001TOC 1> with the instance of TemporalWindow, to define the acceptable reuse time period (e.g., min, max, exact value) of this care element. This is because this care element can be changed or if it is a result of a laboratory test, values may change in a short period of time, based on the patients' dynamic health conditions. The *needMoreAction* data property must be filled in with "true" value, if more action is needed such as more advice is needed, otherwise "false" should be used. If the care element reuse period is still valid, then the *withinTemporalWindow* data property



Pharmaceutical Action CIGHTNPH2

Label	CIGHTNPH2
Needs Modification	
Needs Optimisation	Time Based Optimisation Constraint PECAPP0001TOC1

(a) Excerpt of pharmaceutical action of Hypertension guideline to demonstrate care optimisation transition



Time Based Optimisation Constraint PECAPP0001TOC1

Property	Value
Can Be Reused For	Temporal Window CIGCHFTW1
Comment	
Definition	Reuse pharmaceutical clinical action of CHF
Label	PECAPP0001TOC1
Need More Action	false
Reuse Care Element	Pharmaceutical Care Element CIGCHFPCE1
Reuse Care Element Of	Pharmaceutical Action CIGCHFPH1
Reuse Result	
Reuse Result Of	
Within Temporal Window	true

(b) Time-based optimisation constraint for activity reuse

Figure 5.12: Illustration of activity reuse for care optimisation

needs to be filled with "true" data value, otherwise "false" should be used. In our example, this is stated as "true" since there are two more months left for the termination of <CIGCHFPH1> unless carer makes any changes on this action. The following component of a multiple guideline combination, which is care modification, is demonstrated in the following example.

Care Modification

Example 5.3.2. (Diabetes – Hypertension – Chronic Kidney Disease – Atrial Fibrillation) A multimorbid patient has Hypertension (HTN), and Atrial Fibrillation (AF) as long standing diseases, and Chronic Kidney Disease (CKD), and Diabetes (DB) as newly diagnosed diseases, please see Dumbreck et al. [242] for further information.

In this case study (Example 5.3.2, also see Table 5.2), the combined personal care plan of a multimorbid patient involves the DB, HTN, CKD and AF guidelines, implemented in parallel. However, some of the actions recommended by these guidelines are conflicting when they are combined. To illustrate, pharmaceutical actions of CKD combined with AF and DB cause adverse drug-drug interactions which must be resolved to maintain patient-safety (see the *Care modification algorithm* in Section

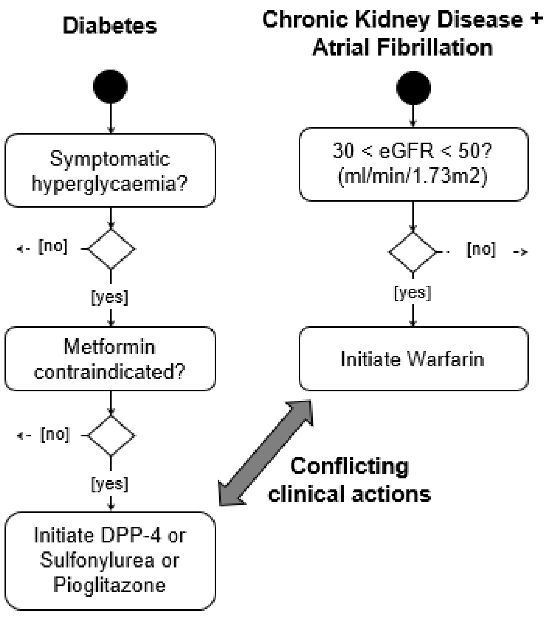
4.4, execution step #8).

Table 5.2: Case Study: Diabetes – Hypertension – Chronic Kidney Disease – Atrial Fibrillation

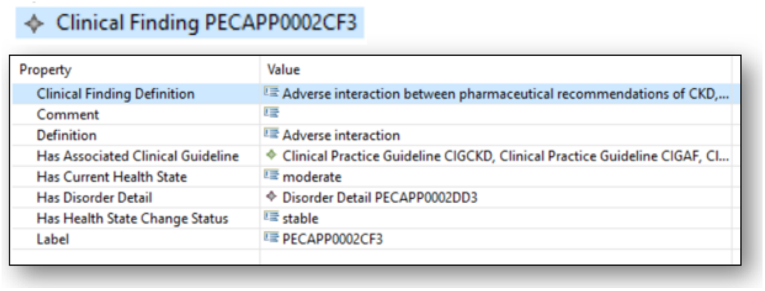
Patient scenario	HTN + AF: long lasting diseases; CKD + DB: newly diagnosed diseases | eGFR = "37ml/min/1.73m2", creatinine = "129mol/L"
CPG sources	CKD: Chronic kidney disease in adults: assessment and management – C3-cloud [67] (chapter 5.4, pp 48-74) and NICE [243]; DB: Type-II diabetes in adults: management – C3-cloud [67] (Chapter 5.2 , pp. 21–42) and NICE [228]; HTN: Hypertension in adults: diagnosis and management – NICE [241]; AF: Atrial fibrillation: management – NICE [244]
Care information	Ramipril, Spironolactone, Bisoprolol, Simvastatin, Amlodipine, Furosemide, Warfarin
New drug treatment	DB: Sulfonylurea
Conflict type	Drug-drug interaction
Conflict solution	DB medication Sulfonylurea is adversely interacted with Warfarin treatment in merged care. Warfarin medication replacement with a different anticoagulant
Clinical ID	Metformin product: 109081006 (STC); Metformin therapy: 1097191000000106 (STC); Sulfonylurea product: 34012005 (STC); Sulfonylurea administration: 432955007 (STC); Warfarin product: 48603004 (STC); Warfarin therapy: 722045009 (STC); Rivaroxaban product: 442539005 (STC); Apixaban product: 703906002 (STC)
Abbreviations	HTN: Hypertension; AF: Atrial Fibrillation; CKD: Chronic Kidney Disease; DB: Diabetes; eGFR: Estimated Glomerular Filtration Rate; STC: SNOMED-CT code; DPP-4: Dipeptidylpeptidase-4

In Figure 5.13a, the DB, CKD and AF guideline steps are shown to illustrate how conflicting care actions may occur in a patient care pathway. Figure 5.13b shows the associated clinical finding record.

The CKD decision (i.e. common action with AF guideline) <CIGCKDDC11> "30 < eGFR < 50 ml/min/1.73m2?", based on the patient value, is given. Accordingly, care continues with the pharmaceutical action <CIGCKDPH3> "Initiate Warfarin treatment" (clinicalID "722045009" (STC)) after the satisfaction of conditional option [30 < eGFR < 50 ml/min/1.73m2= "yes"]. The activity lifecycle status (ActivityLifecycleStatus) of this action is started. DB is the last diagnosed disease in this example. The DB decision <CIGDBDC8> "Systematic hyperglycaemia?" is "yes", based on the patient health status, and the patient has a contradiction on


Diabetes
Chronic Kidney Disease + Atrial Fibrillation
Symptomatic hyperglycaemia?
[no]
[yes]
Metformin contraindicated?
[no]
[yes]
Initiate DPP-4 or Sulfonylurea or Pioglitazone
30 < eGFR < 50? (ml/min/1.73m2)
[no]
[yes]
Initiate Warfarin
Conflicting clinical actions

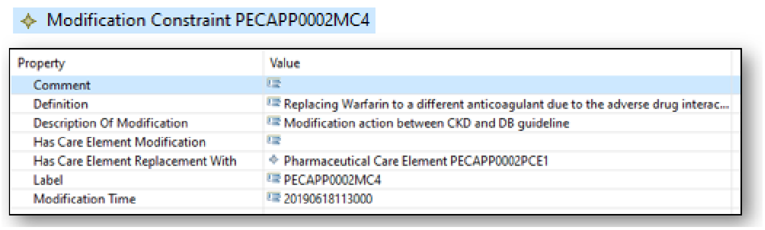
(a) Excerpt of Diabetes, Chronic Kidney Disease and Atrial Fibrillation guidelines to illustrate conflicting care actions



Clinical Finding PECAPP0002CF3

Property	Value
Clinical Finding Definition	Adverse interaction between pharmaceutical recommendations of CKD,...
Comment	
Definition	Adverse interaction
Has Associated Clinical Guideline	Clinical Practice Guideline CIGCKD, Clinical Practice Guideline CIGAF, Cl...
Has Current Health State	moderate
Has Disorder Detail	Disorder Detail PECAPP0002DD3
Has Health State Change Status	stable
Label	PECAPP0002CF3

(b) Example of clinical finding recording



Modification Constraint PECAPP0002MC4

Property	Value
Comment	
Definition	Replacing Warfarin to a different anticoagulant due to the adverse drug interac...
Description Of Modification	Modification action between CKD and DB guideline
Has Care Element Modification	
Has Care Element Replacement With	Pharmaceutical Care Element PECAPP0002PCE1
Label	PECAPP0002MC4
Modification Time	20190618113000

(c) Modification constraint for activity modifications

Figure 5.13: Illustration of care modification

their medication Metformin (clinicalID "109081006" (STC)), as well. Accordingly, the DB decision <CIGDBDC9> "Metformin contraindicated?" is given as "yes". In the following care step, the guideline recommends three alternative pharmaceutical therapy options involving the medications DDP-4, Pioglitazone or Sulfonylurea. The HCP selects the pharmaceutical therapy <CIGDBPH3> "Initiate Sulfonylurea treatment" (clinicalID 432955007 (STC)) whose activity lifecycle status is active.

Before starting a pharmaceutical action, the HCP must discover whether any adverse interaction may occur with such action. In this example, Sulfonylurea (clinicalID "34012005" (STC)) have the potential to cause conflicts (i.e. adverse drug-drug interaction) with Warfarin (clinicalID "48603004" (STC)), which may bring about changes due to anticoagulant and increased hypoglycaemic complications [242]. For this reason, this conflict needs to be resolved to maintain a safe care plan. Thus, the procedural action <PECAPP0002PA11> "Observation of pharmaceutical interactions" must be performed and the associated clinical finding (ClinicalFinding) details (<PECAPP0002CF3>), using the *hasClinicalFinding* object property, must be recorded. Here, the *hasDisorderDetail* object property links this concept with the DisorderDetail (<PECAPP0002DD3>) where disorder type (DisorderType) (e.g., disease, sign, symptom, adverseInteraction, otherDisorder, etc.) can be defined. The *isACauseOfDisorder* object property links this class with the AdverseInteraction, in order to choose the type of adverse interaction (AdverseInteractionType) such as drugDrug, drugDisease, drugPatient, drugFood, timing or otherInteraction (i.e. to capture other interaction types if exist).

In this case, the adverse interaction type is drugDrug. Once the interaction details are recorded, then the modification constraint (ModificationConstraint) is needed to be created and added to the personal care plan, see 5.13c. Accordingly, the modification constraint <PECAPP0002MC4> "Modification action between CKD and DB guidelines" is added to the personal plan. The pharmaceutical care element (PharmaceuticalCareElement) Warfarin (clinicalID "48603004" (STC)) of the pharmaceutical action <CIGCKDPH3> "Initiate Warfarin treatment" (clinicalID "722045009" (STC)) must be modified. Thus, the *needsModification* object property links the pharmaceutical action <CIGCKDPH3> with the modification constraint <PECAPP0002MC4> to perform required modifications. In this example, the care element modification is performed as medication replacement. Accordingly, the *descriptionOfModification* data property is filled with a description of this modification as "Replacing Warfarin to a different anticoagulant due to the adverse drug interaction with DB medication".

The *hasCareElementReplacement* object property links this class with the PharmaceuticalCareElement to define the medication details such as medication name, its associated clinical ID, application dose, method of administration (e.g., inhale, inject, etc.), active ingredient, possible side effects of the medication and the type of therapy (e.g., first line, second line, etc.). Due to the adverse interaction, Warfarin (clinicalID "48603004" (STC)) <CIGCKDPCE3> is replaced with its safe alternative option <PECAPP0002PCE1> Rivaroxaban (clinicalID "442539005" (STC))(or Apixaban (clinicalID "703906002" (STC)). Finally, the *modificationTime* data property of the action is recorded.

5.4 CIG Verification with User Support

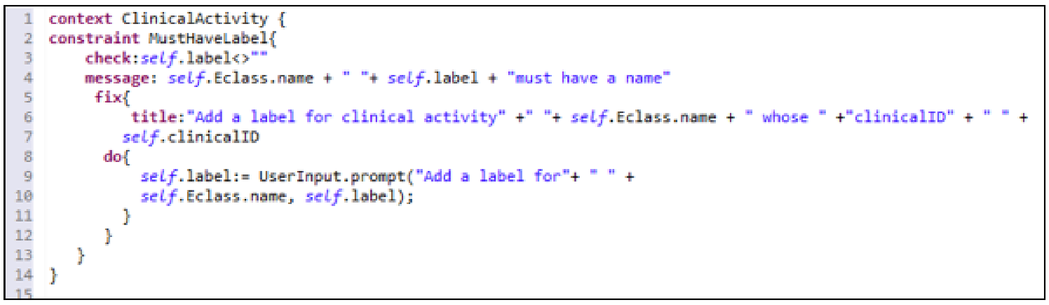
In this section, a set of consistency checking examples which are discussed under three main groups [50], as part of CIG verification is presented. Examples are shown on the depression in adults: recognition and management guideline [67].

The first group of consistency checking involves name and missing information checking while CIG models are created (pre-execution time); and when HCPs add new guideline activities or their interrelations in a patient's personal care plan (execution time). The CIG verification module detects these problems and supplies warnings to HCPs to solve these issues as well as a prompt box where HCP can directly perform his/her correction to proceed the execution. Then, CIG model and/or personal care plan can be automatically updated without the need of finding the related part where the error occurs. To illustrate name consistency and missing information checking, the following EVL codes and their related user messages are presented as follows:

Example 5.4.1. Each instance of a class in the model, must have a name (label). Figure 5.14 presents how missing labels of clinical activities can be detected and supplies user messages regarding how to resolve them.

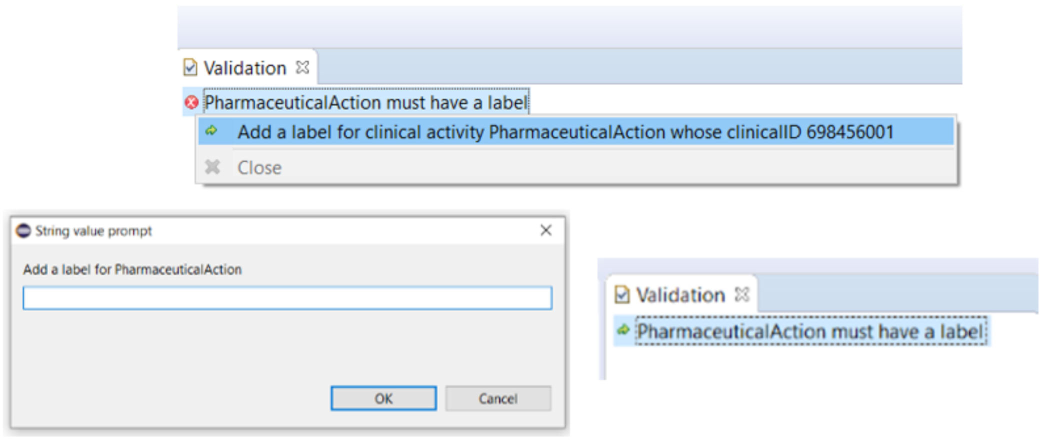
Example 5.4.2. Each label in the model should start with an upper case letter to maintain syntactic consistency. Figure 5.15 illustrates how this inconsistency can be detected, and related user messages can be generated for its resolution.

The second group of consistency checking is to maintain logical consistency. This involves checking transitions between activities and ensure that care flow follows a forward scheme (i.e. no cycles); checking whether there is any missing activities that must be in the personal care plan such as starting activity or conclusion; or checking whether all activities have a target activity (if it is not a final activity) or incoming activity (if it is not a starting activity); or knowledge elements must not exceed the predefined limits (e.g., minimum 2); or must not be missing (e.g., each activity associated with a concurrency constraint must have one activity concurrency status or each pharmaceutical action must have one duration and periodicity). To



```
context ClinicalActivity {
constraint MustHaveLabel{
    check:self.label<>""
    message: self.Eclass.name + " "+ self.label + "must have a name"
     fix{
         title:"Add a label for clinical activity" +" "+ self.Eclass.name + " whose " +"clinicalID" + " " +
        self.clinicalID
       do{
           self.label:= UserInput.prompt("Add a label for"+ " " +
           self.Eclass.name, self.label);
         }
       }
   }
}
```

(a)



Validation
PharmaceuticalAction must have a label
Add a label for clinical activity PharmaceuticalAction whose clinicalID 698456001
Close

String value prompt
Add a label for PharmaceuticalAction

Validation
PharmaceuticalAction must have a label

(b)

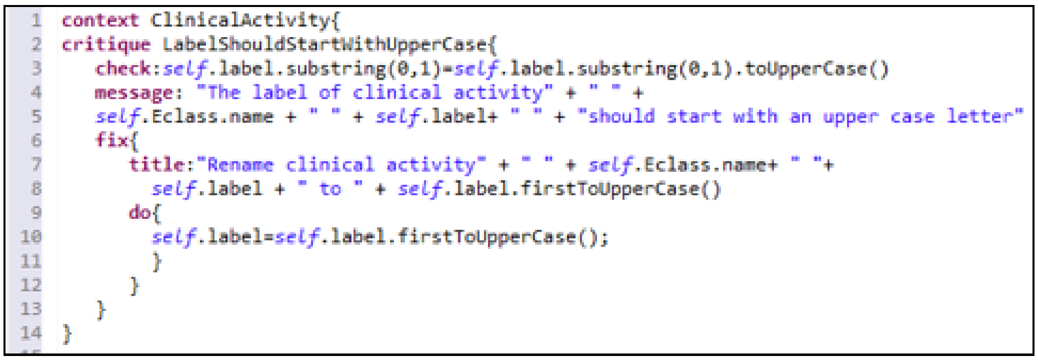
Figure 5.14: Examples of (missing) name checking: (a) code; and (b) user message

illustrate logical consistency checking, the following EVL codes and their related user messages are presented as follows:

Example 5.4.3. Checking whether the care workflow has a cycle. Since clinical activity has a forward flow, cycles are not allowed. Figure 5.16 presents code for how clinical activity transitions that lead to a cycle, can be detected and supplies associated user warning to resolve the problem.

