

A Multiagent System for Workflow-Based Bioinformatics Tool Integration

Bong-Ki Sohn*, Keon-Myung Lee*, Hak-Joon Kim*

* School of Electric and Computer Engineering, Chungbuk National University,
and Advanced Information Technology Research Center(AITrc), Korea
+ Div. of Electronic, Information, Communication Engineering, Howon University, Korea

Abstract

Various bioinformatics tools for biological data processing have been developed and most of them are available in public. Most bioinformatics works are carried out by a composite application of those tools. Several integration approaches have been proposed for easy use of the tools. This paper proposes a new multiagent system to integrate bioinformatics tools in the perspective of workflow since the composite applications of tools can be regarded as workflows. For the easy integration, the proposed system employs wrapper agents for existing tools, uses XML-based messages in the inter-agent communication, and agents are supposed to extract necessary information from the received messages. This allows new tools to be easily added on the integration framework. The proposed method allows various control structures in workflow definition and provides the progress monitoring capability of the on-going workflows. In particular, agents in this system have the rule-based architecture which allows the defined rule set to be a special role agent. This feature provides fast and flexible agent development to aid in managing the complexity of bioinformatics application. This system has been partially implemented and has been proven to be a viable implementation for workflow-based bioinformatics tool integration.

Key words : Multiagent system, workflow, rule-based architecture, tool integration

1. Introduction

With the advent of massive, high-speed sequencing technologies such as short-gun method and efficient assemble softwares, genomes have been being identified for various organisms. In genome projects, very long DNA sequences are randomly segmented into large number of overlapped fragments of size 2 kb or so, and the DNA sequences for the fragments are determined and assembled into long contigs, and then these contigs are combined into a genome map[1]. For the determined genome map, ORFs(open reading frames) likely to encode genes are found and their functions are estimated, if possible, and pieces of related information including gene functions are annotated to the genome database.

Many bioinformatics tools and databases have been developed to support various tasks in genome projects. Depending on target organisms and tasks to be carried out and interests of study, researchers selectively use different tools among the available tools in different ways. In order to choose proper tools, they need to understand what they do, how to use them, and what kinds of pros and cons they have. It is not easy to choose appropriate tools needed to handle given tasks in some situations if the users do not have enough knowledge about the tools and databases. Most bioinformatics tools and databases are provided through the Web and some tools are published to be freely installed at local hosts. In addition, it is sometimes burdensome to figure out the input and output data formats for the tools.

To alleviate this kind of burdens, this paper proposes a multiagent system architecture to provide intelligent bioinformatics tool integration. Usually a sequence of tools are used to process biological data. Therefore, such sequences can be regarded as a workflow[2,3]. The proposed architecture provides the bioinformatics tool integration in a manner of workflow management. For the easy integration of existing tools, their corresponding wrapper agents[4] are supposed to be developed, which make tools act as an agent in the multiagent framework. For the inter-agent communication, XML(extensible Markup Language)-formatted messages are used from which agents are supposed to extract the pieces of information needed without the senders explicit indication of the information intended to the receiver. This kind of communication strategy helps agents integrated in a flexible way. In addition, agents in this multiagent system have the rule-based architecture which allows the defined rule set to be a special role agent. For example, It is possible to develop the Blast wrapper agent by defining the rule set for wrapping the Blast program and loading the rules into itself. This feature provides fast and flexible agent development to aid in managing the complexity of bioinformatics application. Several integration approaches have been proposed and now are in use[5-9]. Most of them take static integration strategy in which it is not so easy to add on new tools.

This paper is organized as follows: Section 2 presents single agent architecture which is a foundation of all agents and section 3 describes the proposed multiagent system for bioinformatics tool integration. Section 4 discusses the implementation issues and an application example of the proposed multiagent framework. In final, Section 5 draws conclusions.

This work was supported by Korea Science and Engineering Foundation (KOSEF) through AITrc.

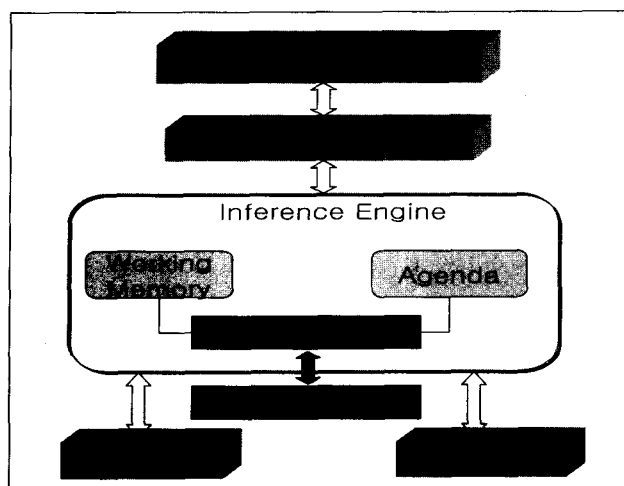


Fig. 1. The rule-based agent architecture

2. Rule-based Agent Architecture

Agents in the proposed multiagent system have the same internal architecture. They adopt rule-based architecture and have only different domain-dependent rule set for modeling agent capabilities. The overall architectural design of a rule-based agent is shown in Fig. 1.

