AUTOMATYKA/ AUTOMATICS • 2013 • Vol. 17 • No. 1

http://dx.doi.org/10.7494/automat.2013.17.1.127

Piotr Szwed*, Igor Wojnicki*, Sebastian Ernst*, Andrzej Głowacz*

Evaluation of a Dynamic Map Architecture with ATAM**

1. Introduction

Development of complex information systems is a difficult process, and the success of the outcome is largely dependent on appropriate decisions made early in the design process. Features such as diversity of data models, platform heterogenity, performance (e.g. real-time) requirements and the need for interoperability make it even more complicated.

Dynamic Map, is one of the key subsystems of the INSIGMA project [2]. It can be considered a complex information system, composed of spatial databases, storing static and dynamic data relevant for urban traffic, as well as a set of software modules responsible for data collection, interpretation and provision. The data originates from a network of sensors, both fixed (e.g. video detectors, acoustic sensors, inductive loops) and on-board GPS receivers installed in vehicles. Clients of the Dynamic Map are various software modules performing such tasks as visualization, route planning, traffic optimization, object tracking and threat detection.

The paper reports the experiences with application of Architecture-based Tradeoff Analysis Method (ATAM) for early evaluation of the Dynamic Map architecture, presenting artifacts created during the method's application: architectural views, utility tree and high priority scenarios, found risks, tradeoffs and sensitivity points.

The paper is organized as follows. A more detailed description of the INSIGMA Project, the concept of the Dynamic Map and challenges related to the design process are given in Section 2. It is followed by an introduction to the ATAM method (Section 3). Section 4 presents a subset of the INSIGMA system being considered, its architecture, scenarios and architectural decisions. It also describes ATAM analysis results. Finally, conclusions and observations are given in Section 5.

* AGH University of Science and Technology, Krakow, Poland

** Work has been co-financed by the European Regional Developement Fund under the Innovative Economy Operational Programme, INSIGMA project no.POIG.01.01.02- 00-062/09

2. Motivation

2.1. About INSIGMA and the Dynamic Map

The INSIGMA project aims at development and implementation of an advanced information system for optimizing road traffic. While more and more cities now are facing the problem of traffic jams and worsening ecological conditions, INSIGMA will provide tools for efficient traffic control and threat detection. In case of existing solutions for map creation and traffic management, the main problem is related to the lack of efficient tools that enable conversion of object location and audiovisual data into dynamic maps. In practice, traffic control is often based on low-frequency components of traffic dynamics. Thus, in case of road accidents or other threats, these systems are incapable of fast response. In such scenarios, traffic problems can be first detected after 30 minutes. Another issue is the limited area of the managed region. Traffic detectors are mostly deployed in major highways but in limited scope in access roads.

On the other hand, vehicle users demand efficient route planning and optimization. Existing navigation solutions consist in analysis of static parameters (e.g. total route length or road type), and even recent solutions rarely use statistical data (e.g. maximum driving speed in selected hours). The dynamic traffic component still remains unaddressed and relevant traffic conditions need to be personally recognized by the user in advance or, worse, in the place of event. Thus, in case of a jammed metropolis, the efficiency of route optimization is far from what is expected. This is particularly important in daily operations of emergency services, fire brigades, police, etc.

The first and foremost feature of INSIGMA is analysis of traffic parameters and further processing on basis of dynamic maps. Dynamic map can be viewed as a set of data describing the state of the road infrastructure. This corresponds to dedicated system services. In the INSIGMA project, these include: route planning, navigation, traffic control or crisis management. Managed data consists of: specifics of road infrastructure, current traffic conditions and preferences. Part of these information can be arranged into static layers (Fig. 1).

The road infrastructure is defined by the road network (road locations, connections, lanes and points of interest) with information about organization of traffic, which contains traffic signs, lights, temporary exclusions (e.g. road repairs) or detours.

The traffic conditions include monitored traffic density and atmospheric phenomenons such as rain or snow, icing, etc. It is also important to consider extraordinary events, e.g. traffic accidents or other dangerous situations.

The scope described above is supplemented by the preferences which correspond to specific needs of particular system. For example, route planning for transport of dangerous materials can operate on reduced map containing only main highways. On the other hand, emergency service may require fastest navigation through jammed city – using all possible connections (also against one-way streets).

