Decision Support System for Rocket Launch Using Semantic Web Services

Seiji Koide, Shohei Misono, Norikazu Shimada, Masanori Kawamura, Hiroshi Nishio
Galaxy Express Corporation
18-16, Hamamatsu-cho 1-chome, Minato-ku, Tokyo, Japan
+81-3-5733-7120
{koide, misono, shimada, kawamura, nishio}@galaxy-express.co.jp

ABSTRACT
In this paper, we describe an overview of the decision support system for rocket launch operation, focusing around Semantic Web Services that allow us to develop complex, large-scale, and globally distributed systems. In rocket launch operation, several parties who share interests and problems participate together in problem solving process against contingent events. The support systems for rocket launch operation that involve different participants must allow different knowledge, diverse business rules, and heterogeneous values of parties. Whereas the Service Oriented Architecture (SOA) once seemed to be a solution, it has become clear that it does not satisfy our requirements. We took up a challenge of building the support systems by means of Semantic Web Services. We have developed an OWL processor, a web agent, and various ontologies for rocket launch operation. We have designed the system architecture in which sub modules are built as Web Services so that user agents integrate them and perform agent tasks in order to help users according to various ontologies.

Categories and Subject Descriptors
D.2.12 [Interoperability]: Distributed Objects – *Web Services, Semantic Webs, and decision support system.*

General Terms
Management, Documentation, Design, Reliability.

Keywords
Web Services, Semantic Webs, decision support system

1. INTRODUCTION
In rocket launch operation, people have a deep concern for smooth launching operation within available time. The time for launch or time window is strictly restricted by various reasons, e.g., satellites maneuvering, the avoidance of disturbance for fishermen's work, weather conditions, etc. In a contingent event, several parties, i.e., satellite makers, rocket makers, and a launch operation service company collaborate to solve problems in the term of launch. Therefore, the corroborative decision support system in rocket launch operation that helps launch operators and expert engineers from different parties who participate in distance is valuable.

Recently the Service Oriented Architecture (SOA) has been promoted and advertised. SOA is a system integration style whose goal is to interact with loosely coupled services by software agents. It allows us to build systems globally distributed over the Internet. The XML and SOAP-RPC Web Services underpin SOA technology. Firstly we developed a prototype system in which every system modules, including an agent, were built as Web Services [1][2]. Whereas we have been convinced of the capabilities of Web Services, we have found that the SOA in enterprise-level does not satisfy our requirements such that participants from different parties collaboratively work in problem solving process and the system must be flexible so that it allows frequent change of parties and sustainable system development required by frequent change of rocket structures. It was obvious in SOA that we must continue to rewrite the main agent program and the interface as long as the launch operation and facilities continue to vary.

The Semantic Web Services are to annotate Web Services with markup languages like OWL and OWL-S so that web agents can discover, compose, and perform Web Services. While Semantic Web Services were still challenging technology at the time, we gained the insight of rationale for us to tackle with it rather than Web Services, because Semantic Web Services appeared to be more flexible upon the firm base of ontologies. In this paper, we describe an overview of the decision support system for rocket launch operation, focusing around Semantic Web Services that enable to develop flexible, complex, large-scale, and globally distributed systems with various kinds of participants from different parties.

2. OVERVIEW OF SUPPORT SYSTEM
2.1 The Global Overview
Figure 1 shows the global overview of the support system, in which several rocket makers and the launch business company

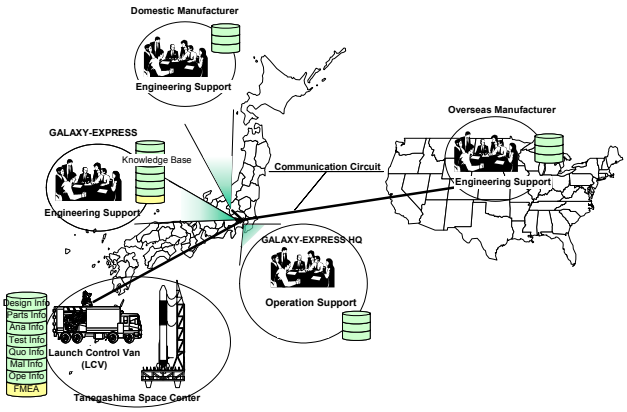

Figure 1. Global Overview of Operation Support System

who share common interests and problems collaboratively work over the network in contingencies. The system delivers sharable data from Tanegashima Space Center to support engineers in distance and ensures collaborative work through online-meeting and the delivery of ITV camera image at the launch spot through the network. Databases that include party specific data are located at each party site, and users can access any database at any site within the limitation of their own authority. Note that the Wide Area LAN and the Internet-VPN secure the communication line in our application.