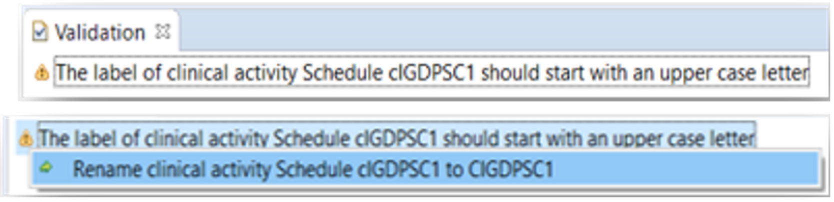
Example 5.4.4. Checking number of conditional options of a decision activity. Each decision activity must have minimum two conditional options. Figure 5.17 presents code regarding how the number of conditional options can be checked and supplies associated user warning to resolve the problem.

Temporal consistency is maintained in this work by involving each activities' start and end time which can be actual or expected (if exact time is not known, and therefore, it is assigned based on the duration of a clinical activity); as well as their



```
context ClinicalActivity{
critique LabelShouldStartWithUpperCase{
   check:self.label.substring(0,1)=self.label.substring(0,1).toUpperCase()
   message: "The label of clinical activity" + " " +
   self.Eclass.name + " " + self.label+ " " + "should start with an upper case letter"
   fix{
      title:"Rename clinical activity" + " " + self.Eclass.name+ " "+
        self.label + " to " + self.label.firstToUpperCase()
      do{
        self.label=self.label.firstToUpperCase();
        }
      }
   }
}
```

(a)



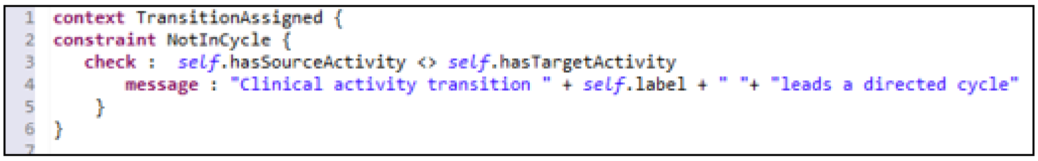
The label of clinical activity Schedule clGDPSC1 should start with an upper case letter

The label of clinical activity Schedule clGDPSC1 should start with an upper case letter

Rename clinical activity Schedule clGDPSC1 to ClGDPSC1

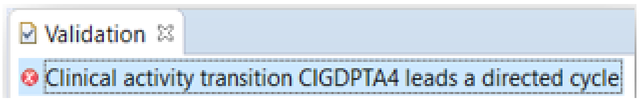
(b)

Figure 5.15: Example of name (typing) checking: (a) code; and (b) user message



```
context TransitionAssigned {
constraint NotInCycle {
   check :  self.hasSourceActivity <> self.hasTargetActivity
      message : "Clinical activity transition " + self.label + " "+ "leads a directed cycle"
   }
}
```

(a)

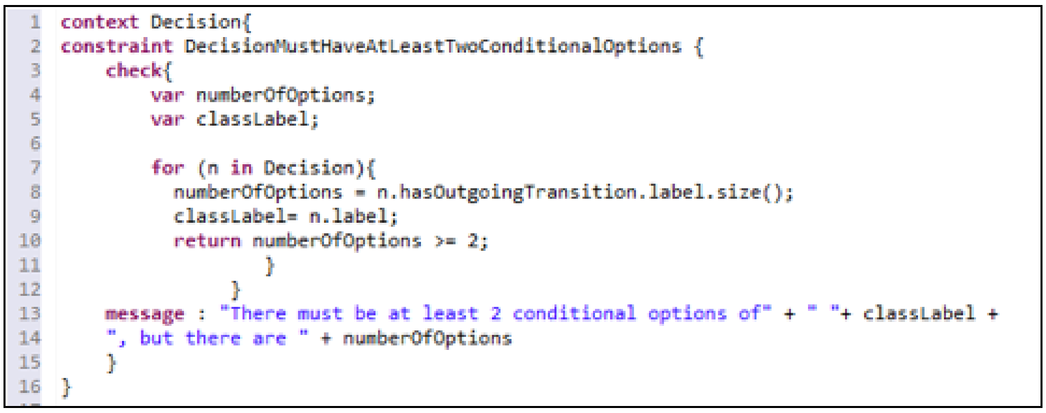


Clinical activity transition ClGDPTA4 leads a directed cycle

(b)

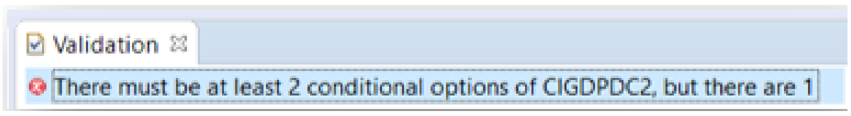
Figure 5.16: Example of checking whether the care workflow has a cycle: (a) code; and (b) user message

activity lifecycle statuses. If an activity has started, then their concurrent activities can be captured through the concurrency management algorithm (see Section 4.4). Thus, this maintains elimination of potential temporal conflicts which may arise between multiple activities. However, more comprehensive temporal checking approaches (e.g., [27, 121]) such as STP framework applications (see Section 2.4.3) and tools like SPIN model checker [161] can be used in MuCIGREF as a future work.



```
context Decision{
constraint DecisionMustHaveAtLeastTwoConditionalOptions {
    check{
        var numberOfOptions;
        var classLabel;

        for (n in Decision){
          numberOfOptions = n.hasOutgoingTransition.label.size();
          classLabel= n.label;
          return numberOfOptions >= 2;
                }
             }
    message : "There must be at least 2 conditional options of" + " "+ classLabel +
    ", but there are " + numberOfOptions
    }
}
```

(a)



```
There must be at least 2 conditional options of CIGDPDC2, but there are 1
```

(b)

Figure 5.17: Example of checking the number of conditional options of decision activities: (a) code; and (b) user message

5.5 Summary

This chapter presents a set of multimorbidity case studies and demonstrates their implementations on MuCIGREF by mainly focusing on combining multiple CIGs to create a personal care plan (step-by-step) for a multimorbid patient and then exemplifies how they can be handled. Combining multiple guidelines to handle multimorbidity care requires advanced management of various knowledge sources to maintain patient safety and consistent care recommendations. Thus, examples of these knowledge management approaches are addressed involving concurrency management; multi-activity merging; care optimisation; and modification.

To begin with, concurrency management is important to handle concurrently implemented clinical actions, especially, for drug combinations. This helps users to handle clinical action recommendations in terms of drug dose level adjustment or to avoid potential adverse drug interactions. MuCIGREF's concurrency constraint (ConcurrencyConstraint) supports users in concurrency management and therefore care planning. This enables users to capture concurrency relations between activity pairs and record them, which may affect the initiating new treatments without causing any drug efficacy reductions, drug overdoses or any patient-safety threatening actions.

Multi-activity merging is vital to eliminate care duplications. These duplications, which may end up unnecessary resource use such as HCPs time to assess the records or performing same lab test repeatedly, meaning additional cost to medical centre, or drug overdose, may lead a patient safety threatening result. However, multi-merging is the limitation of the existing literature [30, 57]. In MuCIGREF, merging is achieved by initially discovering similarities (commonalities) between clinical actions, and then performing synchronisation of multiple activities (TimeBasedSynchronisationConstraint). When these activities are successfully synchronised and there are no restricting conditions for them to follow the next care step, then they can be merged in the subsequent care step. Care modification enables users to have control over the care plan and helps users to resolve detected conflicts (e.g., drug-drug, drug-patient, drug-disease). Conflicts can be resolved by replacement of drug recommendations (ModificationConstraint). In this thesis, drug conflicts are not detected since this requires an additional tool (e.g., drugbank.ca [213] or drugs.com [214]) which is out of scope of this work.

Time-based care optimisation, which is required to optimise health resource use and avoid care duplications. Optimisation can be performed on clinical actions which are recommended by the guideline related with primary disease has been started and the guideline of secondary disease can reuse these actions. Thus, care duplications and unnecessary health resource use (e.g., clinical time) can be eliminated. However, this reuse can be possible if and only if required conditions are satisfied such as whether the clinical activity to be reused still within an appropriate time period (e.g., validity of a lab test result). Thus, the time-based optimisation constraint (TimeBasedOptimisationConstraint) supports users in their care optimisation actions. Following section presents verification and validation results of MuCIGREF.

Chapter 6

Evaluation

6.1 Introduction

The process of evaluation involves performing a set of verification analyses, which means checking whether a knowledge construct satisfies the given requirements to meet the intended objective of the system; and validation analysis, which means checking the applicability of the system for the intended objective [192].

In this section, MuCIGREF is evaluated based on its representation and execution capabilities of multiple concurrently implemented guidelines and their interrelations. The evaluation begins with ontologies of MuCIGREF's multiple CIG representation language, MuCRL, for representing guidelines and their interrelations and then its transformation into a meta model.

Following these, the section describes the evaluation of the execution algorithms of MuCIGREF's run-time execution engine, MuCEE, which allows the combination of multiple CIGs. Knowledge representation power and comprehensiveness of MuCRL and then the outputs of MuCEE are also evaluated, based on a set of CPGs with patient scenarios to represent real world cases.

6.2 Evaluation Method

The developed ontology is initially encoded using Protégé which supports OWL2 format, and then OWL2 ontology is transformed into EMF Ecore metamodel format in Eclipse. This mapping is performed to generate CIG models and perform dynamic execution over them using EOL [65] . However, an ontology must be consistent (i.e. free from contradictions), coherent and syntactically correct to be used for reasoning [245]. To do so, initially, ontology reasoners [92, 93] and tools are used to identify inconsistency, incoherency and syntactic incorrectness in the ontology.

Enhancing semantic interoperability [246] and achieving ontology sharing are also a focus in this thesis. For this reason, the above-mentioned two widely used knowledge modelling platforms are utilised for knowledge representation. Furthermore, an ontology must be complete (i.e. adequate coverage of the domain), clear (i.e. ontologically defined terms represent intended objective) and concise (i.e. free from irrelevant terms) [245] to achieve these targets.

To evaluate the ontologically represented knowledge, a set of evaluation methods has been proposed in the literature [245, 247] that are categorised as follows: (i) golden standard evaluation, which is used to compare the ontology with the gold standard one. Because there is no one gold standard ontology, the MuCRL ontology is compared with existing CIG formalisms to capture whether missing elements exists in knowledge representation of guidelines and their interrelations and/or to support their executions; (ii) application-based evaluation, where the ontology is used for specific applications and its results are evaluated (see ontology evaluation metrics [218, 219, 220, 221, 222]) for many perspectives such as expressivity, correctness, and completeness. A set of case studies are implemented for the evaluation of the ontology and compared to some of the existing works on the same multimorbidity care application; (iii) data-driven (also known as corpus-based) evaluation, which is used to evaluate the ontology by extracting information from the knowledge base. This evaluation is performed after transformation from OWL ontology to the EMF metamodel. Execution algorithms are used for this purpose. Thus, three knowledge representation evaluation approaches are adopted in this thesis.

In the literature, several CIG verification methods have been proposed, such as knowledge-based, theorem proving and model checking (see Section 2.4.3). In this thesis, a variant of knowledge-based CIG verification approach is adopted. This enables users to verify a guideline before real-time execution and if quality requirements are not satisfied, then they can be fixed. Users can also add new knowledge elements in the real-time execution phase, then tool can capture inconsistencies if this new knowledge causes. Developing automated verification techniques is still a challenging area of research for checking the quality of clinical guidelines [158]. For this reason, CIG verification module is developed based on EVL in Eclipse as a verification tool. This supports users to detect missing information and maintain semantic and syntactic consistencies, required for care recommendations. To solve these issues, users are supported with warnings and messages. Its application on a CIG model is shown in Section 5.4.

6.3 Evaluation of MuCIGREF's Representation Language

In this section, the evaluation of MuCIGREF's representation language, MuCRL, is performed. This involves the steps of ontologically represented knowledge evaluation, comparison with existing guideline representation languages and implementation of MuCRL on a set of guidelines.

6.3.1 Evaluation of the Ontology through Ontology Metrics

The ontology is evaluated based on a set of ontology evaluation metrics to discover whether MuCRL ontology achieves the required ontology correctness and quality standards and supply more understanding on the characteristics of the ontology to readers. Ontology can be evaluated in correctness (e.g., completeness, conciseness and consistency) and quality (e.g., computational efficiency (i.e. size), adaptability (i.e. cohesion), and clarity aspects [220]. Thus, the considered metrics [218, 219, 220, 221, 222] are consistency, coherency, comprehension, clarity, conciseness, completeness, structural complexity, cohesion, and conceptualisation.

Initially, ontology is evaluated based on both the consistency, and semantic correctness dimensions. After the ontology is encoded using OWL2 format in Protégé 5.2.0, FaCT++ [93] v.1.6.5 (i.e. tableaux-based reasoner), and HermiT [92] v1.3.8.413 (i.e. a hyper-tableau based reasoner) OWL2 reasoners are used to achieve domain and range checking, identify subsumption relations between classes, checking instances of classes, retrieving instances of classes and to detect and resolve ontology inconsistencies with OntoDebug plugin of Protégé. Thus, a logically coherent and consistent ontology is obtained. The Ontology Pitfall Scanner! (OOPS!)* (*http://oops.linkeddata.es/*) is also used to support identification of the common pitfalls, occurring while developing the ontology.

Subsequently, the semantic correctness of the ontology is assessed based on the following principles [248]: (i) each hierarchy possesses a single root; (ii) each class has minimum one parent; (iii) description of each child is different from its parent class; (iv) the siblings are different from each other; and (v) cycles are not allowed in a is-A hierarchy. As a result, the resulting ontology met the semantic correctness standard. Ontology is then checked whether free from unnecessary classes and duplications, which means there is no similar or same attributes or concepts. This met the conciseness criterion for the ontology.

The completeness metric measures whether the domain can be covered with the ontology entities (e.g., classes, properties) without the need of adding new entities. These are proved by implementing the ontology on a set of CPGs and multimorbidity case studies (see Section 5.3.5) which are discussed in Section 6.3.2. The MuCRL ontology complies with the intended meaning of defined terms and involves objective definitions for the classes and relations with rdfs:label and dc:description; this represents the clarity criterion of the ontology [205]. The ontology is also compared with the existing research and application work, in order to support the completeness dimension (see Section 6.5) as part of the validation process.

In Table 6.1, the size of the ontology is presented involving the number of classes, object and data properties and instances of the enumerated classes with the shares of its ontology groups. The proposed ontology involves are 68 classes, 70 object properties, 126 data properties, 1271 axioms and 115 defined instances (enumerations).

Table 6.1: Summary statistics of MuCRL ontology

	MuCRL (Total)	**Guideline Service Deployment**	**Health Care Service**	**Multi Activity Management**	**Patient Care Personalisation**
Classes	68	11.8%	72.1%	7.4%	8.8%
Object Properties	70	7.1%	71.4%	11.4%	10%
Data Properties	126	13.0%	56.9%	7.3%	22.8%
Instances*	115	35.7%	55.7%	–	8.7%
Size	379	18.6%	62%	6%	13.5%

**Enumerations*

There are 379 ontology entities are introduced in this thesis. HES constitutes 62% of these entities and has the highest share among other groups. However, its connections between other groups imperative to represent and manage guidelines. The numerical values for the following ontology evaluation metrics are presented in Table 6.2 to supply information about the ontology. For the interpretation for them, the following definitions are presented [218, 219, 220, 221, 222]:

- **Structural complexity:** This metric measures the complexity of the ontology. The size (i.e. total number of ontology entities) and relationships between ontology entities are components of this metric. The average number of semantic relation is higher than 1, which shows the connectedness degree of the ontology;
- **Cohesion:** This metric measures the relatedness degree of the ontology. Here, the number of leaf classes (i.e. taxonomy density metric representing classes which has no subclasses) and the average depth of inheritance tree (i.e. the ratio of total path depths (number of nodes) and sum of all paths) are calculated;

Table 6.2: MuCRL ontology evaluation metrics

Evaluation criteria	Metric	Value
Structural	Size of the ontology	379
Complexity	Average number of semantic relations per defined class	1.27
Abstraction	Average depth (path length) of the ontology	2
Cohesion	Number of root classes	4
	Number of leaf classes (external nodes)	55
	Average depth of inheritance tree of leaf nodes	1.24
Conceptualisation	Relationship Richness	0.51
	Attribute Richness	1.85
	Inheritance Richness	4.9

- **Conceptualisation:** This metric measures the semantic richness of the ontology. This involves:
  - **Relationship Richness:** This measures the connections (relations) between classes compared to all connections. The total number of semantic relations assigned to classes, divided by the sum of the number of subclasses and the number of semantic relations. If the ratio is close to one, this represents that most of the relations are between classes whose relations are different from class and subclass relations. In our case, the value of this metric is 0.51;
  - **Attribute Richness:** This measures the total number of attributes, divided by the sum of ontology classes. The higher values of this metric representing classes have high set of information compared to its lower values. The value of this metric is calculated as 1.85;
  - **Inheritance Richness:** This measures classes' average number of subclasses. The higher values of this metric represent the details of a class. The inheritance richness of the ontology is calculated as 4.9.

The afore-mentioned ontology evaluation metrics supply information about the ontology for readers. Thus, they can compare MuCRL ontology with other ontologies (see [249]) when they would like to reuse the ontology. Following this, the ontology is transformed into its metamodel version using EMF in Eclipse, and that might involve some differences in knowledge representations (e.g.,abstract class definitions). In existing literature, several tools (e.g., [232]) have proposed to support this translation.

6.3.2 Evaluation of MuCIGREF's Guideline Representation

To verify that the MuCIGREF's representation approach is adequate for representing and managing CPGs and their interrelations, six CPGs which have different scopes (e.g., diagnosis, treatment, management, etc.) and different numbers of clinical actions, based on NICE guidelines are instantiated. Among them, five CPGs are selected for implementations and compared with the C3-Cloud's CPG flowcharts [67]. These guidelines are selected to combine them in order to implement real-world multimorbid patient scenarios [68, 242] and implement existing works [36], for comparison of this thesis' approach to others.

