BULGARIAN ACADEMY OF SCIENCES
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS • 2022 • Vol. **78**, pp. 23-34
p-ISSN: 2738-7356; e-ISSN: 2738-7364
https://doi.org/10.7546/PECR.78.22.03

Reliable Service-Oriented Architecture for Nasa's Mars Exploration Rover Mission

Puhiza Iseni[1], *Festim Halili*[1]

Author addresses:
[1] *Department of Informatics, Faculty of Natural Sciences and Mathematics, University of Tetovo, North Macedonia*
Email: p.iseni211545@unite.edu.mk

Abstract:
The Collaborative Information Portal (CIP) was created by NASA's Ames Research Center and Jet Propulsion Laboratory (JPL) for NASA's Mars Exploration Rover (MER) mission. Both MER and CIP have performed beyond their expectations. Mission managers, engineers, scientists, and mission researchers conduct CIPs within JPL mission control, and scientists conduct CIPs in their laboratories, homes, and offices. Users are securely connected through the internet. Since the mission took place at the time of Mars, the CIP displayed current time in different time zones of the planet Mars and Earth and introduced staff and event schedules with Martian time scales. In this paper, we present the MER and CIP mission summaries. We show how the CIP helped to meet some of the mission needs and how people used it. We discuss the criteria for choosing its architecture, web services (current time ratio between planet Earth and planet Mars, conversion of Martian (Sol) and Earth Day, and direct connection to planet Mars) and describe how developers made the software so reliable. CIP reliability did not come by chance, but was the result of several key design decisions.

Keywords: SOA, JPL, MER, NASA, CIP, Rover, Web Services.

1. Introduction

In January 2004, the Mars Exploration Rover (MER) mission successfully deployed two robotic geologists – Spirit and Opportunity. Rovers are equipped

with powerful cameras and scientific instruments that send data and photos back to Earth, where they are processed and stored on file servers by ground-based computers. After analyzing the files, NASA scientists have concluded that liquid water existed on the surface of Mars in the distant past.

The mission is not just about rovers and data. On Earth, humans manage missions, send commands to rovers, and analyze downloaded information. NASA's Ames Research Center scientists and engineers collaborated closely with mission management at the Jet Propulsion Laboratory (JPL) to develop mission-supporting applications. The Collaborative Information Portal (CIP) assisted mission personnel in carrying out their day-to-day tasks, regardless of whether they were working within the mission control or in the scientific areas of the JPL, or in their homes, schools, or offices.

With a three-tier, service-oriented architecture (SOA) – client, middleware, and data warehouse – built using Java, industry standards, and commercial software, CIP provides secure access to mission and service schedules. data and images downloaded from Mars. Users may execute CIP client tools on Windows, Unix, Linux, and Macintosh platforms, and it met JPL's mission criteria for capability, scalability, and stability [1, 2, 3].

2. Related Work

The literature research has revealed many scientific papers related to NASA's Mars Rover Exploration Service Oriented Architecture. I will introduce some of them, related to our project.

Ronald Mack, Joan Walton, Leslie Keely, Dennis Heher and Louise Chan present MER and CIP mission summaries. Demonstrate how the CIP aided in meeting some of the mission's needs and how it was used. Criteria for choosing its architecture and describe how the developers made the software so reliable. CIP reliability did not come by chance, but was the result of several key design decisions [1].

The study by Youssef Bassil describes a service-oriented architecture for distributed multi-robot systems based on web services for remote control and manufacturing message specification (MMS) to exchange data between modules of the systems. Its purpose is to monitor and control software design using the concepts of Unified Modeling Language (UML) and MMS [4].

RP Bonasso, D. Kortenkamp, DP Miller and MG Slack introduced one of the earliest models is the 3T robotic design which consists of three independent communication layers: The first layer is the reactive capabilities layer which consists of a set of behaviors reactive handled by a skills manager who is responsible for initiating and stopping specific skills based on several input

sensors. The second layer is the ranking layer which is responsible for generating a set of skill requirements for the reactive skill layer. The third layer is the discussion layer which is responsible for generating plans based on the mission requirements. The 3T software architecture was created to control the effect of autonomous systems in a flexible and robust way [5].