Thus, the Dynamic Map can be perceived as a representation of the road transport infrastructure combined with up-to-date information about traffic intensity and historical traffic data. Such a combination includes information stored in a database and map visualization, which can be presented to the end user via a network or mobile interface. Algorithms for dynamic route optimization are applied to the system; they aid traffic control systems and are particularly useful in urban environments.

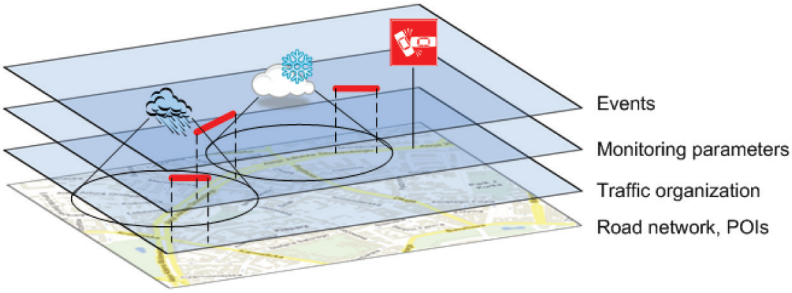

Fig. 1. Logical layers of Dynamic Map

2.2. Dynamic Map design challenges

There are several factors that influence the design of Dynamic Map. They include expected performance, flexibility concerning representation of processed information and seamless integration of various types of sensors, including these already present in the urban traffic infrastructure.

Due to the complexity of the system, its size and key importance of its services to other subsystems we decided to perform architecture evaluation at an early system development stage to identify and correct potential design flaws and limitations, as well as to mitigate risks of not meeting functional and non-functional requirements. The Architecture-based Tradeoff Analysis Method (ATAM) for architecture evaluation [11], was selected for this purpose. ATAM is considered a mature, efficient and non-costly method which can be applied to various types of architectural designs.

This paper discusses key elements of the adopted approach in a case study reduced to a few among high priority requirements (scenarios) and a limited number of components and architectural decisions. Due to high number of components of the Dynamic Map a presentation of the full analysis would have exceeded the expected volume of the article. For the analyzed scenarios the outcomes (sensitivity points, tradeoffs, risks and recommendations) are listed.

3. The ATAM method

The goal of software architecture evaluation methods is to assess whether a system meets or will meet certain requirements concerning quality characterized as quality

attributes. A standardized list of quality attributes is published in ISO/IEC 9126-1 norm [5], which enumerates six groups of quality attributes: Functionality, Reliability, Usability, Efficiency, Maintainability and Portability. This set was extended in the superseding norm ISO/IEC 25010: [6] to 9 groups. Many of the quality attributes were known elsewhere under different names, e.g.

Efficiency as Performance, Changeability (sub-attribute of Maintainability) as Modifiability, etc.

Architecture evaluation methods may bring the greatest benefits to software development if applied early in the software lifecycle, as identified flaws in system design can be corrected at a lower cost [14]. Typically, an assessment is conducted based on the specification of the software architecture (architectural views) and use other sources of information, such as interviews with various stakeholders including owners, future users, architects and development teams. At an early development stage it is difficult to give the ultimate answer whether a particular quality attribute can or cannot be assured. Therefore, assessment methods aim at estimating such characteristics as risk or cost.

Identified high risk to achieve a quality attribute can trigger mitigation actions which consist in revising the design and changing the design decisions. However, it should be emphasized, that even after changes and corrections are applied, some acceptable residual risks can still be present, because the estimated effort required to remove them exceed expected losses.

ATAM (Architecture-based Tradeoff Analysis Method) was developed at the Software Engineering Institute (SEI) in 2000 [11, 4] as a successor of the SAAM method [8]. Its goal is to evaluate architectural decisions against specific quality attributes and to detect:

- *risks* – architectural decisions that may cause problems to assure some quality attributes,
- *sensitivity points* – decisions related to components or their connections that are critical for achieving required level of quality attribute,
- *tradeoffs* – decisions that increasing one quality attribute have negative impact on others.

ATAM produces these results based on requirements expressed as *scenarios* that are elicited and assessed in a formal process divided into phases and steps. These topics are described in subsequent sections.

