Agent Services-Driven Plug-and-Play in F-TRADE[1]

Longbing Cao, Jiarui Ni, Jiaqi Wang, and Chengqi Zhang

Faculty of Information Technology, University of Technology Sydney, Australia
{lbcao, jiarui, jqwang, chengqi}@it.uts.edu.au

Abstract. We have built an agent service-based enterprise infrastructure: F-TRADE. With its online connectivity to huge real stock data in global markets, it can be used for online evaluation of trading strategies and data mining algorithms. The main functions in the F-TRADE include soft plug-and-play, and back-testing, optimization, integration and evaluation of algorithms. In this paper, we'll focus on introducing the intelligent plug-and-play, which is a key system function in the F-TRADE. The basic idea for the soft plug-and-play is to build agent services which can support the online plug-in of agents, algorithms and data sources. Agent UML-based modeling, role model and agent services for the plug-and-play are discussed. With this design, algorithm providers, data source providers, and system module developers of the F-TRADE can expand system functions and resources by online plugging them into the F-TRADE.

1 Introduction

Information technology (IT) has been getting involved in finance more and more inseparably. Both IT and finance are getting more and more professional and technical. It is very hard for financial persons to devote themselves to both financial trading/research and IT for supporting their trading/research. On the other hand, powerful IT support can make financial trading and research more efficient and profitable. This is one way for IT persons to set foot in financial markets. We call it $IT_{R\&D}$-Enabled Finance. In consideration of this, we have built an agent service-based [1] infrastructure called F-TRADE [2], which can support trading and mining.

The main objective of building the F-TRADE is to provide financial traders and researchers, and miners on financial data with a practically flexible and automatic infrastructure. With this infrastructure, they can plug their algorithms into it easily, and concentrate on improving the performance of their algorithms with iterative evaluation on a large amount of real stock data from international markets. All other work, including user interface implementation, data preparation, and resulting output, etc., is maintained by this platform. For financial traders, for instance, brokers and

[1] F-TRADE is a web-based automated enterprise infrastructure for evaluation of trading strategies and data mining algorithms with online connection to huge amount of stock data. It has been online running for more than one year. It gets fund support from CMCRC (Capital Market CRC, www.cmcrc.com) for the Data Mining Program at CMCRC. The current version F-TRADE 2.0 can be accessed by http://datamining.it.uts.edu.au:8080/tsap, information can also be reached from http://www-staff.it.uts.edu.au/~lbcao/ftrade/ftrade.htm.

G.I. Webb and Xinghuo Yu (Eds.): AI 2004, LNAI 3339, pp. 917–922, 2004.

retailers, the F-TRADE presents them a real test bed, which can help them evaluate their favorite trading strategies iteratively without risk before they put money into the real markets. On the other hand, the F-TRADE presents a large amount of real stock data in multiple international markets, which can be used for both realistic back-testing of trading strategies and mining algorithms.

The F-TRADE looks also like an online service provider. As a systematic infrastructure for supporting data mining, trading evaluation, and finance-oriented applications, the F-TRADE encompasses comprehensive functions and services. They can be divided into following groups: (i) trading services support, (ii) mining services support, (iii) data services support, (iv) algorithm services support, and (v) system services support. In order to support all these services, soft plug-and-play is essential in the F-TRADE. It gets involved in plug in of data sources, data requests, trading or mining algorithms, system functional components, and the like. As a matter of fact, it has been a significant feature which supports the evolution of the F-TRADE and the application add-ons on top of the F-TRADE. In this paper, we'll focus on introducing the soft plug-and-play.

The basic idea of soft plug-and-play is as follows. The Agent service-oriented technique [1] is used for designing this flexible and complex software service. We investigate the agent services-driven approach for building a plug-and-play engine. The Agent UML is used for the modeling of the plug-and-play; role model is built for it. Service model for the plug-and-play is presented. The implementation of plug-in algorithms and system modules are illustrated as an instance of plug-and-play.

More than 20 algorithms of trading strategies have been plugged into the F-TRADE using the plug-and-play. All new-coming system modules supporting the migration of the F-TRADE from version 1.0 to 2.0 are logged on using the plug-in support. We have also tested remote plug-in from CMCRC in city of Sydney to the F-TRADE server located at Univ. of Technology Sydney (UTS). With the soft plug-and-play, both researches and applications from finance such as mining algorithms, system modules for technical analysis, fundamental analysis, investment decision support and risk management can be easily embedded into the F-TRADE.

The remainder of this paper is organized as follows. In Section 2, the modeling of the plug-and-play is discussed. Section 3 introduces agent services-driven plug-and-play from the following aspects: role model, agent services, and user interfaces. We conclude this study and discuss the future work in Section 4.

2 Plug-and-Play Modeling

As we have discussed in the above, plug-and-play gets involved in many functions and the evolutionary lifecycle of the F-TRADE. In this paper, we'll take the plug-in of an algorithm as an instance, and introduce the conceptual model, role model [3], agent services [1, 2, 4], and the generation of the user interface to plug-in an algorithm. In this section, we discuss the modeling of the proposed plug-and-play.