Eric Colon, Hichem Sahli and Yvan Baudoin describe the control robots from CORBA known as CoRoBa, a platform designed to allow the integration of distributed robotic control, sensors and computing components. It is basically based on CORBA (Common Object Request Mediation Architecture) which provides a standard medium program for connecting object-oriented components regardless of their internal technologies [6].

The study conducted by the iRobot company describes another CORBA-based software architecture for building and integrating distributed object systems built using different languages [7].

M. Quiqley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng Their study describes ROS (Robot Operating System) as an open-source architecture that supports modular and distributed software components for robotic software. The ROS architecture contains the interaction between entities, the transmission of messages and the concepts of services. The ROS design does not contain a central middle program to coordinate between different nodes of the system [8].

The study conducted by Ivar Jørstad, Schahram Dustdar and Do Van Thanh describes the use of SOA for building collaborative services. The paper describes the Service Oriented Architecture and then the collaboration services. A general model of a collaborative service is analyzed to identify the basic functions of collaboration [9].

Elias Sinderson, Vish Magapu, and Ronald Mac present our approach to mitigating the challenges faced by robots on Mars, concluding with a brief overview of mid-range and web services for the Rovers Mission Collaborative Information Portal. NASA Mars Exploration [10].

3. Service-Oriented Architecture (SOA)

Service-oriented architecture (SOA) is a model for system development based on a loosely integrated set of services that can be used within multiple business areas. SOA also presents an approach and practice for building IT software systems using interoperable services. These services are loosely connected software components that summarize functionality and are available to be accessed remotely from client applications over a network or the Internet. Web services and Enterprise Service Bus (ESB) form the foundation of SOA [4].

3.1. Web Services

W3C defined, a web service is a component of software designed to support interoperable machine-to-machine interaction over a network. Uses SOAP, an XML-based protocol to communicate over HTTP. Characteristics of web services are three key elements: Web service description language (WSDL) is an XML-based description of the web service's operations and functions. It dictates the protocol connections and message formats needed to connect and interact with a web service; Universal Description, Discovery and Integration (UDDI) which is a registry for storing web services WSDLs and a mechanism for registering and localizing web services on the Internet; and the SOAP communication protocol which defines the structure and format of messages exchanged between the customer-represented service provider and the service provider represented by the current web service provider. A client application that needs a specific feature from a service provider, and the service provider is usually a server that hosts and runs the web service.

Figure 1 illustrates how a SOAP-based web service works. There are other types of styles for web services, such as REST, RPC, RMI, .NET Remoting, CORBA, and Network Socket [4].

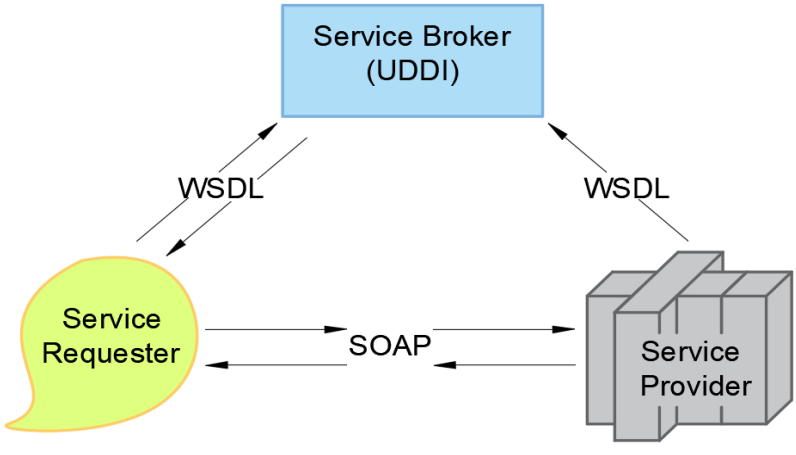

Fig. 1. Typical SOAP-Based Web Service

3.2. Enterprise Service Bus (ESB)

ESB represents the interoperability between its components, SOA often has interoperability between its components, SOA often uses an Enterprise Service Bus (ESB). Essentially, ESB is a piece of software that lies between the various

