Spatial-aware Iterative Integration of Crisis Management Information Systems

Betim Sojeva
Fraunhofer IAIS
Sankt Augustin, Germany
betim.sojeva@iais.fraunhofer.de

Jingquan Xie
Fraunhofer IAIS
Sankt Augustin, Germany
jingquan.xie@iais.fraunhofer.de

***Abstract*—Information systems are playing increasingly more important role in modern crisis management process. An integrated system with capabilities like foresight, prediction and decision support capabilities can provide substantial added-value for decision makers on both tactical and policy-making levels. It is however a challenging task to seamlessly integrate various systems with dedicated functionalities on functional and technical aspects, especially when these systems are developed independently from each other with substantially different design rationale and software technology. In this paper, an iterative system integration approach is proposed by harmonising service-oriented, model-driven and agile system development. Several design principles and best practices from the software engineering community are adopted to facilitate the integration task. In addition, extra attention is paid to provide enhanced support for integrating spatial data into the crisis management workflow. This approach aims to provide a pragmatic system integration methodology to integrate crisis management information systems in a more effective and efficient fashion.**

***Index Terms*—Disaster management, service-oriented architecture, system integration, geographical information system.**

I. Introduction

Modern information systems are essential parts of the crisis management process - both preparedness and response phases. Real-time collaboration systems can support rescue forces to communicate with each during crisis situations. Sensor systems are able to provide information like temperature, smoke concentration, etc. to the central monitoring system for improved situation awareness. Simulation-based systems can help decision makers to assess possible impacts and consequences in the future of certain actions for better decision making. Most of these systems involved in the crisis management process are however developed independently and isolated deployed - for dedicated tasks. Integrating these systems together will provide different roles in the crisis management a seamless user experience in terms of situation awareness and decision making.

This paper proposes an architectural approach for integrating information systems used in a typical crisis management process, which consists of three parts: situational awareness, foresight and prediction, decision making (see Fig. 1). It tries to combine modern software engineering best practices with the specific requirements in crisis management process. An iterative system integration approach is described to facilitate the technical integration work. RESTful mock-up services [1] based on modern Service-Oriented Architecture (SOA) [2] are the essential parts of the integration architecture. Reverse-proxy based solution provides a flexible runtime environment for hiding the technical details of different system implementations. Special design consideration is also given to integrate spatial data into the crisis management process. To maximise the system design flexibility, software containers are used to provide flexible wrappers for the real implementation.

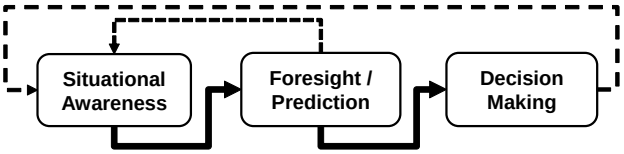

Fig. 1. Typical crisis management process with decision making by considering the current situation and possible future development.

This paper is organised as follows: Section I gives the background and the motivation of the proposed approach. It is followed by Section II where the adoption of RESTful service-oriented architecture, agile iterative integration and software containers with spatial data integration is elaborated. This is the major part of this paper. To help readers better understand the proposed approach, a simplified use case is presented and discussed in Section III. Some related work is discussed in Section IV. Finally, Section V concludes this paper and provides insight on potential future directions.

II. Iterative System Integration

Working with partners from different organisations on the same software project can be difficult, especially when it comes to integrating new system features and providing system maintenance. It can yield unwanted dependencies and slow down the software development process. Therefore, a modular software architecture can help to manage system development and de-couple component dependencies. In the following subsection, four major aspects of the integration approach are elaborated.

A. RESTful service-oriented architecture

Service-Oriented Architecture (SOA) [2] is an architectural design pattern based on isolated and de-coupled software components - each provides dedicated services to the others, focusing on interoperability and re-usability. One approach to

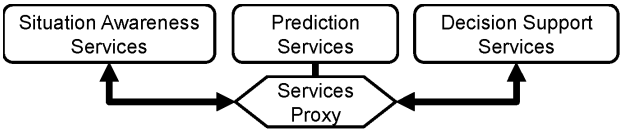

Fig. 2. A service suite with three RESTful web services and one service proxy. Each of them provides dedicated services and can communicate with each other via the proxy.

implement SOA capability is using RESTful web services [1], which provide light-weight and highly scalable solutions. Extensive programming language support and large ecosystem make it ideal for integrating heterogeneous information systems used in crisis management process. Fig 2 illustrates a system with three services and a proxy. All three services can be developed independently by different organisations. They are accessible by exposing themselves via the proxy, which de-couples the service interface and the implementation. This kind of system isolation is of critical for developing different crisis management system components.