We use Agent Unified Modeling Language (AUML) technology [5] to model the agent service-based plug-and-play. In Agent UML, Package is one of the two techniques recommended for expressing agent interaction protocols. We use packages to describe the agent interaction protocols in plug-in support.

There are four embedded packages for supporting the process of plug-in: To Implement, To Input, To Register, and To Generate, as shown in Figure 1. They present the process of agent interactions in the plug-in activities. The package To Implement programs an algorithm in AlgoEditor by implementing the AlgoAPIAgent and ResourceAPIAgent. The To Input package types in agent ontologies of the programmed algorithm and requests to plug in the algorithm. The real registration of the algorithm is done by the package of To Register; naming, directory and class of the algorithm are stored into an algorithm base, and linkage to data resources are set at this moment. Finally, the input and output user interfaces for the algorithm are managed by package To Generate.

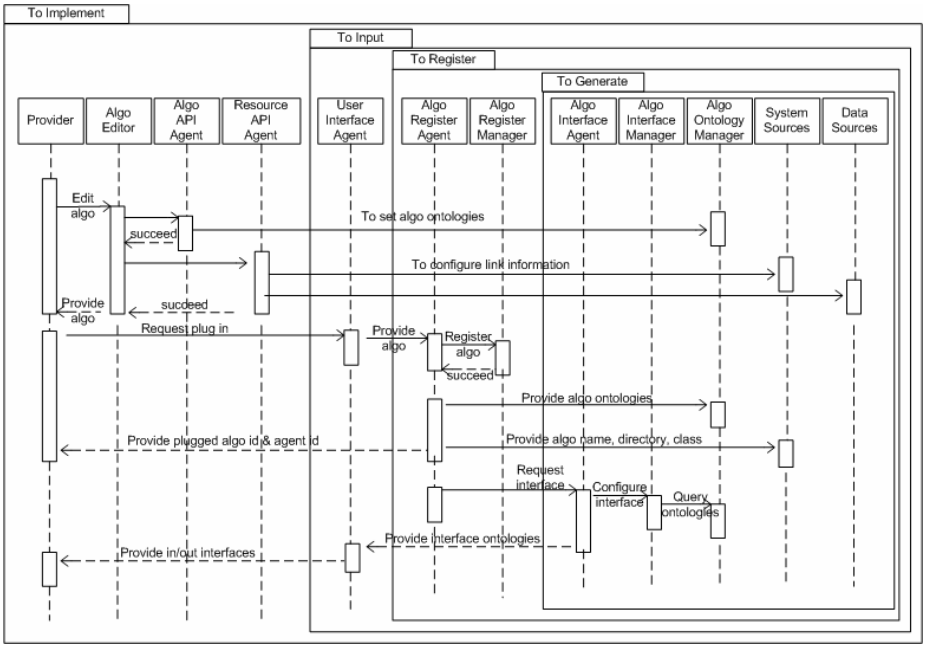

Fig. 1. Package diagram of the plug-and-play

3 Agent Services-Driven Plug-and-Play

In this section, we focus on discussing the analysis and design of the agent services-driven plug-and-play. We first introduce the role model for plug-and-play. Afterwards, the Agent service for plug-and-play is presented in details. Finally, the user interface for algorithm and system module plug-in is presented.

3.1 Role Model for the Plug-and-Play

In the agent-based F-TRADE, there is a role PLUGINPERSON which is in charge of the function of plug-and-play. Role model [2] can be built for the PLUGINPERSON,

which describes attributes of permissions, responsibilities, protocols and activities of the role. Figure 2 shows the role schema of PLUGINPERSON.

Role Schema: PLUGINPERSON

Description:
This preliminary role involves applying registering a nonexistent algorithm, typing in attribute items of the algorithm, and submitting plug in request to F-TRADE.

Protocols and Activities:
ReadAlgorithm, ApplyRegistration, FillinAttributeItems, SubmitAlgoPluginRequest

Permissions:

reads	Algorithms	// an *algorithm will be registered*
changes	*AlgoApplicationForms*	// *algorithm registration application form*
changes	*AttributeItems*	// *all attribute items of an algorithm*

Responsibilities
Liveness:
PLUGINPERSON = (ReadAlgorithm).(ApplyRegistration).
(FillinAttributeItems)+.(SubmitAlgoPluginRequest)

Safety:
- The algorithm agent has been programmed by implementing AlgoInterface agent and ResourceInterface agent, and is available for plug in.
- This algorithm hasn't been plugged into the algorithm base.