components of an SOA, primarily between the service provider and the service provider to enable transparent and seamless communication. Also acts as an intermediary software and a messaging intermediary between different communication parties in the SOA architecture. The main task of the ESB is to support messages routing and to provide a better orchestration and interaction between different interconnected web services built possibly using different technologies, platforms, standards and programming languages [4].

4. Proposed Model Architecture

This section introduces a service-oriented (SOA) architecture for building robotic rover machines for space exploration which uses components of web-based service software. Introduces a distributed model made of loosely interoperable web services and a central enterprise service bus (ESB) which is not located inside the current rover but in an isolated location, possibly a ground control center or space station in low earth orbit. Communication between the rover and Internet services is two-way and is performed remotely using the HTTP protocol with the help of ESB acting as an intermediary program. The way communication is accomplished is by calling the method in which the rover calls or remotely calls the various web services procedures to be executed on the host system and to return the results to the rover. These procedures are known as methods or functions containing the logic and programming instructions that provide the basic functions of the rover.

Basically, the architecture that was proposed consists of three basic levels: The first level is the client represented by the rover tool which calls the internet services methods to perform operations such as mineral examination, geological environment analysis, study and evaluation of rock composition, and capturing and processing images for a variety of applications. The second level is the server represented by the web services which are disconnected from the rover hardware and are cut and executed on server machines located on the ground or at a nearby space station. Web services provide the current code base and rover logic. They contain the algorithm, implementation, and programming instructions needed to provide the rover with its basic operations and functionalities. The third level is the intermediate program represented by ESB, which provides a standard interface and a unified data path for both client and server levels that interact efficiently and exchange data regardless of their platforms. incompatible and implementation technologies, for example, technologies such as REST, SOAP, RPC etc.

Figure 2 illustrates the proposed SOA architecture [4].

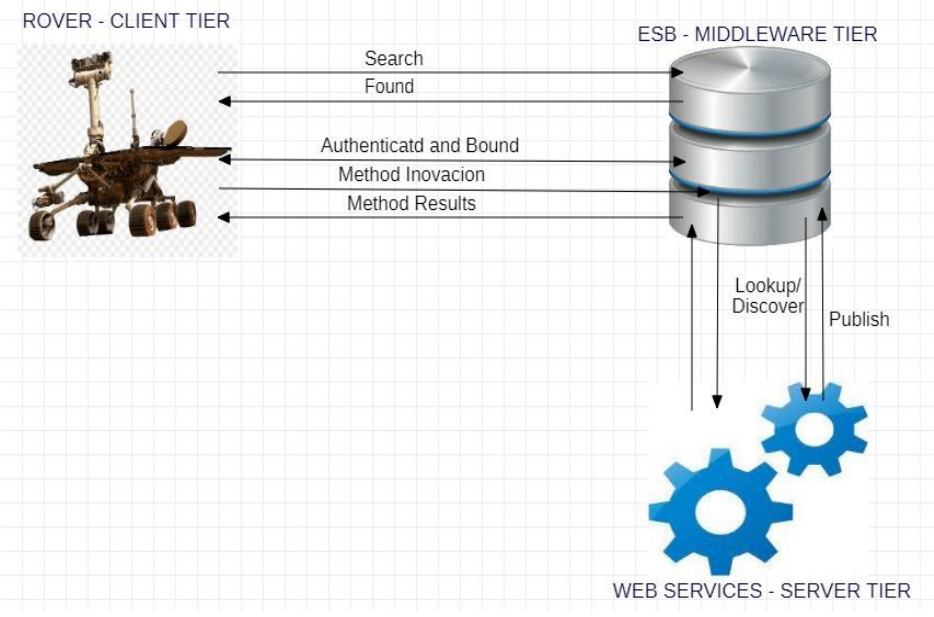

Fig. 2. Tiers of the proposed SOA architecture

4.1. Design Specifications

As discussed earlier, the proposed architecture comprises three tiers: The client, the middleware, and the server tier.