There are several reports on successful applications of ATAM for assessment of a battlefield control system [9], wargame simulation [7], product line architecture [13], control of a transportation system [3] and credit card transactions system [12]. Recently, a few extensions of ATAM were proposed, including combination with the Analytical Hierarchy Process [16] and APTIA [10].

3.1. Utility tree and scenarios

ATAM uses a quality model called the *utility tree*. At the root of the utility tree, an abstract concept *Utility* is placed. Its child nodes are annotated with general quality

attributes, e.g. these specified in the ISO norm (performance, reliability, security, modifiability, etc.); at the next level they can be decomposed into more specific attributes, and finally, scenarios are placed at leaves. Both quality attributes present in the utility tree and scenarios are elicited from various stakeholders and represent their point of view on expected system qualities.

The quality attributes, which are selected to be included into the utility tree are consequences of *business drivers*, i.e. specific goals to be achieved or considerations to be taken: time to market, high security of transactions, or easy integration with external systems.

The advantage of using scenarios is that they turn somehow vague quality attributes into verifiable requirements, that can be submitted to the assessment. Scenario define expected interaction between a stakeholder and the system in form of a chain: trigger, artifacts concerned and expected response.

ATAM distinguishes three types of scenarios:

1. use case scenarios,
2. growth scenarios,
3. exploratory scenarios.

Use case scenarios define expected interaction between users and implemented system. Most often they are assigned in the utility tree to such quality attributes, as: performance (response time, throughput), usability (a user can easily perform a specific task) or reliability (actions to be taken in case of failures or exceptional conditions).

Growth scenarios capture anticipated future changes in the system. Such scenarios are particularly important, if the system development is conducted in builds or iterations and each phase in the system development should deliver a functional instance. In such iterative models it is highly important that architectural decision made in earlier phases will not prevent or hinder the possibility to introduce some foreseen functions in the future. Scenarios belonging to this group usually fall into such categories as modifiability, scalability, interoperability, portability.

Exploratory scenarios are not likely to occur. However, they can be identified and analyzed to detect implicit assumptions and communicate to the stakeholders limits in the architectural design. They can be assigned to analogous quality attributes as the growth scenarios.

3.2. The ATAM process

According to ATAM, the architecture assessment process is a group effort of various stakeholders involved in the system development. It deploys typical group techniques as brainstorming, assigning priorities and voting. The course of evaluation is divided logically into four phases and nine steps:

1. *Presentation*: (1) presentation of the ATAM method, (2) business drivers and (3) the assumed software architecture.

2. *Investigation and Analysis*: (4) identification of architectural approaches, (5) generation of quality attribute tree and (6) analysis of the architectural approaches.
3. *Testing*: (7) brainstorming and prioritization of scenarios, (8) repeated analysis of the architectural approaches with reference to high priority scenarios.
4. *Reporting*: (9) presenting results of the analysis: risks, sensitivity points and tradeoffs.

4. Evaluation of the Dynamic Map

ATAM-based evaluation of the Dynamic Map was performed in accordance to the general method guidelines, with some minor deviations. The INSIGMA project is developed by several distributed teams and it was extremely difficult to gather all stakeholders to participate in brainstorming sessions. Direct communication was replaced by teleconference sessions and wiki-based voting (e.g. in step 7 of ATAM).

This section presents information prepared for the 8th step of the process, i.e. repeated analysis of the architectural approaches with reference to elicited earlier high priority scenarios.

4.1. Architecture

The architecture of the Dynamic Map is functionally decomposed into three subsets of components: Static Map (corresponding to the lower two layers in Fig. 1), the Traffic Repository (the third layer) and the Event Repository (the top layer). These subsets are collections of active components and interfaces grouped around databases.

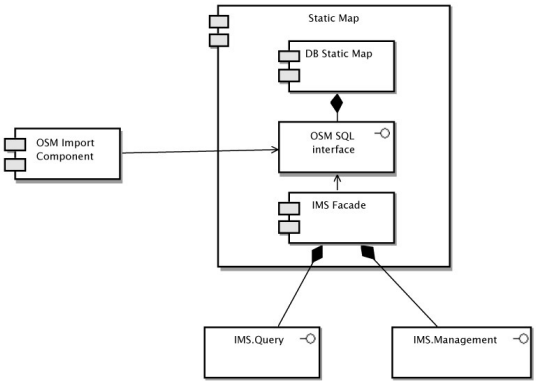

Fig. 2. Static Map architecture

1. The Static Map is responsible for storing information about road connections and other objects appearing on the map, e.g. Point of Interests (POIs). These data originate from Open Street Map project [1] and have similar representation. Static Map contains also

information about traffic organization and uses additional structures: Lanes, Turns and Crossroads. Their role is twofold: they are introduced to support needs of route planning and traffic optimization, but also they serve as intermediary objects linking traffic monitoring parameters and events with elements of OSM.