The Communication Module receives and sends messages. It extracts information from the received XML messages by parsing it. It also encodes information into XML messages. The Agent Control Module handles the overall agent control flow. If external message is normal, agent control is passed to Inference Engine and a rule instance is executed.

The Inference Engine is the core of the agent components. It evaluates the rules against the objects that are in the Working Memory and stores all matching rule instance in Agenda. And then the most appropriate rule instance is selected and executed. The Rule base includes IF-THEN rules to model agent capabilities. The simple rules are directly executed, but the plan rules have control flow which orders the rules to accomplish the goal. The plan rules are pre-defined plans which are created by Rule Editor or Planner.

Agent maintains the Operator Set similar to STRIPS operator[10], which an operator corresponds to a simple rule or a plan rule. A rule is automatically transformed into an operator by some conversion mechanisms. When there are no rules satisfying the external goal, the Planner begins planning to find plan rules with existing rules. If the Planner succeed to find plans, the plan is converted into rule format and stored in Rule base. Otherwise, the agent cannot process the external goal and send error message. The Rule Editor is important component for developing, maintaining and debugging agent-dependent rules. It is possible to create simple rules and plan rules. The created rules are dynamically stored in Rule base without restarting the agent.

The proposed multiagent system includes a large number of agents which have the same internal architecture and different role within system by defining rules for specific task. This

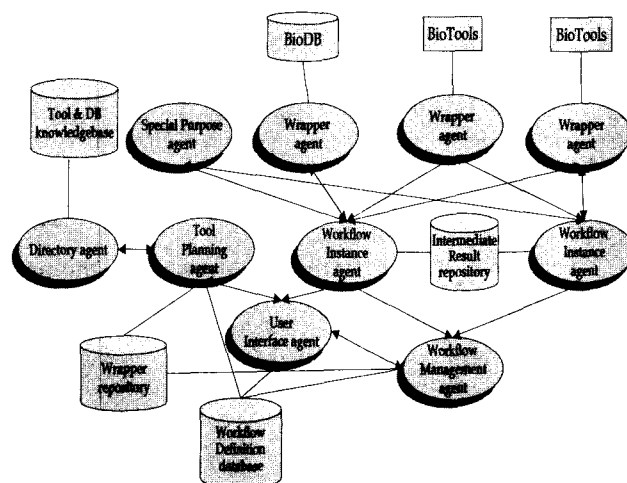


Fig. 2. The proposed multiagent architecture

feature provides fast and flexible agent development and management for complex bioinformatics application.

3. A Multiagent System for Workflow-Based Bioinformatics Tool Integration

3.1 The Architecture of the Proposed Multiagent System

The proposed multiagent system is comprised of a network of different role agents communicating by means of XML-format messages. Fig. 2 shows the proposed multiagent framework that consists of user-interface agents, a workflow management agent, a planning agent, a directory agent, workflow instance agents, and task agents like wrapper agents and special purpose agents. Through a user-interface agent, a user (i.e., biologist or bioinformatician) accesses and controls tools to be used for bioinformatical processing, and retrieves the intermediate results of the processing.

The directory agent maintains information about how and where tools and databases are used and what they do. The planning agent plays the role of taking the users task specification about what to do and suggesting a workflow definition about how to process the given task. The workflow management agent is charge of coordinating and monitoring all workflow instance agents. It instantiates a workflow instance agent for each workflow. If requested by the user, it may terminate an ongoing workflow. It also directs requests from users to their corresponding workflow instance agent. It collects statistical data for workflows and involving agents, which can be helpful in future decision-making. A workflow instance agent takes care of an instantiated workflow. It plays the role of activating proper task agents according to the workflow definition, coordinating communication among task agents, providing the retrieval service of the intermediate processing results, updating some parameters of the current workflow. Task agents refer to both wrapper agents and special purpose agents. The wrapper agents play the role of exposing the existing tools or applications as an agent that behaves like an autonomous entity that can communicate with

