From GeneWeaver to Agmial

K. Bryson[1], M. Luck[2], M. Joy[3], D.T. Jones[4], P. Nicolas [1], P. Bessières [1], J-F. Gibrat [1]

1. Unité Mathématique Informatique et Génome, INRA, Versailles 78026, France
2. Electronics and Computer Science, University of Southampton, SO17 1BJ, UK
3. Department of Computer Science, University of Warwick, CV4 7AL, UK
4. Department of Computer Science, University College London, WC1E 6BT, UK

ABSTRACT

We report on our experiences of developing two multi-agent systems for bioinformatics called GeneWeaver and Agmial. The concepts behind these systems are described and the results of experimenting with different agent communities are given.

The GeneWeaver system provides one possible view of an architecture which may be used to deal with organizing and managing bioinformatics data and methods in a distributed, dynamic environment. Experiments with the GeneWeaver system reveal benefits both in terms of keeping data up-to-date, and also in terms of analysis method accuracy.

However, one clear limitation of the GeneWeaver system is that it has never actually been deployed as an open system between laboratories. A number of reasons exist for this lack of true open deployment. These include the lack of mechanisms to deal appropriately with security and trust between agents, and the lack of a standard, widely adopted agent communication language for inter-agent communication.

Some of these aspects are considered in the second system, called Agmial, which is currently under development. It uses some of the standards and technologies which have arisen since the development of GeneWeaver to provide a system which, at least theoretically, could provide the basis for an open community of bioinformatics agents. Its approach is one based on the Web services framework, each agent being deployed as a Java servlet, using XML Protocol for inter-agent communication and HTTP for human-agent communication.

1. INTRODUCTION

Imagine biologists studying a particular human neurodegenerative disease called Huntington's disease. They have a cell-culture model of the disease which shows decreased cell death compared to that of a control when a particular neurotrophic factor, IGF-1, is applied [12].

The vital question, since it may reveal clues for a treatment, is why does this factor decrease cell death ? They decide to investigate and start intensively crawling through the literature, querying various molecular biology and disease databases, and carrying out computational analysis on relevant sequences. After a number of weeks they have a huge repository of knowledge on this specific topic, now insight is required to relate their findings to established results and possibly determine a link.

Now picture a world, in the future, where the biologist goes to a computer and puts in the query: 'Discover general links between IGF-1 and Huntington's disease'. The next day they receive a message saying that a huge repository of interesting data is sitting on their disk. Their disk contains a list of all the tasks which were carried out for them, and the goals which these tasks were trying to accomplish. Among these tasks are a number of database searches of IGF-1 and Huntington's disease to establish a base of knowledge. A defective gene is followed to a mutant protein, which is then analyzed using a variety of sequence analysis tools. A signaling pathway database establishes that IGF-1 results in the activation of a number of kinases, which receive further attention. Unknown to the biologist, the data and tools are distributed world-wide, together with the programs which embody the knowledge of how to effectively use the resources to accomplish goals. All the biologist sees is a structured summary of the findings with justifications for all the conclusions reached, for they are still slightly suspicious of all this technology and want to manually check anything useful.

Now back to the present. It has been recognized that the word 'agent' is starting to have a buzzword status, often accompanied by promises of amazing feats such as the above [15]. This buzzword status is slightly unfortunate, since it overshadows some of the real contributions that agent research is making in developing cooperative software, which we believe will play an important role in the above vision of integrated biological resources. The concepts which agent-based software promotes seem ideal for this purpose: tasks driven to accomplish specific goals, autonomy leading to flexibility and robustness, cooperation between open distributed components, using meta-data to discover and use each others abilities to accomplish specific goals. Whether these distributed components are eventually called 'agents', or semantic Web services, or something else should not concern us, the programming paradigm is agent-based.

It is clear that a large number of research disciplines will be involved in accomplishing the above vision. Prominent amongst these are studies involving the semantic integration of heterogenous biological databases [14]. Within the data integration field, a significant role is being played by

NETTAB 2O02 Bologne, Italy

biological ontology development, such as the Gene Ontology[23] and TAMBIS [5, 6]. These permit biological data to be given an accurate semantic meaning, facilitating their comparison and integration.

The Semantic Web [7, 24] is also playing a significant role, and languages such as DAML-S [2] are being developed to encode the above ontologies into a format suitable for the Web. These ontologies provide semantic meaning to the XML encoded data which provides the input and output of Web services, thus permitting the functionality of Web services to be described. These are technologies that should bring the world-wide integration of bioinformatics resources much closer.

The Web services framework does not put any restrictions on the behavior of the distributed components themselves. This is reasonable since this framework is for generic use, and should not prescribe a too restrictive model for the architecture. Services simply use other services knowing that if certain inputs are provided, then the required result may be obtained.