2. The Traffic Repository adopts a simple data model: it stores Types of monitoring parameters, typed Instances linked indirectly to locations on the map and roads and current Values. Traffic Repository has two functions coupled in one database: it provides a discoverable directory of monitoring parameters and the storage supporting high volume of feeds and queries about values of instances.
3. The Event Repository follows the approach taken for the Traffic Repository, it defines various types of events (e.g. traffic jams, accidents, weather conditions) and information on their occurrences.

Such approach defines clear separation of concerns, however causes that certain queries are distributed between databases, what may introduce risks to performance. It should be also mentioned, that all data structures, including dictionaries representing types of monitoring parameters and events, are defined in ontologies, that were used at the development stage to model domain and provide further semantic reference supporting integration.

The architecture of Dynamic Map is presented in form of architectural views prepared in the ArchiMate [15] language.

The Static Map architecture is given in Figure 2. It consists of a relational database (DB Static Map) with an SQL-based interface (OSM SQL interface). It also provides an interface façade: IMS (Insigma Map Static) Facade. The Façade is used by two separate interfaces: IMS.Query and IMS.Management for querying and managing Static Map data respectively. There is also the OSM Import Component which feeds data from the Open Street Map Project into the Static Map using the SQL interface directly.

The Traffic Repository architecture (Fig. 3) introduces several new components. The External Subscriber Facade allows to subscribe to chosen traffic parameter data feeds to have such information delivered as soon as it is available in the repository. The RT Event Interpreter analyzes traffic parameters and infers events that are caused by them. The repository is fed with data through the Feed Interface from diverse sensors. The RT Sensor State Analyzer verifies if sensors are on-line. Furthermore, it also supports the GPS Tracker which enables gathering traffic information from mobile GPS-based devices.

The Event Repository architecture (Fig. 4) consists of a database, native SQL-based interface and several façades with corresponding interfaces (IMD stands for Insigma Map Dynamic). It also introduces a discovery interface which enables to identify what kind of events are actually stored in the repository. The event source is twofold: either Dedicated Event Reporting Software (i.e. used by law-enforcement or public safety agencies) or Mobile Event Reporting Clients. Since events are geo-localized the Query Facade and the Event Reporting Facade utilize the Static Map interfaces (OSM SQL interface, IMS.Query).