In Table 6.3, patient scenarios, associated with guidelines representing their health conditions, are presented. Patient scenario implementations and information about guideline sources are previously discussed in Section 5.3.5, under each scenario. These implementations demonstrated that the MuCRL ontology can represent all the considered CPGs and their interrelations (e.g., medications, lab test results) without the need of adding new classes or relations. This result supports its adaptability (i.e. the ability of using the ontology in different domains, for example, in diagnosis or disease management, and its extendibility and completeness. Hitherto, this is derived from the current CPG implementations, and therefore more CPGs need to be used to further this result, in a future research.

Table 6.3: Patient scenarios used for testing

Clinical practice guideline	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Atrial fibrillation: management				✓
Chronic heart failure in adults: management			✓	
Chronic kidney disease in adults: assessment and management				✓
Depression in adults: recognition and management	✓			
Hypertension in adults: diagnosis and management		✓	✓	✓
Type-II diabetes in adults: management		✓	✓	✓

6.4 Evaluation of MuCIGREF's Execution Engine

This section discusses the evaluation of MuCIGREF's execution engine, MuCEE. The evaluation begins with interpreting the implementations of MuCEE's modules in multimorbid case studies and then compared with existing CIG execution engines and tools.

6.4.1 Evaluation of CIG Acquisition Module

MuCEE supports acquisition of multiple CIGs, which means transferring computerised guidelines based on patient health conditions to the personal care plan. This plan is the resultant care plan of a multimorbid patient, where all the updates are performed on this plan. CIGs can be acquired simultaneously or when the disease occurs. Before CIG acquisition, CIGs must involve all the required information and they must be consistent, as well. Pre-CIG execution verification needs to be performed before the CIG acquisition phase.

In this thesis, CIG acquisition is realised by the satisfaction of required executional constraints. CIG actions can only be acquired if activity lifecycle status of a clinical activity is active which denotes that a clinical activity is executable. If it is not active, then the activity lifecycle status must be passive. This means that the activity is not executable because it is absolute (not currently in use) or not able to be implemented with the available health care resources (e.g., CT scanner). The guideline acquisition approach is presented in Section 4.3.1 and its implementation in Section 5.3.2 and Section 5.3.5, where multiple computerised guidelines (e.g., diabetes, depression, chronic heart failure, and hypertension) are acquired (transferred) to the personal care plan.

Thus, the multiple CIG acquisition module successfully acquired all the guidelines associated with patients' health conditions for the considered multimorbid case studies. Following section presents the parallel CIG execution module evaluation results.

6.4.2 Evaluation of Parallel CIG Execution Module

In this section, the coverage of execution requirements (see Section 4.2) of MuCEE, is evaluated based on the implementation results of CIG executions on the multimorbidity case studies in Section 5.3. The application steps of real-time execution engine are as follows:

Step 1. MuCEE supplies MuCIGREF's CIG representation language, MuCRL, for CPG encoding and deals with different guidelines (e.g., diabetes, chronic heart failure, obesity, etc.) with different guidance, such as management, diagnosis, or risk assessment. Care initialisation begins with the creation of a personal care plan. In this plan, all the updates related with patient-HCP encounter and progress of clinical activities are recorded (see Section 5.3.1);

Step 2. After the creation of personal care plan and recording clinical findings, then the HCP can acquire CIG models (see Section 5.3.2) to transfer them to the personal care plan based on patient health conditions and satisfying required conditions (e.g., transferring active labelled clinical activities). MuCEE records each implementations of CIG activities at the knowledge base and supplies a history of patient care progress. MuCEE also recommends HCPs data entry (e.g., patient health status, life-style information) when needed, to follow the next care step based on the clinical options (as part of the verification process); and, additionally, supports them to query information from the knowledge base. Furthermore, MuCEE involves clinical IDs (e.g., SNOMED-CT) for data standardisation. Use of clinical IDs enhances semantic interoperability, since they eliminate impreciseness in the definition of clinical actions, parameters (e.g., good or bad result), and care elements (e.g., medication);

Step 3. HCPs can select the first clinical activity to initialise care based on the MuCEE recommendation. Clinical activities are sequenced (see Section 5.3.3) based on the workflow control patterns (see Section 3.5.5). However, their sequence can change based on the care progress. For instance, an HCP can ignore some care steps as they wish;

Step 4. Activity sequencing is performed step-by-step based on the satisfaction of required conditions (e.g., preconditions) and constraints (e.g., execution states). MuCEE executes care steps involving activity execution states such as started, cancelled, done, in correspondence with the care progress, and manages different condition types (see Section 3.5.3). In this thesis, four condition types are considered as precondition (e.g., lab test result must be ready before observation), activating condition (e.g., decision conditions), outcome condition (e.g., starting care based on the patient health condition such as signs of high blood pressure) and end condition (e.g., medication must be ended when a patients' health condition reached a certain level);

Step 5. The patient can have more diseases in time; therefore, more CIG models can be acquired and combined in the personal care plan, at any time point during care (see Section 5.3.5). These may need to be mapped to each other for handling multimorbidity related relations. For instance, multiple clinical actions of different CIGs may have concurrency relations. These actions need to be mapped to record their concurrency relations and eliminate care duplications;

Step 6. Multiple CIGs may need to be merged, if they satisfy required conditions (e.g., time-based synchronisation) and constraints at some point of the care cycle (e.g., common clinical actions such as on certain medication);

Step 7. HCPs can update, modify or ignore the CIG recommendations. Care optimisation can be performed to eliminate unnecessary care duplications by supporting reusing of care activities;

Step 8. HCPs can modify care actions to improve patient safety by replacing a medication with its safe alternative.

Each step maps one or multiple CIG execution requirements, as presented in Section 4.2, which are satisfied by MuCEE. Among them, steps 3 – 8 address the parallel CIG execution module. This thesis concluded that MuCEE has adequate execution capabilities in managing multiple concurrently implemented CIGs.

6.4.3 Evaluation of CIG Verification Module

CIGs must involve required properties (e.g., medical practice, guideline goal, patient-specific clinical condition, time, patient groups) to satisfy the intended objective (e.g., diagnosis, management) and supply consistent and error-free recommendations when they are executed [10, 152, 153].

CIG models (pre-execution) and personal care plan (real-time execution) need to be verified to eliminate possible model errors, and missing information (e.g., label, relation) which are necessary to supply care recommendations. For this purpose, the EMF model validation tool [250] is initially used for automated checking models of individual CIGs and personal care plans (where multiple CIGs are combined). This produces warnings for incorrectness and maintaining each instance of the class has the required values (i.e. ensures that there is no missing value), and relations (e.g., *hasExamination*) based on the cardinality restrictions (e.g., min 1).

Secondly, EVL [102] is used for the development of verification constraints to detect inconsistencies in models and help users how to repair them involving the importance of the error (e.g., critique is used to designating not crucial errors). CIG verification module of MuCEE can be used, for example, preserving directedness between connections of nodes (classes); ensuring required workflow constructs are correctly defined and checking whether any unvisited node is remained; ensuring semantic consistency (e.g., labels of model entities should start with upper case letters); ensuring there is no missing information (e.g., all the clinical activities must have source and target activities); and ensuring knowledge consistency such as each guideline implementation must have one starting activity, each guideline implementation must have minimum one conclusion, each decision activity must have minimum two conditional options, a pharmaceutical action must have one pharmaceutical care element, an examination action must have one examination; or a procedural action must have one procedure and many others.

User messages are also developed to support users in fixing the errors with user prompt messages and enable users flexibility in developing task-specific verification constraints, which are the major motivation of using EVL rather than using the built-in model verification tool (e.g., SPIN [161]) but they can be used for comparison with EVL in future. In Section 5.4, real-time CIG verification implementation on a depression guideline [67] is presented involving some example verification codes and related user messages, showing the inconsistencies and how to resolve them.

6.5 Comparison with Existing CIG Languages and Execution Engines

In this section, MuCIGREF is compared with a set of well-known CIG languages; and their CIG execution engines which are required to represent and simulate CPGs with the patient data to generate care recommendations for patients.

Initially, existing well-known CIG Languages which involve Asbru, EON, GLIF, and PRO*forma*, are compared with MuCRL, in supporting control-flow patterns [30, 57]. In Section 3.5.5, the MuCRL's supported control flow patterns are presented. These are important to control knowledge elements required to create a care pathway for a patient. In Table 6.4, the comparison of Asbru, EON, GLIF and PRO*forma* formalisms, in supporting control-flow patterns, as performed by Mulyar et al. [30], with MuCRL is presented. These patterns are categorised based on their roles in workflow systems as follows: (i) basic control flow patterns; (ii) advanced branching and synchronisation; (iii) state-based patterns; (iv) cancellation and force completion patterns; (v) structural patterns which involves termination and iteration patterns; and (vi) trigger patterns. These patterns are initially introduced in Russell et al. [57] and in the official web site *workflowpatterns.com* [215] where each pattern involves a pattern number, related definition. They are used in pattern-based analysis of CIGs in existing works (see [30, 128]). Section 2.4.2 presents the general descriptions on main elements of workflow patterns but for further information please see [57, 215]. MuCRL is included in this comparison table presented by Mulyar et al. [30]. In addition, the support of some PRO*forma*'s control-flow patterns (e.g., simple merge and arbitrary cycle) is updated on this table based on the work of Grando et al. [128].

Table 6.4: Comparison of supported control-flow patterns in existing CIG Languages [30] with MuCRL

Workflow pattern	**Asbru**	**EON**	**GLIF**	**PRO*forma***	**MuCRL**
Basic control-flow					
Sequence	+	+	+	+	+
Parallel split	+	+	+	+	+
Exclusive choice	+	+	+	+	+
Simple merge	+	+	+	-	+
Advanced branching & synchronisation					
Multichoice	+	+	+	+	+
Multi-merge	-	-	-	-	+
Local synchronising merge	-	-	-	+	+
General synchronising merge	-	-	-	-	+
Structured partial (n-out-of m) join	+	-	+	+	+
Structured discriminator (1-out-of- m join)	+	+	+	+	+
State-based patterns					
Deferred choice	+	-	+	+	+
Interleaved (parallel) routing	+	-	-	-	+
Milestone	-	-	-	+	+
Critical section	+	-	+	-	+
Cancellation & force completions					
Cancel activity	+	+	+	+	+
Cancel case	+	-	+/-	+	+
Cancel multiple instance activity	+	-	+	+	+
Complete multiple instance activity	+	-	-	+	+
Cancelling n-out-of-m join	-	-	-	+	+
Structural patterns					
Arbitrary cycles	-	+	+	+	-
Termination	+	+	+	+	+
Structured loop	+	+	+	+	-
Recursion	+	-	-	-	-
Trigger					
Transient Trigger	-	-	-	+	+
Persistent Trigger	-	-	+	+	+

+: *supported* ; -: *not supported*; +/-: *partially supported*

The major contribution of this thesis is to manage concurrently implemented multiple CIGs and their actions which are the limitation of existing works, see Section 2.4.1 and Section 2.6.1. Among control-flow patterns, representing advanced branching and synchronisation patterns support this goal. These involve multi-merge, general synchronising merge, and local synchronising merge patterns which are the common patterns that are not involved in existing formalisms [30, 128] are covered in this work (see Section 3.5.5) with the support of multi-activity management ontology group.

Because of the design principles, arbitrary cycles are not supported in this work, because cycles are not allowed in our system. Structured loop is also not supported in MuCRL. Clinical activities are performed always in forward way. Recursion is also not supported because any action cannot invoke itself. User must start the recommended activity from the system. Consequently, the coverage of these patterns affects the execution capabilities of run-time execution engines. Based on this table, MuCRL has the highest workflow control patterns coverage. Then, PRO*forma*, Asbru, GLIF and EON follows this, respectively.

Secondly, existing real-time CIG execution engines are compared. This comparison was adopted from Isern and Moreno [19] and involved MuCEE in Table 6.5.

Table 6.5: List of CIG languages and their execution engines

CIG Language	**CIG Elements**	**CIG Execution**	**Tool**
PRO*forma*	Action, decision, enquiry, plan	Rule-based	Arezzo
		Event-based	HeCaSe2
Asbru	Time-oriented skeletal plans, plan, precondition, plan type	Event-based	DeGeL
GLIF3	Action, decision, branch synchronisation, patient state	Event-based & rule-based	GLEE
Graph-like	Query, work, decision, conclusion	Rule-based	META-GLARE
GUIDE	PetriNet	Rule-based	NewGuide
EON	Context, decision, plan, enquiry	Event-based	SAGE
SAGE Guideline model	Action, decision, action, branch synchronisation, scenario	Event-based	SAGE
Arden Syntax	Medical logic modules	Rule-based	Arden Syntax IDE
MuCRL	Query, activity, conditions, activity management constraints	Event-based & rule-based	MuCEE

While rule-based approaches [251] (e.g., query or encoded knowledge) have been used for continuous systems, event-based approaches have been used to manage nonsynchronous actions which requires a triggering action to start a CIG activity (e.g., clinical events such as waiting result of lab tests or HCP requests; or system events such as when a care action is completed, then the next action can be followed if and only if there is no limiting condition over this action) [15, 19]. Accordingly, execution engines that have rule-based execution mechanism involves Arezzo, META-GLARE, NewGuide and Arden Syntax; event-based execution mechanisms involve HeCaSe2 and SAGE; and lastly, hybrid (rule + event) execution mechanisms involve GLEE and MuCEE.

Existing CIG execution engines and tools meet common execution requirements addressed in Section 4.2. Nevertheless, the major limitations of the existing works are mainly about combined management of multiple CIGs, which involves merging multiple CIGs and their concurrently implemented multiple clinical actions; managing care inefficiencies (e.g., care duplications) affecting care; and then generating automatic personalised care plans involving patient context. In Section 2.6, the limitations of existing works, se per the complex requirements of multimorbidity care management, are extensively discussed.

In the light of the foregoing, this thesis contributes on managing multimorbidity care through combining multiple concurrently implemented CIGs while handing the afore mentioned complexities. Thus, a new knowledge representation approach, MuCRL, is developed to support multi-activity management and then mapped to modelling environment where execution is performed over models. Table 6.6 presents multi-activity management constraints, with their functionalities used in managing multimorbidity care. Each MAN constraint is associated with a functionality type as dynamic or flexible. Dynamic here refers to a functionality that enables users to add, remove or modify actions during the solution process (e.g., care recommendation). Flexible refers to a functionality that enables users to add user preferences on choosing alternative care options.

Table 6.6: Multi-activity management constraints supporting combination of multiple CIGs

	Constraints			
	Concurrency Constraint	**Time-based Synchronisation Constraint**	**Modification Constraint**	**Time-based Optimisation Constraint**
Functionalities				
Eliminating care duplications	✓			✓
Merging multiple clinical actions		✓		
Enabling care modifications and updates			✓	
Handling of synchronisation relations		✓		
Constraint types				
	Dynamic	Dynamic	Dynamic, Flexible	Dynamic, Flexible

✓: *supported*

The concurrency constraint can be used to eliminate duplications, since they record concurrency of activity pairs until the end of their concurrency period. Likewise, time-based optimisation constraints can be used to reuse completed clinical activities, if they are within an acceptable time period. This will eliminate care duplications and therefore resolve potential conflicts. For instance, drug overdoses, due to the duplicated same medication recommendations, can be eliminated.

The time-based optimisation constraint is defined as a dynamic constraint when care reuse must be performed, and a flexible constraint that an optimisation action can be performed onto, for which its violation does not affect the patient safety.

The time-based synchronisation is a key constraint that must be added to the personal care plan when multiple activities are merged in their common (similar) subsequent clinical activities. Synchronisation relations of multiple clinical activities can also be handled with this constraint. Merging of these activities are not forced after their synchronisation. Synchronisation of clinical activities are performed in parallel. Care modifications are required, when conflicting clinical actions occur or patient allergy or personal context (e.g., preferences) need to be considered.

The modification constraint is defined as both a dynamic and a flexible constraint. It can be considered as a dynamic constraint when the care modifications, such as drug replacement with its safe alternative or adjusting timing of a clinical action must be performed to maintain patient safety. It can also be considered as a flexible constraint when HCPs (user) would like to modify care steps or care options, based on their or patient preferences. Thus, this constraint supplies flexibility to users.

6.6 Summary

MuCIGREF was evaluated for knowledge representation of CPGs and their interrelations and its execution engine capabilities. The evaluation order of MuCIGREF is as follows. Initially, the evaluation of MuCRL- MuCIGREF's CIG Representation Language was performed. This involved evaluation of MuCRL ontology using a set of ontology evaluation metrics; comparison of MuCRL with existing CIG languages; MuCRL's transformed metamodel version is evaluated by its instantiating this metamodel on different patient scenarios using different CPGs to demonstrate the coverage of the knowledge representation approach in different domains (e.g., diabetes, chronic heart failure). Afterwards, the evaluation of MuCEE- MuCIGREF's Execution Engine was performed. This involved evaluation of pre and post CIG execution; comparison of MuCEE with existing CIG execution engines; and evaluation of MuCEE in multimorbidity case studies mainly focusing on multi-activity management which is the major focus of this thesis.

- **Evaluation of MuCRL**

MuCRL ontology was developed using OWL2 in Protégé. For the evaluation of this, a set of ontology reasoners and tools were used to maintain consistencies and semantic coherency of the ontology. A set of ontology metrics was used to show that the ontology meets required quality standards and supply intuitiveness on the ontology aspects. Afterwards, ontology encoded in OWL2 is transformed to EMF in Eclipse. These systems have different strengths in knowledge representation, execution and evaluation.

Because MuCRL involves new concepts and properties for managing multimorbidity care, there is no gold standard ontology for the comparison. Yet, it is compared with existing CIG languages to cover the required control flow constructs. The results demonstrated that MuCRL covers workflow patterns required for managing multiple concurrently implemented CIGs. The major contribution of MuCRL is on representing advanced branching and synchronisation relations of clinical actions and their associations (e.g., care elements).

To evaluate MuCRL's representation power, a set of CPGs were represented with this formalism. As a result, MuCRL successfully represented all the considered guideline knowledge constructs and their interrelations. This was deduced from the use of CPG flowcharts of C3-cloud [67] where a group of HCPs developed the CPG flowcharts. Thus, this thesis ensured that all the guidelines were successfully covered.

- **Evaluation of MuCEE**

MuCEE involves three modules as multiple CIG acquisition module, CIG execution module and CIG verification module.