Fig. 2. Schema for role PLUGINPERSON

The objective of this role is to plug in an algorithm into the F-TRADE, which is nonexistent in the algorithm base. The agent playing this role will execute the protocol ReadAlgorithm, followed by the activities ApplyRegistration and FillinAttributeItems, and then executes the protocol SubmitAlgoPluginRequest. The role has rights to read the algorithm from non-plug in directory, and changes the application content for the registration and the attributes of the algorithm. As pre-conditions, the agent is required to ensure that two constraints in safety responsibility are satisfied.

3.2 Agent Services for the Plug-and-Play

In reality, many agents and services are involved in the plug-and-play in order to make it successful. There are three directly related agent services which handle the plug-and-play. They are the *InputAlgorithm*, *RegisterAlgorithm* and *Generate AlgoInterface* services, respectively. Here, we just take one service named *RegisterAlgorithm* as an example, and introduce it in details in Figure 3. More information about agent service-oriented analysis and design and about the plug-and-play can be reached from [1].

AgentService
RegisterAlgorithm(algoname;inputlist;inputconstraint;outputlist;outputconstraint;)
Description:
This agent service involves accepting the registration application submitted by role PluginPerson, checking the validity of attribute items, creating the name and directory of the algorithm, and generating a universal agent identifier and a unique algorithm id.
Role: PluginPerson
Pre-conditions:
- A request of registering an algorithm has been activated by protocol SubmitAlgoPluginRequest
- A knowledge base storing rules for agent and service naming and directory

Type: algorithm.[datamining/tradingsignal]
Location: algo.[algorithmname]
Inputs: inputlist
InputConstraints: inputconstraint[;]
Outputs: outputlist
OutputConstraints: outputconstraint[;]
Activities: Register the algorithm
Permissions:
- Read supplied knowledge base storing algorithm agent ontologies
- Read supplied algorithm base storing algorithm information

Post-conditions:
- Generate a unique agent identifier, naming, and a locator for the algorithm agent
- Generate a unique algorithm id

Exceptions:
- Cannot find target algorithm
- There are invalid format existing in the input attributes

Fig. 3. Agent service RegisterAlgorithm

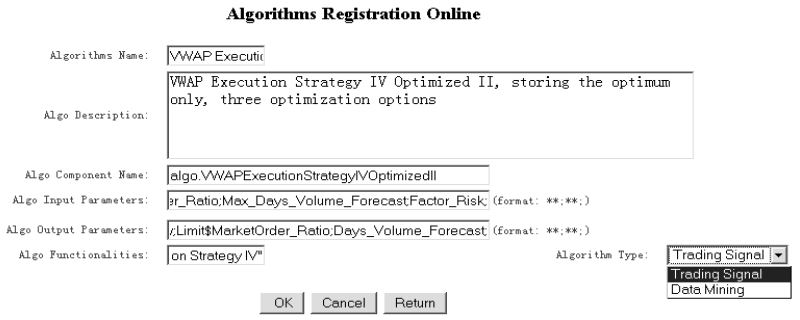

Fig. 4. User interface for an algorithm plug-in

3.3 Implementation

User interfaces must be implemented in association with the plug-and-play. Figure 4 shows the user interface for plugging an algorithm into the F-TRADE. Ontologies include all parameters of the algorithm, and specifications for and constraints on

every ontology element must be defined and typed here. After submitting the registration request, the input and output interfaces for this algorithm will be generated automatically, respectively. As we discussed before, plug-and-play can be used not only for algorithms, but also for data sources and functional agents and services.

4 Conclusions and Future Work

We have built an agent services-driven infrastructure F-TRADE, which supports trading and mining in international stock markets. It can be used as a virtual service provider offering services such as stock data, trading and mining algorithms, and system modules. In this paper, we have focused on introducing a key function called soft plug-and-play provided in the F-TRADE. We have studied the agent services-driven approach to building this function. Agent UML-based conceptual model and role model for the plug-and-play have been discussed. We also have presented the agent services and user interfaces for the plug-and-play. Our experiments have shown the agent services-driven approach can support flexible and efficient plug-and-play of data sources, trading and mining algorithms, system components, and even top-up applications for the finance to the F-TRADE.

Further refinements will be performed on how to make the plug-and-play more intelligent. The first issue is to support more user-friendly human agent interaction; ontology profiles will be enhanced in the interaction with user agents. The second investigation is to develop more flexible strategies for the mediation of agents and services, which can help search and locate the target agents and services efficiently.

References

1. Cao, L.B.: Agent service-oriented analysis and design. PhD thesis, University of Technology Sydney, Australia (2005) (to appear)
2. Cao, L.B., Wang, J.Q., Lin, L., Zhang, C.Q.: Agent Services-Based Infrastructure for Online Assessment of Trading Strategies. In Proceedings of the 2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IEEE Computer Society Press (2004)
3. Zambonelli, F., Jennings, N. R., Wooldridge, M.: Developing multiagent systems: the Gaia Methodology. ACM Trans on Software Engineering and Methodology, 12 (3) (2003) 317-370
4. Java Community Process. Java agent services specification. 5 Mar (2002)
5. Agent UML: www.auml.org