We take the view that it is beneficial to place further restrictions on the behavior of the distributed components for the development of large open systems. Restricting the behavior of the components makes it more likely that they will successfully interact with each other. We believe that adopting agent-based design principles for the distributed components aids the development of these types of systems.

In particular, the distributed components, which we call agents, should communicate using a suitable agent communication language (ACL), which prescribes particular semantic meaning to the communication acts (or speech acts). Also certain characteristics should be promoted within the agents [25]: reactivity to changes in the external environment, proactiveness at carrying out tasks to accomplish internally generated goals and cooperation within an agent community.

These multi-agent concepts have been used recently to develop systems for the integration, analysis and annotation of genomic information: including DECAF [10, 11] and EDITToTrEMBL [19, 20].

In this paper we examine a community of agents developed under the GeneWeaver project [8, 9], and the benefits which might be achieved by allowing bioinformatics data resource and analysis methods to be integrated within such a framework. We try to steer clear of the complexities of semantic data integration, as mentioned above, and set up a simple framework which permits sufficient experimentation with these concepts to answer the basic question 'Do we gain any benefits from using an agent-based approach ?'

Further to this, we then introduce another system, developed under the Agmial project, the intention of which is to field-test the ideas developed with GeneWeaver for the annotation of a number of *Lactobacillus* genomes.

The paper is organized as follows. First we provide a broad picture of the community of agents present within the GeneWeaver system. The communication language and its semantics are then described before giving some scenarios of interaction. An experiment demonstrating the increased performance from the community of agents when cooperation is present is then described.

This provides evidence that the adoption of such ideas in a truly open system may provide substantial benefits. GeneWeaver lacks certain aspects which are essential for true open deployment between laboratories. The system developed under the Agmial project is hoping to fill in some of these missing details using various Web technologies.

2. AN OVERVIEW OF GENEWEAVER

GeneWeaver is a multi-agent system aimed at addressing concerns with the management of data and analysis methods for bioinformatics. It comprises a community of agents that interact with each other, each having a particular role, in an overall effort to automate processes involved in bioinformatics. In particular, the community was targeted at genome annotation, but the system should not really be viewed as satisfying a single need, with each agent being able to deliver its own expertise at solving particular problems.

Figure 1 illustrates the overall perspective of GeneWeaver in that it contains the different classes of agents and shows the principal interactions between them. These interactions are built dynamically, as will be discussed in Section 3.4.

At the top is the Broker Agent, a *facilitator* which provides a yellow-pages service of meta-data about other agents in the community. Each agent is required to register information about itself with the broker to be recognized within the community. Conversely, agents can query the broker for information about other agents in the community.

At the left side, Pdb Agent, Pir Agent and Swiss Agent manage the protein structure and sequence databases indicated by their names. They maintain their local versions of these databases up-to-date using suitable Internet sites. They also map the data contained in these databases into a relational representation which is suitable for querying and manipulation using the inter-agent communication language, BAL, which will be outlined in Section 3.2. These are primary database agents, with one agent for each type of database. In general they wrap information sources and in an extended community may deliver a variety of information, only constrained by the encoding of the agent language.

In the center there is the *non-redundant database* agent (Nrdb Agent). Using the information contained in the Broker Agent, the Nrdb Agent subscribes to all the protein sequence data, of a given quality, currently held by agents present in the community. In this example it subscribes to protein sequences held by the primary database agents. From this data it builds a non-redundant database, currently using a definition of equal residue strings for redundancy. Clearly, different Nrdb Agents may be constructed with different definitions of what community data to integrate and the definition of redundancy. In more general terms, these agents can be viewed as integration agents, obtaining information from different sources, filtering and combining it.

At the right side of Figure 1 are the Psi-Blast and Memsat Agents. These apply different bioinformatics analysis methods, in this case PSI-BLAST [1] and MEMSAT [13, 17]. These are just two of a potentially large number of calculation agents. They wrap pre-existing methods so that they can be provided to the community using the inter-agent language BAL. These also advertise meta-data about their services with the broker so that other agents may make use of them. But this is not the only purpose of these agents. The agent-based methodology promotes that agents should be proactive, dynamic and autonomous. These agents should not only provide an interface to allow the calculation method to be employed, but they should proactively manage under-

lying aspects of the service.

So like the primary database agents, the calculation agents also attempt to automate the 'updating' of themselves. This updating usually takes two forms: updating any underlying databases which an analysis tool depends upon, and retraining the analysis tool with newly available data. In these respects, the PSI-BLAST AGENT subscribes to proteins managed by the NRDB AGENT, and it updates its underlying PSI-BLAST databases when it decides a new version of the database is justified (automatic rules for this decision process are not dealt with in this paper). Potentially the PSI-BLAST AGENT could provide searching over any database held by the community, but in this example it subscribes to only one. In a similar manner, the MEMSAT AGENT subscribes to the SWISS AGENT for information about transmembrane proteins contained in SWISSPROT. It uses this information to retrain itself when it determines this is justified. The dynamic updating of these agents will be examined in the experimental communities given in Section 4.