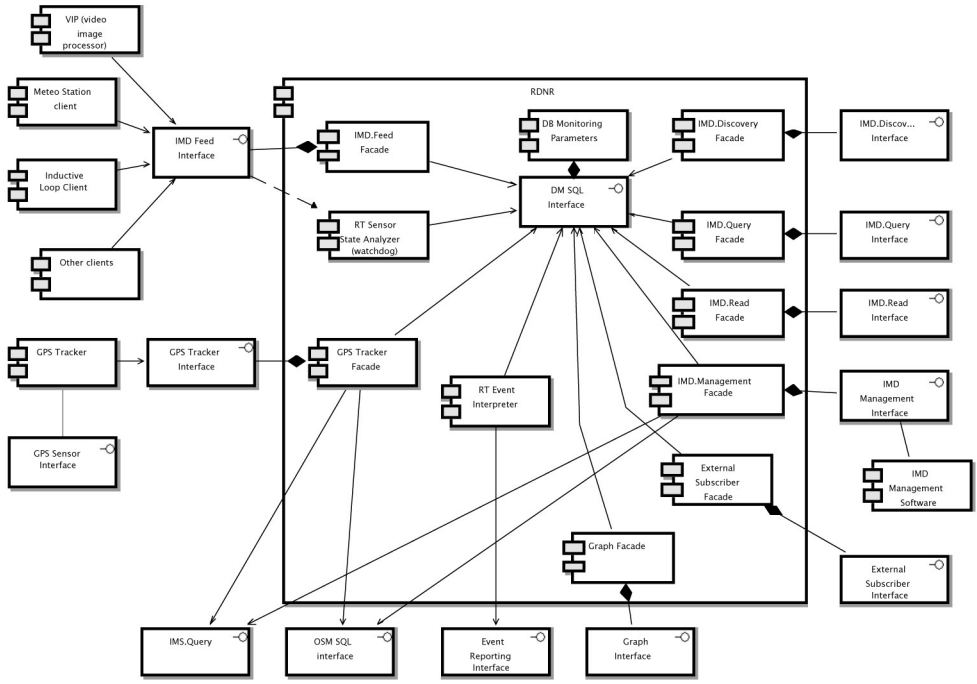

Fig. 3. Traffic Repository architecture

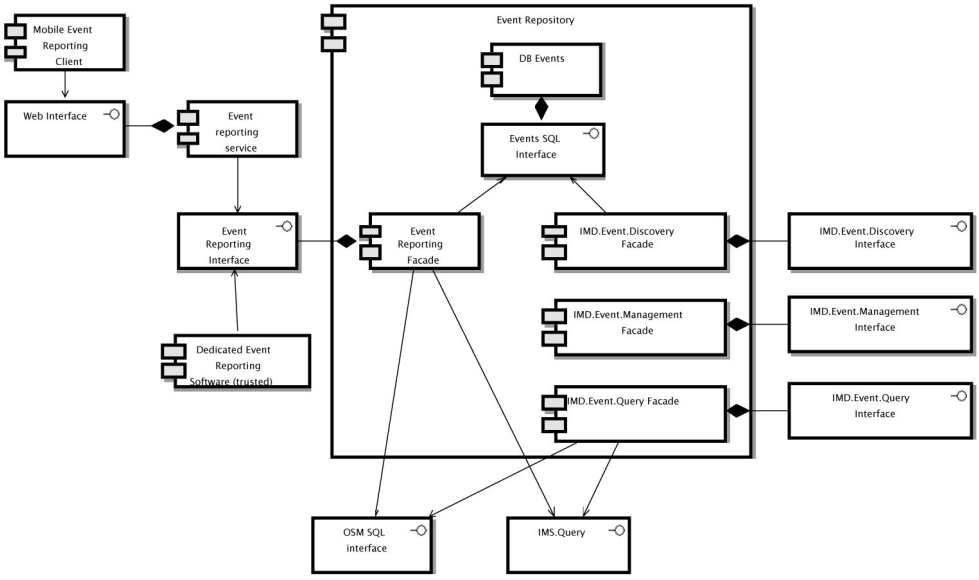

Fig. 4. Event Repository architecture

4.2. High priority scenarios

High priority scenarios, which are usually identified in a brainstorming session, were elicited in a few sessions performed with participation of smaller teams, documented on the project wiki pages and submitted to voting. Below their list is presented. The related quality attributes are marked in the parenthesis.

1. New sensor types and sensor processing software will be integrated in <6 days> (Modifiability).
2. Dynamic Map will be capable of accepting feeds from up to 1000 sensors occurring every <60> seconds (Performance).
3. Dynamic Map will be capable of processing up to <200> read queries per second (Performance).
4. New map features can be introduced and made discoverable with built-in functionality (Usability).
5. System operation will be restored within 3 hours after a single component fails (Reliability).
6. Updating of data defining the road network can cause a loss of at most 10 values of a single monitoring parameter (Reliability).
7. Sensor failures will be detected in a configurable period ranging from 3 to 20 minutes (Reliability).
8. New event types and corresponding customizable detection routines can be introduced in 1 day (Modifiability).
9. System will be able to accommodate and interpret location data from 5000 GPS tracker units updated every 5 seconds (Performance).
10. A Dynamic Map component can be ported to another platform within 20 days (Portability).

4.3. Results of evaluation and mitigation strategies

Scenario #1: New sensor types and sensor processing software will be integrated in <6 days>

The goal of the scenario is to integrate within the Dynamic Map a new sensor, e.g. a videodetector, with developed and tested processing software capable of calculating several types of traffic parameters. It is assumed, that the sensor software will be augmented by a ready to use communication component using web service interface over a SSL secured network to feed values.

