10 MULTI-AGENT BASED FRAMEWORK FOR LARGE SCALE VISUAL PROGRAM REUSE

Mika Karaila
Energy & Process Automation, Research & Technology Department
Metso Automation Inc. FIN-33101 Tampere, FINLAND
+358-40-7612563 mika.karaila@metso.com
Ari Leppäniemi
Ari.leppaniemi@metso.com

Today's application engineers are committed to the reuse of programs for performance and economic reasons. Moreover, they increasingly have to complement application programs with less information and in shorter time. The reuse of already implemented programs is therefore fundamental. We have implemented a process automation specific framework that supports reuse of our domain specific visual language. The visual Function Block Language is used for power plant and paper machine controls.
The reuse framework discussed in this paper relies in identification and usage of templates, which are used for generating actual application software instances.
The framework automates data mining with software agents collecting metadata. The metadata is send ahead to the receiver agent that stores the data into the central database. Another agent analyzes stored data and performs template matching. Again another agent is called for more detailed template match comparison. Although the database is centralized, the agents can be distributed and run in intranet. The framework implementation is pure java based and runs on JADE-FIPA agent platform

1. INTRODUCTION

Normally computer programs are written using textual programming languages. The more sophisticated or domain specific environment programming can be done in visual way. CAD-like programming environment will support different kinds of symbols and connections describing methods or relationships between the actual objects or instances.

The process automation specific visual language is used for making customer specific process control software (mass customization). The application software is created with visual Function Block Language (Figure 1. An example program). Later on function block loops are compiled to byte-code that is executed on the control system.

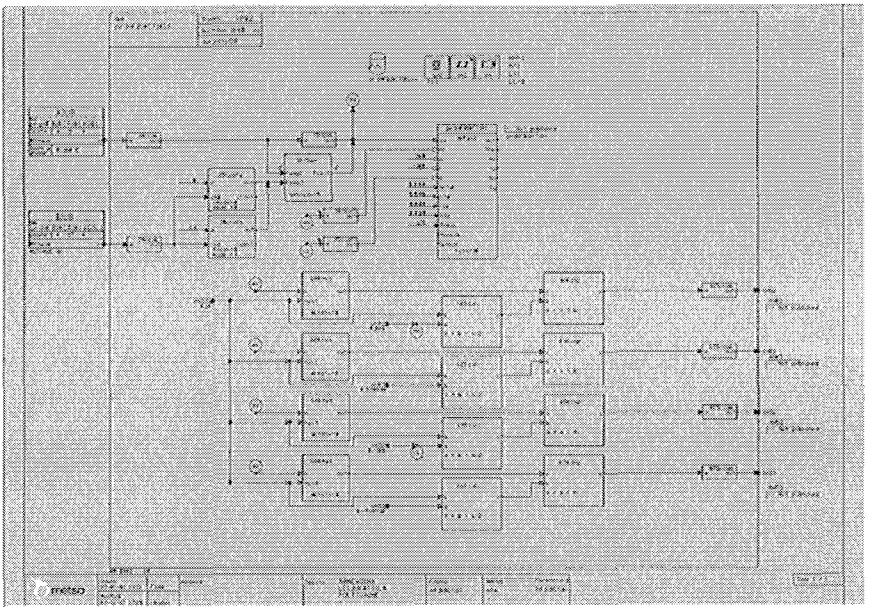

Figure 1 – An example program.

A function block is a capsulated subroutine. It will run functions according the given parameters and connections. Each parameter value reflects to component's functionality and connections are binding dynamic values to a function block. Each function block will allocate only the predefined amount of memory, because in process industry controls the real-time response and functionality must be predictable.

One function block diagram may represent actually many programs and document efficiently one program entity. If the program is larger, the program can be divided into multiple diagram pages with references.

In process industry each delivered project contains customer specific data and field devices. The program interface for the field devices is implemented with varying project specific addressing convention. An average project contains 5000-6000 loops / function block diagrams and over 20000 input/output-connections for the field devices.

When dealing with such a large amount of data, an efficient and successful project requires mass customization. Normally an engineer uses his/her own knowledge and earlier implemented programs in each project.

The basis of effective application implementation relies on usage of so called templates. Templates are application entities describing individual parts of process control software, without project specific definitions. Actual application instances are created when project specific data is combined to template. Our framework utilizes a practical way to identify and search these templates and implemented instances for project reuse.

