BIOINFORMATICS ORIGINAL PAPER

Vol. 21 no. 14 2005, pages 3155–3163
doi:10.1093/bioinformatics/bti491

Databases and ontologies

A criticality-based framework for task composition in multi-agent bioinformatics integration systems

Konstantinos A. Karasavvas[1,*], Richard Baldock[2] and Albert Burger[1,2]

[1]School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, UK and [2]Human Genetics Unit, Medical Research Council, Edinburgh EH4 2XU, UK

Received on May 26, 2004; revised on April 12, 2005; accepted on May 6, 2005
Advance Access publication May 12, 2005

ABSTRACT

Motivation: During task composition, such as can be found in distributed query processing, workflow systems and AI planning, decisions have to be made by the system and possibly by users with respect to how a given problem should be solved. Although there is often more than one correct way of solving a given problem, these multiple solutions do not necessarily lead to the same result. Some researchers are addressing this problem by providing data provenance information. Others use expert advice encoded in a supporting knowledge-base. In this paper, we propose an approach that assesses the importance of such decisions with respect to the overall result. We present a way of measuring decision criticality and describe its potential use.

Results: A multi-agent bioinformatics integration system is used as the basis of a framework that facilitates such functionality. We propose an agent architecture, and a concrete bioinformatics example (prototype) is used to show how certain decisions may not be critical in the context of more complex tasks.

Contact: ceekk@macs.hw.ac.uk

INTRODUCTION

To solve a complex problem, systems typically break it down into more manageable smaller problems, which are then executed following some partial ordering. In general, there is a tradeoff between protecting the user from unnecessary details of these (de)composition activities (i.e. providing a high level of transparency) and giving him/her some control over the answer-finding process.

In many cases there is more than one way to solve a particular problem. For example, what scoring matrix should be used for a sequence comparison, and what database should be used to find tissue-specific gene expression data. We use the term decision point (DP) to refer to situations where such choices exist. Providing some measure of the criticality of a DP can aid the system, as well as the biologist in pruning the possible solution space for any given problem.

For providing such a measure, we compare the results acquired after making one choice to the results acquired from the other available choices of the DP. Alternative choices are executed according to user preferences and available computational resources. The higher the differentiation between the result sets over a number of queries that contain a certain DP, the more critical that DP becomes.

The framework we propose supports three execution modes for each DP: (1) automatic, where the DP is resolved by the system; (2) interaction, where the system asks for advice from the user while presenting known benefits/drawbacks of the available choices; and (3) comparison, where alternative choices are executed and the acquired results are compared to calculate criticality. In addition, after execution, the framework allows the querist to request an explanation for the choices made during execution—we call that global explanation.

The basic concept of DPs and their criticality is applicable to a variety of technologies, such as distributed query composition, workflow composition and hierarchical task network planning.

Agent technology has been applied successfully in the past for system integration (Bayardo *et al*., 1997; Sycara *et al*., 2001; Carey *et al*., 1995; Garcia-Molina *et al*., 1997). Furthermore, it has been argued that it is particularly suitable for integrating bioinformatics resources (Karasavvas *et al*., 2004). This paper proposes a multi-agent framework and architecture that enhances the traditional mediation approach to integration. We specify a new agent protocol and define two new agent roles that will enable measurement of decisions' criticality in a distributed environment.

The remainder of the paper is organized as follows: an overview of the multi-agent system is given in the next two sections; then, decision criticality is explained in more detail, while the following section describes the experiments that were carried out; the final sections discuss related work, evaluation and conclusions.

MULTI-AGENT SYSTEM OVERVIEW

Our system is a purely communicative multi-agent system: there is no external environmental influence and the agents communicate only by means of messages. The system is based on the FIPA specifications (http://www.fipa.org), and a FIPA-compliant development tool, JADE (Bellifemine *et al*., 1999), is used for implementation. Messages exchanged between agents are formed in a high-level language, FIPA Agent Communication Language (ACL), and the ACL content language is SL0—a subset of the FIPA suggested Semantic Language (SL). In turn, the SL0 content conforms to specified ontologies, discussed later in this section.

Architecture

The agent framework that we will discuss extends the well-established mediation (Wiederhold, 1992) approach to integration. This approach is also used by a number of successful integration systems using agent technology (Bayardo *et al*., 1997; Sycara *et al*., 2001; Carey *et al*., 1995; Garcia-Molina *et al*., 1997)—not including

*To whom correspondence should be addressed.

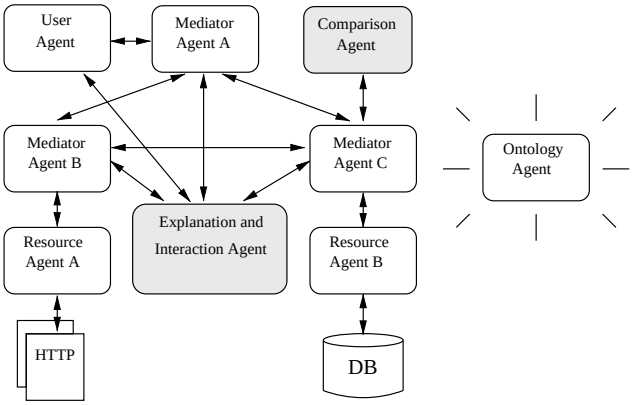

Fig. 1. Mediation-based agent integration architecture including the two new agent types that we propose.

'Comparison Agent' and 'Explanation and Interaction Agent', which are introduced in our framework.

The basic agent roles—and in this case agents—in a mediation-based integration system are shown in Figure 1. They are:

User agent (UA). This is the entry point of a request. Users interact with this type of agent to formulate queries and view final results or any errors that occurred. An UA knows how to translate an interface query into an ACL message and send it, and how to display an ACL message reply received.

Mediator agent (MA). This type of agent has mediating capabilities. It accepts a task, decomposes it to sub-tasks as necessary, sends them to other agents and then integrates the results before returning them. Various planning techniques could be used here.