Introduction of a new sensor type may result in adding descriptions of a sensor and measured parameters into the system ontology, inserting new values to the dictionaries and the directory of sensors and instances in TM database. Adding a new sensor indirectly

increases required capacity of components that store or allow access to instances of measured parameters and their values.

- Senistivity point: definitions of monitoring parameter types appear in the ontology and dictionaries within TM database.
- Tradeoff: coupling of directory and storage functions within TM database is a tradeoff between performance and modifiability. While feeding values, their ranges are to be checked within real-time constraints. This precludes examining the directly the ontology, due to the lack mature of Semantic Web tools at .NET platform. Access to the ontology is wrapped by a web service, what would introduce an overhead influencing the performance. On the other hand, the directory in TM database has a fixed structure, what may hinder changes.
- Risk: there is a potential risk of loosing cohesion between the ontology and dictionaries as no decision is made concerning the rules for updating the dictionaries within TM database to reflect changes in the ontology.
- Non-risk: data structures and interfaces are designed to support seamless integration.
- Recommendation: Make a decision on the ontology storage and define a workflow for updating the dictionaries based on ontology.

Scenario #2: Dynamic Map will be capable of accepting feeds from up to 1000 sensors occurring every <60> seconds

The scenario specifies a typical performance requirement. The approach taken assumes that a single sensor can feed multiple values of monitoring parameters (typically 20) at time using a synchronous web service that place them into a common memory buffer. Values collected in memory are entered to the database by bulk queries executed by a periodical process.

- Sensitivity points: query granularity, use of synchronous web services and buffering.
- Non-risk: the initial tests of the communication overhead and data base performance indicate that there is no substantial risk to achieving the scenario response. Data originating from the assumed number of sensor can be published in the database with the latency limited to 10 seconds (assuming 5 second period of a thread executing bulk inserts of maximum 20000 values).

Scenario #4: New map features can be introduced and made discoverable with built-in functionality

The scenario encompasses both creating objects and modifying those already present, e.g. closing a road to perform some works. New map elements can either be new OSM element types (introduced via the *OSM Map Features*) or new traffic organization objects not available in OSM (such as a more detailed description of a complex road network element). These objects are stored in SM database and their types are defined in the ontology. Clients, with exception of the current graph interface, discover map objects using identifiers

of concepts (URIs) defined in the ontology. After introducing new objects client applications should be made aware of the new features, and that certain parameters pertain to them.

- Sensitivity points: coherence between the ontology and SM database content.
- Non-risk: SM interface (and facade) support feature discovery.
- Risk: new types of objects (combinations of tags and values) can be embedded in imported OSM data. To make the map clients aware of such new objects they must be defined in the ontology.
- Tradeoff with performance: specific clients, e.g. route planning systems can use their own optimized representation of the map (e.g. memory based). It is up to client to decide when these representations should be updated. Frequent updates would decrease the performance.
- Recommendation: define rules for updating the OSM ontology (describing map features and their representations).
- Recommendation: implement a versioning mechanism to allow client checking if it uses the uptodate map version (polling).
- Recommendation: consider implementing a notification mechanism to inform clients about changes.

Scenario #6: Updating of data defining the road network can cause a loss of at most 10 values of a single monitoring parameter

The specification of the scenario constraints the influence of the changes in the road network structure (which should be updated on a regular basis) on the process of collecting and accessing the values of monitoring parameters.

- Sensitivity point: distribution of data objects and their relations.
- Non-risk: the most important types of monitoring parameters are not linked with OSM structures directly. OSM data defining the road network and associative structures: lanes, turns and crossroads are stored in a separate data base. Processes of updating the map and collecting values of monitoring parameters run in isolation.
- Risk: the only exception are direct links between monitoring parameters and OSM's area records. Updating them may cause loss of data.
- Risk: non defined business logic for *TM discovery facade* may cause erroneous implementation ignoring intermediate inconsistency between the map and monitoring parameters. In particular, to avoid the impact of such inconsistency, the queries should be based on locations of instances of monitoring parameters and not on their links to roads.
- Recommendation: introduce an associative data structure wrapping links to area records.
- Recommendation: add functionality allowing to validate and update associations (lanes, turns, etc.) in case of changes to the map data.