A second consideration when managing these analysis tools is the presence of dependencies between the calculation methods. For example, the MEMSAT method uses an internal version of the PSI-BLAST method during its processing. The version of PSI-BLAST used, and its underlying database, are hidden within the MEMSAT method itself. Two problems arise from this: the embedded tools used will not be automatically updated, and subtle inconsistencies may arise since different versions are implicitly being used within the wider community. In order to deal with this consistency problem, the calculation agents attempt to make these dependencies explicit. Hence the MEMSAT AGENT interacts with the PSI-BLAST AGENT whenever it requires to apply the PSI-BLAST method.

Finally, at the top of the figure, is the ECOLI GENOME AGENT. This is one of a number of genome agents, each one managing the genome information for a particular organism, and with a particular configuration for its analysis. Again they try to be autonomous, with regular automatic updating of the genome data from suitable Internet sites, and automatic reanalysis by the community when changes occur.

3. AGENT COMMUNICATION LANGUAGE

3.1 Introduction

The agent communication language (ACL) is at the heart of constructing cooperative communities of agents. It promotes a single shared language for interaction, in constrast to distributed systems which are based on a collection of remote procedure calls.

In general, each ACL message uses a *performative*, such as **tell** or **ask**, to indicate the attitude of the agent to the content of the message in accordance with speech-act theory [4]. So an ACL message using the **tell** performative indicates that the agent sending the message wants to tell its content to the agent receiving the message. With the shared underlying meaning given to the performatives, the community can have a global understanding of each others motives. For example, under KQML semantics [16], the agent sending the **tell** believes the content and also wishes the other agent to believe it. Each agent is responsible for mapping the modalities implicit within the ACL, such as belief, into suitable functionality for its context. This underlying shared language provides the basis for agent cooperation.

At a higher level of communication are conversations [15], which involve the sending and receiving of a number of messages following a particular conversation protocol, with a particular purpose for the conversation.

3.2 The BioAgent Language (BAL)

Agents in the GeneWeaver community communicate with each other using the BioAgent Language (BAL), which is based on KQML [16], with similar performatives, but is both dramatically reduced in range, and tailored to the particular kinds of interactions used within GeneWeaver. A more detailed description of the language can be found in [9].

BAL *interactions* lie at the highest level of communication, and consist of a number of messages being passed between agents following a fixed protocol for a particular type of interaction. These are prescriptive conversations, in a similar manner to that considered by InfoSleuth [21].

An example of a BAL *message* is as follows:

```
Sender:  Swiss008a6641ab
Receiver:  Broker008a6641ab
Transport:  rmi
Language:  BACL
Ontology:  BAMO
Perform:  register
Ref:  Swiss008a6641ab_0
Content:
AgentInfo(
ID = 0,
OWNER = "Swiss008a6641ab",
TYPE = "PrimaryDB",
MOD_TIME = 2002-06-08T18:24:15+0100,
LOCAL_MOD_TIME = 2002-06-08T18:24:15+0100,
AGENT_URL = "http://mig171.versailles.inra.fr/swiss/",
DESCRIPTION = "Swiss agent which manages the SWISSPROT
database."
).
```

This message is sent from the SWISS AGENT to the BROKER AGENT using the 'rmi' transport protocol. Messages within GeneWeaver may be encoded in a variety of transport protocols, promoting 'open' communication. The *language* and *ontology* fields specify the language which is used to encode the content of the message and the semantic meaning of any symbols employed. In this study a simple prototype language and ontology are employed since it is not our overall objective to study data representation.

The performative of this message is **register**, indicating the intention that the SWISS AGENT wants to register with the BROKER. In doing so, it includes meta-data about itself, which the BROKER may use to determine whether to allow it to register with the community. The performatives used within BAL are provided in Table 1. These are fairly standard for ACLs, the only exceptional one being **derive**, which indicates that the sender wants the other agent to use its skills and plans to derive suitable results matching the query. When using **derive** the sender does not know how the receiver will derive the required result, this is in contrast to the **do** performative, for which the sender is required to know precisely the structure of the input and output parameters of the specified skill.