In addition to building the above corroborative network environment, there are two main R&D themes and six sub themes in our project.

- Large-scale Knowledge Databases
  - Distributed-database Retrieval
  - Multi-media Data Management
  - Ontology of Rocket Launch Operation
- Trouble-shoot Algorithm
  - Case-Based Monitoring Algorithm
  - Model-Based Diagnosing Algorithm
  - Fusion Algorithm

In this paper, we mainly describe Web Services, ontologies, and a web agent out of the above development issues.

2.2 Architecture for Operation Support

We have designed the whole architecture of the operation support systems that will be deployed mainly at Tanegashima Space Center. **Figure 2** illustrates the software modules and the connections among them in the support system. In the figure, most of small square boxes connected to other small boxes in other modules are Web Services and an agent system is pictured in the center of the figure. Please grasp the whole complexity rather than the details.

In this system, the Demand and Notification Accept Service (at the bottom in the agent system) keeps to run without the rule of the agent and with several sub modules, e.g., the Data Delivery System, the Advanced Monitoring System, the Model-Based inference System, and the Case-Management System (each is located at the bottom). Note that the measurement and control data are transmitted to the Data Delivery System from the Off-site Monitoring System (at the right and bottom corner) of the Launch Control Ground Facilities.

The Advanced Monitoring System monitors the behavior of the launch ground facilities and the control system. When it detects some anomaly in the plant behavior, it informs the agent about the anomaly, and then the agent (actually the Demand and