Resource agent (RA). This type of agent has all the required knowledge to access a specific information resource. It knows how to translate a request it receives in an ACL to the appropriate resource query language and how to construct the resource's answer back to a proper ACL message—to send back to the requesting agent.

Ontology agent (OA). This type of agent can contain one or more ontologies that the system uses. It can be used as a common vocabulary of the system so that other agents can refer to it and resolve potential semantic heterogeneities. An OA can usually communicate with any domain-dependent agent.

These are the agents dependent on the application domain. The shaded components are system agents responsible for the proposed framework's functionality. It comprises:

Explanation and interaction agent (EIA). This type of agent has the important role of providing the user with either a global explanation or an interaction—the DP with the available choices together with argumentation on the benefits of each—to allow the user to decide. Upon initialization, MAs will notify EIA of their DPs, which the latter uses to build a table containing the MAs, their DPs and the possible choices for each of these decisions. During execution it also receives partial execution traces from MAs, so that in the end a complete execution trace can be compiled, to be used in the global explanation (see Adjustable autonomy protocol section).

Comparison agent (CA). This type of agent accepts requests for comparisons between different result sets. It is responsible for estimating criticality of decisions—and/or of agents', based on all their available DPs—and storing the results. It also informs other agents of these results upon request. The criticality will be expressed in an easily understandable—by the user—format, like a percentage.

Other system components/agents, that for the sake of simplicity are not illustrated, deal with agent brokering, life-cycle and message transport services—Directory Facilitator (DF), Agent Management System and Message Transport System, respectively, according to the FIPA specifications.

Other multi-agent integration systems may use different names for agents, e.g. planning agents, or task composition agents instead of mediator agents, but their functionality is similar to the one we describe here.

Prototype: gene-expression case study

We have implemented and tested a prototype system to demonstrate the ideas mentioned above. It is a multi-agent bioinformatics system integrating gene expression resources for mouse. It comprises GXD (Ringwald *et al*., 2001), a mouse gene expression database, EMAGE (Davidson *et al*., 1997), a mouse gene expression database with mappings to a mouse embryo 3D model and BLAST (Altschul *et al*., 1990) at NCBI (http://www.ncbi.nlm.nih.gov/BLAST), an Internet sequence matching tool.

Previously, we examined the functionality of each type of agent available to the system. In this section, we will discuss the specific application-dependent Resource and Mediator agents used.

Resource agents

BLASTAgent. Converts a request from SL0—from the content of an ACL message—to an appropriate format understood by BLAST and connects (to BLAST) via a socket connection. It then converts the reply to SL0 again and sends an ACL message back to the requesting agent. BLASTAgent accepts a protein or DNA sequence and returns the genes that match the given sequence. Other parameters include the sequence data source to be used, the organism and the scoring matrix.

EMAGEAgent. Converts the SL0 content of an ACL message to the appropriate CORBA request(s)—supported by EMAGE—and then re-converts the reply to SL0, which is then sent back to the requester. EMAGEAgent requires genes and a developmental stage as inputs and returns the tissues that the given genes are expressed in.

GXDAgent. Converts the SL0 content of an ACL message to the appropriate SQL statements and then uses them to query the GXD database; it then converts the result sets back to SL0 and returns them to the requester. Similar to EMAGEAgent, GXDAgent requires genes and a developmental stage as inputs and returns the tissues that the given genes are expressed in.

Mediator agents

GenesAgent is responsible for dealing with requests that search for genes. It mediates on what resources can be used (BLASTAgent) and it knows what kind of input they require, as well as what kind of decisions have to be resolved for that request. It accepts requests to acquire gene names, given a sequence (protein or DNA). It is in this agent where the decisions on which parameters, a BLAST request for example, are selected. Currently, we focus on the scoring matrix parameter.

TissuesAgent is responsible for dealing with requests that search for tissues in a particular developmental stage. It mediates on what resources can be used (EMAGEAgent, GXDAgent) and it knows what kind of input they require, as well as what kind of decisions have to be resolved for that particular request. The decision that we focus on this agent is data source selection; currently, 'TissuesAgent' can acquire tissues, given gene names, from two gene expression data sources (EMAGE and GXD).

BioMouseAgent is responsible for high-level requests. The 'User Agent' would usually contact this agent for mouse gene-expression related queries. It mediates what RA and/or MA (GenesAgent, TissuesAgent) can be used to tackle a particular request; it knows the kind of input they require, as well as what kind of decisions have to be resolved for that request. An example of a high-level query, which makes use of other MAs, will be presented later in the Experiments section.

Our prototype makes use of three ontologies. At the lower level, `fipa-agent-management` is a FIPA specified ontology that provides semantics for all necessary low-level system functions, such as brokering and life-cycle services. We defined `adjustable-autonomy-ontology`, which specifies the appropriate ontological objects and functions to understand the Adjustable Autonomy Protocol (AAP) that constitutes the skeleton of the framework presented. Finally, we specified a simple application domain ontology, called `biomouse-ontology`, which semantically defines the concepts used in the prototype. This domain ontology was used for simplicity instead of other more complete ontologies, like Gene Ontology (GO) (Ashburner *et al*., 2000).

Configuration

In our integration system the user is allowed to customize the system's behaviour according to his/her needs. She/he can decide if a DP is to be resolved automatically by the system or if she/he wants to make the choice (interaction mode), or when a comparison is required.

The user is able to configure the above agents so as to achieve the functionality needed. More specifically she/he can specify two flags for each DP known by the system to indicate execution with respect to interaction and comparison modes, respectively.

To automate whether or not a decision is executed interactively, we include another configuration parameter, called criticality threshold. It is expressed in the same format as criticality, i.e. as a percentage, which then indicates that a DP is to immediately initiate an interaction—overriding the interaction flag—, if its criticality is above the threshold.