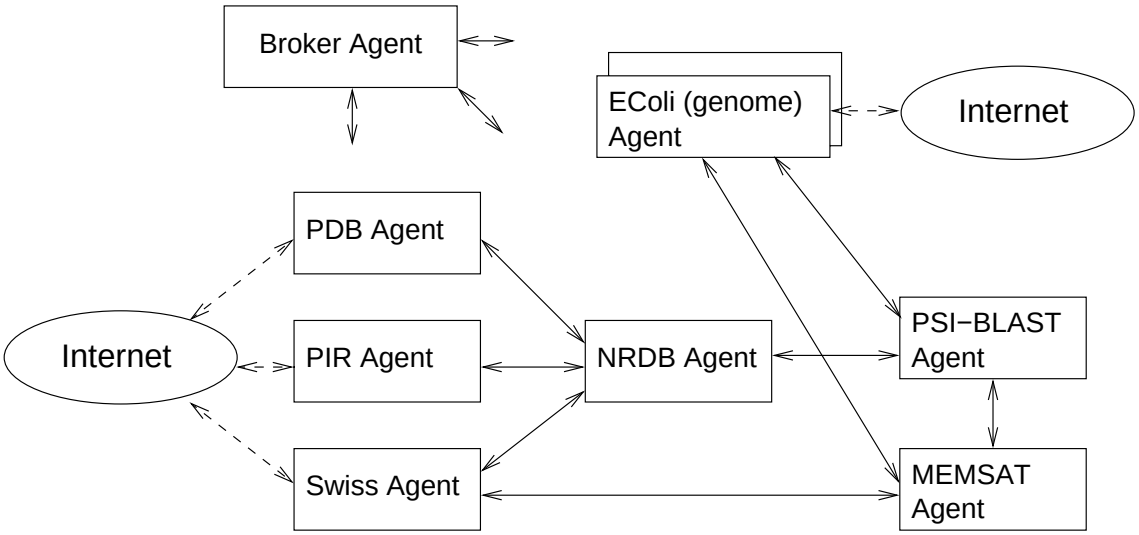

Figure 1: GeneWeaver example agent community. Dashed lines indicate agents communicating with the Internet using HTTP or FTP protocols. Solid lines represent the main communication paths between the agents using an agent communication language. The arrows from the BROKER AGENT indicate it communicates with all other agents.

Performative	Description
`ask`	Sender wishes to know if any data exist in the receiver's data store (or meta-store) which matches the given data query.
`tell`	Indicates that the given data exist in the data store (or meta store) of the sender.
`deny`	Indicates that the data given no longer exist in the data store (or meta store) of the sender.
`subscribe`	Sender wishes to subscribe to data (or meta-data) matching the given query and be informed about future changes.
`unsubscribe`	Remove the subscription.
`do`	Sender wishes the receiver to use the specified skill with the given content as input.
`derive`	Sender wishes the receiver to derive data satisfying the given data query using its plans and skills.
`ok`	Confirms an action has been successful.
`error`	Terminates a conversion since either an agent did not understand the contents or the message protocol was not followed.
`sorry`	Terminates a conversion. The agent understood the message but did not have any response to it.
`register`	Registers an agent with the broker. This provides the broker with a symbolic name and a description of the agent.
`unregister`	Cancels a register (and any commitments of the agent).

Table 1: Description of BAL performatives.

3.3 Meta-Data and Domain Data Employed

The community of agents could not interact using just the meta-data AgentInfo, more meta-data is required for successful interaction. The BioAgent Meta-Ontology (BAMO) provides meaning for the data outlined in Table 2. This provides information about the transport protocols, skills and motivations of the different agents. This prescribes a particular meaning for the structural components from which GeneWeaver agents are constructed. More detail about the actual internal architecture of GeneWeaver agents can be found in [8].

The TransportInfo, LanguageInfo and OntologyInfo provide information about what transport protocols the agent supports (RMI, CORBA, etc), what content languages it can handle and what ontologies for data it can understand. Two agents which wish to communicate require to establish a common transport protocol for the messages together with a common content language and ontology. These cater for extensions to the community to include new transport protocols, languages and ontologies.

RequesterInfo and ProviderInfo relate to the types of interaction which an agent can engage in, both as a requester for an interaction and a provider of an interaction. These permit an agent to determine a common conversation policy to interact effectively with another agent.

The SkillInfo and PlanInfo provide information about the local skills and plans which an agent possesses. Whereas MotivationInfo provides information about the local motivations the agent holds. Motivations essentially describe internal goal-directed behavior of the agent, which might range from simply updating databases using suitable Web sites, to the complex analysis of genome data. Essentially, motivations generate local goals which initiate skills, plans and interactions to satisfy these goals.

The WebSiteInfo is used to describe the outside environment known to the agent, simply consisting of relevant Web sites and data that they may provide.

Finally, RelationInfo, ActivePlanInfo and ActionInfo give information about the current ongoing interactions which the agent has with other agents, together with the current active instances of plans and skills employed by the agent.