5. Conclusions

Experience gathered during analysis of the Dynamic Map architecture indicates that ATAM is a valuable method, as it helps to detect real flaws in the design and identify potential risks. We did not use ATAM to gain a false conviction that the system is optimally designed, but to collect requirements that were not expressed earlier, which represent expectations of various stakeholders, and to assess that they can be satisfied with the proposed system architecture (including adaptations and changes at limited costs). The presented list of risks and recommendations proves, that the initial system architecture was not perfect in every detail and several issues were detected by using ATAM.

Faced with the task of applying ATAM to Dynamic Map architecture assessment, we realized that the method provides excellent guidelines on how to organize the entire process and how to specify requirements and describe the outcomes. However, due to its generic character, ATAM lacks supporting tools and techniques which would allow to describe impacts of scenarios on particular components and to collect various properties (design decisions) of architectural elements.

Another observed problem is the selection of architectural decisions that should be considered during the evaluation. Implementation of the each complex system, including Dynamic Map, relies on several technologies related to databases, communication, semantic web and security. Identification of key design decisions (properties) is up to experts' knowledge and experience. Gathering such knowledge related to particular technologies, e.g. SOA, as a support to ATAM would be beneficial for the efficiency and reliability of evaluation.

References

[1] OpenStreetMap wiki. http://wiki.openstreetmap.org/wiki, Last accessed: Jan 2012.
[2] INSIGMA project. http://insigma.kt.agh.edu.pl, Last accessed: Mar 2012.
[3] Bouck'e N., Weyns D., Schelfthout K., Holvoet T., *Applying the ATAM to an Architecture for Decentralized Control of a Transportation System*. Vol. 4214, Springer, 2006, pp. 180–198.
[4] Clements P., Kazman R., Klein M., *Evaluating Software Architectures: Methods and Case Studies*. Addison-Wesley Professional, 2001.
[5] ISO/IEC. ISO/IEC 9126, *Software engineering – Product quality*. ISO/IEC, 2001.
[6] ISO/IEC. ISO/IEC CD 25010.3, *Systems and software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Software product quality and system quality in use models*. ISO/IEC, 2009.
[7] Jones L.G., Lattanze A.J., *Using the architecture tradeoff analysis method to evaluate a wargame simulation system: A case study*. Technical Report CMUSEI2001TN022 Software Engineering Institute Carnegie Mellon University Pittsburgh PA, (December):33, 2001.
[8] Kazman R., Bass L., Abowd G., Webb M. *SAAM: a method for analyzing the properties of software architectures*, vol. 16, pp, IEEE Comput. Soc. Press, 1994, pp. 81–90.
[9] Kazman R., Barbacci M., Klein M., S Jeromy Carriere, Woods S.G., *Experience with performing architecture tradeoff analysis*. Proceedings of the 21st international conference on Software engineering ICSE 99, 1999, pp. 54–63.

[10] Kazman R., Bass L., Klein M., *The essential components of software architecture design and analysis*. Journal of Systems and Software, 79(8), 2006, pp. 1207–1216.
[11] Kazman R., Klein M., Clements P., *ATAM: Method for architecture evaluation*. Technical report, Carnegie Mellon University, Software Engineering Institute, 2000.
[12] Lee J., Kang S., Chun H., Park B., Lim C., *Analysis of VAN-core system architecture- a case study of applying the ATAM*. [in:] Proceedings of the 2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, SNPD '09, Washington, DC, USA, 2009, IEEE Computer Society, pp. 358–363.
[13] Ferber S., Heidl P., Lutz P., *Reviewing product line architectures: Experience report of ATAM in an automotive context*. vol. 2290, Springer, 2001, pp. 364–382.
[14] Banani R., Graham T.C.N., *Methods for evaluating software architecture : A survey*. Computing, 545(2008-545):82, 2008.
[15] Van Den Berg H., Bosma H., Dijk G., Van Drunen H., Van Gijsen J., Langeveld F., Luijpers J., Thé Nguyen, Oosting R., Slagter G. et al., *Archimate made practical*. Work, 2007.
[16] Wallin P., Froberg J., Axelsson J., *Making decisions in integration of automotive software and electronics: A method based on ATAM and AHP*. Fourth International Workshop on Software Engineering for Automotive Systems SEAS 07, 2007, pp. 5–5.