To further automate a comparison requirement for a decision, we include another configuration parameter, called comparison threshold. It is expressed as a simple number that indicates the least number of comparisons we want on a particular DP. If the comparisons on a DP are less than the threshold, it will attempt to run in comparison mode, irrespective of whether or not the comparison flag of the decision is set (it might still not compare in case the agent/system is overloaded).

The configurable flags and thresholds that we have mentioned can be configured at several levels of granularity:

- global: for all DPs;
- decision type: applies to all the DPs of a specific type encountered during execution of a query;
- MA applies to all the DPs reached during execution of the query that reside on a specific MA—that would be especially useful when the system makes use of MAs of other organizations;
- a combination of the above two: applies to all DPs of a specific type on a specific MA.

The configuration is delegated to the MAs by using a user-defined slot in the ACL message, according to the FIPA specifications, which does not interfere with agents that do not understand the `adjustable-autonomy-ontology`.

ADJUSTABLE AUTONOMY PROTOCOL

To achieve the kind of functionality referred to in the Introduction section, our agents have to follow certain protocols. On the lower level, for purposes of agent communication, the FIPA interaction protocols are sufficient—like FIPA-Request, or FIPA-Query.

While this could be sufficient for requesting a task, it cannot specify the overall cooperation needed to achieve user intervention—adjustable autonomy—and/or how an explanation is constructed for planning and decision-making. For that purpose we developed a higher-level protocol, one that coordinates interactions among the MAs, the RAs, the EIA and the CA. We call this protocol AAP.

The AAP can be requested from an MA. The protocol is illustrated, using AUML (Odell *et al*., 2001), in Figure 2. The diagram shows the interactions between agents (horizontal arrows specifying direction) during execution of the protocol. The vertical dashed lines represent time. The numbered boxes in the diagram represent sub-protocols, initiated according to user configuration and system planning. The protocol can be described in stages:

(1) The first MA, *root* MA, receives a request/query—which contains a label that uniquely identifies the query—and constructs an execution plan. It then checks the user preferences to see if the plan contains DPs, and for each DP it checks whether:
   - the user wants to take control of it, or its criticality exceeds the criticality threshold, in which case it sends a request to the EIA (1)—see stage 3, or
   - the user wishes to measure its criticality, or the number of comparisons that were made on it are less than the comparison threshold, in which case a number of possible choices follow are executed and when the results return they are forwarded to the CA (6) to measure their differentiation/similarity (the greater the similarity in the results, the less critical the decision is).

   When the plan is finalized, the MA notifies the EIA of the (partial) plan (3)—including the unique query identifier as a reference—and the MA executes it. The plan contains requests to RAs (4) and/or other MAs (5).

(2) Non-root MAs also act according to the previous stage. At the bottom of the MAs-tree, the *leaf* MAs, send requests only to RAs (4).

(3) Upon request for user interaction, the EIA forwards the DP to the UA (2), together with comments on the possible effects that this decision could have on the result and, if available, the criticality of the decision. The user makes his/her choice through the UA and the latter informs the EIA of the decision taken, which in turn informs the MA.