B. Iterative Integration

An iterative approach of system integration can be separated into three stages (see Fig. 3):

1) Defining specification and requirement of the service. This includes developing use cases, formal specification, etc.
2) Writing service mock-ups and deploy them to the server for automated testing. After this stage, all unit tests should pass as required in classical Test-Driven Development (TDD) [3].
3) Iteratively replace mock-ups by real implementations. Each time, if a service mock-up is replaced, all unit tests must be executed to guarantee that the service implementation meets the requirements defined in the specification.

The first step requires a complete specification description of each RESTful endpoint including:

- Uniform Resource Locator (URL) - a unique ID of the service.
- Request method type - indicating the character of the service whether it is for reading, writing or deleting operations.
- Header information - some meta information that is not suitable to be encoded in the URL.
- Payload - additional information that is two large to be encoded in the URL.
- Expected responses - the result delivered by the service implementation.

Finally, after all the mock-ups are replaced by real implementation, it is always a good practice to have some high-level tests running like functional tests and even human-guided tests.

C. Embracing Software Container

Component-based development is a technique to manage software artefacts on a single or multiple host machines.

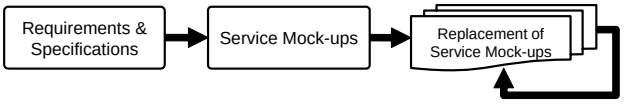

Fig. 3. Iterative integration of different RESTful services. It starts with the service specification and followed by the mock-up services. The major effort devotes to the replacement of the mock-ups by the real implementation, which is iterative, i.e. services can be replaced one by one.

A software container is isolated and independent software that works as infrastructure software to host other software components [4]. After deployed, a software container can be considered as a running application with all the dependency it needs. To build such a container, a configuration file is needed, which comprises the blueprint of the software container. It includes specific instructions for the Container-Engine to determine the runtime behaviour of the container. In the proposed iterative approach, the following configuration is recommended to have a skeleton of the container deployment (see Fig. 4):

- Web Server - the web server is responsible for handling the requests from the service consumers. On the other hand, the responses generated by RESTful services are forwarded by the web server to the service consumers.
- Documentation Server - this serve as a descriptive website of all of the services. It can host the information for both developers and the system end-users.
- Continuous Integration - this ensures a sound software development workflow by seamlessly integrate the software modifications into the deployment.
- Mock-up services - this container consists of the mock-up services as described in the previous section.
- WMS and WFS - the mapping services for exposing spatial data. More details see Section II-D.

D. Spatial Data Integration

Spatial data integration is an essential part in modern crisis management process. Most of the objects that are of interest to the crisis management team have geographical location - like a street, a electrical substation, a telecommunication router, etc. crisis managers and situation operators need sufficient information about states of these objects, in order to make reasonable decision like whether to evacuate a certain region. Modern geographical information systems consist of a set of standards [5] like Web Map Service (WMS) and Web Feature Service (WFS) to facilitate the modelling of these objects. The status of these objects can be encoded directly as attributes of certain features - an object in both WMS and WFS. As illustrated in Fig. 4, dedicated map server can be setup as containers to provide spatial data support. The objects that

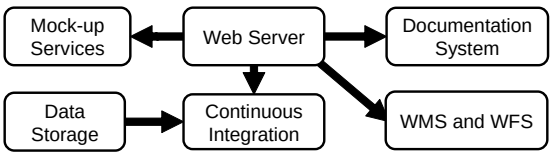

Fig. 4. Integrated Service using software containers.

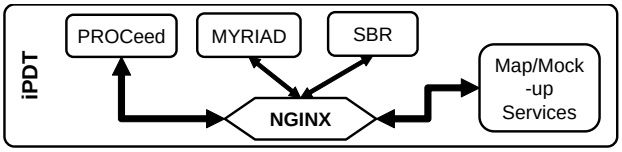

Fig. 5. The integrated PREDICT tool suite consisting of three major components - PROCeed, MYRIAD and SBR. The mapping service and the service mock-ups are also part of the container deployment.

need to be rendered in the map server can be extracted from the Data Storage container.