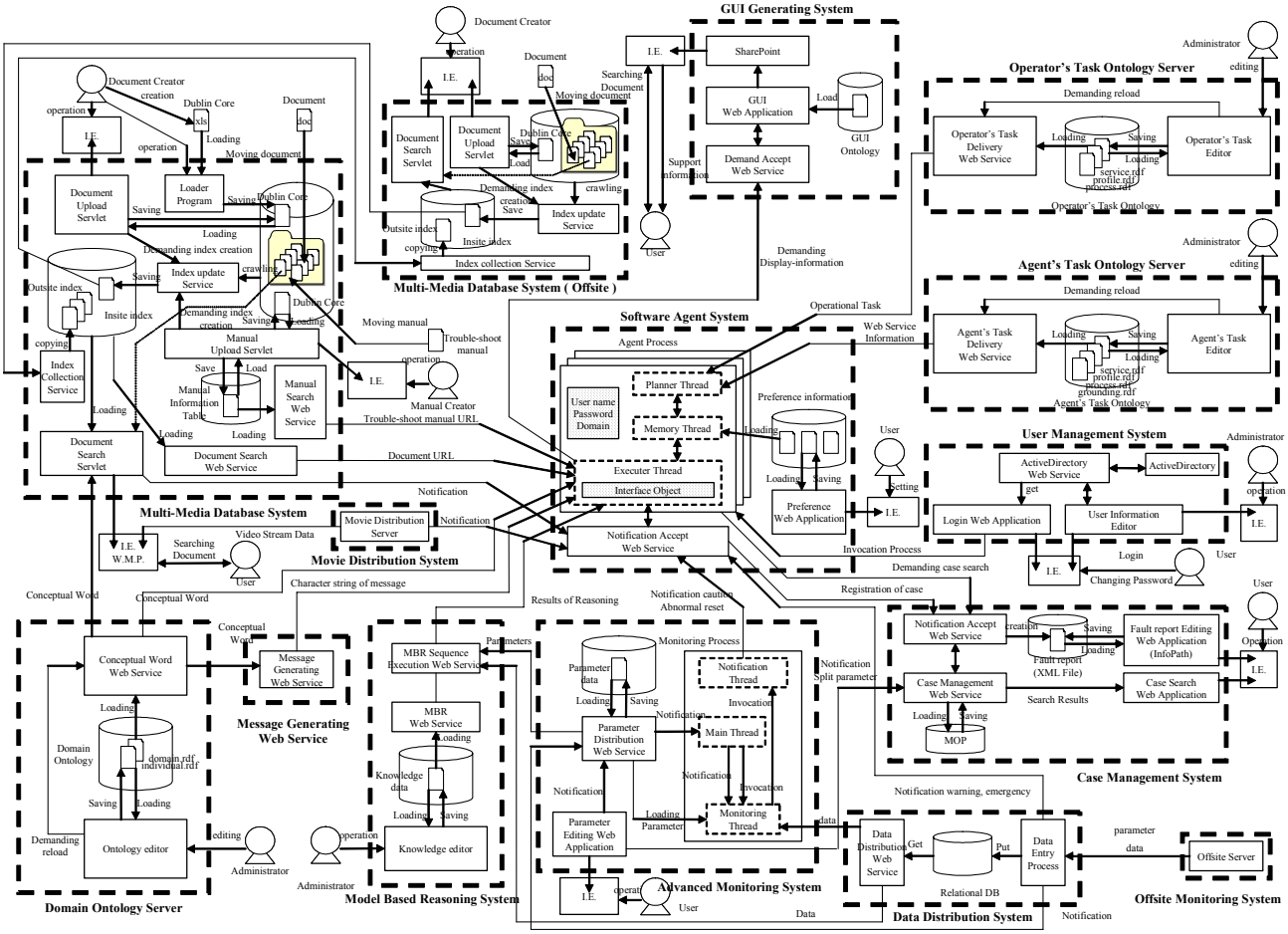

Figure 2. Whole System Architecture for Operation Support System

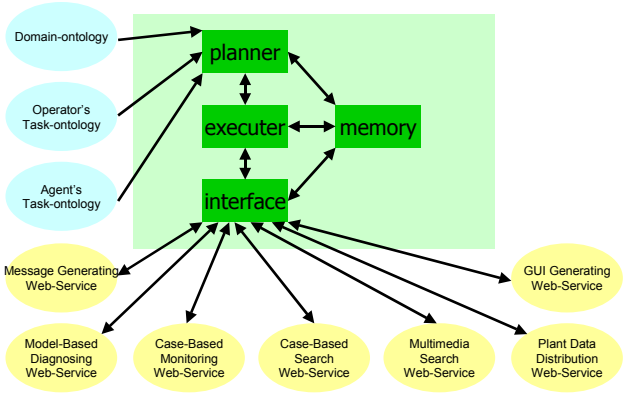

Figure 3. Agent Architecture

Notification Accept Service) invokes the Model-Based inference System. The Model-Based inference System diagnoses the anomaly with abnormal measurement and control data received from the Data Delivery System. The result of diagnosing is also notified the agent of.

When a user in distant logs in to the User Management System (the right side in the middle-high) through a usual web browser, a user-dedicated agent process is spawned and the agent mediates the user and the Demand and Notification Accept Service. The single sign-on and the user management functionality are realized on top of MS Active Directory. The agent process inherits user's authority. We are implementing the user preference functionality and the access control functionality with user authority. Ontology servers (pictured at the right-top corner and left-bottom corner) provide various ontologies, i.e., operator-task ontology, agent-task ontology, domain ontology, and so on.

The Multimedia Database System at Tanegashima Space Center (pictured at the left and top corner) houses and manages the operation manuals, the trouble-shooting manuals, and the past records of troubleshoot. Another Multimedia Database System at another site is pictured at the next of the one at the Space Center. We have developed the desktop information indexing and retrieval system named GXFINDER on top of Lucene-Ja, (Lucene is a search engine from Jakarta.apache.org). Each of the distributed GXFINDER shares the indices at each site and the retrieval is carried out in peer-to-peer fashion, while each indexer crawls within its owl site only.