To test the CIG execution engine and acquisition module, clinical pathways of multimorbid patients were computerised based on the flowcharts of C3-Cloud [67] and NICE, and patient scenarios were created based on the suggestions of the existing literature [28, 36, 114, 242] and C3-Cloud [68] as well. CIG models were successfully acquired in accordance with the patient scenarios. In Section 6.4.2, how MuCEE satisfies CIG execution requirements to achieve run-time multimorbidity care management based on the algorithms proposed in Section 4.4 was discussed.

Based on the review of the existing execution approaches (see Section 2.4.1), multimorbidity care management approaches and their limitations (see Section 2.6) and the comparison of MuCEE with well-known execution engines (see Section 6.5), this thesis concluded that MuCEE fills the gap of execution requirements in combining multiple concurrently implemented CIGs. Because the focus of this thesis is multimorbidity care management, multi-activity management approaches are the contributions of this thesis to the literature. To prove this, Section 5.3 exemplifies multimorbidity management using MuCEE. The following multi-activity management approaches that were successfully implemented in Section 5.3.5 with the given guideline combinations which are as follows: (i) concurrency management; (ii) multi-activity merging ; and (iii) time-based care optimisation with the Diabetes – Hypertension – Chronic Heart Failure guideline combinations; and (iv) care modification with the Diabetes – Hypertension – Chronic Kidney Disease – Atrial Fibrillation guideline combinations. The associated execution algorithms are presented in Section 4.4.

The evaluation of CIG verification module involves checking the implementations above-mentioned guideline implementations before execution (installation level) and execution time to ensure that naming and labelling conventions [253] are preserved, there is no missing information, semantic incorrectness, cycles and many others (see Section 6.4.3) in a care plan. Verification before CIG execution is required for CIG acquisition phase to transfer guideline instantiations and their associations to patients' personal care plans. Execution time CIG verification is required to ensure that if any instances are added to the personal care plan may not cause any inconsistencies (e.g., missing information, missing connections, directedness violations) affecting a care flow. A set of EVL constraints were developed to detect these inconsistencies and generate user messages regarding how to fix them interactively.

Chapter 7

Conclusions and Future Works

7.1 Conclusions

Clinical practice guidelines are evidence-based knowledge sources that supply care recommendations to healthcare professionals in managing mainly a specific health condition. Traditional CPGs face difficulties in presenting a detailed consideration of strategies and recommendations to coordinate conditions of a multimorbid patient since they mainly single disease centred. The major challenges of using guidelines are their inability to adapt to dynamic changes in patient health conditions as well as their manual management of their recommendations, which may adversely interact with each other when multiple of them concurrently implemented.

Multimorbidity management is an increasingly relevant topic of research in the health informatics community, due to its care challenges and concerns of providing personalised therapies [26, 51, 136, 177, 203, 254, 255] for each patient, which is the main aim of patient-centred care [6, 256]. For personalisation, diverse information related to all a patient's health conditions, clinical history, health records, as well as personal context need to be consolidated. Evaluation and amendment of many co-existing care plans, as well as coping with possible adverse interactions, make multimorbidity care much more challenging. HCPs struggle with supplying care to patients under such complexities, without causing any treatment conflicts or making any inconsistent and/or unnecessary recommendations. When these complexities are poorly managed, they may negatively affect the duration of care and the healing process of a patient, which may result in several undesired outcomes. Moreover, eliminating care interruptions of patients after their encounters with their HCPs and sustaining their adherence to the agreed care plan are also crucial for their outcome.

This thesis initially introduced a multiple CIG representation language by developing a generic ontology as a guideline formalisation method, which represents knowledge constructs of CPGs and their interrelations and to establish multimorbidity relations between multiple of them to handle their concurrent implementations. Afterwards, ontologically represented knowledge is transformed to the EMF Ecore metamodel for generating CIG models to achieve dynamic knowledge execution. Afterwards, a real-time multiple CIG execution engine is introduced to combine multiple concurrently implemented CIGs to generate personal care plans for multimorbid patients. The major challenge towards combining multiple concurrently implemented CIGs is merging of multiple clinical actions, which need advanced activity synchronisation and concurrency management and conciliation of different knowledge sources (e.g., medication, duration of care, patient characteristics such as allergy and health state), such that the interaction of them should not cause any conflicts which may threat the patient safety or care redundancies which may cause unnecessary health resource uses.

As a result, MuCIGREF can be used to support HCPs in a CDSS setting to generate unified and personalised care recommendations for multimorbid patients by avoiding care duplications, modification of care actions to supply safe care, and supplying time-driven optimised health resource management by enabling reuse of clinical elements if they are within the acceptable reuse period. This may improve operational and cost efficiency in multimorbidity care management.

Following sections present discussion of thesis results, thesis contributions, limitations and future works.

7.2 Meeting the Research Objectives

This section discusses the results related to the development of a new CIG language, MuCRL, and its execution engine, MuCEE, by addressing the research challenges and objectives raised in the introduction, respectively.

7.2.1 Discussion of Results Related to CIG Language Development

MuCRL is designed to deal with the challenges of mainly single-disease centredness of currently available CPGs that supply general care statements to manage a patient's health condition(s) without considering his/her multimorbid conditions.

To date, guidelines have been represented using different CIG languages which have different modelling approaches to develop CIGs. The common features of these languages are to represent knowledge constructs of guidelines (e.g., activities, decisions, etc.), their associated conditions (e.g., precondition) and constraints (e.g., task duration, etc.), organisational workflow control patterns required to manage clinical tasks (e.g., sequencing, synchronising, branching, etc.), and executional semantics required to control and store the states of clinical tasks. However, existing works mainly struggle with combining multiple concurrently implemented CIGs and generate unified care recommendations, which must be safe and personalised. These limitations are found out by performing a gap analysis through conducting a comprehensive systematic literature review [31]. Thus, MuCRL aimed to fill the gaps of the existing literature and it can be used as a reference CIG representation language.

MuCRL can be used to represent knowledge constructs of CPGs and their interrelations and to create required mappings between multiple CPGs to meet patients' multimorbid health needs. To achieve this, a generic ontology is developed based on the synthesis of well-known ontology building lifecycle methodologies. Subsequently, the ontologically represented knowledge is conceptualised under four groups, as follows: (i) Guideline Service Deployment, represents guideline information (e.g., author, guideline name, etc.) and information about the HCP who performs the care; (ii) Health Care Service, represents required clinical elements to create a care pathway (e.g., temporal constraints, clinical actions, condition parameters, etc.) and executional information; (iii) Multi-activity Management, represents required multi-activity management constraints required for the management of concurrent presentation of multiple CIGs; and lastly, (iv) Patient Care Personalisation, represents patient-specific information (e.g., name, weight, life-style information, admission date, etc.). These groups are linked with each other (e.g., task-network [9]), to develop a combined care pathway for a multimorbid patient. MuCRL partially reuses some knowledge entities from existing clinical terminologies and codes (for standardisation and enhancing semantic interoperability), ontology portals and existing CIG languages which are addressed in Section 3.4 since these elements are common elements for almost all CIG languages.

The novelty of MuCRL is to present a set of new classes and properties that are required to represent elements of concurrently implemented multiple CIGs and manage their multiple activities in a CDSS setting. To illustrate, during the care, new disorders may occur due to allergic conditions, and/or other disorders or existing care may need to be updated. Overlapping activities may occur and cause drug overdose, which may have life threatening impact upon the patient or cause extra health resource use (e.g., physicians' time, lab tests, etc.) that means costs for the health organisation. When many guidelines, and their interrelations, are considered, delays in care flow may occur as well. MuCRL supplies required semantics for enabling dynamic and flexible management of multiple CIGs through the use of multi-activity management constraints. These constraints are designed for performing activity concurrency management, time-based synchronisation, time-based optimisation and modification. Thus, they support management of multiple care recommendations (e.g., concurrently implemented or with delays) which needs to be merged after their synchronisations to eliminate care duplications, and handling conflicting care goals with modification.

MuCRL substantially supports the workflow control patterns presented in [9, 30, 57] (see Section 3.5.4 for how it supports these patterns and see Section 6.5 for comparison with existing CIG languages). MuCRL mainly contributes on advanced synchronisation and branching patterns (e.g., multi-merging organisational workflow control pattern, which is required to manage concurrent implementation of multiple CIGs). A wide range of temporal constraints as duration, periodicity, temporal distances, limits (e.g., deadlines), windows (i.e. allowed time periods to maintain patient safety) and temporal uncertainties (e.g., expected start time) to manage CIG actions are considered. Allen's temporal constraints [227] and like GLARE, advanced temporal information by supplying required semantics to handle temporally exact, imprecise and unknown information in implementing care activities are considered. Moreover, while in existing works decisions can have two options [36, 188, 189], in MuCRL, decisions (Decision) have XOR split property and can have minimum two conditional options.

Lastly, MuCRL is tested based on a set of evaluation criteria which are presented in Section 6.3.2. These involve using ontology reasoners and ontology evaluation metrics, several CPG implementations and comparisons with existing CIG languages. This thesis concluded that MuCRL supplies consistent, and coherent ontology. Ontology metrics are also presented for readers to supply more insights to the ontology if they would like to reuse it. MuCRL successfully represented knowledge elements of selected six CPGs with different clinical guidance (e.g., management, diagnosis, prevention, etc.) and their interrelations without the need of creating new classes and/or their properties.

MuCRL is initially implemented using the Protégé ontology development environment which supports OWL2 [63, 89] format from the Semantic Web. To increase flexibility in CIG management and verification, MuCRL is also implemented in Eclipse environment. Thus, OWL ontology is transformed to EMF metamodels [64] to generate CIG models and codes required for their management. Thus, the same knowledge can be easily structured under two environments that have different strengths and capabilities, such as knowledge sharing and execution. OWL2 file of the work is also shared which is a contribution to the literature.

7.2.2 Discussion of Results Related to Real-time CIG Execution Engine

MuCEE is designed to deal with the complexity in handling and merging actions of multiple, concurrently implemented, single-disease CIGs and their interrelations, in order to generate personalised care plans for a multimorbid patient. Initially, a systematic literature review on existing CIG execution engines is performed. These involve discovering of CIG execution requirements and analysis of execution capabilities of existing works in managing multimorbidity care (see Section 4.2). Accordingly, this thesis aims to fill the gap of the existing literature, which is to combine multiple (more than two) concurrently implemented CIGs and generate a unified personal care plan for a multimorbid patient. The issue of combining multiple CIGs and handling associated complexities are defined in this thesis as a Multiple CIG Combination Problem (MCCP) and MuCEE supplies solution for this problem (see Section 4.3).

MuCEE involves three modules with different specific objectives, which are as follows. Firstly, the multiple *CIG Acquisition module*, enables users to acquire multiple CIGs based on patients' health conditions. This can be achieved by the satisfaction of a designated model-level state conditions (e.g., activity lifecycle status) to transfer CIG models to the personal care plan. This also supplies CIG version control [40, 41] since users can only transfer up-to-date clinical activities.

Secondly, the *parallel CIG execution module*, enables users to execute multiple CIG models in parallel under the personal care plan, where care personalisation is iteratively realised based on the completion of care steps and satisfaction of the related conditions and constraints. To achieve this, a specialised MCCP solving algorithm is developed. This can adopt to dynamic changes occurred in the CIG actions and their associated interrelations. Existing literature [36, 188, 238, 252] especially designed for detecting inconsistencies (contradictory treatments) when reconciling multiple CPGs through the analysis of a collection of constraints associated with a set of knowledge interactions (e.g., drug-drug, drug-disease, etc.). The major limitations of existing works include the lack of computerised adaptation of patient context into their model and there is no evidence on how to handle more than two concurrently implemented clinical activities of multiple CIGs at a time [9, 30]. In this thesis, we generate a unified personal care plan for each patient where all diseases of the patient are reconciled. Under this plan, we are able to perform (multi) merging of clinical actions of multiple CIGs with a specialised MCCP solving approach that has several sub-algorithms that allow handling different types of complexity (e.g., concurrency and synchronisation relations and care duplications) in managing the multimorbidity care.

Lastly, the *CIG verification module* enables users on how to detect inconsistencies, missing values and errors in pre- and real-time CIG execution phases for the verification purpose. Accordingly, a novel knowledge-based verification method based on a variant of Object Constraint Language [66] is proposed. In the existing literature, many different CIG verification approaches (see Section 2.4.3) have been proposed. These are mainly on temporal verification [27, 50, 168], however, there is a gap in the literature in verification tools which support dynamic execution of CIG languages [158] and their combined CIG implementations. This thesis supplies a new verification approach by applying verification on pre- and real-time CIG execution (where verification is performed for all executed CIGs under the personal care plan) involving customised user messages and interactive update mechanism on the detected inconsistencies.

To evaluate MuCEE, a set of verification and validation analyses is performed. Initially, CIG acquisition module is evaluated by acquiring (transferring) these models from the knowledge base to the personal care plan ensuring that the model requirements are met. CIG acquisitions are successfully performed. Subsequently, parallel CIG execution module is evaluated based on a set of real-life multimorbidity case studies involving different guideline combinations representing different patient scenarios. Here, the main focus is on handling challenges such as care duplications and conflicts while combining multiple CIGs. Thus, several examples are given which address concurrency management, multi-activity merging, care modification and optimisation aspects of multi-activity management required for handling multimorbidity care. As a result, this module successfully handles these aspects in line with the algorithms provided in Section 4.4. Lastly, the CIG verification module is used to verify pre-execution of six CIG models where each of them represent a specific disease and their execution-time verifications under the personal care plan. This module substantially facilitates CIG verification by enabling users dynamic missing information and inconsistency detection and supporting them with customised error messages which supply information regarding the detected inconsistency and how it can be solved.

MuCEE is also compared with existing CIG execution engines and tools. As a consequence, MuCEE substantially satisfies the CIG execution requirements and its comprehensive execution mechanism which involves three modules that designed for supporting the execution of multiple CIGs. The majority of the CIG execution mechanisms bind to the Protégé environment, yet MuCEE does not bind to this environment and supply more flexibility in handling multiple CIG management. For the CIG model acquisition and parallel CIG execution modules, EOL is used, and for the CIG model checking module EVL is used. These languages are direct communications between each other and CIG models. As a result, the limitations of existing literature (see Section 2.6) in combining multiple CIGs are covered such as lack of functionalities for achieving or handling multi-merging of clinical activities [128, 181, 193], temporal constraints [181, 187], concurrency management [28, 203], complex decisions which have more than two options [36, 188, 189, 191], and supplying computerised patient-specific intervention plans [177]. In the following section, specific contributions of MuCIGREF are presented.

7.3 Contributions

MuCIGREF is presented as an approach to represent and execute multiple concurrently implemented CIGs for generating combined care plan in order to support HCPs for managing multimorbid patients. Real-world multimorbidity case studies are used, with associated CPGs and patient data, and compared with existing works to demonstrate MuCIGREF's applicability and contributions. MuCIGREF contributions are analysed under two categories: (i) CIG representation language, and (ii) real-time CIG execution engine, respectively.

7.3.1 Contributions of CIG Representation Language

The specific contributions of MuCRL are as follows:

Representation. MuCRL as a language introduces a generic knowledge representation approach to represent knowledge constructs of multiple CPGs and their interrelations, and to create multimorbidity relations between them. The gaps of existing literature mainly on representing advanced branching and synchronisation patterns required to manage concurrent multiple CIGs and creating mappings between them, are filled in MuCRL. Through developing generic ontology, this thesis also aimed to eliminate semantic heterogeneity problem (i.e. representing the same knowledge differently) of the literature. The use of clinical terminology standards is important for achieving semantic interoperability for knowledge sharing and reusing. Thus, existing CIGs which are built on different CIG formalisms can be easily mapped with MuCRL through the clinical IDs. For instance, SNOMED-CT codes are used in this thesis. Thus, clinical activities and their interrelations can be easily integrated with clinical standards (e.g., UMLS Semantic Network, HL7 RIM) by mapping with their clinical identification codes to maintain semantic interoperability while implementing CIGs.

Reusability. MuCRL is implemented using OWL2- the modelling standard of the Semantic Web and W3C; and then Emfatic to represent EMF Ecore metamodels in Eclipse. The link of OWL file is presented in the Section 7.5 to support reusing of this work.

In the following section, the contributions of MuCIGREF's real-time CIG execution engine are presented.

7.3.2 Contributions of Real-time CIG Execution Engine

MuCEE is designed to combine multiple CIGs through performing real-time execution. This engine is built on MuCRL and involves three modules as CIG acquisition, parallel CIG execution and CIG model checking. The contributions of this engine are as follows:

Care combination approach. Introducing a novel real-time CIG execution mechanism based on a set of execution algorithms (see Section 4.4) is the major contribution of this thesis. This mechanism has three specific objectives and associated contributions, which are as follows:

- The first objective is about acquiring (transferring) multiple CIGs (see Section 4.3.3) based on satisfaction of given constraints which maintains CIGs are up-to-date (i.e. supplies CIG version control) and can be directly adapted to the existing personal care plan of a patient in any point of his/her care pathway;
- The second objective is about execution of parallel CIGs (see Section 4.3.2 which contributes to the literature by introducing a set of specialised algorithms to achieve (i) multiple clinical activity concurrency management; (ii) multiple activity merging through performing time-based synchronisation of activities; (iii) care modification to perform required updates on clinical activities and their care elements; and (iv) time-based care optimisation to eliminate care repetitions and maintain reusing them to eliminate unnecessary health resource use. Their functionalities and constraint types are also addressed in Section 6.5;

- The third objective is about CIG verification as part of the verification analysis. A new verification approach is introduced which generates dynamic custom-built user messages to fix the missing information, semantic and syntactic inconsistencies. This module supports pre- and real-time CIG execution phases and also supply customised messages that directly updates CIG models and associated personal care plans.

Dynamic user support. Direct adaptation of CIGs to personal care plans and supplying dynamic user supports also other contributions of the execution approach. Thus, MuCEE can be used when simultaneous management of care flows of different CPGs are needed.

Flexibility. MuCEE enables users to override and modify care steps and do not force them to bind only the CIG recommendations. Thus, users can make changes as he/she wishes.

7.4 Limitations and Future Work

This section presents the limitations of this thesis with suggested solutions to handle these limitations in future research work.

7.4.1 CIG Language Limitations and Solutions

The limitations of MuCRL and suggestions to handle them are as discussed below:

Computerising CPG encoding. The future work will be computerising the manual encoding of the guidelines and supporting this with appropriate natural language processing techniques, such as for finding semantic likeness [257]. For instance, embedding WordNet (*https://wordnet.princeton.edu/*) to the system may help to group of similar knowledge elements (e.g., [258]) and may reduce the instance creation time. Moreover, transformation from OWL to EMF is performed manually. This process can be automatically implemented using existing tools [232] to facilitate mapping between two platforms and reduce the transformation time but this is out of scope of this work.