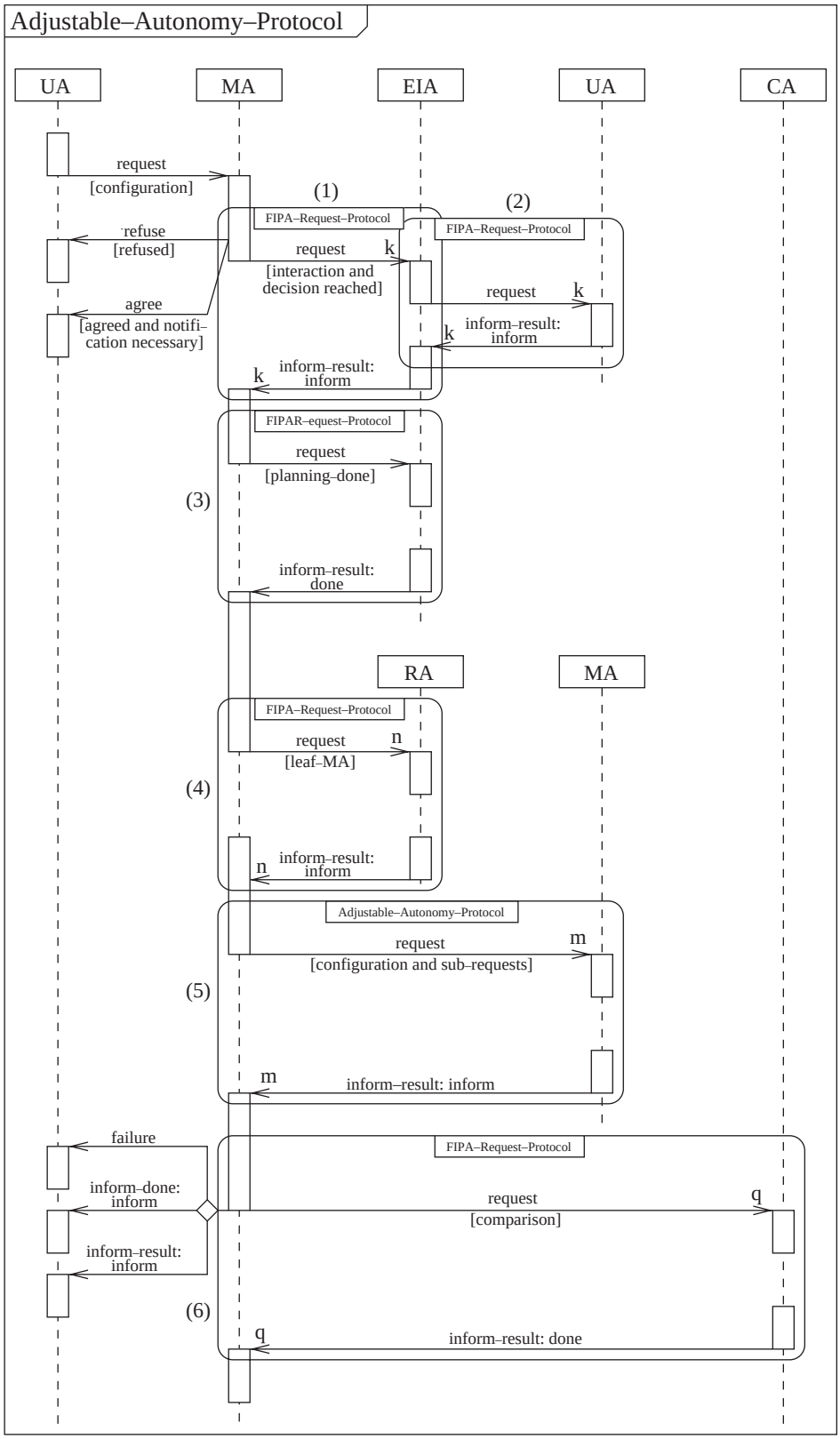

Fig. 2. Adjustable Autonomy Protocol expressed in AUML.

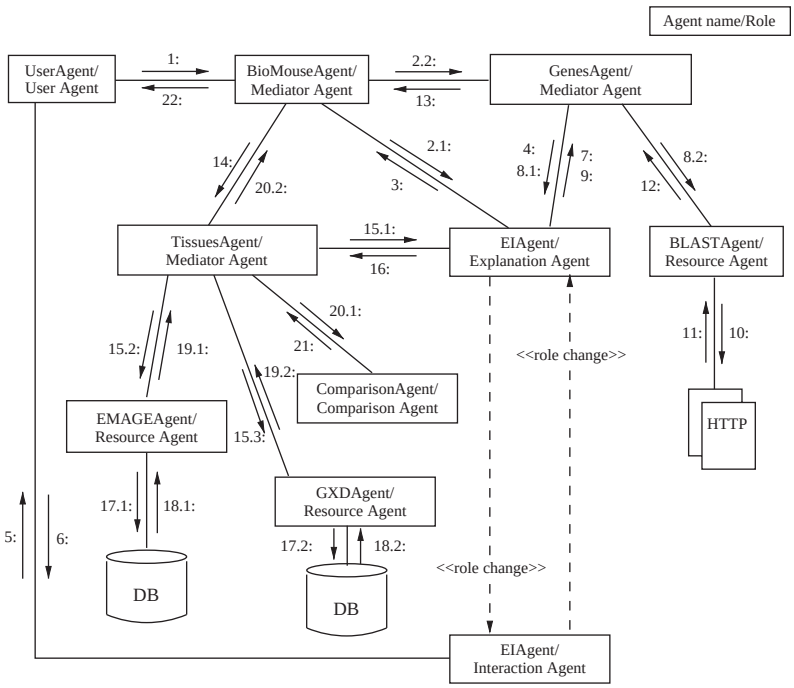

Fig. 3. AUML collaboration diagram showing agent interactions.

(4) Answers are integrated—when appropriate—in MAs and sent back to the requesters. Then, the root MA receives the answers to its requests and constructs a final answer.

(5) In the last stage, the root MA sends the answer to the initial requester.

Each UA query is labelled with a unique identifier so that we can later refer to it. Thus, after the user—via the UA—views the results she/he can ask for a global explanation from the EIA by providing this unique query identifier. The EIA would by now be on par with the protocol, have a complete trace assembled ready for use in the construction of the global explanation.[1]

To better understand the functionality of the protocol, we describe what happens during initialization and execution of the system.

Initialization and execution

During start-up, all agents that offer services (such as RA, MA, EIA and CA) have to register these services with the DF. When a service is needed, agents have only to ask the DF to retrieve a set of agents able to cope. In addition, all MAs that deal with critical decisions have to register with the EIA. The registration message contains the type of decision and the possible choices, and comments (benefits/drawbacks) on each choice. An UA will request the possible critical decisions from the EIA, so that it can present them to the user.

In Figure 3 we use an AUML collaboration diagram that illustrates the interactions of a sample execution from our prototype system, according to AAP. In order to demonstrate the use of all agents, each of the MAs operates in a different mode. 'BioMouseAgent' operates in automatic mode. It decomposes the initial request received (1:) into two sub-tasks—we assume that the second sub-task requires information from the first and thus, they cannot run concurrently. It makes all the necessary decisions automatically and notifies EIA of the partial plan (2.1:). It simultaneously requests 'GenesAgent' to deal with one of the sub-tasks (2.2:). The latter operates in interaction mode, so it first requests user intervention from EIA (4:)—which in turn requests the UserAgent (5:)—before finalizing its plan and notifying EIA (8.1:). Finally, after 'BioMouseAgent' obtains the results from the first sub-task (13:) it requests the second sub-task from 'TissuesAgent' (14:), which operates in comparative mode. That means it will have to execute more than one decision choice—we assume that the choice is between two data sources, EMAGE and GXD. By default it also operates automatically so it notifies the EIA of the partial plan (15.1), and follows it up with a request for each choice, one to 'EMAGEAgent' (15.2) and the other to 'GXDAgent' (15.3). After receiving the results (19.1: and 19.2:), 'TissuesAgent' will send them for comparison (20.1:), while also sending a reply back to 'BioMouseAgent' (20.2:). The final result is then integrated in the 'BioMouseAgent' and sent back to the UserAgent (22:). Note that the requests to the RAs are not part of the AAP but are included to provide a complete example.