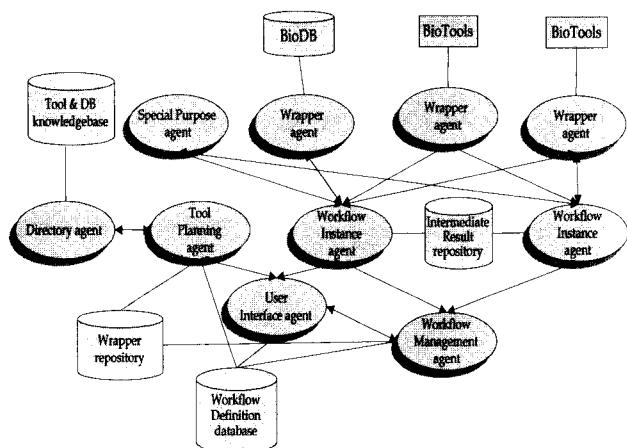


Fig. 2. The proposed multiagent system architecture

other agents and perform some processing. In addition, the wrappers enable the users to use tools without thorough understanding of their data formats and options. The special purpose agents are agents devoted to special data processing to which we cannot find appropriate tools. Using these kinds of task agents, bioinformatics works are carried out.

In this framework, there are several repositories as follows: Tools/DBs information repository contains the information about task agents which will be used in workflows. It is actually a database for the directory agent. Wrapper repository maintains the codes of wrappers and other task agents. Only when such task agents are needed, they are instantiated and start to work. Workflow definition database stores the already-developed workflow definitions. Intermediate result repository plays role of a shared memory among task agents. Task agents store their processing results to Intermediate result repository, allows user to retrieve intermediate results from the repository, and uses it to exchange messages like a mailbox.

3.2 Agent Design and Roles

Task agents

A task agent is instantiated by a workflow instance agent, receives messages about assigned tasks from a workflow instance agent and replies to it after processing the tasks. It maintains default parameter values for the designated tools or tasks. The default parameter values might be overridden by the messages from a workflow instance agent.

The messages to be handled in task agents are as follows: Control messages to assign a new task to a task agent, and to force an agent to terminate its processing. Query message to be used to retrieve a processing state of an agent. A task agent is in one of the following states: *Finished with Success*, *Finished with Failure*, *Timer-Out*, *Terminated*, *Processing*, *Failed with no proper input information*, and *Idle*.

Communication Module receives and sends messages and parses the received messages and extracts relevant information from them. Agent Control Module coordinates the processing of the task agent that works in goal-driven manner where a goal corresponds to a kind of messages handled by the task agent. For the intelligence processing, task agent contains

Inference Engine and thus it has both a Rule base to store the rules and an Working Memory to maintain the state information of the agent. Wrapper agent includes the rules which send a request to the wrapped tool and wait until it gets response from the tool. In case of task specific agents, they may contain multiple task-specific rules to perform specific processing.

Workflow Instance agents

A workflow instance agent takes care of a workflow instance. It receives a workflow description from the workflow management agent and produces its corresponding internal data structure and representation. It parses the XML message for the workflow description, generates its corresponding directed graph which is used to keep track of the progress of the workflow and to allocate constituent tasks to task agents, and represents the control structure information into a Rule base. The Rule base is used to make decision about what actions it will take. The states of tasks and local information like parameters are maintained in the Working Memory. According to the given workflow description, a workflow instance agent instantiates task agents. For each instantiated task agent, it creates an object containing the information about its task agent ID, current state, starting time, time-out flag, time-out value, associated rules IDs, and the pointer to the produced output. It passes the related information to the instantiated task agents. The information to be handed over a task agent is the merged result of successfully finished preceding task agent(s), and parameters set up by the user. A workflow instance agent finally generates an output by merging the outputs of successfully finished task agents.

A workflow instance agent handles the following messages: Control messages to assign a new workflow, to terminate a workflow in process, and to deactivate a workflow instance agent. Query messages to query the status of a workflow or some tasks composing a workflow. A workflow instance agent is in one of the following states: Finished with Success, Finished with Failure, Timer-Out, Terminated, Processing, and Idle.

Agent Control module instantiates task agents according to the workflow definition and monitors the progress of workflow. Working Memory maintains the state information of workflow. Rule base stores the rules about control flows of workflow.

Workflow Management agent

The workflow management agent creates a new workflow instance agent for a new workflow to be managed, passes the workflow information to the created agent, and deactivates the created agent later.

The messages to be handles are as follows: Control messages to hand over a new workflow instance to be managed, to terminate an ongoing workflow, and to deactivate a workflow instance agent. Query messages to ask the state of a specific workflow.