Knowledge comprehension. Knowledge comprehension can be improved by adding the following items to MuCRL:

- HCP information is represented in MuCRL. Each clinical activity is associated with minimum one HCP. However, semantics regarding agent delegations [259], involving the number of required clinicians for a specific clinical process and their utilisations or equipment delegations and their availability, are not addressed. Thus, these can be a future extension of this thesis;
- If there are multiple drugs available to treat a same disease, which are appropriate for the patient, making a drug recommendation to a HCP that has the least acquisition cost, can be a future extension of this thesis as well.

Knowledge evaluation. More CPGs can be used for the verification of the expressivity of the ontology. Lastly, the re-usability of this work needs to be tested by domain experts on whether it is adequately clear and understandable by them as part of the validation process. These can be future implementations of this thesis.

7.4.2 CIG Execution Engine Limitations and Solutions

The limitations of MuCEE and suggestions to handle them are as follows:

Exception handling. To maintain patient-safety, potential medical errors or harms must be discovered and resolved. Thus, more CIG verification constraints can be introduced such as to detect adverse drug-patient or drug-food interactions. In addition, resource unavailability (e.g., doctor, laboratory equipment, operating room, etc.) is not considered in automated care activity recommendations. The inclusion of them can be a future implementation of this thesis. Existing verification tools (e.g., SPIN ([161] model checker) can be used for performance comparison with the proposed CIG verification module as well.

Patient involvement. For the validation of MuCEE, some patient scenarios from real-world case studies are considered in this thesis. However, future implementation can be involving real patients in testing the system by including their preferences (such as therapy choices) in care-personalisation process as a result of HCP-patient shared decision making [260]. In this thesis, patient preferences are not directly implemented. However, HCPs can include them while generating care recommendations using modification functionality of the system.

Conflict detection. In this thesis, drug conflicts are not automatically detected, but the modification algorithm is introduced which helps users to add a modification constraint to the personal care plan for performing resolution (e.g., replacement with a safe medication alternative) on the detected conflicts. To achieve this, the official websites such as drugbank.ca [213] or drugs.com [214] where different types of interactions such as drug-drug, drug-disease, drug-food with their different conflict degrees (e.g., moderate, severe, etc.) can be checked, and then embedded to the CIG execution system to automatically discover clinical actions that will possibly adversely interact. Future work will be improving the conflict discovery process.

System Requirements. The development of user-friendly interfaces to handle personal care plans, maintain interactions and communications between HCPs while coordinating a multimorbid patient care is the limitation of this thesis. Thus, a future interface implementation may enable HCPs to receive recommendations and inputs from other HCPs. Moreover, MuCEE is not integrated with EMRs. Patient data is manually supplied but it mimics the real-world implementation because real-world case studies are used (see Section 5.3.5). Likewise, clinical terminologies are used like SNOMED-CT codes in guideline implementations where appropriate, but they are not automatically applied. Thus, this integration can be computerised in a future implementation.

Execution evaluation. Reusability and interoperability in combining multiple CIGs can be further tested using more case studies involving domain experts' views which are future works of this thesis.

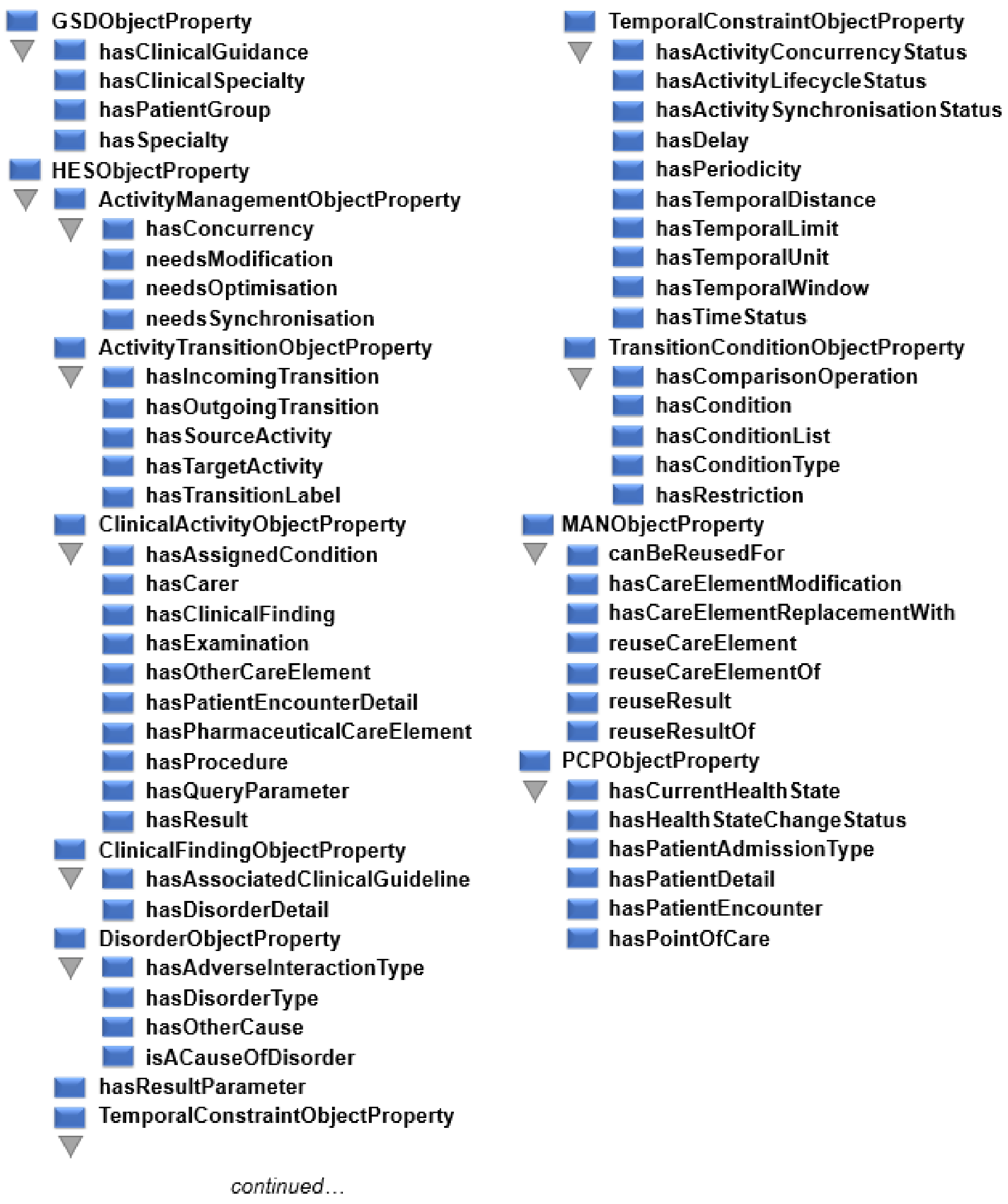
7.5 Data and Material Sharing

MuCRL knowledge representation which is implemented in OWL2 format in Protégé can be accessed via Web Protégé, please see Özyiğit [261].

Appendix A

Supplementary Material

Figure A.1 and Figure A.2 presents lists of object and data properties per ontology group of MuCRL, respectively.

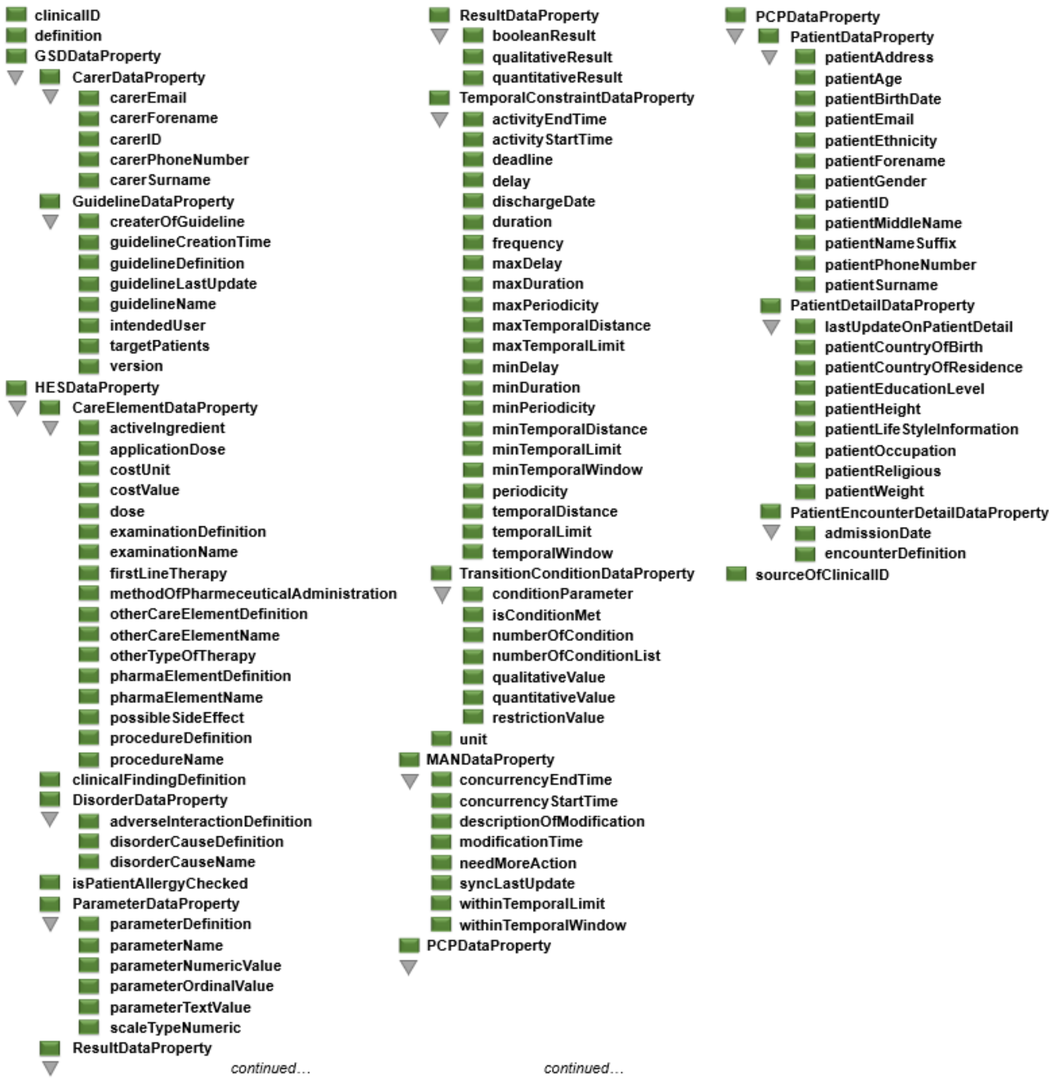


- GSDObjectProperty
  - hasClinicalGuidance
  - hasClinicalSpecialty
  - hasPatientGroup
  - hasSpecialty
- HESObjectProperty
  - ActivityManagementObjectProperty
    - hasConcurrency
    - needsModification
    - needsOptimisation
    - needsSynchronisation
  - ActivityTransitionObjectProperty
    - hasIncomingTransition
    - hasOutgoingTransition
    - hasSourceActivity
    - hasTargetActivity
    - hasTransitionLabel
  - ClinicalActivityObjectProperty
    - hasAssignedCondition
    - hasCarer
    - hasClinicalFinding
    - hasExamination
    - hasOtherCareElement
    - hasPatientEncounterDetail
    - hasPharmaceuticalCareElement
    - hasProcedure
    - hasQueryParameter
    - hasResult
  - ClinicalFindingObjectProperty
    - hasAssociatedClinicalGuideline
    - hasDisorderDetail
  - DisorderObjectProperty
    - hasAdverseInteractionType
    - hasDisorderType
    - hasOtherCause
    - isACauseOfDisorder
  - hasResultParameter
  - TemporalConstraintObjectProperty

continued...

- TemporalConstraintObjectProperty
  - hasActivityConcurrencyStatus
  - hasActivityLifecycleStatus
  - hasActivitySynchronisationStatus
  - hasDelay
  - hasPeriodicity
  - hasTemporalDistance
  - hasTemporalLimit
  - hasTemporalUnit
  - hasTemporalWindow
  - hasTimeStatus
- TransitionConditionObjectProperty
  - hasComparisonOperation
  - hasCondition
  - hasConditionList
  - hasConditionType
  - hasRestriction
- MANObjectProperty
  - canBeReusedFor
  - hasCareElementModification
  - hasCareElementReplacementWith
  - reuseCareElement
  - reuseCareElementOf
  - reuseResult
  - reuseResultOf
- PCPObjectProperty
  - hasCurrentHealthState
  - hasHealthStateChangeStatus
  - hasPatientAdmissionType
  - hasPatientDetail
  - hasPatientEncounter
  - hasPointOfCare

Figure A.1: List of object properties per ontology group of MuCRL



- clinicalID
- definition
- GSDDataProperty
  - CarerDataProperty
    - carerEmail
    - carerForename
    - carerID
    - carerPhoneNumber
    - carerSurname
  - GuidelineDataProperty
    - createrOfGuideline
    - guidelineCreationTime
    - guidelineDefinition
    - guidelineLastUpdate
    - guidelineName
    - intendedUser
    - targetPatients
    - version
- HESDataProperty
  - CareElementDataProperty
    - activeIngredient
    - applicationDose
    - costUnit
    - costValue
    - dose
    - examinationDefinition
    - examinationName
    - firstLineTherapy
    - methodOfPharmeceuticalAdministration
    - otherCareElementDefinition
    - otherCareElementName
    - otherTypeOfTherapy
    - pharmaElementDefinition
    - pharmaElementName
    - possibleSideEffect
    - procedureDefinition
    - procedureName
  - clinicalFindingDefinition
  - DisorderDataProperty
    - adverseInteractionDefinition
    - disorderCauseDefinition
    - disorderCauseName
  - isPatientAllergyChecked
  - ParameterDataProperty
    - parameterDefinition
    - parameterName
    - parameterNumericValue
    - parameterOrdinalValue
    - parameterTextValue
    - scaleTypeNumeric
  - ResultDataProperty

continued…

- ResultDataProperty
  - booleanResult
  - qualitativeResult
  - quantitativeResult
- TemporalConstraintDataProperty
  - activityEndTime
  - activityStartTime
  - deadline
  - delay
  - dischargeDate
  - duration
  - frequency
  - maxDelay
  - maxDuration
  - maxPeriodicity
  - maxTemporalDistance
  - maxTemporalLimit
  - minDelay
  - minDuration
  - minPeriodicity
  - minTemporalDistance
  - minTemporalLimit
  - minTemporalWindow
  - periodicity
  - temporalDistance
  - temporalLimit
  - temporalWindow
- TransitionConditionDataProperty
  - conditionParameter
  - isConditionMet
  - numberOfCondition
  - numberOfConditionList
  - qualitativeValue
  - quantitativeValue
  - restrictionValue
- unit
- MANDataProperty
  - concurrencyEndTime
  - concurrencyStartTime
  - descriptionOfModification
  - modificationTime
  - needMoreAction
  - syncLastUpdate
  - withinTemporalLimit
  - withinTemporalWindow
- PCPDataProperty

continued…

- PCPDataProperty
  - PatientDataProperty
    - patientAddress
    - patientAge
    - patientBirthDate
    - patientEmail
    - patientEthnicity
    - patientForename
    - patientGender
    - patientID
    - patientMiddleName
    - patientNameSuffix
    - patientPhoneNumber
    - patientSurname
  - PatientDetailDataProperty
    - lastUpdateOnPatientDetail
    - patientCountryOfBirth
    - patientCountryOfResidence
    - patientEducationLevel
    - patientHeight
    - patientLifeStyleInformation
    - patientOccupation
    - patientReligious
    - patientWeight
  - PatientEncounterDetailDataProperty
    - admissionDate
    - encounterDefinition
- sourceOfClinicalID

Figure A.2: List of data properties per ontology group of MuCRL

Bibliography

[1] M. Field and K. Lohr, Clinical Practice Guidelines: Directions for a new program, Washington DC: The National Academies Press, 1990.

[2] M. Field and K. Lohr, Guidelines for clinical practice: From development to use, Washington DC: The National Academies Press, 1992.

[3] R. Lenz, R. Blaser, M. Beyer, O. Heger, C. Biber, M. Bäumlein, et al., "IT support for clinical pathways - Lessons learned," *Int. J. Med. Inform.*, vol. 76, pp. 397–402, 2008.

[4] M. Peleg, V. L. Patel, V. Snow, S. Tu, C. Mottur-Pilson, E. H. Shortliffe, et al., "Support for guideline development through error classification and constraint checking," in *AMIA Annual Symp. Proc.*, 2002.

[5] S. Woolf, R. Grol, A. Hutchinson, M. Eccles and J. Grimshaw, "Potential benefits, limitations, and harms of clinical guidelines," *BMJ*, vol. 318, no. 7182, pp. 527–530, 1999.

[6] Institute of Medicine (IOM), Crossing the quality chasm: A new health system for the 21st century, Washington DC: The National Academies Press, 2001, pp. 39–54.

[7] I. Sim, P. Gorman, R. A. Greenes, R. B. Haynes, B. Kaplan, H. Lehmann, et al., "Clinical decision support systems for the practice of evidence-based medicine," *J. Am. Med. Inform. Assoc.*, vol. 8, no. 6, pp. 527–534, 2001.

[8] E. K. Genco, J. E. Forster, H. Flaten, F. Goss, K. J. Heard, J. Hoppe, et al., "Clinically inconsequential alerts: The characteristics of opioid drug alerts and their utility in preventing adverse drug events in the emergency department," *Ann. Emerg. Med.*, vol. 67, no. 2, pp. 240–248, 2016.

[9] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R. A. Greenes, et al., "Comparing computer-interpretable guideline models: A case-study approach," *J. Am. Med. Inform. Assoc.*, vol. 10, no. 1, pp. 52–68, 2003.

[10] M. Peleg, "Computer-interpretable clinical guidelines: A methodological review," *J. Biomed. Inform.*, vol. 46, no. 4, pp. 744–763, 2013.