2.3 Agent System Architecture

Figure 3 depicts the agent architecture and the relation between several ontologies. A web agent consists of four modules, i.e., a planner, an executer, the memory, and the interface. The planner inputs the domain ontology and the operator task ontology. Thus, the agent captures the operation task flow. The tacit work of agent should be described onto the agent task ontology, which may include support tasks as counterparts of operation tasks in the operator task ontology. When a user demands a specific task to the agent, the planner makes a plan that achieves the goal from scratch (if there is no evidence in memory) or retrieves a similar abstract plan from memory (if exists) and adaptively modifies it for the current situation. The plan made is stored into the memory. We have concluded that the planner must be a partial order planner like UCPOP or SIPE-2 through the development of STRIPS-like planner for service composition in our study. The Hierarchical Task Network Planner (HTN) may be useful to avoid the plan break with the interaction to users.

The memory is a Case-Based memory like MOP (Memory Organization Package) [3] by Schank, et al. The plan cases and the results of the performance in the past are stored in memory. We are developing the MOP functionality on top of SWCLOS [4], a Semantic Web Processor on top of CLOS (Common Lisp Object System).

The executer retrieves a plan from memory and performs tasks according to the plan. Through the effort of making OWL and OWL-S files of 6,600 lines for the ISWC2004 exhibition presented by the Galaxy Express Corporation, we have found that writing OWL-S and letting the agent directly perform OWL-S ontology is very labor consuming. Note that the validity in OWL semantics is automatically justified by SWCLOS. The labor-consuming problem is in OWL-S. It is nonsense that the executer directly reads and performs three OWL-S files, i.e., Service, ServiceProfile, and ServiceModel which are written in RDF/XML. Therefore we have decided the development of alternative task language for OWL-S for agent's direct performance. In the new task language, a task is an ensemble of one Service, relevant ServiceProfile, and relevant ServiceModel in OWL-S. Terminal nodes in task taxonomy are atomic tasks that are callable as Web Services. The task has the interface definitions of service, i.e., inputs, outputs, preconditions, and effects, but may have no body. A task that has no body is an abstract task like SimpleProcess, and the task that has a body is a sort of a CompositeProcess.

The executer for the new task language will be developed as Scheme language-like interpreter. It not only evaluates and executes the concrete tasks but also instantiates the abstract plans that is generated by the planner, depending on the current data in memory, then creates and executes the instantiated tasks. It can be supposed that the abstract tasks make a task flow or a business flow, and the instantiated tasks that underlie the current world are a sort of program.

The interface part reads the OWL-S grounding file for Web Services and hides the grounding details from the executer and the memory. Various Web Services are actually invoked by the interface part. The resource accessing management is going to be implemented in the interface.

3. Ontology for Operation Support

3.1 Approach and Usage

3.1.1 Domain Ontology

We have two approaches to build the domain ontology and two usages. One of building approaches comes with the target system analysis for monitoring and diagnosing system and the other is from keyword collection out of a number of technical documents.

In order to develop the Advanced Monitoring System and Model-Based diagnosing system, we analyze the control system and the ground facilities. While we are constructing the qualitative plant models of facilities, this work process is quite same as the knowledge acquisition process, i.e., reading documents, interviewing design experts, and so on. Thus, we can obtain the diverse domain ontologies, i.e., the device ontology (for example 770 items from one apparatus unit), process ontology (for example 330 items from one apparatus unit), abnormality

ontology (for example 130 items from one apparatus unit), as bi-products.

On the other hand, we have developed GXFINDER, a flexible desktop information retrieval system, in which we use ontology for flexible conceptual information retrieval as well as keyword retrieval and bibliographic meta-data (extended Dublin Core classes) information retrieval. With the morphological analysis of contents of 10,500 technical documents such as design requirements, design specifications, design charts, technical reports, memos, etc., we have obtained 24,000 Japanese words. Those keywords include general nouns and verves that are appeared in the documents. Now we are developing a semi-autonomous ontology construction tool, which is originated from DODDLE by Yamaguchi et al., that arranges words to IS-A hierarchy with referencing EDR, Japanese electronic dictionary, and WordNet.