DECISION CRITICALITY

Criticality of a DP is calculated by comparing the results received after execution of two or more possible choices. To measure the similarity of two sets, we take their intersection (common elements) and divide the cardinality of the resulting set by the average cardinality of the two sets (other criticality measurements may be explored); for

[1]Global explanation could simply be a beautified version of the trace or alternatively a natural language explanation using the reconstruction approach (Wick and Thompson, 1992).

sets A_1 and A_2 their similarity s_{12} is:

$$s_{12} = \frac{|A_1 \cap A_2|}{\frac{|A_1|+|A_2|}{2}} = 2 \cdot \frac{|A_1 \cap A_2|}{|A_1| + |A_2|}. \quad (1)$$

In the case of more than two sets, we need to make all possible comparisons in pairs. A result of k sets consists of $n = k(k-1)/2$ pairs, and thus will make that many comparisons. Then, the mean similarity of k sets is calculated by:

$$\overline{s_k} = \frac{1}{n}\left(\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} s_{ij}\right) \quad (2)$$

Now we also have the mean differentiation of the k sets, $\overline{d_k} = 1 - \overline{s_k}$. Finally, we can express both, similarity and differentiation means, as percentages with $\overline{S_k} = \overline{s_k} \cdot 100$ and $\overline{D_k} = \overline{d_k} \cdot 100$, respectively.

As an example, consider the three sets, $A_1 = \{a, b, c\}$, $A_2 = \{a, b\}$ and $A_3 = \{a, b, d\}$. First, we find all possible pairs:

$$n = \frac{k(k-1)}{2} = \frac{3(3-1)}{2} = 3$$

Then we apply Equation (2) and we get:

$$\begin{aligned}\overline{s_k} &= \frac{1}{n}(s_{12} + s_{13} + s_{23}) \\ &= \frac{1}{n}\left(\frac{2 \cdot |A_1 \cap A_2|}{|A_1| + |A_2|} + \frac{2 \cdot |A_1 \cap A_3|}{|A_1| + |A_3|} + \frac{2 \cdot |A_2 \cap A_3|}{|A_2| + |A_3|}\right) \\ &= \frac{1}{3}\left(\frac{2\times 2}{5} + \frac{2\times 2}{6} + \frac{2\times 2}{5}\right) = 0.75\end{aligned}$$

Thus, sets A_1, A_2 and A_3 differ by 25% [$\overline{D_k} = (1 - \overline{s_k}) \cdot 100$].

Of course, $\overline{D_k}$ reflects the differentiation of a DP over only one query. While this is useful for analysing the possible fluctuation on a specific query, we would also like to have a general differentiation measure for a DP over the history of all queries. We can thus calculate the criticality of a decision d, which is simply the mean:

$$\overline{C_d} = \frac{1}{h}\sum_{i=1}^{h} \overline{D_k^i}, \quad (3)$$

where h is the history—number of queries—of that DP.

So far, we measured the criticality of one DP. A query usually contains more than one such DP. More formally, a query Q can be said to consist of a set of DPs [$Q : (\mathrm{DP}_1, \mathrm{DP}_2, \ldots, \mathrm{DP}_n)$]—each one with its own local criticality—that influences its global (overall) criticality. Often, a DP depends on another DP that precedes it and thus depends on the data/criticality of that DP. To specify dependences between DPs we use '$\rightarrow$', as in: $Q : (\mathrm{DP}_1 \rightarrow \mathrm{DP}_2)$, where DP_2 needs data acquired after DP_1 is resolved.

Further analysis of the query result's criticality relative to its DPs' criticality could provide important insight on the DPs' influence/importance during the planning process. Thus, except from local criticality of DPs, we can identify global criticality, which is based on the influence of the criticality of one DP onto the criticality of its dependent DP (see next section for a detailed example).

EXPERIMENTS

The framework's functionality is responsible for the distributed orchestration of the agents so as to facilitate user interaction with explanation and criticality calculation for the DPs. All appropriate information (plans, criticality results, etc.) is stored in a database, during execution. It requires further tools to data-mine through these data in order to analyse the DPs' influence on our queries and/or provide other statistical information concerning the data sources. We have implemented basic statistical and analysis tools to analyse the data acquired for the following experiments.

The experimental results presented here are based on the following query: 'Find the mouse tissues that express the genes which match the given protein sequence'. This query can be decomposed into two sub-queries or steps:

(a) which mouse genes correspond to the protein sequence given as input, and

(b) which mouse tissues express these genes at a particular developmental stage.

We tested 44 queries, where the input of each was a protein sequence randomly taken from known existing genes. We collected these sequences from the Swiss-Prot (part of Uniprot) protein sequence database.

In our integration system we use an online BLAST sequence tool for sub-query (a). There are many parameters to be considered when querying BLAST, e.g. sequence database, scoring matrix and gap costs. Each parameter could constitute another DP during the task composition process. For this experiment we examined only one, the scoring matrix, and more specifically we only considered two matrix choices, BLOSUM62 and PAM70. The criteria on the basis of which we picked these matrices were that they are not of the same series, and that their default gap costs are neither identical nor very different.

For sub-query (b) we use two different gene-expression resources, GXD and EMAGE. Thus, the decision to be made is, which of the two databases should be used to acquire the final results.

In each sub-query, either automatically (by the system), or interactively (by the user) one choice would be selected and followed through to acquire the final results—mouse tissues for this query. For example, matrix BLOSUM62 could be used to obtain the mouse genes and the GXD resource to obtain the mouse tissues. A different combination may provide different results, both intermediate and final. Indeed, comparing the results would enable us to calculate their differences and consequently how critical a decision is.

Our example query can be written as: $Q_e : (\mathrm{DP}_1 \rightarrow \mathrm{DP}_2)$, where DP_1 and DP_2 are the DPs for the scoring matrix and the gene expression database, respectively, and the arrow specifies dependence, i.e. an execution ordering of DPs.