[11] H. Karadimas, V. Ebrahiminia and E. Lepage, "User-defined functions in the Arden Syntax: An extension proposal," *Artif. Intell. Med.*, vol. 92, pp. 103–110, 2018.

[12] A. Seyfang, S. Miksch and M. Marcos, "Combining diagnosis and treatment using Asbru,"*Int. J. Med. Inform.*, vol. 68, no. 1, pp. 49–57, 2002.

[13] S. Tu and M. Musen, "Modeling data and knowledge in the EON guideline architecture," *Stud. Health Technol.*, vol. 84, no. 1, pp. 280–284, 2001.

[14] P. Terenziani, S. Montani, A. Bottrighi, M. Torchio, M. G. and G. Correndo, "The GLARE approach to clinical guidelines: main features," *Stud. Health Technol. Inform.*, vol. 101, pp. 162–166, 2004.

[15] D. Wang, M. Peleg, S. W. Tu, A. A. Boxwala, O. Ogunyemi, Q. Zeng, et al., "Design and implementation of the GLIF3 guideline execution engine," *J. Biomed. Inform.*, vol. 37, no. 5, pp. 305–318, 2004.

[16] D. R. Sutton and J. Fox, "The syntax and semantics of the PROforma guideline modeling language," *J. Am. Med. Inform. Assoc.*, vol. 10, no. 5, pp. 433–443, 2003.

[17] S. Tu, J. Campbell and M. Musen, "The SAGE guideline modeling: motivation and methodology," *Stud. Health Technol. Inform.*, vol. 101, pp. 167–171, 2004.

[18] R. Studer, V. R. Benjamins and D. Fensel, "Knowledge engineering: Principles and methods," *Data Knowl. Eng.*, vol. 25, no. 1-2, pp. 161–197, 1998.

[19] D. Isern and A. Moreno, "Computer-based execution of clinical guidelines: a review," *Int. J. Med. Inform.*, vol. 77, no. 12, pp. 787–808, 2008.

[20] M. E. Salive, "Multimorbidity in older adults," *Epiodemiol. Rev.*, vol. 35, no. 1, pp. 75–83, 2013.

[21] R. Navickas, V. K. Petric, A. B. Feigl and M. Seychell, "Multimorbidity: What do we know? What should we do?," *J. Comorbidity*, vol. 6, no. 1, pp. 4–11, 2016.

[22] S. P. Bell and A. A. Saraf, “Epidemiology of multimorbidity in older adults with cardiovascular disease,” *Clin. Geriatr. Med.*, vol. 32, no. 2, pp. 215–226, 2016.

[23] C. Salisbury, L. Johnson, S. Purdy, J. M. Valderas and A. A. Montgomery, “Epidemiology and impact of multimorbidity in primary care: A retrospective cohort study,” *Br. J. Gen. Pract.*, vol. 61, no. 582, pp. e12–e21, 2011.

[24] C. Salisbury, “Multimorbidity: Redesigning health care for people who use it,” *The Lancet*, vol. 380, no. 9836, pp. 7–9, 2012.

[25] M. Vassilaki, J. Aakre, R. H. Cha, W. Kremers, S. J. L. St, M. Mielke, et al., “Multimorbidity and risk of mild cognitive impairment,” *J. Am. Geriatr. Soc.*, vol. 63, no. 9, pp. 1783–1790, 2015.

[26] N. Lasierra, A. Alesanco, S. Guillen and J. Garcia, “A three stage ontology-driven solution to provide personalised care to chronic patients at home,” *J. Biomed. Inform.*, vol. 46, no. 3, pp. 516–529, 2013.

[27] L. Anselma, A. Bottrighi, L. Giordano, A. Hommersom, G. Molino, S. Montani, et al., “A hybrid approach to the verification of computer interpretable guidelines,” in *Foundations of Biomedical Knowledge Representation*, Heidelberg, Springer, 2015, pp. 287–315.

[28] D. Riaño and A. Collado, “Model-based combination of treatments for the management of chronic comorbid patients,” in *Lecture Notes in Computer Science*, vol. 7885, N. Peek, R. Marìn Morales and M. Peleg, Eds., Heidelberg, Berlin, Springer, 2013, pp. 11–16.

[29] D. Kodner, “All together now: A conceptual exploration of integrated care,” *Healthc. Q.*, vol. 13, pp. 6–15, 2009.

[30] N. Mulyar, W. M. Van der Aalst and M. Peleg, “A pattern-based analysis of clinical computer-interpretable guideline modeling languages,” *J. Am. Med. Inform. Assoc.*, vol. 14, no. 6, pp. 781–787, 2007.

[31] E. Bilici, G. Despotou and T. Arvanitis, “The use of computer-interpretable clinical guidelines to manage care complexities of patients with multimorbid conditions: A review,” *Digital Health*, vol. 4, pp. 1–21, 2018.

[32] D. Sittig, A. Wright, J. Osheroff, B. Middleton, J. Teich, J. Ash, et al., “Grand challenges in clinical decision support,” *J. Biomed. Inform.*, vol. 41, no. 2, pp. 387–392, 2008.

[33] E. Kilsdonk, L. Peute, R. Riezebos, L. Kremer and M. Jaspers, “Uncovering healthcare practitioners’ information processing using the think-aloud method: From paper-based guideline to clinical decision support system,” *Int. J. Med. Inform.*, vol. 45, no. 6, pp. 10–19, 2016.

[34] P. De Clercq, K. Kaiser and A. Hasman, “Computer-interpretable guideline formalisms,” *Stud. Health Technol. Inform.*, vol. 139, pp. 22–43, 2008.

[35] A. Gonzàlez-Ferrer, A. Ten Teije, J. Fdez-Olivares and K. Milian, “Automated generation of patient-tailored electronic care pathways by translating computer-interpretable guidelines into hierarchical task networks,” *Artif. Intell. Med.*, vol. 57, no. 2, pp. 91–109, 2013.

[36] S. Wilk, M. Michalowski, W. Michalowski, M. Hing and K. Farion, “Reconciling pairs of concurrently used clinical practice guidelines using constraint logic programming,” in *AMIA Annu. Symp. Proc.*, 2011.

[37] A. Kovalov and J. Bowles, “Avoiding medication conflicts for patients with multimorbidities,” in *Integrated Formal Methods*, vol. 9681, E. ÀÁbrahàm and M. Huisman, Eds., Springer, Cham, 2016, pp. 376–390.

[38] C. Breuker, O. Abraham, L. Di Trapanie, T. Mura, V. Macioce, C. Boegner,et al., “Patients with diabetes are at high risk of serious medication errors at hospital: Interest of clinical pharmacist intervention to improve healthcare,” *Eur. J. Intern. Med.*, vol. 38, pp. 38–45, 2017.

[39] J. Aronson, “Medication errors: Definitions and classification,” *Br. J. Clin. Pharmacol.*, vol. 67, no. 6, pp. 599–604, 2009.

[40] F. Grandi, F. Mandreoli and R. Martoglia, “Efficient management of multi-version clinical guidelines,” *J. Biomed. Inform.*, vol. 45, no. 6, pp. 1120–1136, 2012.

[41] F. Grandi, “Dynamic class hierarchy management for multi-version ontology-based personalisation,” *J. Comput. Syst. Sci.*, vol. 82, no. 1, pp. 62–90, 2016.

[42] M. A. Steinman, C. S. Landefeld, G. E. Rosenthal, D. Berthenthal, S. Sen and P. J. Kaboli, “Polypharmacy and prescribing quality in older people,” *J. Am. Geriatr. Soc.*, vol. 54, no. 10, pp. 1516–1523, 2006.

[43] R. Dagli and A. Sharma,“Polypharmacy: A global risk factor for elderly people,” *J. Int. Oral Health*, vol. 6, no. 6, pp. 1–2, 2014.

[44] D. Bates, D. Cullen, N. Laird, L. Petersen, S. Small, D. Servi,et al., “Incidence of adverse drug events and potential adverse drug events: Implications for prevention,” *JAMA*, vol. 274, no. 1, pp. 29–34, 1995.

[45] S. Ghibelli, A. Marengoni, C. Djade, A. Nobili, M. Tettamanti, C. Franchi, et al., “Prevention of inappropriate prescribing in hospitalized older patients using a computerised prescription support system,” *Drugs Aging*, vol. 30, no. 10, pp. 821–828, 2013.

[46] T. Morimoto, T. Gandhi, A. Seger, T. Hsieh and D. Bates, “Adverse drug events and medication errors: Detection and classification methods,” *Qual. Saf. Health Care*, vol. 13, no. 4, pp. 306–314, 2004.

[47] D. Bates, N. Spell, D. Cullen, E. Burdick, N. Laird, L. Petersen, et al., “The costs of adverse drug events in hospitalised patients,” *JAMA*, vol. 277, no. 4, pp. 307–311, 1997.

[48] M. Pirmohamed, S. James, S. Meakin, C. Green, A. Scott, T. Walley, et al., “Adverse drug reactions as cause of admission to hospital: Prospective analysis of 18820 patients,” *BMJ*, vol. 329, no. 7456, pp. 15–19, 2004.

[49] M. Akçura and Z. Ozdemir, “Drug prescription behavior and decision support systems,” *Decis. Support. Syst.*, vol. 57, pp. 395–405, 2014.

[50] G. Duftschmid, S. Miksch and W. Gall, “Verification of temporal scheduling constraints in clinical practice guidelines,” *Artif. Intell. Med.*, vol. 25, no. 2, pp. 93–121, 2002.

[51] D. Isern, A. Moreno, D. Sànchez, Á. Hajnal, G. Pedone and L. Varga, “Agent-based execution of personalised home care treatments,” *Appl. Intell.*, vol. 34, no. 2, pp. 155–180, 2011.

[52] L. Anselma, P. Terenziani, S. Montani and A. Bottrighi, “Towards a comprehensive treatment of repetitions, periodicity and temporal constraints in clinical guidelines,” *Artif. Intell. Med.*, vol. 38, no. 2, pp. 171–195, 2006.

[53] M. Uschold and M. Gruninger, “Ontologies and semantics for seamless connectivity,” *ACM SIGMod Record*, vol. 33, no. 4, pp. 58–64, 2004.

[54] A. Malhotra, E. Younesi, M. Gündel, B. Müller, M. Heneka and M. Hofmann-Apitius, “ADO: A disease ontology representing the domain knowledge specific to Alzheimer’s disease,” *Alzheimers Dement.*, vol. 10, no. 2, pp. 238–246, 2014.

[55] M. Waqialla and M. Razzak, “An ontology-based framework aiming to support cardiac rehabilitation program,” *Procedia Comput. Sci.*, vol. 96, pp. 23–32, 2016.

[56] Y. Zhang and Z. Zhang, “Preliminary result on finding treatments for patients with comorbidity,” in *Knowledge Representation for Health Care*, vol. 8903, S. Miksch, D. Riaño and A. ten Teije, Eds., Springer, Cham, 2014, pp. 14–28.

[57] N. Russell, A. Ter Hofstede, W. Van Der Aalst and N. Mulyar, “Workflow control-flow patterns: A revised view,” BPM Center Report BPM-06-22, Eindhoven, 2006.

[58] BioPortal, National Center for Biomedical Ontology, [Online]. Available: `https://bioportal.bioontology.org/`. [Accessed March 2019].

[59] Health Level Seven International, “HL7 Reference Information Model,” [Online]. Available: `https://www.hl7.org/implement/standards/product_brief.cfm?product_id=77`. [Accessed Apr 2019].

[60] M. Fernández-López, A. Gómez-Pérez and N. Juristo, “Methontology: From ontological art towards ontological engineering,” in *AAAI-97 Spring Symposium Series*, Stanford University, 1997.

[61] N. Noy and D. McGuinness, “Ontology development 101: A guide to creating your first ontology: Knowledge systems laboratory,” Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI, 2001.

[62] G. Antoniou and F. van Harmelen, “Web Ontology Language: OWL,” in *Handbook on Ontologies*, S. Staab and R. Studer, Eds., Springer, Berlin, Heidelberg, 2004, pp. 67–92.

[63] W3C, “OWL2 Web Ontology Language,” [Online]. Available: `https://www.w3.org/TR/owl2-overview/`.[Accessed Aug 2019].

[64] The Eclipse Project, “Eclipse Modeling Framework,” [Online]. Available: `https://www.eclipse.org/modeling/emf/`. [Accessed May 2019].

[65] D.S. Kolovos, L. Rose, R.F. Paige and A. Garcìa-Domınguez, The Epsilon Book, 2013.

[66] Object Management Group, “Object Constraint Language,” Feb 2014. [Online]. Available:https://www.omg.org/spec/OCL/About-OCL/. [Accessed Mar 2019].

[67] C3-Cloud (A federated collaborative care cure cloud architecture for addressing the needs of multi-morbidity and managing poly-pharmacy), “D7.1 Evidence based clinical guideline definitions and flowcharts for individual chronic conditions,” Dec 2016. [Online]. Available: https://c3-cloud.eu/wp-content/uploads/2019/06/D7.1_v1.pdf. [Accessed Jan 2019].

[68] C3-Cloud (A federated collaborative care cure cloud architecture for addressing the needs of multi-morbidity and managing poly-pharmacy), “D4.1 Guidance for the development of new patient pathways and corresponding care plans,” 7 Jun 2017. [Online]. Available: https://c3-cloud.eu/wp-content/uploads/2019/06/D4.1_v1.pdf. [Accessed Jan 2019].

[69] E. Bilici, S. Lim Choi Keung, G. Despotou and T. Arvanitis, “Computer-interpretable guidelines driven clinical decision support systems: An approach to the treatment personalisation routes of patients with multi-diseases,” in *WIN 2017*, Coventry, UK, 2017.

[70] E. Bilici, G. Despotou and T. Arvanitis, “Ontology-driven representation and management of clinical practice guidelines and their associated knowledge constructs: A case study for multimorbidity,” *J. Biomed. Semant.*, submitted, 2019.

[71] E. Bilici, G. Despotou and T. Arvanitis, “Concurrent execution of multiple computer-interpretable clinical practice guidelines and their interrelations,” *Stud. Health Technol. Inform.*, vol. 262, pp. 7–10, 2019.

[72] J. Pan, S. Staab, U. Aßmann, J. Ebert and Y. Zhao, Ontology-driven software development, Springer Science and Business Media, 2012.

[73] C. Roussey, F. Pinet, M. Kang and O. Corcho, “An introduction to ontologies and ontology engineering,” in *Ontologies in Urban Development Projects*, London, Springer, 2011.

[74] A. Rector, J. Rogers, P. Zanstra and E. V. D. Haring, “OpenGALEN: Open source medical terminology and tools,” in *AMIA Annu. Symp. Proc.*, 2003.

[75] R. Stevens, C. Goble and S. Bechhofer, “Ontology-based knowledge representation for bioinformatics,” *Brief. Bioinform.*, vol. 1, no. 4, pp. 398–414, 2000.

[76] J. Euzenat and P. Shvaiko, "The matching problem," in *Ontology Matching*, Berlin, Springer, Heildelberg, pp. 25–54, 2007.

[77] M. Fox, "The TOVE project towards a common-sense model of the enterprise," in *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, F. Belli and F. Radermacher, Eds., Springer, Berlin, Heidelberg, pp. 25–34, 1992.

[78] Y. Sure, S. Staab and R. Studer, "On-To-Knowledge Methodology," in *Handbook on Ontologies*, S. Staab and R. Studer, Eds., Springer, Berlin, Heidelberg, pp. 117–132, 2004.

[79] A. Schreiber, G. Schreiber, H. Akkermans, A. Anjewierden, N. Shadbolt, R. de Hoog, et al., "Knowledge engineering and management: The CommonKADS methodology," MIT Press, 2000.

[80] M. Suárez-Figueroa, A. Gómez-Pérez and M. Fernández-López, "The NeOn Methodology framework: A scenario-based methodology for ontology development," *Appl. Ontol.*, vol. 10, pp. 107–145, 2015.

[81] Object Management Group, "OMG Unified Modeling Language," Dec 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1/PDF. [Accessed Aug 2019].

[82] S. Cranefield and M. Purvis, "UML as an ontology modelling language," in *New Zealand: Department of Information Science*, University of Otago, 1999, pp. 46–53.

[83] W3C Recommendation, "OWL 2 Web Ontology Language: Structural specification and functional-style Syntax," Dec 2012. [Online]. Available: `https://www.w3.org/TR/owl2-syntax/`. [Accessed Jun 2019].

[84] W3C, "Semantic Web," 2015. [Online]. Available: `https://www.w3.org/standards/semanticweb/`.[Accessed Aug 2019].

[85] W3C, "Extensible markup language (XML)," 2016. [Online]. Available: `https://www.w3.org/XML/`.[Accessed Aug 2019].

[86] K. Munir and M. Anjum, "The use of ontologies for effective knowledge modelling and information retrieval," *Applied Computing and Informatics*, vol. 14, pp. 116–126, 2018.

[87] J. Pan, “Resource description framework,” in *Handbook on Ontologies*, Springer, Berlin, Heidelberg, 2009, pp. 71–90.

[88] D. Bricklet and R. Guha, “Resource Description Framework (RDF) Schema 1.1,” W3C Recommendation, 25 Feb 2014. [Online]. Available: `https://www.w3.org/TR/rdf-schema/`.

[89] B. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider and U. Sattler, “OWL 2: The next step for OWL,” *Web Semant.*, vol. 6, no. 4, pp. 309–322, 2008.

[90] A. Gòmez-Pèrez and O. Corcho, “Ontology languages for the semantic web,” *IEEE Intell.*, vol. 17, no. 1, pp. 54–60, 2002.

[91] B. Grosof, I. Horrocks, R. R. Volz and S. Decker, “Description logic programs: Combining logic programs with description logic,” in *12th Int. Conf. on World Wide Web*, Budapest, 2003.

[92] R. Shearer, B. Motik and I. Horrocks, “HermiT: A highly-efficient OWL reasoner,” in *OWLED*, 2008.

[93] D. Tsarkov and I. Horrocks, “FaCT++ description logic reasoner: System description,” in *Automated Reasoning*, vol. 4130, U. Furbach and N. Shankar, Eds., Berlin, Springer, 2006, pp. 292–297.

[94] J. Bock, P. Haase, Q. Ji and R. Volz, “Benchmarking OWL reasoners,” in *Workshop on Advancing Reasoning on the Web: Scalability and Commonsense*, Tenerife, 2008.