4.1.1. The Client Tier

Actually, the client is the robotic rover vehicle. It has an onboard computer that can discover external web services using the ESB interface, which explains the various functions wrapped within linked web services. In order to communicate, the rover has to bind to the ESB interface. This binding authenticates the rover (requester) and allows it to send requests to the ESB (provider) using the remote procedure invocation approach. The ESB then forwards the rover's request to the intended web service. The results that are returned by the web service are first received by the ESB then forwarded to the rover. The provider does all of the work, and the requester only gets the results. Communication between requester and provider is done solely using the HTTP protocol through the Deep Space Network (DSN) that relays transmission between the earth where the provider is located and the outer space where the rover is located.

Figure 3 illustrates the sequence of interactions between the rover client and the rest of the entities [1, 4].

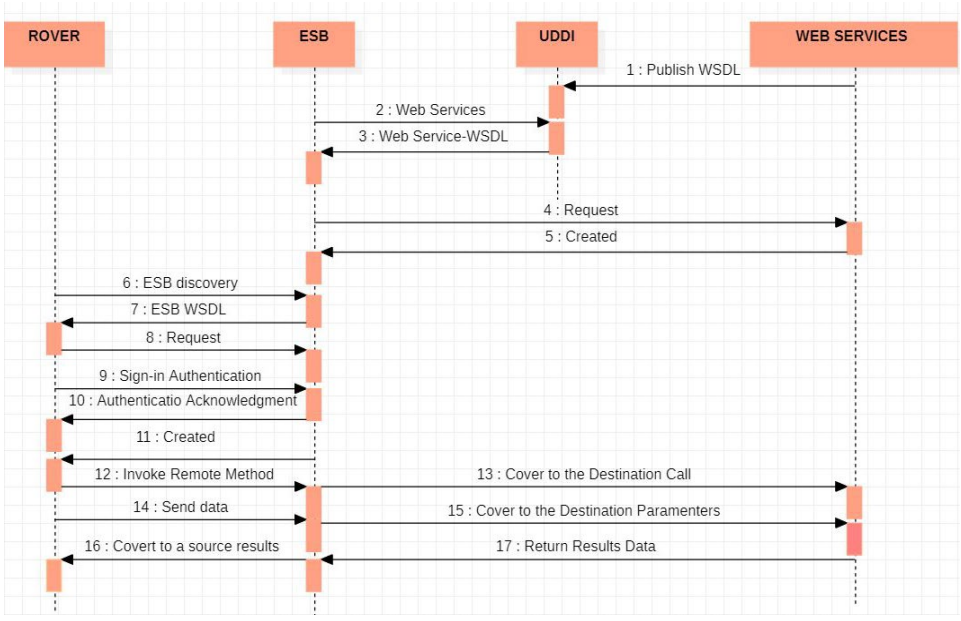

Fig. 3. The Rover's Sequence of Interactions

4.1.2. The Middleware Tier

The ESB acts as a conduit for data between the rover unit and the various web services. It serves as a data transmission medium, simulating a messaging middleware that connects the rover on one side and the various distributed services on the other, allowing them to send and receive data back and forth. It also automates inbound and outbound communications between all parties involved, organizes their activities, and provides for message storage, routing, and transformation during inter-system interactions.

The proposed ESB is cross-platform and cross-network, allowing the rover to send requests and receive responses from multiple types of web services, some of which may be incompatible, and which are constructed using different platforms, standards, technologies, and programming languages.