The former ontology includes a lot of instance-level terminology which is specific to our domain, and these terminology are closely coupled each other. On the other hand, the latter ontology includes upper-level words and they are arranged hierarchically. We are going to describe the former ontology principally with owl:ObjectProperty and owl:onProperty restrictions, and the latter ontology principally with rdfs:subClassOf. We will integrate the widely spreading instance-level ontology and the vertically spreading class-level ontology.

3.1.2 Task Ontology

The operator task ontology is the ontology of operation task flow of some kinds of operators. The operation clue members are classified to Launch Commander (LCDR), Task Commander (TC), Task Leader (TL), and field workers. TCs and TLs are dispatched for one of ground systems such as Mechanical System, Electronic System, Facility, and Safety. These members' tasks are different but closely relate each other through the control process and the operation procedures. The minimal size tasks in our operation are designed as so-called two-page plans. Therefore, the minimal operator task in task ontology is formalized according the two-page plans. Each two-page plan has some constraints, i.e., before-task constraints, preconditions, time constraints, etc., and effects. Thus, the operation task ontology can be formalized with OWL-S and/or the alternative task ontology language.

The agent who serves an operator reads the specific operator's task ontology according to classified operator roles, and captures the operation task flow of the user. With the progress of launch process according to the task schedule, the agent traces the scheduled process through monitoring the measurement and control data. Thus, the agent can understand what is going on in the current process and what should be done for the user.

On the other hand, the agent task ontology for operators is abstract business flow for agents. As the agent task flow is closely coupled with the user's task flow, the agent task ontology contains counterparts of the operator task ontology. Individual agent task flows are different at different operator classes. However, we guess that the meta-level descriptions of individual agent task flow are same in the sense of decision support. Namely, it is the upper ontology of the decision support agent. It is very challenging to make the upper ontology of support agent clear through the operational task analysis and the operation task ontology.

The agents for support engineers do not have any fixed task ontology. Q&A or Command & Execution functionality is realized and utilized, while the monitoring and diagnosing results are alarmed to support engineers, too.

4. CONCLUSIONS AND FUTURE WORK

We summarized the decision support system for rocket launch operation that is built with a number of Web Services and an agent. The motivation of Semantic Web Services is a flexibility to ensure the sustainable development and integrate heterogeneous Web Services in future. We expect that a new function is made available by simply adding a new Web Service into the environment, or only a Web Service plus small piece of ontology.

We have developed an on-demand image delivery system, a flexible desktop information retrieval system, a Model-Based diagnosing system, and a Semantic Web processor. We have designed the operation support network environment, the whole system architecture, and agent architecture. We are now constructing the web agent, diverse domain ontologies and task ontologies.

5. ACKNOWLEDGMENTS

We thank Prof. Mizoguchi at Osaka University, who is a co-researcher at the ontology development, and Prof. Gofuku at Okayama University, who is a co-researcher for Cause-Identifier with Model-Based Reasoner. We also thank Emeritus Prof. Ohsuga at Tokyo University, who is the chairperson of the Technology Evaluation Committee of this project. Prof. Yamaguchi at Keio University has originally developed DODDLE, and we are co-working to develop DODDLE-J and the document-oriented ontology building tool.

This paper has been prepared as a part of the Japanese IT project entitled "Building a Support System for the Large-Scale Operation System using Information Technology" under the contract with the Ministry of Education, Culture, Sports, and Technology (MEXT).

6. REFERENCES

[1] Koide, S., et al. Operation-Support System for Large-Scale System Using Information Technology. *Proceedings of the 5th Int. conference on Enterprise Information Systems (ICEIS2003),* 4 (2003), 430-437.

[2] Koide, S., et al. Coping with Large-Scale Complex Systems by Agents, Semantic Web Services, and Ontology. *Proceedings of the 5th Int. conference on Enterprise Information Systems (CSIMTA'04),* (2004), CSG1.1.

[3] Riesbeck, C. K. and Schank, R. C. *Inside Case-based Reasoning.* LEA, NJ, 1989.

[4] Koide, S. and Kawamura, M. SWCLOS: A Semantic Web Processor on Common Lisp Object System. *Int. Semantic Web Conference (ISWC2004)*, demo paper, http://iswc2004.semanticweb.org/demos/32/.