[95] W3C, “SPARQL query language for RDF,” 2008. [Online]. Available: `https://www.w3.org/TR/rdf-sparql-query/`.

[96] Eclipse, “Emfatic: A textual synctax for EMF Ecore (meta-) models,” 2004. [Online]. Available: `https://www.eclipse.org/emfatic/`. [Accessed Jul 2019].

[97] J. Warmer,and A. Kleppe, A., The Object Constraint Language: Getting Your Models Ready for MDA, Reading, MA: Addison-Wesley, 2003.

[98] P.J. Papajorgji, and P.M. Pardalos, Software engineering techniques applied to agricultural systems: An object oriented and UML approch, pp. 121–122, Springer, 2014.

[99] Epsilon. [Online]. Available: `http://www.eclipse.org/epsilon`. [Accessed Jul 2019]

[100] Epsilon, “Epsilon Object Language,” [Online]. Available: `https://www.eclipse.org/epsilon/doc/eol/`. [Accessed Sep 2019]

[101] D. Kolovos, R. Paige and F. Polack, “The Epsilon Object Language (EOL),” in *ECMDA-FA*, 2006.

[102] Epsilon, “Epsilon Validation Language,” [Online]. Available: `https://www.eclipse.org/epsilon/doc/evl/`. [Accessed Sep 2019].

[103] D.S. Kolovos, R.F. Paige, F.A. Polack, “On the evolution of OCL for capturing structural constraints in modelling languages,” in *Rigorous Methods for Software Construction and Analysis*, Springer, Berlin, Heidelberg, 2009, pp. 204–218.

[104] Y. F. Zhang, L. Gou, Y. Tian, T.C. Li, M. Zhang and J.S. Li, “Design and development of a sharable clinical decision support system based on a semantic web service framework,” *J. Med. Syst.*, vol. 40, no. 5, p. 118, 2016.

[105] M. Goldstein, B. Hoffman, R. Coleman, S. Tu, R. Shankar, M. O’Connor, et al., “Patient safety in guideline-based decision support for hypertension management: ATHENA DSS,” *J. Am. Med. Inform. Assoc.*, vol. 9, no. 6, pp. S11–6, 2002.

[106] R. Goud, A. Hasman, S. A. and N. Peek, “A parallel guideline development and formalisation strategy to improve the quality of clinical practice guidelines,” *Int. J. Med. Inform.*, vol. 78, no. 8, pp. 513–520, 2009.

[107] R. N. Shiffman, A. Agrawal, A. M. Deshpande and P. Gershkovich, “An approach to guideline implementation with GEM,” *Stud. Health Technol. Inform.*, vol. 84, no. 1, pp. 271–275, 2001.

[108] P. De Clercq, A. Hasman, J. Blom and H. Korsten, “Design and implementation of a framework to support the development of clinical guidelines,” *Int. J. Med. Inform.*, vol. 64, no. 2-3, pp. 285–318, 2001.

[109] M. Hussain, M. Afzal, T. Ali, R. Ali, W. A. Khan, A. Jamshed, et al., “Data-driven knowledge acquisition, validation, and transformation into HL7 Arden Syntax,” *Artif. Intell. Med.*, vol. 92, pp. 51–70, 2018.

[110] R. Jenders, K. Adlassnig, K. Fehre and P. Haug, "Evolution of the Arden Syntax: key technical issues from the standards development organisation perspective," *Artif. Intell. Med.*, vol. 92, pp. 10–14, 2018.

[111] V. Koutkias, V. Kilintzis, G. Stalidis, K. Lazou, J. Niès, L. Durand-Texte, et al., "Knowledge engineering for adverse drug event prevention: on the design and development of a uniform, contextualized and sustainable knowledge-based framework," *J. Biomed. Inform.*, vol. 45, no. 3, pp. 495–506, 2012.

[112] P. Ciccarese, E. Caffi, S. Quaglini and M. Stefanelli, "Architectures and tools for innovative health information systems: The Guide project," *Int. J. Med. Inform.*, vol. 74, no. 7–8, pp. 553–562, 2005.

[113] S. Skonetzki, H. Gausepohl, M. van der Haak, S. Knaebel, O. Linderkamp and T. Wetter, "HELEN, a modular framework for representing and implementing clinical practice guidelines," *Methods Inf. Med.*, vol. 43, no. 4, pp. 413–426, 2004.

[114] J. López-Vallverdú, D. Riaño and A. Collado, "Rule-based combination of comorbid treatments for chronic diseases applied to hypertension, diabetes mellitus and heart failure," in *Process Support and Knowledge Representation in Health Care*, vol. 7738, Berlin, Lenz, R.; Miksch, S.; Peleg, M.; Reichert, M.; Riaño, D.; ten Teije, A., 2012, pp. 30–41.

[115] M. Samwald, K. Fehre, J. Bruin and K.-P. Adlassnig, "The Arden syntax standard for clinical decision support: experiences and directions," *J. Biomed. Inform.*, vol. 45, pp. 711–718, 2012.

[116] L. Piovesan, G. Molino and P. Terenziani, "An ontological knowledge and multiple abstraction level decision support system in healthcare," *Decis. Anal.*, vol. 1, p. 8, 2014.

[117] A. Bottrighi, G. Molino, S. Montani, P. Terenziani and M. Torchio, "Supporting a distributed execution of clinical guidelines," *Comput. Methods Programs. Biomed.*, vol. 112, no. 1, pp. 200–210, 2013.

[118] A. Bottrighi and P. Terenziani, " META-GLARE: A meta-system for defining your own computer interpretable guideline system—Architecture and acquisition," *Artif. Intell. Med.*, vol. 72, pp. 22–41, 2016.

[119] Open Clinical, "GLARE," [Online]. Available: `http://www.openclinical.org/gmmsummaries.html#glare`. [Accessed Dec 2018].

[120] P. Terenziani, G. Molino and M. Torchio, "A modular approach for representing and executing clinical guidelines," *Artif. Intell. Med.*, vol. 23, no. 3, pp. 249–2769, 2001.

[121] L. Anselma, L. Piovesan and P. Terenziani, "Temporal detection and analysis of guideline interactions," *Artif. Intell. Med.*, vol. 76, pp. 40–62, 2017.

[122] World Health Organisation, "History of the development of the ICD," [Online]. Available: `https://www.who.int/classifications/icd/en/HistoryOfICD.pdf`. [Accessed Jan 2019].

[123] A. Bottrighi, S. Rubrichi and P. Terenziani, "META-GLARE: a meta-engine for executing computer interpretable guidelines," in *AIME*, 2015.

[124] Sutton, P. Taylor and K. Earle, "Evaluation of PROforma as a language for implementing medical guidelines in a practical context," *BMC Med. Inform. Decis. Mak.*, vol. 6, no. 1, p. 20, 2006.

[125] A. Grando, M. Peleg and D. Glasspool, "A goal-oriented framework for specifying clinical guidelines and handling medical errors," *J. Biomed. Inform.*, vol. 43, no. 2, pp. 287–299, 2010.

[126] J. Fox, A. Alabassi, V. Patkar, T. Rose and E. Black, "An ontological approach to modelling tasks and goals," *Comput. Biol. Med.*, vol. 36, no. 7–8, pp. 837–856, 2006.

[127] D. Isern, D. Sánchez and A. Moreno, "Ontology-driven execution of clinical guidelines," *Comput. Methods. Programs Biomed.*, vol. 107, no. 2, pp. 122–139, 2012.

[128] M. Grando, D. Glasspool and J. Fox,"A formal approach to the analysis of clinical computer-interpretable guideline modeling languages," *Artif. Intell. Med.*, vol. 54, no. 1, pp. 1–13, 2012.

[129] Y. Shahar, S. Miksch and P. Johnson, "The Asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines," *Artif. Intell. Med.*, vol. 14, no. 1–2, pp. 29–51, 1998.

[130] Y. Shahar, O. Young, E. Shalom, M. Galperin, A. Mayaffit, R. Moskovitch, et al. "A framework for a distributed, hybrid, multiple-ontology clinical-guideline library, and automated guideline-support tools," *J. Biomed. Inform.*, vol. 37, pp. 325–344, 2004.

[131] K. Donnelly, "SNOMED-CT: The advanced terminology and coding system for eHealth," in *Stud. Health Technol. Inform.* , vol. 121, pp. 279–290, 2006.

[132] Logical Observation Identifiers, Names and Codes (LOINC), [Online]. Available: `https://loinc.org/`. [Accessed Mar 2019].

[133] O. Young and Y. Shahar, "The spock system: Developing a runtime application engine for hybrid-Asbru guidelines," in *AIME*, Berlin, 2005.

[134] O. Young, Y. Shahar, Y. Liel, E. Lunenfeld, G. Bar, E. Shalom, et al., "Runtime application of Hybrid-Asbru clinical guidelines," *J. Biomed. Inform.*, vol. 40, no. 5, pp. 507–526, 2007.

[135] C. Eccher, A. Seyfang and A. Ferro, "Implementation and evaluation of an Asbru-based decision support system for adjuvant treatment in breast cancer," *Comput. Methods Programs Biomed.*, vol. 117, no. 2, pp. 308–321, 2014.

[136] M. Peleg, Y. Shahar, S. Quaglini, A. Fux, G. García-Sáez, A. Goldstein, et al., "MobiGuide: a personalized and patient-centric decision-support system and its evaluation in the atrial fibrillation and gestational diabetes domains," in *User Modeling and User-Adapted Interaction*, vol. 27, Springer, Netherlands, 2017, pp. 159–213.

[137] A. A. Boxwala, M. Peleg, S. Tu, O. Ogunyemi, Q. T. Zeng, D. Wang, et al., "GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines," *J. Biomed. Inform.*, vol. 37, no. 3, pp. 147–161, 2004.

[138] M. Peleg, S. Keren and Y. Denekamp, "Mapping computerized clinical guidelines to electronic medical records: Knowledge-data ontological mapper (KDOM)," *J. Biomed. Inform.*, vol. 41, no. 1, pp. 180–201, 2008.

[139] E. Miller, "An introduction to the resource description framework," *Bull. Am. Soc. Inf. Sci.*, vol. 25, no. 1, pp. 15–19, 1998.

[140] M. Peleg, A. Boxwala, E. Bernstam, S. Tu, R. Greenes and E. Shortliffe, "Sharable representation of clinical guidelines in GLIF: relationship to the Arden Syntax," *J. Biomed. Inform.*, vol. 34, no. 3, pp. 170–181, 2001.

[141] M. Peleg, J. Fox, V. Patkar, D. Glasspool, I. Chronakis, M. South, et al., "A computer-interpretable version of the AACE, AME, ETA medical guidelines for clinical practice for the diagnosis and management of thyroid nodules," *Endocr. Pract.*, vol. 20, no. 4, pp. 352–359, 2014.

[142] Protégé, Stanford University, [Online]. Available: `https://protege.stanford.edu/products.php`. [Accessed Oct 2018].

[143] S. Tu and M. Musen, "The EON model of intervention protocols and guidelines," in *Proc. AMIA Annu. Fall. Symp.*, 1996.

[144] S. Tu, J. Campbell, J. Glasgow, M. Nyman, R. McClure, J. McClay, et al., "The SAGE Guideline Model: achievements and overview," *J. Am. Med. Inform. Assoc.*, vol. 14, no. 5, pp. 589–598, 2007.

[145] D. Berg, P. Ram, J. Glasgow and J. Castro, "SAGEDesktop: An environment for testing clinical practice guidelines," in *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, pp. 3217–3220, 2004.

[146] K. Zheng, R. Padman, M. Johnson and S. Hasan, "Guideline representation ontologies for evidence-based medicine practice," in *Handbook of Research on Advances in Health Informatics and Electronic Healthcare Applications: Global Adoption and Impact of Information Communication Technologies*, pp. 234–254, 2010.

[147] W. van der Aalst, K. van Hee and K. van Hee, Workflow management: Models, methods, and systems, MIT Press, 2004.

[148] A. Boxwala, S. Tu, M. Peleg, Q. Zeng, O. Ogunyemi, R. Greenes, et al., "Toward a representation format for sharable clinical guidelines," *J. Biomed. Inform.*, vol. 34, pp. 157–169, 2001.

[149] G. Schadow, D. Russler and C. McDonald, "Conceptual alignment of electronic health record data with guideline and workflow knowledge," *Int. J. Med. Inform.*, vol. 64, no. 2–3, pp. 259–274, 2001.

[150] D. Boaz and Y. Shahar, "A framework for distributed mediation of temporal-abstraction queries to clinical databases," *Artif. Intell. Med.*, vol. 34, pp. 3–24, 2005.

[151] A. Preece, "Evaluating verification and validation methods in knowledge engineering." In *Industrial Knowledge Management*, 2001, pp. 91–104.

[152] A. Hommersom, P. Groot, P. Lucas, M. Balser and J. Schmitt, "Verification of medical guidelines using background knowledge in task networks," *IEEE T. Knowl. Data. En.*, vol. 19, no. 6, pp. 832–846, 2007.

[153] B. Pérez and I. Porres, "Authoring and verification of clinical guidelines: A model driven approach," *J. Biomed. Inform.*, vol. 43, no. 4, pp. 520–536, 2010.

[154] Y. Zhang, L. Gou, Y. Tian, T. Li, M. Zhang and J. Li, "Design and development of a sharable clinical decision support system based on a semantic web service framework," *J. Med. Syst.*, vol. 40, no. 118, pp. 1–14, 2016.

[155] S. Bäumler, M. Balser, A. Dunets, W. Reif and J. Schmitt, "Verification of medical guidelines by model checking–a case study," in *SPIN*, vol. 3925, pp. 219–233, 2006.

[156] L. Giordano, P. Terenziani, A. Bottrighi, et al., "Model checking for clinical guidelines: An agent-based approach," *AMIA Annu. Symp. Proc.*, vol. 2006, pp. 289–293, 2006.

[157] A. Ten Teije, M. Marcos, M. Balser, J. van Croonenborg, C. Duelli, F. van Harmelen, et al., "Improving medical protocols by formal methods," *Artif. Intell. Med.*, vol. 36, no. 3, pp. 193–209, 2006.

[158] A. Hommersom, P. Lucas and P. Van Bommel, "Checking the quality of clinical guidelines using automated reasoning tools," *Theor. Pract. Log. Prog.*, vol. 8, no. 5–6, pp. 611–641, 2008.

[159] G. Duftschmid and S. Miksch, "Knowledge-based verification of clinical guidelines by detection of anomalies," *Artif. Intell. Med.*, vol. 22, no. 1, pp. 23–41, 2001.

[160] C. Baier and J.-P. Katoen, Principles of model checking, The MIT Press, 2008.

[161] G. Holzmann, "The model checker SPIN." *IEEE Trans. Softw. Eng.*, vol. 23, no. 5, pp. 279–295, 1997.

[162] M. Balser, W. Reif, G. Schellhorn, K. Stenzel and A. Thums, "Formal system development with KIV," in *FASE*, pp. 363–366, 2000.

[163] A. Preece and R. Shinghal, "Foundation and application of knowledge base verification," *Int. J. Intell. Syst.*, vol. 9, no. 8, pp. 683–701, 1994.

[164] M. Iannaccone and M. Esposito, "Formal specification of temporal constraints in clinical practice guidelines," in *Knowledge, Information and Creativity Support Systems: Recent Trends, Advances and Solutions*, vol. 364, A. Skulimowski and J. Kacprzyk, Eds., Springer, Cham, 2016, pp. 373–386.

[165] L. Piovesan and P. Terenziani, “A mixed-initiative approach to the conciliation of clinical guidelines for comorbid patients,” in Knowledge Representation for Health Care, vol. 9485, D. RiaRiaño, R. Lenz, S. Miksch, M. Peleg, M. Reichert and A. ten Teije, Eds., Springer, Cham, 2015, pp. 95–108.

[166] M. Gospodarowicz and B. O’Sullivan, “Patient management scenario: a framework for clinical decision and prognosis,” *Semin. Surg. Oncol.*, vol. 21, pp. 8–12, 2003.

[167] R. Dechter, I. Meiri and J. Pearl, “Temporal constraint networks.” *Artif. Intell.*, vol. 49, no. 1, pp. 61–95, 1991.

[168] R. Kosara and S. Miksch, “Visualization methods for data analysis and planning in medical applications.” *Int. J. Med. Inform.*, vol. 68, no. 1–3, pp. 141–153, 2002.

[169] E. van Roon, S. Flikweert, M. le Comte, P. Langendijk, W. Kwee-Zuiderwijk, P. Smits, et al., “Clinical relevance of drug-drug interactions,” *Drug Safety*, vol. 28, no. 12, pp. 1131–1139, 2005.

[170] Stanford University, “GLINDA: Guideline INteraction Detection Architecture,” [Online]. Available: `http://glinda-project.stanford.edu/`.

[171] K. Corrie and J. Hardman, “Mechanisms of drug interactions: Pharmacodynamics and pharmacokinetics,” *Anaesth. Intensive Care*, vol. 18, no. 7, pp. 331–334, 2017.

[172] M. Yeh, Y. Chang, P. Wang, Y. Li and C. Hsu, “Physicians’ responses to computerized drug–drug interaction alerts for outpatients,” *Comput. Methods Programs Biomed.*, vol. 111, no. 1, pp. 17–25, 2013.

[173] Y. Tan, Y. Hu, X. Liu, Z. Yin, X. Chen and M. Liu, “Improving drug safety: From adverse drug reaction knowledge discovery to clinical implementation,” *Methods*, vol. 110, pp. 14–25, 2016.

[174] L. Sacchi, S. Rubrichi, C. Rognoni, S. Panzarasa, E. Parimbelli, A. Mazzanti, et al., “From decision to shared-decision: Introducing patients’ preferences into clinical decision analysis,” *Artif. Intell. Med.*, vol. 65, no. 1, pp. 19–28, 2015.

[175] M. Murad, “Clinical practice guidelines: a primer on development and dissemination,” *Mayo Clin. Proc.*, vol. 92, no. 3, pp. 423–433, 2017.

[176] A. Christensen, Patient adherence to medical treatment regimens: Bridging the gap between behavioral science and biomedicine, New Haven, CT: Yale University Press, 2004, p. 3.

[177] D. Riaño, F. Real, J. Lòpez-Vallverdú, F. Campana, S. Ercolani, P. Mecocci, et al., "An ontology-based personalisation of health-care knowledge to support clinical decisions for chronically ill patients," *J. Biomed. Inform.*, vol. 45, no. 3, pp. 429–446, 2012.