Figure 4 displays the planned ESB's design as well as its inner workings [4].

The ESB actually has two public interfaces: The rover's first interface provides a unified single SOAP-based end-point through which the rover can communicate with the ESB. The second interface comes from the web services side, which offers a set of adapters that serve as end-point connectors for various web services [4].

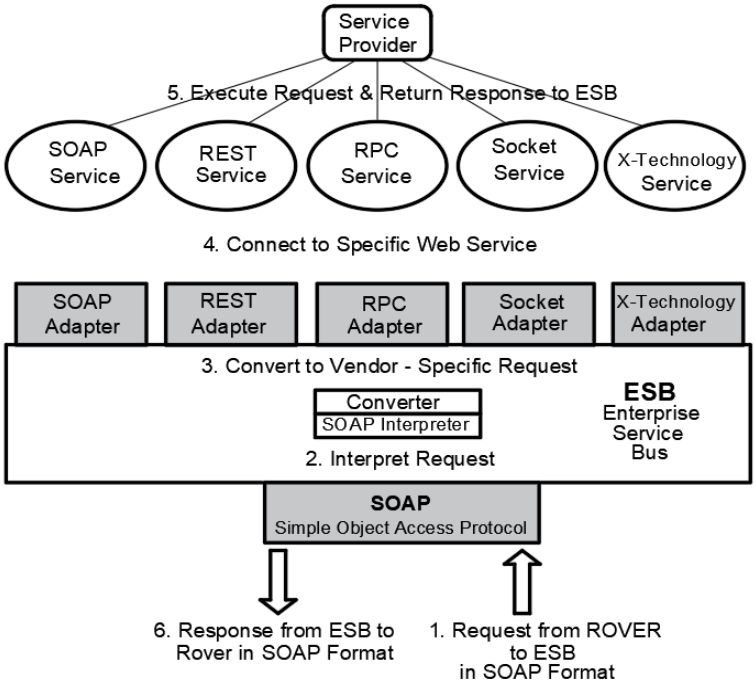

Fig. 4. The Architecture of the proposed ESB [4]

4.1.2. The Server Tier:

The server tier is where the web services are performed. It consists mainly of several large computer servers located either in the ground control centers or in a space station in low earth orbit. These servers are tasked with delivering online services, processing rover requests, executing business logic, and performing intensive calculations on behalf of the rover. Web services can be: protocols or versions and they interact with the ESB. Whenever a new web service is integrated into the system, it publishes its WSDL on the ESB, which stores it inside an internal registry along with other important details. The ESB then exposes the WSDL to the rover vehicle allowing it to remotely call all available functions. Web services can provide all kinds of functionality, including computer vision functions that analyze captured images and recognize objects within those images; navigation functionalities that allow the rover to move and move over the planet's surface; sensory functionalities that measure the atmospheric properties surrounding the rover; microscopic functionalities to analyze and inspect the nature of rocks and soils and their structure, and scanning functionalities to detect the presence of certain elements within the planetary terrain [4].

5. Implementations

A robotic rover simulation software was built capable of performing various actions. Besides, it is capable of sending requests to and reading results from the ESB using the SOAP protocol. The software is a web application built using C# ASP.NET. Figure 5 depicts the GUI interface of the rover simulation software. Additionally, three web services were developed. SOAP-based web services built using C# ASP.NET with a .asmx file extension and is responsible for performing functionalities: the current time ratio between planet Earth and planet Mars, the calculation and conversion of the Martian (Sol) and Earth Day, and the direct connection to planet Mars (by the Rover robot).