Planning agent

The planning agent provides an environment to generate workflow plans. Due to the inherent difficulty of workflow

planning, the current design of the planning agent keeps a collection of predefined workflow definitions for bioinformatics works, recommends candidates tools for tasks in a selected workflow definition using a rule-based system containing domain-expert knowledge, and allows users to update and edit existing workflow definitions and their control structures.

Directory agent

The directory agent registers the information about available tools and answers to the queries about tools. A new tool can be integrated into the multiagent system by registering its information to the directory agent and its corresponding wrapper agent to the wrapper repository.

User-Interface Agents

Through the user-interface agents, users are allowed to initiate a new workflow definition, to activate, terminate, and deactivate a workflow instance, to monitor the progress of a workflow instance, to retrieve intermediate processing results, and to tune the workflow instance while it is in progress. All these functions are embodied by exchanging messages with other agents.

3.3 XML-based Workflow Definition and Communication

To deal with the data format mismatches among tools and databases, the proposed approach employs an XML-based data representation. Both workflow definitions and messages among agents are all represented by a XML-based format. In a workflow definition, the employed representation allows the following control structures to accommodate flexible structures: sequence, fork, join, choice, merge, fork with choice(s), choice with fork(s), join with merge(s), and merge with join(s). Fig. 3 shows XML description to express a fork with choices structure, where three threads are forked up after finishing task ti but one of tj2 and tj3 is processed according to their conditions.

```
<fork>
<base> <tid> ti </tid> </base>
<outgoing nthreads = 3>
<thread> <tid> tj1 </tid> </thread>
<thread> <choice nested = true>
<flow><condition>condj2</condition><tid>tj2</tid></flow>
<flow><condition>condj3</condition><tid>tj3</tid></flow>
</choice> </thread>
<thread><tid>tj4</tid></thread>
</outgoing>
</fork>
```

Fig. 3 An XML description of workflow

The tag names of data entities for biological information are adopted from those used in NCBI system, and other tag names are defined by us. Each message has a unique message ID with which the response message is identified in the receiver agent. Even though task agents communicate with each other by sending messages, they do not tell which information is intended to the receiver agent. Task agents

know which pieces of information are needed to be extracted from the message. This strategy enables to develop wrapper agents without consideration of which agents to be invoked after the task agent. Thanks to this strategy, existing tools and new tools can be easily added on the multiagent framework.

4. Implementation

Each agent needs to respond to messages at any moment even though it is taking care of some pre-occupying task. Therefore, agents are implemented by Java as a process with multiple threads, two of which threads are for communication module and for agent control module. For the communication among task agents, the proposed architecture employs the strategy to inform the receiver of a new message by using Java Remote Method Invocation (RMI) and then the receiver pulls the message from the intermediate results repository. All messages have a unique ID and the ID is used to retrieve messages and to respond to the corresponding message.

To see the applicability of the proposed architecture, we have been implementing a prototype system to integrate bioinformatics tools for gene annotation, especially of a bacterium called *Sphingomonas Chungbukensis* DJ77[11]. Fig. 4 shows a workflow for gene annotation, which represents a sequence of bioinformatics tools to accomplish the work.

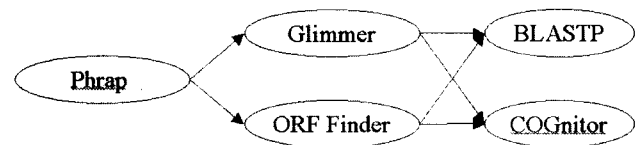


Fig. 4. A workflow for gene annotation

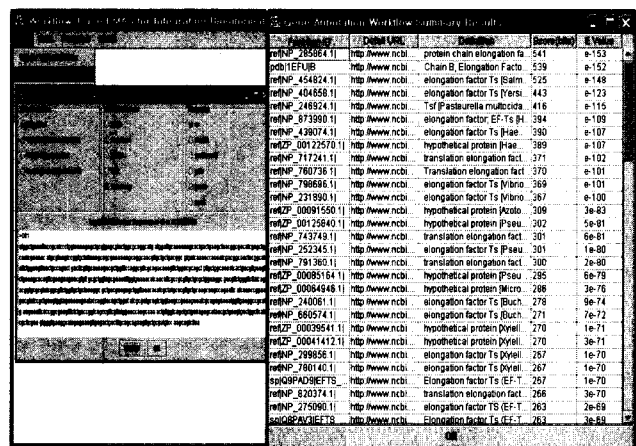


Fig. 5. An example of gene annotation

Fig. 5 shows an input for a sequence data of the above bacterium and tool options and its output for candidate gene functions. When a user selects the defined gene annotation workflow through the user-interface agent, the workflow management agent creates a new workflow instance agent for the gene annotation workflow to be managed and passes the

workflow information to the workflow instance agent. The gene annotation workflow instance agent takes care of the workflow execution. It takes inputs of protein sequence and tool options to use during workflow. The results of gene annotation workflow instance agent are candidate gene functions and this workflow ends.