[178] S. Abidi and S. Abidi, "Towards the merging of multiple clinical protocols and guidelines via ontology-driven modeling," in *AIME*, 2009.

[179] S. Abidi, J. Cox, M. Shepherd and S. Abidi, "Using OWL ontologies for clinical guidelines based comorbid decision support," in *45th Hawaii International Conference on System Sciences*, 2012.

[180] B. Jafarpour and S. Abidi, "Merging disease-specific clinical guidelines to handle comorbidities in a clinical decision support setting," in *Lecture Notes in Computer Science*, vol. 7885, N. Peek, R. Marìn Morales and M. Peleg, Eds., Springer, Berlin, Heidelberg, 2013, pp. 28–32.

[181] B. Jafarpour, S. Abidi and S. Abidi, "Exploiting semantic web technologies to develop OWL-based clinical practice guideline execution engines," *IEEE J. Biomed. Health Inform.*, vol. 20, no. 1, pp. 388–398, 2016.

[182] B. Jafarpour, S.R. Abidi, W. Van Woensel, S.S. Abidi, "Execution-time integration of clinical practice guidelines to provide decision support for comorbid conditions," *Artif. Intell. Med.*, vol. 94, pp. 117–137, 2019.

[183] E. Lozano, M. Marcos, B. Martínez-Salvador, A. Alonso and J. Alonso, "Experiences in the development of electronic care plans for the management of comorbidities," in *KR4HC*, vol. 5943, D. Riaño, A. ten Teije, S. Miksch and M. Peleg, Eds., Springer, Berlin, Heidelberg, 2009, pp. 113–123.

[184] W3C, "SWRL: A semantic web rule language combining OWL and RuleML," 2004. [Online]. Available: `https://www.w3.org/Submission/SWRL/`. [Accessed April 2019].

[185] B. Motik, U. Sattler and R. Studer, "Query answering for OWL-DL with rules," *J. Web. Semant.*, vol. 3, no. 1, pp. 41–60, 2005.

[186] Apache Jena, [Online]. Available: `https://jena.apache.org/`. [Accessed Aug 2019].

[187] V. Zamborlini, M. Da Silveira, C. Pruski, A. ten Teije, E. Geleijn, M. van der Leeden, et al. “Analyzing interactions on combining multiple clinical guidelines.” *Artif. Intell. Med.*, vol. 81, pp. 78–93, 2017.

[188] S. Wilk, W. Michalowski, M. Michalowski, K. Farion, M. Hing and S. Mohapatra, “Mitigation of adverse interactions in pairs of clinical practice guidelines using constraint logic programming,” *J. Biomed. Inform.*, vol. 46, no. 2, pp. 341–353, 2013.

[189] S. Wilk, M. Michalowski, W. Michalowski, D. Rosu, M. Carrier and M. Kezadri-Hamiaz, “Comprehensive mitigation framework for concurrent application of multiple clinical practice guidelines,” *J. Biomed. Inform.*, vol. 66, pp. 52–71, 2017.

[190] M. Michalowski, S. Wilk, X. Tan and W. Michalowski, “First-order logic theory for manipulating clinical practice guidelines applied to comorbid patients: A case study,” *AMIA Annu. Symp. Proc.*, vol. 2014, pp. 892–898, 2014.

[191] M. Michalowski, S. Wilk, W. Michalowski, D. Lin, K. Farion and S. Mohapatra, “Using constraint logic programming to implement iterative actions and numerical measures during mitigation of concurrently applied clinical practice guidelines,” in *Artif. Intell. Med.*, vol. 7885, N. Peek, R. Marín Morales and M. Peleg, Eds., Springer, Berlin, Heidelberg, 2013, pp. 17–22.

[192] A. Preece, “Evaluating verification and validation methods in knowledge engineering,” in *Industrial Knowledge Management*, R. Roy, Ed., Springer, London, 2001, pp. 91–104.

[193] V. Zamborlini, R. Hoekstra, M. Da Silveira, C. Pruski, A. Ten Teije and F. Van Harmelen, “Inferring recommendation interactions in clinical guidelines,” *Semantic Web*, vol. 7, no. 4, pp. 421–446, 2016.

[194] V. Koutkias, P. McNair, V. Kilintzis, K. Andersen, J. Niès, J. Sarfati,et al., “From adverse drug event detection to prevention,” *Methods Inf. Med.*, vol. 53, no. 6, pp. 482–492, 2014.

[195] J. Jaffar and M. Maher, “Constraint logic programming: A survey,” *J. Logic Prog.*, vol. 19, pp. 503–581, 1994.

[196] M. Beccuti, A. Bottrighi, G. Franceschinis, S. Montani and P. Terenziani, “Modeling clinical guidelines through Petri Nets,” in *Artif. Intell. Med.*, vol.

5651, C. Combi, Y. Shahar and A. Abu-Hanna, Eds., Springer, Berlin, Heidelberg, 2009, pp. 61–70.

[197] M. Peleg, S. Tu, A. Manindroo and R. Altman, "Modeling and analysing biomedical processes using workflow/Petri Net models and tools," in *Medinfo*, Amsterdam, 2004.

[198] X. Tan, "Towards a formal representation of clinical practice guidelines for the treatment of comorbid patients," in *Int. Conf. on Bioinformatics and Biomedicine*, Shanghai, China, 2013.

[199] E. Rood, R. Bosman, J. Van Der Spoel, P. Taylor and D. Zandstra, "Use of a computerised guideline for glucose regulation in the intensive care unit improved both guideline adherence and glucose regulation," *J. Am. Med. Inform.*, vol. 12, no. 2, pp. 172–180, 2005.

[200] D. Riaño, "The SDA* model: A set theory approach," in *Int. Symp. on Computer-Based Medical Systems*, 2007.

[201] J. Bohada, D. Riaño and J. López-Vallverdú, "Automatic generation of clinical algorithms within the state-decision-action model," *Expert. Syst. Appl.*, vol. 39, no. 12, pp. 10709–10721, 2012.

[202] L. Piovesan, G. Molino and P. Terenziani, "Supporting multi-level user-driven detection of guideline interactions," in *HEALTHINF*, 2015.

[203] M. Peleg, Y. Shahar and S. Quaglini, "Making healthcare more accessible, better, faster, and cheaper: The MobiGuide Project," *European J. ePractice*, vol. 20, pp. 5–20, 2014.

[204] A. Hatsek, Y. Shahar, M. Taieb-Maimon, E. Shalom, D. Klimov and E. Lunenfeld, "A scalable architecture for incremental specification and maintenance of procedural and declarative clinical decision-support knowledge," *Open Med. Informat. J.*, vol. 4, pp. 255-–277, 2010.

[205] M. Grüninger and M. Fox, "Methodology for the design and evaluation of ontologies," in *Proc. of the Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[206] T. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?," *Int. J. Hum.-Comput. St.*, vol. 43, no. 5–6, pp. 907–928, 1995.

[207] U.S. National Library of Medicine, "Unified Medical Language System (UMLS)," May 2019. [Online]. Available: `https://www.nlm.nih.gov/research/umls/`. [Accessed Jul 2019].

[208] U.S. National Library of Medicine, "The UMLS Semantic Network," [Online]. Available: `https://semanticnetwork.nlm.nih.gov/`. [Accessed Jul 2019].

[209] The World Health Organization (WHO), "10th revision of the International Statistical Classification of Diseases and Related Health Problems (ICD-10)," [Online]. Available: `https://www.who.int/classifications/icd/icdonlineversions/en/`. [Accessed Mar 2019].

[210] U.S. National Library of Medicine, "RxNorm," [Online]. Available: `https://www.nlm.nih.gov/research/umls/rxnorm/`. [Accessed Jul 2019].

[211] P. Whetzel, N. Noy, N. Shah, P. Alexander, C. Nyulas, T. Tudorache, et al., " BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications," *Nucleic. Acids Res.*, vol. 39, pp. W541–5, 2011.

[212] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, et al., "The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration," *Nature Biotechnol.*, vol. 25, pp. 1251-–1255, 2007.

[213] DrugBank, [Online]. Available: `https://www.drugbank.ca/`. [Accessed May 2019].

[214] Drugs.com, [Online]. Available: `https://www.drugs.com/`. [Accessed Jan 2019].

[215] Workflow Patterns, "The Workflow Patterns," [Online]. Available: `http://www.workflowpatterns.com/`. [Accessed Jun 2019].

[216] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *The Knowledge Engineering Review*, vol. 11, no. 2, pp. 93–136, 1996.

[217] A. Hameed, A. Preece and D. Sleeman, "Ontology reconciliation," in *Handbook on ontologies*, S. Staab and R. Studer, Eds., Berlin, Springer, Heidelberg, 2004, pp. 231–250.

[218] Y. Zhe, D. Zhang and Y. Chuan, "Evaluation metrics for ontology complexity and evolution analysis," in *ICEBE*, Shanghai, 2006.

[219] H. Yao, A. Orme and L. Etzkorn, “Cohesion metrics for ontology design and application,” *Journal of Computer Science*, vol. 1, no. 1, pp. 107–113, 2005.

[220] H. Hlomani and D. Stacey, “Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: A survey,” *Semantic Web Journal*, vol. 1, no. 5, pp. 1–11, 2014.

[221] R. Djedidi and M. Aufaure, “ONTO-EVOAL an ontology evolution approach guided by pattern modeling and quality evaluation,” in *FoIKS*, Berlin, 2010.

[222] S. Tartir, I. Arpinar, M. Moore and A. Sheth, “Ontology evaluation and validation,” in *Theory and Applications of Ontology*, Amsterdam, 2010.

[223] OWL-Time, “Time Ontology in OWL,” W3C , October 2017. [Online]. Available: `https://www.w3.org/TR/owl-time/`.

[224] BioPortal, “The Drug-Drug Interactions Ontology (DINTO),” BioPortal, [Online]. Available: `https://bioportal.bioontology.org/ontologies/DINTO`.

[225] M. Peleg and S.W. Tu, “Design patterns for clinical guidelines,” *Artif. Intell. Med.*, vol. 47, no. 1, pp. 1–24, 2009.

[226] National Institute for Health and Care Excellence (NICE), “Chronic kidney disease in adults: assessment and management,” Sep 2018. [Online]. Available: `https://www.nice.org.uk/guidance/ng106`. [Accessed Mar 2019].

[227] J. Allen and G. Ferguson, “Actions and events in interval temporal logic,” *J. Log. Comput.*, vol. 4, no. 5, pp. 531–579, 1994.

[228] National Institute for Health and Care Excellence (NICE), “Type 2 diabetes in adults: management,” 04 Jun 2019. [Online]. Available: `https://www.nice.org.uk/guidance/ng28`. [Accessed Feb 2019].

[229] D. Elenius, D. Martin, R. Ford, and G. Denker, “Reasoning about resources and hierarchical tasks using OWL and SWRL,” in *Int. Semantic Web Conf.*, 2009 , pp. 795–810).

[230] V. Fortineau, T. Paviot, L. Louis-Sidney, S. Lamouri, “SWRL as a rule language for ontology-based models in power plant design,” in *IFIP Int. Conf. on Product Lifecycle Management*, 2012, pp. 588–597.

[231] C. Schneider, “Towards an Eclipse ontology framework: Integrating OWL and the Eclipse Modeling Framework,” in *TWOMDE*, 2010.

[232] T. Rahmani, D. Oberle and M. Dahms, “An adjustable transformation from OWL to Ecore,” in *MODELS*, 2010.

[233] E. Davis, “Constraint propagation with interval labels,” *Artif. Intell.*, vol. 32, no. 3, pp. 281–331, 1987.

[234] N. Mamoulis and K. Stergiou, “Constraint satisfaction in semi-structured data graphs,” in *Int. Conf.on Principles and Practice of Constraint Programming*, pp. 393–407, 2004.

[235] P. Johnson, S. Tu, N. Booth, B. Sugden and I. Purves, “Using scenarios in chronic disease management guidelines for primary care,” *Proc. AMIA Symp.*, pp. 389–393, 2000.

[236] P. Kolesa, J. Spidlen and J. Zvárová, “Obstacles to implementing an execution engine for clinical guidelines formalized in GLIF,” *Stud. Health Technol. Inform.*, vol. 116, pp. 563–568, 2005.

[237] J. Fox, M. Beveridge and D. Glasspool, “Understanding intelligent agents: analysis and synthesis,” *AI Communications*, vol. 16, no. 3, pp. 139–152, 2003.

[238] M. Hing, M. Michalowski, S. Wilk, W. Michalowski and K. Farion, “Identifying inconsistencies in multiple clinical practice guidelines for a patient with comorbidity,” in *IEEE Int. Conf. on Bioinformatics and Biomedicine Workshops*, 2010.

[239] I. Meiri, “Combining qualitative and quantitative constraints in temporal reasoning,” *Artif. Intell.*, vol. 87, pp. 343–385, 1996.

[240] National Institute for Health and Care Excellence (NICE), “Depression in adults: recognition and management,” 28 Oct 2009. [Online]. Available: `https://www.nice.org.uk/guidance/cg90`. [Accessed Dec 2018].

[241] National Institute for Health and Care Excellence (NICE), “Hypertension in adults: diagnosis and management,” 2011. [Online]. `Available:https://www.nice.org.uk/guidance/cg127`. [Accessed Feb 2019].

[242] S. Dumbreck, A. Flynn, M. Nairn, M. Wilson, S. Treweek, S. Mercer, et al., “Drug-disease and drug-drug interactions: Systematic examination of recommendations in 12 UK national clinical guidelines,” *BMJ*, vol. 350, p. h949, 2015.

[243] National Institute for Health and Care Excellence (NICE), "Chronic kidney disease in adults: assessment and management," 23 Jul 2014. [Online]. Available: `https://www.nice.org.uk/guidance/cg182`. [Accessed Jan 2019]

[244] National Institute for Health and Care Excellence (NICE), "Atrial fibrillation: management," 18 Jun 2014. [Online]. Available: `https://www.nice.org.uk/guidance/cg180`. [Accessed Mar 2019].

[245] J. Raad and C. Cruz, "A survey on ontology evaluation methods," in *Proc. of the Int. Conf. on Knowledge Engineering and Ontology Development (KEOD)*, Lisbon, Protugal, 2015.

[246] A. Tolk, S. Diallo and C. Urnitsa, "Applying the levels of conceptual interoperability model in support of integratability, interoperability, and composability for system-of-systems engineering," *Journal of Systems, Cybernetics, and Informatics*, vol. 5, no. 5, pp. 65–74, 2007.

[247] J. Brank, M. Grobelnik and D. and Mladenic,"A survey of ontology evaluation techniques," in *Proc. of the Conf. on Data Mining and Data Warehouses(SiKDD)*, Ljubljana, Slovenia, 2005.

[248] O. Bodenreider, B. Smith, A. Kumar and A. Burgun, "Investigating subsumption in SNOMED-CT: An exploration into large description logic-based biomedical terminologies," *Artif. Intell. Med.*, vol. 39, no. 3, pp. 183–195, 2007.

[249] P.I. Dissanayak, T.K. Colicchio and J.J. Cimino, "Using clinical reasoning ontologies to make smarter clinical decision support systems: A systematic review and data synthesis,"*J. Am. Med. Inform. Assoc.*, vol. 27, no. 1, pp. 159–174, 2020.

[250] Eclipse Modeling Framework (EMF), "Eclipse Modeling Framework Model Validation," [Online]. Available: `https://www.eclipse.org/emf-validation/`. [Accessed May 2019].

[251] F. Parreiras, S. Staab and A. Winter, "On marrying ontological and metamodeling technical spaces," in *The 6th Joint Meeting on European Software Eng. Conf. & the ACM SIGSOFT Symp. on the Foundations of Software Eng.*, 2007.

[252] M. Michalowski, M. Hing, S. Wilk, W. Michalowski and K. Farion, "A constraint logic programming approach to identifying inconsistencies in clinical practice guidelines for patients with comorbidity," in *Artif. Intell. Med.*, vol.

6747, M. Peleg, N. Lavrač and C. Combi, Eds., Springer, Berlin, Heidelberg, 2011, pp. 296–301.

[253] E. Montiel-Ponsoda, D. Vila Suero, B. Villazón-Terrazas, G. Dunsire, E. Escolano Rodríguez and A. Gómez-Pérez, "Style guidelines for naming and labeling ontologies in the multilingual web.," in *Int. Conf. on Dublin Core and Metadata Applications*, Hague, Holanda, 2011.

[254] M. Peleg, Y. Shahar, S. Quaglini, T. Broens, R. Budasu, N. Fung, et al., "Assessment of a personalized and distributed patient guidance system," *Int. J. Med. Inform.*, vol. 101, pp. 108–130, 2017.

[255] S. Quaglini, Y. Shahar, M. Peleg, S. Miksch, C. Napolitano, M. Rigla, et al., "Supporting shared decision making within the MobiGuide project," *AMIA Annu. Symp. Proc.*, vol. 2013, pp. 1175-–1184, 2013.

[256] G. Elwyn, C. Dehlendorf, R. Epstein, K. Marrin, J. White and D. Frosch, "Shared decision making and motivational interviewing: Achieving patient-centered care across the spectrum of health care problems," *Ann. Fam. Med.*, vol. 12, no. 3, pp. 270–275, 2014.

[257] D. Sánchez, M. Batet, D. Isern and A. Valls, "Ontology-based semantic similarity: A new feature-based approach," *Expert Syst. Appl.*, vol. 39, pp. 7718–7728, 2012.

[258] E. Bilici and Y. Saygin, "Why do people (not) like me?: Mining opinion influencing factors from reviews," *Expert Syst. Appl.*, vol. 68, pp. 185–195, 2017.

[259] A. Bottrighi, L. Piovesan and P. Terenziani, "Supporting the distributed execution of clinical guidelines by multiple agents," *Artif. Intell. Med.*, vol. 98, pp.87–108, 2019.

[260] G. Elwyn, S. Laitner, A. Coulter, E. Walker, P. Watson and R. Thomson, "Implementing shared decision making in the NHS," *BMJ*, vol. 341, p. c546, 2010.

[261] E. Bilici Özyiğit, "MuCIGREF (Multiple Computer-interpretable Guideline Representation and Execution Framework) Ontology", Available: `https://webprotege.stanford.edu/#projects/edf817ad-b084-4da4-8b6e-c5f3f6e7e8a3/edit/Classes`.