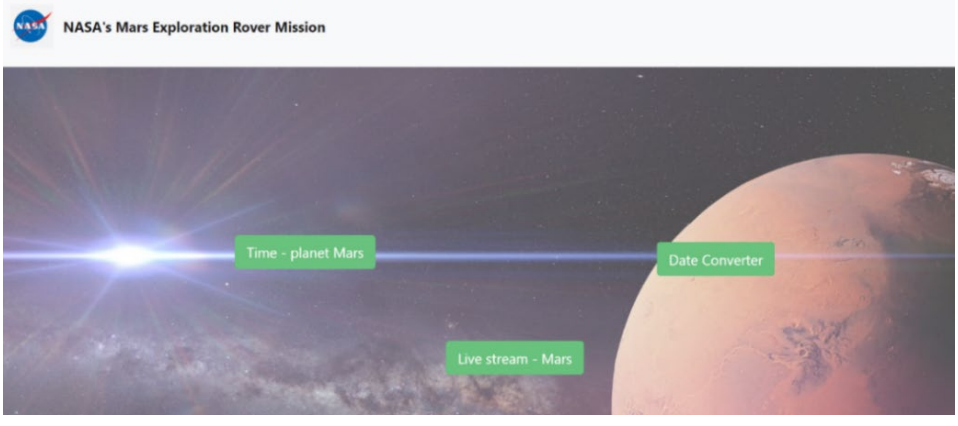

Fig. 5. Rover's GUI interface

ESB was built to act as a middleware between the rover and the different web services. As the ESB is the service broker, it is responsible for exposing the different functionalities of the web services to the rover [4]. Figure 6 delineates the list of functionalities exposed by the ESB and originally implemented in the web services.

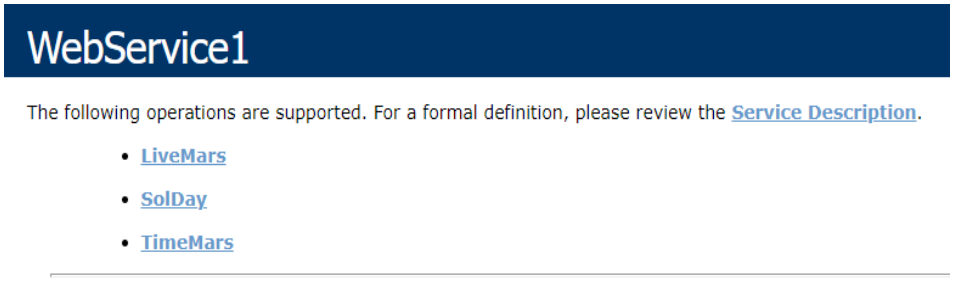

Fig. 6. Web Services

6. Reliability

The CIP was extremely reliable. During the first seven months of the nominal and extended rover missions, its medium program stayed on for over 99.9% of the time and operated non-stop for up to 77 days at a time. Some of the industry standards were followed and COTS software was used.

The application server contributed to the reliability by constantly monitoring the behavior of the EJBs and did automatic repetition or error recovery whenever necessary. On development servers, extensive mid-program stress testing is done before deploying CIP and even during mission. CIP usage patterns had a sharp rise, as many users became very active immediately after rovers downloaded new data and images. Stress testing showed us how the secondary program would behave during such increases and noted performance barriers. they were able to adjust the system settings accordingly to enable the intermediate program to better handle heavy loads. It was developed as an independent, interactive tool to perform stress testing by simulating any number of users performing various client functions, such as accessing schedules or downloading files (see Fig. 7).

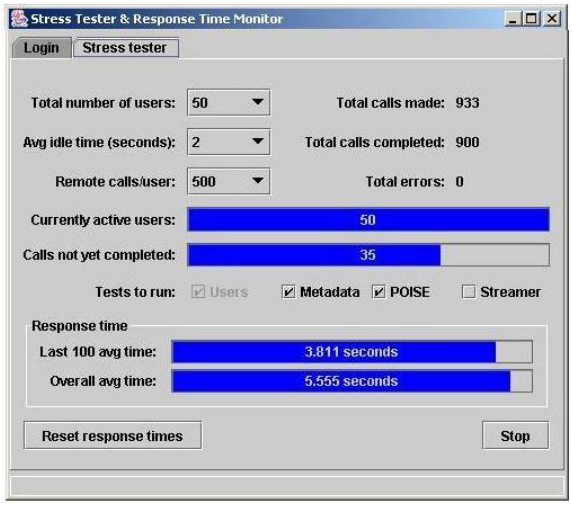

Fig. 7. The Middleware Stress Tester [1]