The upper panel diagram in Figure 4 provides an overview of the execution of the sample query. At the bottom we have all possible result sets for this two-step query. By acquiring both result sets at a DP and comparing them, we can calculate the local criticality for that DP. At step one, we use BLAST to get the genes that correspond to the input protein sequences. Using a different matrix results in a different gene set. Comparing the gene sets for each query and getting the mean of their differentiation gives us the local criticality of that decision, 14% for our example. We only compare very close matches, i.e., results where the BLAST expect value is $<1 \times 10^{-10}$ (please see

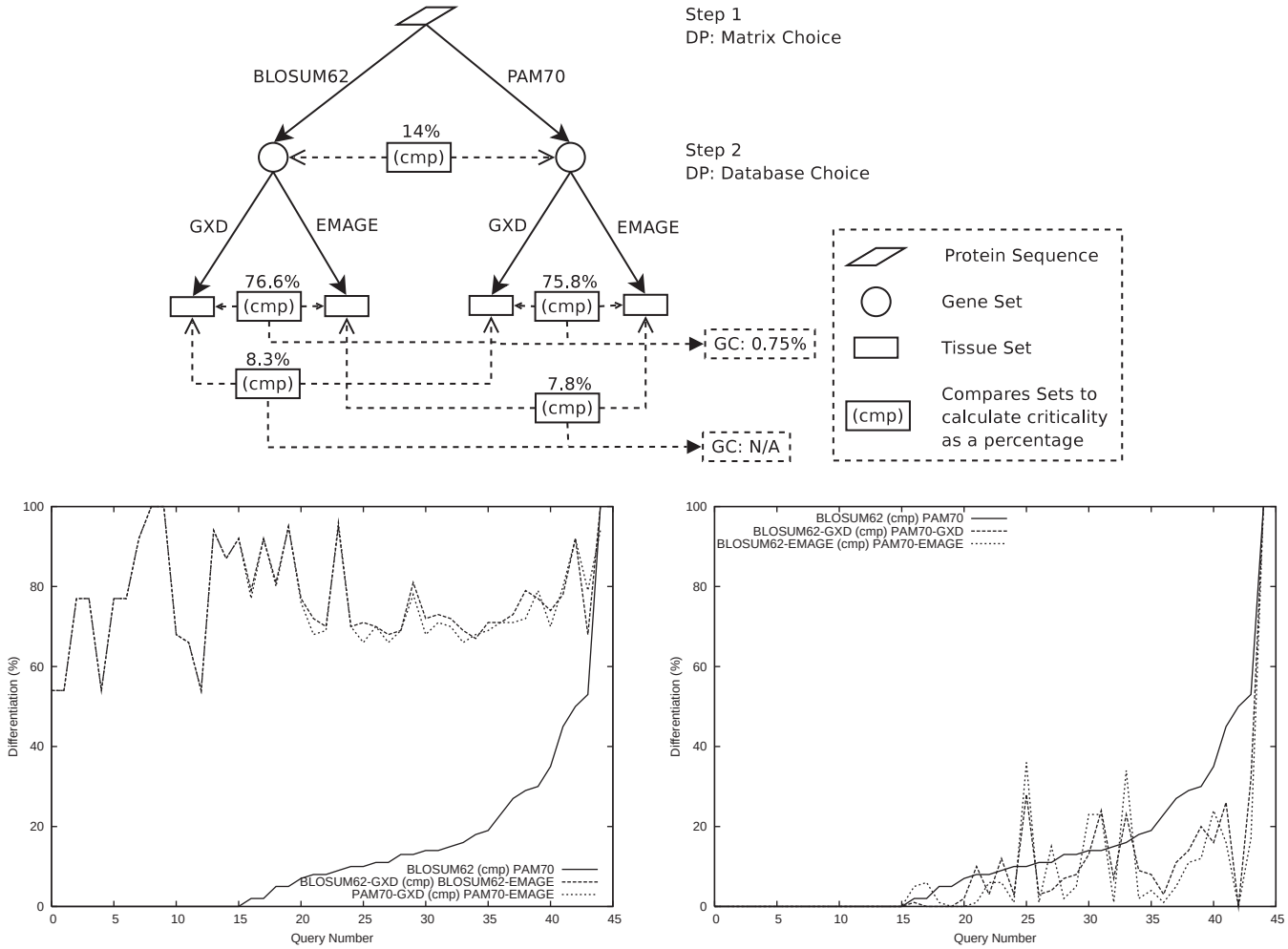

Fig. 4. Upper panel: Overview of experiment's DPs and criticalities; lower left panel: (Step 1) differentiation between BLOSUM62 and PAM70. (Step 2) Differentiation between same matrix and different data sources; lower right panel: (Step 1) differentiation between BLOSUM62 and PAM70. (Step 2) Differentiation between different matrices and same data sources.

http://www.ncbi.nlm.nih.gov/blast/blast_FAQs.shtml#Expect for an explanation). Both lower panel diagrams in Figure 4 illustrate the differentiation (sorted in ascending order) for each query between BLOSUM62 and PAM70 matrices with the normal lines; note that although the mean criticality is 14% the standard deviation is very high. In brief, for some of the given protein sequences (represented by the queries along the x-axis) it mattered which matrix was used, while for others it did not. One can conclude that given in isolation, this DP should be given some attention.

At step two, we use both the gene sets that we got from the matrix DP and continue execution for each one. For the mouse genes returned from the BLOSUM62 choice, we acquired results from both the GXD and the EMAGE databases and then compared the results—BLOSUM62-GXD (cmp) BLOSUM62-EMAGE (see normal dashed line in the lower left panel diagram). The average of the differentiations yields a 76.6% criticality. Similarly, we compared the tissue results acquired from GXD and EMAGE using the PAM70 genes result set as an input—PAM70-GXD (cmp) PAM70-EMAGE (see the bold dashed line in the diagram)—which yielded 75.8% when we calculated the mean of the differentiations.

Criticality in both cases was high but comparing the two dashed lines, we observed only minor differences between them, no matter how high the matrix criticality was. The results suggest that even though the local criticality of the matrix choice varies, i.e. result sets can be quite different at times; when used to further query for the mouse tissues, the impact of this difference is less significant, i.e. its global criticality[2] (with respect to the given query) is relatively low. Hence, even though DP_2 depends on the results of DP_1, the local criticality of the latter does not influence the final results significantly. This suggests that, for the given example, most of the genes that are expressed in mouse were found by both BLOSUM62 and PAM70.