In a similar way the BioAgent Domain Ontology (BADO) is a simple ontology used to prescribe shared semantic meaning to problem domain data items such as protein sequences and transmembrane topologies.

BAMO type	Property of agent
AgentInfo	General features of the agent.
TransportInfo	Transport protocols supported.
LanguageInfo	Content languages supported.
OntologyInfo	Content ontologies supported.
RequesterInfo	Interaction types the agent can request.
ProviderInfo	Interaction types the agent can provide.
SkillInfo	Skills which may be employed.
PlanInfo	Plans which may be employed.
MotivationInfo	Motivations held by the agent.
WebSiteInfo	Web sites known by the agent.
RelationInfo	Current ongoing relations with agents.
ActivePlanInfo	Current ongoing instances of plans.
ActionInfo	Current ongoing actions being carried out.

Table 2: Types of meta-data present in the BioAgent Meta Ontology.

3.4 BAL Interactions

We will briefly describe the types of BAL interactions to provide a context for the explanation of the example communities given in Section 4. All interactions take place between two agents: the requester which initiates the interaction and the provider which satisfies it. A more detailed explanation is given in [9].

3.4.1 Register Interaction

Each agent (other than brokers) has a motivation to register with a broker, which will result in a **register** message being sent providing AgentInfo meta-data about the agent. The broker in turn may demand more information from the agent before permitting registration by replying with an **ok** message. Finally the requester agent can send an **unregister** message to terminate the interaction.

3.4.2 Data Interactions: Query, Subscribe, Tell

A number of types of data interaction exist within the community. The Query interaction permits a single-shot query for information held by the other agent, either meta-data or domain data, using the **ask** performative. The Subscribe interaction is similar but the requester will be kept informed about changes in the data using **tell** and **deny** performatives, until they send an **unsubscribe** performative to terminate the relationship.

For instance the NRDB AGENT may subscribe to the BROKER AGENT to be kept informed about new protein databases which appear in the community. When the SWISS AGENT registers with the broker, the broker would then send a **tell** message to the NRDB AGENT informing it about the new SWISSPROT database present.

Both the Query and Subscribe interactions are solicited forms of data sharing. The Tell interaction permits agents to engage in unsolicited telling of information to other agents without the data being explicitly demanded.

3.4.3 Action Interactions: Do and Derive

The Do interaction is initiated with a **do** performative with the content providing both the meta-data of the skill to perform, and also the ordered input data for this skill. If successful, this will result in an action being generated within the provider agent to carry out the given task.

The Do interaction requires the requester agent to completely know the input data and its semantics for the skill involved, in addition to knowing how to interpret the resulting data. The Derive interaction permits agents to be given tasks in a more declarative manner. In this case the requesting agent wishes to derive data which satisfies a query given in the content, without actually knowing how the provider agent will accomplish this. In addition to the query, the requester agent may also attach data which it believes may be relevant. If the provider agent requires additional data from the requester, it can obtain it using a Query interaction.

For instance, the ECOLI GENOME AGENT can use the following message to derive the membrane topology of a given protein without actually knowing how the MEMSAT AGENT carries out the task:

```
Sender: EColi008a6641ab Receiver: Memsat008a6641ab
Transport: rmi Language: BACL Ontology: BADO
```

```
Perform:  derive
Ref:  EColi008a6641ab_42
Content:
MemTopol(
N_LOC?, OTHER_LOC?, N_LOC_SIDE?,
SPANS?, QUALITY?, DERIVED?,
SEQ_REF = 8_EColi008a6641ab
).
```

4. EXAMPLE COMMUNITIES

From the previous discussion it is hoped that the general architecture of the GeneWeaver agent community, and how the agents interact, has been established. In this section we describe an experiment with the community shown in Figure 1. This should give an overview of the cooperation that takes place and also provide evidence, by way of results, of some benefits of this cooperation.

4.1 Method

Four independent communities of agents were set up, each containing the agents shown in Figure 1. The first community of agents worked in a way that we believe provides cooperative interaction, reactive updating and fine-grained integration. What do we mean by fine-grained integration? Usually systems carry out data integration once different analysis methods have carried out their calculations, with each of the methods being treated as essentially a black box. Little consideration is given to what data is being used to train the methods, what data is used during the calculation in the form of dependent databases and also what analysis methods may be employed during the calculation. By fine grained integration we mean that the methods should train themselves using data derived from the community, the databases they employ should be obtained from the community, and if any internal analysis tools are used they should be provided by the community to ensure that they also have their dependencies consistent with the community view. We call this approach the self-consistent community model, and believe that it should enhance consistency when integrating analysis results.