2. FRAMEWORK ARCHITECTURE

The reuse framework is based on delivered project archives. These project library archives contain all implemented application solutions. Application instances and templates used are stored as DXF-files (Data eXchange Format) on directory structure corresponding to the projects process hierarchy. These archives are accessible for project engineers as mounted network disks.

The reuse framework developed binds these detached project libraries under single content management entity. The centralized content management solution stores only the essential application metadata from diagrams to content management server and allows the archived files remain in local project libraries. The stored metadata includes also links to actual application solution files.

The content management server contains search interface for finding appropriate application solutions for reuse. The stored metadata is used as search conditions and the desired solutions can be downloaded from local project libraries through the file links (Figure 2. Reuse framework, basic architecture).

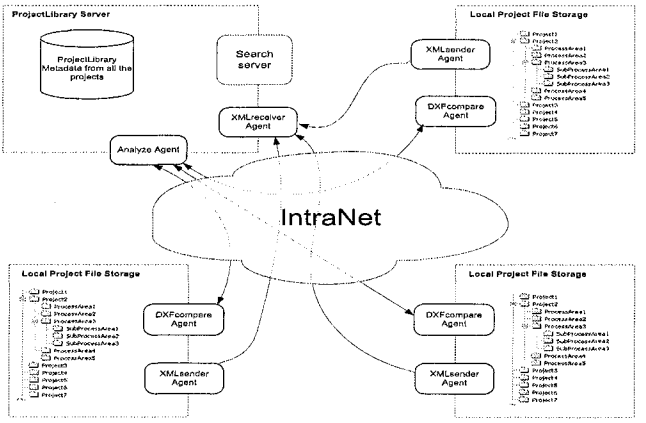

Figure 2 – Reuse framework, basic architecture

The metadata consists of the general part of data included to the diagram header: creator, modifier, creation time, modification time and other basic description fields. The general data is informative and practical as search conditions later on.

Moreover some data is read or calculated from the diagram objects: object count, primary function and statistical amounts of the following:

- Entities,
- Function blocks,
- Analog inputs / outputs,
- Digital inputs / outputs, and
- Connection type.

In addition, to a search criteria this kind of metadata is used for analyze and compare diagrams comprehensively in the database. Comparing actual diagrams files does the final and more detailed comparison. Since the actual file comparison is rather heavy process the preliminary comparison is essential for better performance.

Another performance related problem was solved by distributing tasks to agents running on local computers instead of centralized everything on content management server.

3. JADE-FIPA AGENT PLATFORM

The developed reuse framework is implemented on JADE-FIPA agent platform. JADE (Java Agent Development Framework) is a software development framework aimed at developing multi-agent systems and applications conforming FIPA standards for intelligent agents. It includes two main products: a FIPA compliant agent platform and a package to develop Java agents (JADE, 2004) (Bellifemine et al., 2004).

The agent platform can be split to several hosts, as has been done in developed reuse framework implementation. Only one Java application, and therefore only one Java Virtual Machine (JVM), is executed on each host. Each JVM is a basic container of agents that provides a complete run-time environment for agent execution and allows several agents to concurrently execute on the same host.

The JADE Agent class represents a common base class for user-defined agents. Therefore, from a programmer's point of view, a JADE agent is simply an instance of a user defined Java class that extends the base Agent class (Figure 3). This implies the inheritance of features to accomplish basic interactions with the agent platform (registration, configuration, remote management...) and a basic set of methods that can be called to implement the custom behavior of the agent (e.g. send/receive messages, use standard interaction protocols, register with several domains...).

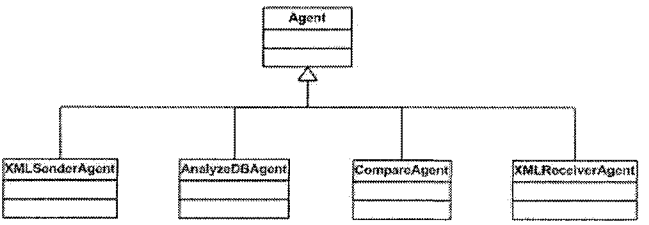

Figure 3 – Implemented application agents.