Another approach to analyse the results of this two-step query would be to make comparisons between the first and the third tissue

[2] Note that the global criticality, GC, is not the difference of the two alternative criticalities at the second step, but rather the mean of the global differentiations of the second step. The global differentiation of DP_1 is calculated as the standard deviation of the local differentiations of DP_2 that resulted from the gene sets we got from step one. A detailed explanation of global criticality calculation is not included due to space limitations.

sets and between the second and the fourth ones. That would mean we could use the partial result sets of the matrix DP (gene sets acquired from different scoring matrices) to the same gene-expression database; thus, in the second step there is no DP and no choice selection, as the partial result sets are passed over to the same choice: GXD to get the first and the third tissue sets and EMAGE to get the other two.

We can see the relevant calculations in the upper panel diagram of Figure 4; and in the lower right panel diagram the detailed differentiations for each query. The percentages calculated in the second step (8.3 and 7.8%) do not represent the criticality of any DP, rather the difference between two result sets. This should be obvious, as there is no DP in the second step; the two results sets acquired from the first step are both passed to GXD, and then again to EMAGE. Thus, the two bottom percentages in the diagram cannot be used to calculate the global criticality, as the latter is meant to provide the deviation in criticality that the partial results of a DP have from the next dependent DP (detailed explanation of how GC is calculated is not included due to space limitations).

What these calculations really show, is the influence that a particular gene-expression database has on the decision criticality of the matrix DP. The difference that matrix DP result sets introduced decreases when passing them to a gene-expression database; 8.3 and 7.8%, respectively. This shows that only a very small amount of significant genes were matched by one matrix and not by the other—a conclusion which is in agreement with our analysis made in the previous section.

To summarize, the first approach shows the influence that the choices of the first step can have on the DP criticality of the second step, while the second approach provides an estimate of the influence that a specific choice (and not choices) at the second step can have on the DP criticality of the first step.

RELATED WORK

The focus of mixed-initiative systems (Ferguson *et al*., 1996; Ferguson and Allen, 1998) is cooperation between humans and computer systems. The primary concern of these systems is the discourse/dialogue model (Allen, 1999; Horvitz, 1999; Guinn, 1999); the system engages the user in a dialogue and 'discusses' the planning process. They provide enhanced functionality as far as collaboration—between users and systems—is concerned, but collaboration is in most cases obligatory. Our framework allows the user to take control of decisions, if she/he needs to but can also execute independent of the user.

Decision Support Systems (DSSs) (Sprague and Watson, 1986) aim to help users make certain decisions. They are typically used in abstract problems such as high-level management decisions (Adelman, 1992). At first glance one might notice similarities to our framework but there are fundamental differences. A DSS supports the user in making a decision. The user queries the system to obtain help and then he/she has to make the choice. Our framework comprises a reasoning module able to replicate human behaviour—expert system—while extending it by providing explanation of and user control over DPs. The system queries the user on DPs—according to preferences—and the latter may or may not make the choices.

Although in recent years the work on agents' adjustable autonomy (American Association for Artificial Intelligence, 1999) is increasing, it is still in the very early stages. As the need to increase users' trust of agents becomes more important so does adjustable autonomy. However, work is focused on whether to transfer control to the user and when rather than how (Scerri *et al*., 2002). Identification of the DPs and calculation of their criticality provide us with the knowledge of whether to transfer control and when, and thus we can focus on how.

Workflow composition systems, such as Geodise (Chen *et al*., 2003), use domain knowledge to guide the user during the composition process. In contrast to our framework, this is not automated and the user has to manually construct the workflow with the help of the system.

Finally, our approach, in addition to domain knowledge uses an independent measure, criticality, which is based on the actual data of the resources involved in the composition. This data-centric approach of calculating criticality differentiates our system substantially from all the other aforementioned approaches.

EVALUATION

Since its conception, this project has benefited from the guidance of expert bioinformaticians and biologists, feedback from whom has helped to refine the data-centric framework and requirements introduced in the previous sections. Further sources used to revise the system include feedback received during workshops and conferences, as well as from the implementation and testing of the theoretic framework and the agent communication protocol in the prototype. The latter demonstrated that the communication and interaction between the agents worked as was expected.

In order to further assess the quality of our research and the usefulness of the latter to bioinformaticians, we also decided to conduct an evaluation. It is important to point out here that since the implementation of our framework was but a prototype to demonstrate the work presented in this thesis rather than a fully functional software tool, the aforesaid evaluation has not been aimed at the implementation itself—as is usually the case with more formal appraisals—but rather at the concepts behind the framework and the usefulness of the functionality provided.

To obtain opinions of the experts, we opted for a combination of an informal presentation/interview and a questionnaire. During the presentation stage of the process we explained the motivation and rationale behind this work and familiarized the interviewees with the functionality provided by our framework—including the criticality calculation and analysis and choice delegation and explanation. We demonstrated this functionality with the results of the experiments, also presented in this work. Following the preliminary discussion, we presented the users with a questionnaire comprising statements, each related to some aspect of the ideas presented in this paper. For each statement, the users had to rate their agreement using a scale from '1' meaning 'strongly disagree' to '5' meaning 'strongly agree'. We are not going to describe the questionnaire in detail, but rather focus on three very important statements that the users had to rate:

S1. Showing the criticality of a DP to the expert user helps him/her to decide whether he/she wants to intervene to the decision-making and when not.

S2. Criticality thresholds, set by expert users, should be used to determine when to delegate decision-making to the user.

S3. The experiment demonstrated that the matrix choice between BLOSUM62 and PAM70, in the first step, does not influence

the criticality of the second step. On this evidence, it is safe to automate the matrix choice (first step) without substantially affecting the final results.