How does this relate to our community of agents? In this community the Psi-Blast Agent is configured to subscribe to the Nrdb agent's non-redundant database, which in turn is constantly kept up-to-date by subscriptions to the primary database agents. Hence the database of Psi-Blast is kept up-to-date with the 'community view'. The Memsat Agent subscribes to the Swiss Agent for all known transmembrane proteins, which it uses to retrain itself whenever the Swiss Agent records new transmembrane structures.

Also the Memsat Agent employs the Psi-Blast Agent, using the Do interaction, since there is a strong coupling between these methods and it knows exactly what inputs are needed and what outputs it wants. In this way it derives protein profiles using PSI-BLAST which are up-to-date with respect to the community data available.

Finally the EColi Genome Agent uses the Psi-Blast and Memsat Agents to predict protein homologues and transmembrane information for the proteins contained in the *E. coli* genome it is managing. It does this using the Derive interaction since it has no information about the details of using the PSI-BLAST or MEMSAT skills. This allows the genome agent to employ a number of agents registered with the broker as able to derive various types of information for protein sequences, without actually knowing the exact details about how to use the skills.

So this is the first community of agents. The other three communities are simply set up with less cooperation between the community members. A less cooperative Memsat agent exists in the second community, which does not retrain itself as new data arises in SWISSPROT. The third community consists of a non-reactive Psi-Blast Agent which does not update the database it employs as the non-redundant database is updated. In the fourth community, neither the Memsat or Psi-Blast Agents react to changes in the community, either by retraining themselves or updating their underlying data.

We assess each community's ability to predict protein homologues and transmembrane results for the proteins within the *E. coli* genome. In doing this study, we have had to accelerate time, to permit significant changes to be observed due to updating. Hence we took the current version of SWISSPROT and by using the sequence last modified date we created a Jan. 1998 version of the database and a Jan. 2002 version. We started the systems with SWISSPROT, PIR and PDB data from 1998, and then after some time presented the new 2002 data for SWISSPROT and examined how each system reacted.

4.2 Results and Discussion

Virtually all homologue results in the EColi Agent were modified for the communities with the reactive Psi-Blast Agent, no updates to the homologue data occurred in the other communities. Figure 2 shows a comparison for the number of homologues detected for proteins in 2002 compared with 1998. Only proteins with less than 20 homologues detected in 1998 are shown, since these reflect proteins where new homologues detected are most important. Note in particular that some proteins which had no homologues detected in 1998, had over 40 homologues detected in 2002.

Somewhat surprizingly, not a single change in the predicted transmembrane topology results occurred, even though the PSI-BLAST profiles changed. Neither did the communities which retrained the MEMSAT method using new SWISSPROT membrane data show any changes in their transmembrane predictions even though the underlying scoring matrices were updated. In retrospect this may not be so surprizing, since the MEMSAT method is based on average amino-acid scores and hence small changes in the input parameters may be averaged away and not significantly affect the fairly strong transmembrane signals.

This highlights an important observation. For some methods it is important that the underlying data is updated regularly, since the outputs are sensitive to these input parameters, for example iterative methods such as PSI-BLAST. For some other methods the outputs are much less sensitive to small variations in the inputs. Hence policies need to be established, for the agents, based on knowledge of what magnitude of changes in the underlying data justifies the updating or retraining of the method.

5. FUTURE WORK - AGMIAL

The GeneWeaver project has developed and demonstrated the feasibility and advantages of using multi-agent systems within the field of bioinformatics. It has focused on the gains