An important measure of software reliability is how long it stays on and running. An application may crash suddenly, or system administrators may remove it for maintenance. Reconfiguring a service to suit a change in an operational parameter, such as the time it took for a signal to travel from Earth to Mars, was a frequent maintenance procedure for CIP (the time of one-way light). Dynamic reconfiguration was a key feature that allowed the CIP to stay in

operation and operate for long periods without scheduled server maintenance interruptions.

The CIP secondary software design and application server allowed individual services to be "set hot": that we could add, remove, replace, or restart a service while the rest of the secondary program (and CIP as a whole) continues to function. To reconfigure a service, a system administrator first edited the service configuration file and then redistributed the service. When the service resumed, it read in its new configuration. Restoring the service usually takes a few seconds and often users did not notice any interruptions [1].

7. Conclusion

This paper introduced an architecture that enables the building of exploration robotic rover systems using distributed software components called web services. This architecture consists of three levels: the client level, the server level, and the ESB that acts as an intermediary program. The analyzes performed showed that this architecture is reliable, scalable, interoperable, reusable and maintainable, that can adapt to unforeseen circumstances and withstand various obstacles that may be encountered during their missions.

References

1. Mak. R., Walton. J., Keely, L., Heher, D., Chan, L.: A reliable service–oriented architecture for NASA's Mars exploration rover mission. April (2005), https://www.researchgate. net/publication/ 4204299_Reliable_Service_Oriented_Architecture_for_NASA%27s_Mars_Exploration_Rover_Mission
2. Mak, R., Walton, J.: The collaborative information portal and NASA's Mars Rover Mission. IEEE Computer Society, January – February (2005)
3. Ai-Chang, M., Bresina, J., Charest, L., Hsu, J., Jonsson, A. K., Kanefsky, B., Maldague, P., Morris, P., Rajan, K., Yglesias, J.: MAPGEN Planner: Mixed-initiative activity planning for the Mars Exploration Rover mission. February 2003, https://www.researchgate. net/publication/24157887_MAPGEN_Planner_Mixed-Initiative_Activity_Planning_for_the_Mars_Exploration_Rover_Mission
4. Bassil, Y.: Service-oriented architecture for space exploration robotic rover systems. Int. Journal of Science & Emerging Technologies (IJSET) 3(2), http://ojs.excelingtech.co.uk/index.php/IJSET/article/download/426/239 (2012)
5. Bonasso, R. P., Kortenkamp, D., Miller, D. P., Slack, M.G.: Experiences with an Architecture for Intelligent Reactive Agents. In: Proc. of the International Joint Conference on Artificial Intelligence (1995)

6. Colon, E., Sahli, H., Baudoin, Y.: CoRoBa, A multi mobile robot control and simulation framework. International Journal of Advanced Robotic Systems 3(1), (2006)
7. iRobot, Mobility Integration Software User's Guide (2002)
8. Quiqley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A.: ROS: An open-source Robot Operating System. In International Conference on Robotics and Automation, Workshop on Open-Source Robotics (2009)
9. Jørstad, I., Dustdar, S., Do Van Thanh: A service-oriented architecture framework for collaborative services. 2005, Link: https://www.researchgate.net/publication/4208018_A_service_oriented_architecture_framew ork_for_collaborative_services
10. Sinderson, E., Magapu, V., Mak, R.: Middleware and Web Services for the Collaborative Information Portal of NASA's Mars Exploration Rovers Mission, October (2004), https://www.researchgate.net/publication/221461222_Middleware_and_Web_Services_for_the_Collaborative_Information_Portal_of_NASA%27s_Mars_Exploration_Rovers_Mission