The questionnaire was completed by five expert bioinformaticians and biologists, and the average of their appraisal was that they agree with all of the above statements (rated 4 and above). The only objection raised was that words like 'high' or 'very low' would possibly be more meaningful to users and, thus, be a better way of indicating a score than percentages. However, it is our belief that attaching specific adjective-labels to specific percentages would obscure the fact that one single percentage is not necessarily equally critical, for every query. Criticality, and by implication, the adjective that can successfully replace a percentage varies, depending on the experiment conducted. All in all, the users' endorsement of statements that are central to the evaluation of the framework shows that the latter is not only well understood, but also that it will provide a good practical solution.

CONCLUSIONS

Bioinformatics integration systems need facilities that can help users to better understand and control the consequences of certain choices made during task composition. We have introduced the concepts of DPs and their local and global criticality, as well as a multi-agent framework that facilitates the realization of these concepts. A concrete bioinformatics example was used to show how certain decisions may not be critical in the context of more complex tasks. The work was carried out using a prototype multi-agent system for mouse gene expression research, developed at the MRC Human Genetics Unit in Edinburgh.

REFERENCES

Adelman,L. (1992) Evaluating Decision Support and Expert Systems. John Wiley and Sons, New York.

Allen,F.J. (1999) Mixed-initiative interaction. *IEEE Intell. Syst.*, **14**, 14–16.

Altschul,F.S. *et al*. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

American Association for Artifical Intelligence (1999) In *Proceedings of the AAAI Spring Symposium on Agents with Adjustable Autonomy*, March 22–24, Stanford, CA, p. 1.

Ashburner,M. *et al*. (2000) Gene Ontology: tool for the unification of biology. *Nat. Genet.*, **25**, 25–29.

Bayardo,R., Bohrer,W., Brice,R., Cichocki,A., Fowler,J., Helal,A., Kashyap,V., Ksiezyk,T., Martin,G., Nodine,M., Rashid,M., Rusinkiewicz,M., Shea,R., Unnikrishnan,C., Unruh,A. and Woelk,D. (1997) InfoSleuth: agent-based semantic integration of information in open and dynamic environments. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 1997)*, May 13–15, Tuscon, AZ. ACM press.

Bellifemine,F., Rimassa,G. and Poggi,P. (1999) JADE—a FIPA-compliant agent framework. In *Proceedings of the Fourth International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents* (PAAM 99), London, UK, April 19–21, pp. 97–108.

Carey,M., Hass,L., Schwarz,P., Aryo,M., Cody,W., Fagin,R., Flickner,M., Lahiewski,A., Niblack,W., Petkovic,D., Thomas II,J., Williams,J. and Wimmers,E. (1995) *Towards heterogeneous multimedia information systems: the garlic approach*. In *Proceedings of the Fifth International Workshop on Research Issues in Data Engineering—Distributed Object Management*, Taipet, Taiwan, March 6–7, *IEEE Computer Society Press*, pp. 124–131.

Chen,L., Shadbolt,R.N., Goble,C., Tao,F., Cox,J.S., Puleston,C. and Smart,R.P. (2003) Towards a knowledge-based approach to semantic service composition. Lecture Notes in Computer Science, *LNCS* 2870, pp. 319–334.

Davidson,D. *et al*. (1997) The mouse atlas and graphical gene-expression database. *Semin. Cell Dev. Biol.*, **8**, 509–517.

Ferguson,G. and Allen,J. (1998) TRIPS: an intelligent integrated problem-solving assistant. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence, (AAAI-98)*, Madison, WI, 26–30 July, pp. 567–573.

Ferguson,G., Allen,J. and Miller,B. (1996) Trains-95: towards a mixed-initiative planning assistant. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems, (AIPS-96)*, Edinburgh, Scotland, 29–31 May, pp. 70–77.

Garcia-Molina,H., Papakonstantinou,Y., Quass,D., Rajaraman,A., Sagiv,Y., Ullman,J., Vassal,K. and Widom,J. (1997) The TSIMMIS approach to mediation: data models and languages. *Journal of Intelligent Information Systems*, **8**(2), 117–132.

Guinn,I.C. (1999) Evaluating mixed-initiative dialog. *IEEE Intell. Syst.*, **14**, 21–23.

Horvitz,E. (1999) Uncertainty, action, and interaction: in pursuit of mixed-initiative computing. *IEEE Intell. Syst.*, **14**, 17–20.

Karasavvas,K.A. *et al*. (2004) Bioinformatics integration and agent technology. *J. Biomed. Inform.*, **37**(3), 205–219.

Karp,P.D. *et al*. (2001) Database verification studies of SWISS-PROT and GenBank. *Bioinformatics*, **17**, 526–532.

Odell,J., Panurak,H. and Bauer,B. (2001) Representing agent interaction protocols in UML. In Ciancarini,P. and Wooldridge,M. (eds), *Agent-Oriented Software Engineering*, Berlin, London, Springer, pp. 121–140.

Ringwald,M. *et al*. (2001) The mouse gene expression database (GXD). *Nucleic Acids Res.*, **29**, 98–101.

Scerri,P., Pynadath,D.V. and Tambe,M. (2002) Why the elf acted autonomously: towards a theory of adjustable autonomy. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems* (*AAMAS* 2002), Bologna, Italy, July 15–19. ACM Press, pp. 857–864.

Sprague,R.H., Jr and Watson,H.J. (1986) *Decision Support Systems: Putting Theory into Practice*. Prentice-Hall.

Sycara,K., Paolucci,M., van Velsen,M. and Giampapa,J. (2001) The RETSINA MAS infrastructure. Technical Report *CMU-RI-TR-01-05*, Robotics Institute, Carnegie Mellon.

Wick,M.R. and Thompson,W.B. (1992) Reconstructive expert system explanation. *Artif. Intell.*, **54**, 33–70.

Wiederhold,G. (1992) Mediators in the architecture of future information systems. *IEEE Computer*, **21**, 38–50.