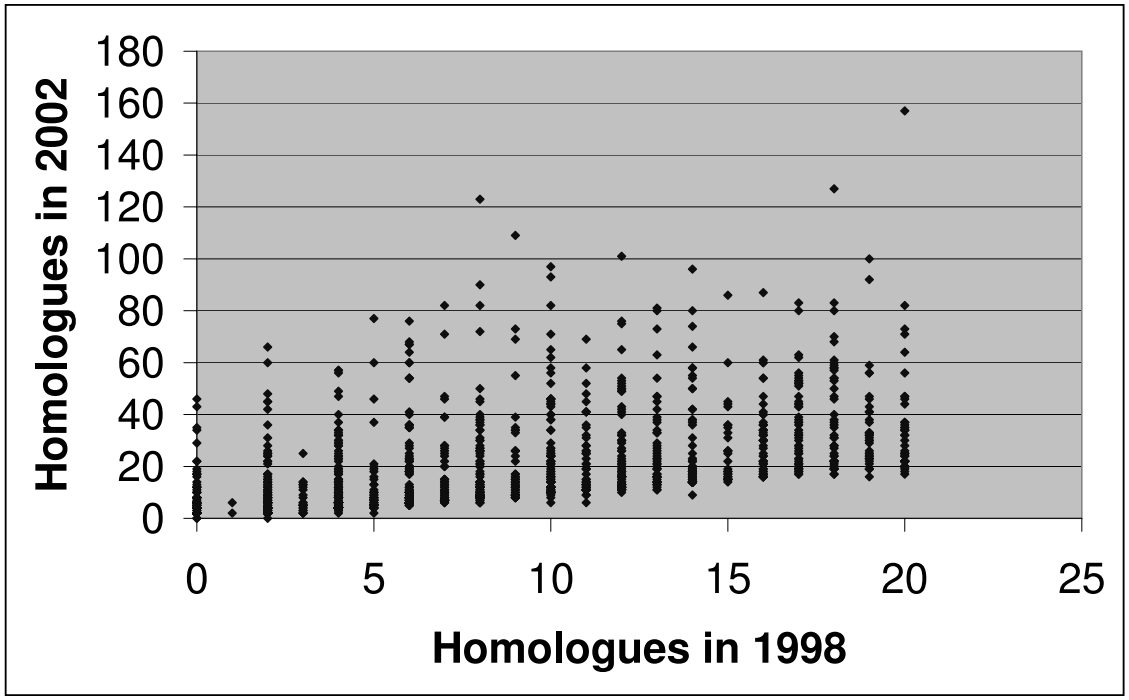

Figure 2: Comparison of the number of significant homologues detected in 2OO2 against those detected in 1998 for proteins where less than 20 homologues were initially detected.

to be obtained when different components can inter-operate and improve their abilities with time, by reactively updating themselves.

This system, however, only employs a small number of calculation agents. Within an open environment, in which third-parties can deploy a large number of information and calculation agents, it is envisaged that huge benefits for bioinformatics may arise. In such an environment, where many untested analysis tools and data sources could become available, additional mechanisms may be needed for maintaining accurate meta-data about the different services. Additional types of agents, for instance data verifying and benchmarking agents, may need to be deployed. One could imagine a number of types of benchmarking agents for different types of derived data. For instance, a transmembrane benchmark agent would test and record, using an experimentally verified test set, the sensitivity and selectivity of different agents which advertise their abilities at deriving transmembrane results. The benchmarking agents could also determine more detailed information for particular skills, for instance some transmembrane prediction methods which generally produce good results show particularly poor performance for G-coupled receptors [18]. For accurate benchmarking results, it is envisaged that calculation agents would need to provide facilities for other agents to create specific training sets so that benchmarking agents could knifejack their results.

Although GeneWeaver has demonstrated benefits from cooperation, a number of reasons exist why it could not be deployed in an open environment. Foremost amongst these is lack of security and trust. GeneWeaver does not currently have any mechanism to ensure that the identities of other agents engaged in an interaction are accurate, neither does it have any policy on whether the data or methods of external agents should be trusted. Another aspect is that GeneWeaver uses its own languages for inter-agent communication and data representation. Clearly widely-adopted standards need to be employed for true open deployment. GeneWeaver has served us well to demonstrate the concepts, but a radical shift in the technology employed seems to be required for the further development of these ideas. The Web standards as outlined in the introduction seem to provide technologies which can accomplish this vision.

The Agmial project involves a federation of loosely-coupled laboratories at INRA, France, all of which are interested in the analysis of different *Lactobacillus* bacteria and their genomes. It is within this framework that our second multi-agent system is being developed. This system takes the GeneWeaver ideas further by using technologies which have been developed since GeneWeaver was conceived. In particular, Agmial agents are being deployed using the Java servlet technology and the Apache Tomcat [3] servlet container. This provides a robust and flexible system for the management of these agents. Also dynamic human-agent interaction is provided via the Web by simply using HttpServlets. Agent-agent interaction is provided by using the Web services framework and allowing the agents to communicate using the widely adopted XML Protocol standard [26].

A facilitator service within this community is provided by UDDI, in a similar way to the broker in GeneWeaver. It is also hoped that the current advances in Web based ontologies, their languages and reasoning systems, will provide benefits for this project.

This approach also very easily provides secure and trusted inter-agent interaction by using secure sockets and electronic signature authorization between agents, together with the

basic password mechanism for human-agent interaction, which are all supported by Tomcat.

Finally this project is employing, to a greater extent, the agent-based principle of wrapping pre-existing software components using a thin layer which provides a suitable interface between the component and the speech-acts of the agent language. In fact we wrap the entire Artemis genome annotation system [22] using an XML Protocol layer within the framework. In a sense, we have given Artemis an attitude to its data, when different speech-acts are employed.

An initial prototype of the system is currently under deployment at a number of different laboratories within INRA. For us, we hope a first tentative step towards true inter-laboratory open systems.

5.1 Acknowledgements

The GeneWeaver work described in this paper was funded by the Bioinformatics programme of the UK's EPSRC and BBSRC.