III. Use Case - the integrated PREDICT tool suite

The integrated PREDICT tool suite - iPDT for short - developed in the PREDICT project [6] is an example that comes with the proposed integration approach. The fully integrated system iPDT (see Fig. 5) combines the integration clusters on both conceptual and technical level. Each of the blocks in Fig. 5 correspond to a Docker container [4] - a proprietary implementation of software containers. Services provided by components like PROCeed or MYRIAD are specified at the beginning and replaced iteratively by implementations provided by different organisations. This kind of isolation and de-coupling make the distributed development and deployment more efficient. Moreover, information generated within iPDT can also be fed into other systems. For instance, the information forecasted by PROCeed can also be fed into other system by providing the standard mapping services on top of the Web. Currently a working group in the PREDICT project is focusing on integrating the Dutch national crisis management LCMS with iPDT by applying this kind of spatial-aware integration approach. Finally, all the services are exposed by using the reverse proxy server NGINX [7], which provides high performance and scalable solutions for exposing RESTful web services.

IV. Related Work

System integration is not a new topic for managing information systems. Several approaches have already been proposed during the last decades. Applying grid computing technology for general purpose system integration is discussed in [8]. With modern Web infrastructure this kind of integration is well suited. A similar approach as proposed in this work is also discussed in [9], which however more focus on the health care information systems. System integration on the semantics level is also investigated. A survey paper [10] presents several approaches using state-of-the-art Semantic Web technologies. This is one of the limitations of the proposed approach - lacking the support for the semantic data exchange between different system components.

V. Conclusion and Outlook

Modern crisis management process requires a sophisticated combination of different information systems. Seamlessly integrating them to provide a holistic view of the crisis situations is of great added-value to rescue forces and decision makers, especially for real crisis where time is critical for saving human lives. This paper proposes an iterative system integration approach tailored for crisis information management systems. Special attention is also given to integrate spatial data, which is nowadays an essential part to provide advanced situation awareness. Best practices from modern software engineering like Service-Oriented Architecture (SOA), agile development, and container-based system deployment provide a solid foundation for operationalise the proposed approach. A simplified use case is also presented to demonstrate the potential benefits of the proposed approach. One of the major limitations of the proposed approach is its sophistication. System developers need to be familiar with modern software engineering technology. Moreover, software containers are still an emerging technology and not widely used; therefore deploying existing systems into containers requires additional overhead for adopting the proposed approach.

In the future, more efforts will be devoted to provide extensive semantic interoperability, i.e. information exchanged are not only represented as syntactical artefacts, but also the context and the meta information will be shared as well. This is extremely useful for cross-border crisis management use cases where different participants from different countries (with different background information) can be sure that they are talking about the same thing even syntactically represented differently - semantic consistency. Technologies from the Semantic Web community like linked data, formal ontology provide promising solutions to tackle this problem. In addition, high-level standard crisis management mock-up services for integration purposes can be summarised and provided to reduce the efforts for integrating new information systems.

Acknowledgement

This research was funded by the European Commission within the Seventh Framework Programme project *PREDICT*. The authors would like to thank all of the project partners.

References

[1] R. Fielding and R. Tylor, "Principled design of the modern web architecture," *ACM Transactions on Internet Technology*, vol. 2, no. 2, pp. 15–150, 2002.

[2] E. Newcomer and G. Lomow, *Understanding SOA with Web services*. Addison-Wesley, 2005.

[3] D. Janzen and H. Saiedian, "Test-driven development: Concepts, taxonomy, and future direction," *Computer*, no. 9, pp. 43–50, 2005.

[4] C. Boettiger, "An introduction to docker for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, 2015.

[5] C. D. Michaelis and D. P. Ames, "Web feature service (wfs) and web map service (wms)," in *Encyclopedia of GIS*. Springer, 2008, pp. 1259–1261.

[6] PREDICT, "The EU PREDICT Research Project," 2014. [Online]. Available: http://www.predict-project.eu

[7] W. Reese, "Nginx: the high-performance web server and reverse proxy," *Linux Journal*, vol. 2008, no. 173, p. 2, 2008.

[8] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "Grid services for distributed system integration," *Computer*, vol. 35, no. 6, pp. 37–46, 2002.

[9] W. Wang, M. Wang, and S. Zhu, "Healthcare information system integration: A service oriented approach," in *Proceedings of ICSSSM'05. 2005 International Conference on Services Systems and Services Management, 2005.*, vol. 2. IEEE, 2005, pp. 1475–1480.

[10] N. F. Noy, "Semantic integration: a survey of ontology-based approaches," *ACM Sigmod Record*, vol. 33, no. 4, pp. 65–70, 2004.