5. Conclusions

This paper introduces a multiagent architecture for bioinformatics tool integration. In the proposed architecture, new tools can be easily added on by developing their corresponding wrapper agents. For the improvement of flexibility in agent communication, it uses XML-format messages in which all information needed by any receiver agent is contained. It manages assigned tasks as workflows and provides mechanism to control those workflows and to answer the queries. The proposed architecture is very flexible to integrate new tools thanks to its open multiagent architecture. We have been developing a prototype system based on the proposed architecture. In the experimental applications to a bacterium gene annotation, several biologists have expressed strongly positive expectations.

References

- [1] L. Hunter, *Molecular Biology for Computer Scientists, Artificial Intelligence and Molecular Biology* (L. Hunter, eds.), AAAI Press.
- [2] H. Schuschel, M. Weske, *Integrated Workflow Planning and Coordination. Proc. of the 14th International Conference on Database and Expert Systems Applications*, 2003.
- [3] F. Wan, S. K. Rustogi, J. Xing, M. P. Singh, *Multiagent Workflow Management, International Journal of Intelligent Systems in Accounting, Finance and Management*, Vol. 8, pp.105-117, 1999.
- [4] L. Chen, H. M. Jamil, *On using remote user-defined functions as wrappers for biological database interoperability, International Journal of Cooperative Information Systems*, Vol.12, No.2, pp.161-195, 2003.
- [5] K.-H. Cheung, P. Miller, A. Sherman, S. Strtmann, M. Schultz, et al. *Graphically-Enabled Integration of Bioinformatics Tools Allowing Parallel Execution, Proc. of the 2000 AMIA Annual Symposium*, 2000.
- [6] S. Moller, U. Leser, W. Fleischmann, R. Apweiler, *EDITtoTrEMBL: A Distributed Approach to High-Quality Automated Protein Sequence Annotation, Proc. of the German Conference on Bioinformatics*, 1998.
- [7] D. Frishman, K. Albermann, J. Hani, K. Heumann, A. Metanowski, A. Zollner, H.-W. Mwes, *Functional and Structural Genomics Using PEDANT, Bioinformatics*, Vol. 17, No. 1, pp.44-57, 2001.
- [8] K. Bryson, M. Luck, M. Joy, D.T. Jones, *Agent Interaction for Bioinformatics Data Management, Applied Artificial Intelligence*, Vol. 15, No. 10, pp. 917-947, 2001.
- [9] P. G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, R. Stevens, *TAMBIS Transparent Access to Multiple Bioinformatics Information Sources, Proc. of the Sixth International Conference on Intelligent Systems for Molecular Biology, ISMB98, Montreal, 1998.*
- [10] R. E. Fikes, N. J. Nilsson, *STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, Artificial Intelligence*, Vol. 2, pp. 189-208, 1971.
- [11] S.-J. Kim, J. Chun, K. S. Bae, Y.-C. Kim, *Polyphasic assignment of an aromatic-degrading Pseudomonas sp., strain DJ77, in the genus Sphingomonas as Sphingomonas chungbukensis sp. nov., International Journal of Systematic and Evolutionary Microbiology*, Vol.50, pp.1641-1647, 2000.

Bong-Ki Sohn

He received the B.S. degree in Department of Computer Science from Seowon University, Cheongju, Korea, in 1998. He received the M.S. degree in Department of Computer Science from Chungbuk National University, Cheongju, Korea, in 2000. He is currently working towards the Doctor degree in Department of Computer Science, Chungbuk National University, Cheongju, Korea. His research interests are in agent systems, machine learning and workflow system.

Keon-Myung Lee

He received B.S., M.S., and Ph.D. in Dept. of Computer Science from Korea Advanced Institute of Science and Technology(KAIST) in 1990, 1992 and 1995, respectively. He joined Chungbuk National University, Korea in 1996 and is currently an associate professor at School of Electronic and Computer Engineering. His research interests include data mining, agent systems, soft computing, bioinformatics, and computer security.

Hak-Joon Kim

He received B.S. in Dept. of Mathematics Education from Seoul National University, and received M.S. in Dept. of Computer Engineering from Samsil University. He had worked for DACOM from 1985 to 1997. He joined Howon University in 1997 as an assistant professor. He is pursuing Ph.D. in Dept. of Computer Science, Chungbuk National University from 2001. His research interests include artificial intelligence, and software engineering.