The Agmial work is supported by INRA and an Individual Marie-Curie Fellowship awarded to KB by the European Union.

We thank people involved in the Agmial project at the laboratory of Dr Monique Zagorec within the Unité Flore Lactique et Environnement Carné, and also those at the laboratory of Dr Emmanuelle Maguin in Génétique Microbienne, both at the Jouy-en-Josas campus of INRA. In particular we wish to thank Dr Stéphane Chaillou in the former laboratory, and Dr Maarten van de Guchte in the latter, as well as Mark Hoebeke and Dr Hélène Chiapello at the Unité Mathématique Informatique et Génome, all for extensive and useful discussion about the project.

6. REFERENCES

[1] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25:3389–402, 1997.

[2] A. Ankolenkar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, and K. Sycara. DAML-S: Web service description for the semantic web. (Presented at The First International Semantic Web Conference (ISWC)), 2002.

[3] Apache Tomcat. http://jakarta.apache.org/tomcat/.

[4] J. Austin. *How To Do Things With Words*. Harvard University Press, 1975.

[5] P. Baker, C. Goble, S. Bechhofer, N. Paton, R. Stevens, and A. Brass. An ontology for bioinformatics applications. *Bioinformatics*, 15:510–20, 1999.

[6] P. G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens. TAMBIS: Transparent access to multiple bioinformatics information sources. In J. Glasgow, T. Littlejohn, F. Major, R. Lathrop, D. Sankoff, and C. Sensen, editors, *6th Int. Conf. on Intelligent Systems for Molecular Biology*, pages 25–34, Montreal, Canada, 1998. AAAI Press, Menlo Park.

[7] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[8] K. Bryson, M. Luck, M. Joy, and D. Jones. Applying agents to bioinformatics in GeneWeaver. In *Cooperative Information Agents IV*, volume 1860 of *LNAI*, pages 60–71, 2000.

[9] K. Bryson, M. Luck, M. Joy, and D. Jones. Agent interaction for bioinformatics data management. *Applied Artificial Intelligence*, 15(10):917–947, 2001.

[10] K. Decker, S. Khan, C. Schmidt, and D. Michaud. Extending a multi-agent system for genomic annotation. In *Cooperative Information Agents*, pages 106–117, 2001.

[11] K. Decker, X. Zheng, and C. Schmidt. A multi-agent system for automated genomic annotation, 2001.

[12] S. Humbert, E. A. Bryson, F. P. Cordelires, N. C. Connors, S. R. Datta, S. Finkbeiner, M. E. Greenberg, and F. Saudou. The IGF-1/Akt pathway is neuroprotective in Huntington's disease and involves huntingtin phosphorylation by Akt. *Developmental Cell*, 2:831–837, 2002.

[13] D. Jones, W. Taylor, and J. Thornton. A model recognition approach to the prediction of all-helical membrane protein structure and topology. *Biochemistry*, 33:3038–3049, 1994.

[14] P. D. Karp. A strategy for database interoperation. *Journal of Computational Biology*, 2(4):573–583, 1996.

[15] Y. Labrou. Standardizing agent communication. In *EASSS*, pages 74–97, 2001.

[16] Y. Labrou and T. W. Finin. A semantics approach for KQML - a general purpose communication language for software agents. In *CIKM*, pages 447–455, 1994.

[17] L. McGuffin, K. Bryson, and D. Jones. The PSIPRED protein structure prediction server. *Bioinformatics*, 16:404–5, 2000.

[18] S. Möller, M. D. Croning, and R. Apweiler. Evaluation of methods for the prediction of membrane spanning regions. *Bioinformatics*, 17:646–653, 2001.

[19] S. Möller, U. Leser, W. Fleischmann, and R. Apweiler. EDITtoTrEMBL: a distributed approach to high-quality automated protein sequence annotation. *Bioinformatics*, 15:219–227, 1999.

[20] S. Möller and M. Schroeder. Conflict resolution for the automatic annotation of transmembrane proteins. *Journal of Computing and Chemistry*, 46(1):41–49, 2001.

[21] M. H. Nodine and A. Unruh. Facilitating open communication in agent systems: The infosleuth infrastructure. In *Agent Theories, Architectures, and Languages*, pages 281–295, 1997.

[22] K. Rutherford, J. Parkhill, J. Crook, T. Horsnell, P. Rice, M.-A. Rajandream, and B. Barrell. Artemis: sequence visualization and annotation. *Bioinformatics*, 16:944–945, 2000.

[23] The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[24] Web Services Activity. http://www.w3.org/2002/ws/.

[25] M. Wooldridge. Agent-based software engineering. *IEE Proceedings Software Engineering*, 144(1):26–37, 1997.

[26] XML Protocol Working Group. http://www.w3.org/2000/xp/Group/.