4. AGENTS & TASKS

The implementation of the developed multi-agent based reuse framework uses only simple JADE agent behavior classes. The agent communication is implemented as

common JADE agent communication language (ACL) that is based on java remote method invocation (RMI) -communication.

The developed framework includes four types of agents. XMLsender-agents are executed on project library hosts. They agents detect new directories in the local project library disks. Agent will automatically process zip-compressed files searching essential application metadata. The XML-coded metadata is enveloped into an agent message and passed ahead to XMLreceiver-agent on content management server (Figure 4). The XMLreceiver-agent will receive metadata messages and store the data into the database.

Periodically executed Analyzer-agent performs analyzes in content management database. Analyses include project template summary counts, template identification and template matching to generated instances. When the Analyzer-agent identifies a matching template it will inform Compare-agent that will then compare the template with generated instance files locally on project library hosts. After the comparison Compare-agent replies to Analyzer-agent that updates comparing results into the database.

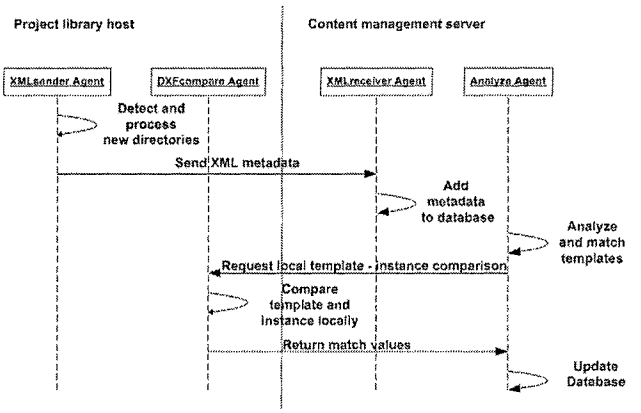

Figure 4 – Agent interaction.

4.1 XMLsender Agent

The XMLreport-agent detects new directories in the given environment according to last modification date of the file. Special JADE WakerBehavior is used to execute new directory search at regular intervals just after a given timeout is elapsed. New archived files are identified and processed. The searched metadata is enveloped into an agent message and passed ahead to XMLreceiver-agent over intranet. The essential data is also stored locally to XML-files for possible later use.

4.2 XMLreceiver Agent

XMLreceiver-agent receives XMLsender-agent's XML-coded metadata messages. Special JADE CyclicBehaviour class is used to control the message receiving process. The XML-messages are parsed to common attribute-value pairs and stored

to content management database. XMLreceiver-agent is also able to update already existing metadata entries into the database.

4.3 Database Analyzer Agent

Periodically executed Analyzer-agent is used to process the database-stored metadata. Agent's main task is to identify and match templates used to generated instances. The identification process uses the similarities between primary function blocks, function block amounts, and certain function block attributes to match templates to instances. When Analyzer-agent identifies matching template-instance pairs it will request Compare-agent for more accurate template match comparison. Analyzer-agent will get comparison results from the Compare-agent and update the result value to the database.

Analyzer-agent is also used to perform certain project specific analyses. For example, the project summary analyses include etc. different loop type and IO connection counts and complexity numbers that can be used to support decision-making.

4.4 DXF Compare Agent

Compare-agent receives DBanalyzers matching requests. The agent compares the actual template and instance files and calculates match values. When no structural changes between the template and instance exist the match value equals 100. That is, only different parameter values may exist. Each structural chance diminishes match value with a certain amount. For example by deleting and adding one symbol the match value is decreased by two to 98.

Function blocks are compared first at element level: new and removed elements are identified. For the common existing elements, parameter values are compared. Most critical changes are structural changes that are actually viewable as added or removed elements. The comparison can also be visualized with different colors indicating added and removed elements and changed parameter values.

5. SEARCH CAPABILITIES

The versatile search tool is essential for engineers to find and download good application solutions for reuse. The developed framework contains Tomcat server based search tool enabling versatile search options. The search tool implementation takes advantage of java-bean, JSP-page and applet technologies and thus the users can access the search tool without any external program installations by using only a web browser.

The search interface (Figure 5) allows users to search application solutions according to collected metadata and agent performed analysis. The search can be focused to certain process areas and projects. The more detailed search criteria can include e.g. the main function of the program (function block like pid-controller or motor controller), the IO connection type used and the application creation time.

The search results include all the matching application solutions or templates. Each search match can be taken to more detailed inspection. The more detailed view represents all the relevant metadata of the current loop.

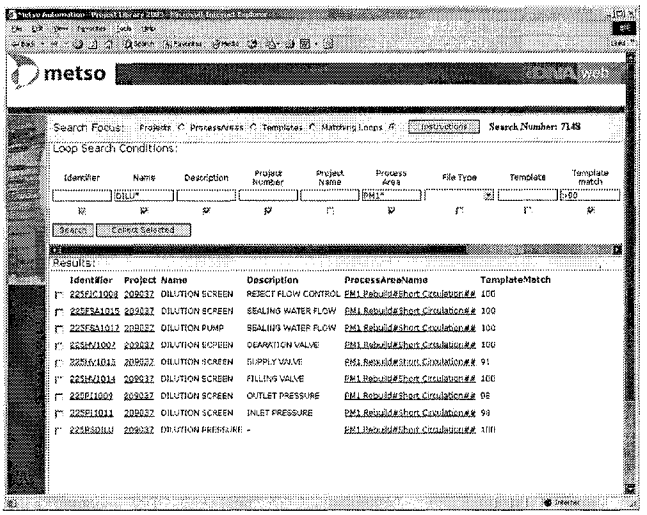

Figure 5 – Search interface.

The templates and instances used can also be graphically examined by using the DXF-viewer applet. DXF-viewer functionalities include also panning and zooming.

The search interface contains also possibilities to inspect complete project and process area analyses. Analyses include information about different kind of implemented IO connection and application loop amounts. These analyses serve also as the project summaries when complete projects are archived.

Analyses also contain essential key figures estimating project complexities and implementation methods. For example, the project summary analyses include complexity values that can be used to support decision-making concerning schedules and workloads for similar projects in future. Also marketing may use complexity figures as a support when pricing future projects.

6. CONCLUSIONS

The agent based reuse framework developed has enabled an efficient way for users to archive and share implemented solutions and knowledge. The automated agent-based application solution filing process together with search tool has proven to be an efficient and practical solution.

The current content management database size exceeds now 800 Mega bytes. Database contains over 200 projects and links together over 30 Giga bytes of compressed files. Metadata has been archived from approximately 600000 function block diagrams. The usage of search tool has become a part of application engineers working manners. Approximately 2000 searches are performed monthly.

Our experiences with JADE-FIPA agent platform have illustrated the flexibility and suitability of the multi-agent technologies to function in local and distributed environment. Moreover, the agent platform has been very stable. Although the current agent messaging has been tested only in local office network, the JADE supports also HTTP communication that also enables communication over Internet.

The analyses and template-matching processes implemented have allowed us to study more the real problem of finding a higher abstraction level for mass customization.

7. RELATED WORK

A similar agent framework is used also for traditional software reuse (Erdur et al, 2000). The framework is more advanced and contains more agents than this implementation. Another good study is related more to our visual language and the metadata handling. Younis and Frey have made a survey how existing PLC programs can be formalized (Younis et al., 2003). Even our template matching is on general level it binds instances and templates together for reuse and in future also for reverse engineering capabilities.

8. FUTURE DEVELOPMENT

The future development work of the reuse framework includes uploading new feature templates and instances to project library hosts. With this feature we are striving to get applications reused more quickly and efficiently.

The analyses methods will be further developed to use enhanced algorithms to match a template. Also, the differences between instance and templates can be already visualized in the CAD based engineering tools and it should be added also to search interface's applet window. This can be very good way to make first analysis from the variation similarities.

9. ACKNOWLEDGMENTS

We would like to thank professor Tarja Systä (Tampere University of Technology) for the support, Timo Kuusenoksa for the coding and many other people who provided helpful comments on previous versions of this document.

10. REFERENCES

1. Bellifemine Fabio, Caire Giovanni, Trucco Tiziana, Rimassa Giovanni, Jade Administrator's Guide http://sharon.cselt.it/projects/jade/ Accessed 23.1.2004.
2. Erdur Riza Cenk, Dikenelli Oguz, Agent Oriented Software Reuse (June 2000)
3. JADE-FIPA, http://sharon.cselt.it/projects/jade/ Accessed 23.1.2004.
4. Younis M. Bani, Frey G., Formalization of existing PLC Programs: A Survey (July 2003)