Application of New ATAM Tools to Evaluation of the Dynamic Map Architecture *

Piotr Szwed, Igor Wojnicki, Sebastian Ernst, and Andrzej Glowacz

AGH University of Science and Technology
{pszwed,wojnicki,ernst,aglowacz}@agh.edu.pl

Abstract. The paper reports an application of Architecture-based Trade-off Analysis Method (ATAM) for early evaluation of the Dynamic Map architecture. The Dynamic Map is 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. Due to the complexity of the system, its size and key importance of its services to other subsystems, we decided to perform architecture evaluation using the ATAM method. To facilitate the task new tools supporting ATAM based assessment are proposed: Scenario Influence Matrix and Architectural Decision Matrix. Taking as example an excerpt from the system architecture, we present how they were used during the architecture evaluation. The gathered experience confirm usefulness of the tools, enabling ATAM to help detecting real flaws in a design and identify potential risks.

Keywords: dynamic map, ATAM, architecture, evaluation

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.

The Dynamic Map is one of the key subsystems of the INSIGMA project [1].It can be considered a complex information system, composed of spatial databases, storing static and dynamic data related to urban traffic, as well as a set of software modules responsible for data collection, interpretation and provision to clients. 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 include various software modules performing such tasks as visualization, route planning, traffic optimization, object tracking and threat detection.

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

The paper reports the experiences gained by using the Architecture-based Tradeoff Analysis Method (ATAM)[12] for early evaluation of the Dynamic Map architecture. The main contribution of this paper is an extension of ATAM through introduction of additional supporting tools, as well as adaptation of the method to fit the characteristics of the Dynamic Map.

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). Application of the proposed approach is given in Section 4. Section 5 presents the actual subset of the INSIGMA system under consideration, its architecture, scenarios and architectural decisions. It also describes the results of ATAM analysis. Finally, conclusions and observations are given in Section 6.

2 Motivation

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.

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.

There are several factors that influence the design of the Dynamic Map. They include the expected performance, flexibility concerning representation of processed information and seamless integration of various types of sensors, including those 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 the risks of not meeting functional and non-functional requirements. The ATAM[12], was selected for this purpose, as it is considered a mature, efficient and non-costly method which can be applied to various types of architectural designs.

Faced with the task of applying the 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, the ATAM lacks supporting tools and techniques which would allow to describe the impacts of scenarios on particular components and to collect various properties (design decisions) of architectural elements.

During the assessment, we developed and used two such tools: the Scenario Influence Matrix (SIM) and the Architectural Decision Matrix (ADM), which extended the architectural views described in the ArchiMate [17] language.

This paper discusses the key elements of the adopted approach in a case study reduced to just few of the high priority requirements (scenarios) and a limited number of components and architectural decisions. Due to the high number of components of the Dynamic Map, a presentation of the full analysis would have exceeded the expected volume of the paper. For the analyzed scenarios, the

outcomes (sensitivity points, tradeoffs, risks and recommendations) have been 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 [7] and ISO/IEC 25010 [8] standards, which define nine attribute categories, e.g. Functionality, Reliability, Usability, Efficiency, Maintainability, Modifiability and Portability.

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 such as owners, future users, architects and development teams.

The ATAM was developed at the Software Engineering Institute (SEI) in 2000 [12], [4]. 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 a quality attribute and *tradeoffs* – decisions that increasing one quality attribute have negative impact on others.

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, which can be decomposed into more specific attributes at the next level; finally, scenarios are placed at leaves.

The advantage of using scenarios is that they turn somehow vague expectations into verifiable requirements. ATAM distinguishes three types of scenarios: use case, growth and exploratory.

Use case scenarios define the expected interactions between users and an 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. Scenarios belonging to this group usually fall into such categories as modifiability, scalability, interoperability or 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.

There are reports on successful applications of ATAM to assessment of a battlefield control system [10], wargame simulation [9], product line architecture [5], control of a transportation system [3] and credit card transactions system[13].

Recently, a few extensions of ATAM were proposed, including combination with the Analytical Hierarchy Process [18] and APTIA [11].

4 Supporting tools for the ATAM method

The ATAM has many obvious benefits: it precisely defines the quality model based The on the utility tree, enumerates the expected outcomes, indicates the participants and provides an organizational framework for the evaluation.

Nevertheless, due to its generic character, the method can cause problems related to gathering and representing information that can be used for an architecture assessment. When preparing the assessment of the Dynamic Map according to ATAM, we realized that architectural views that can be the input to the ATAM do not provide constructs do describe the influence of a given scenario on a particular component. This regards in particular growth scenarios, achievement of which may require modifications, redesign of components, but may also increase resource utilization or generate demand on data capacity.

We have decided to design and develop two specifications (artifacts) that helped us conduct the evaluations: the Scenario Influence Matrix and the Architectural Decision Matrix.

4.1 Scenario Influence Matrix

The Scenario Influence Matrix (SIM) describes an influence of a scenario on components using a standard set of values defined in a vocabulary established for evaluation purposes. At this point, a remark should be made. Typically, an architecture description encompasses components and their connections. In most cases we omit connections during analysis, because in the architectural view we used ArchiMate [17] diagrams, which distinguish an *Interface* from a module providing it. The interface specification (e.g. a Web Service interface) very often decides on the nature of a connections that use it. Typically, an interface can be realized by a Façade – a module responsible for its implementation.

The Scenario Influence Matrix is defined as a partial function that assigns actions or influences to scenarios and components:

$$SIM : S \times C \rightarrow Mod \times Act \cup Inf, \tag{1}$$

where: S – is a set of scenarios, C – is a set of components, Mod – is an access mode Mod = $\{M, A\}$, where M stands for manual and A for automatic, Act – is a set of actions on components, Inf – is a set of influences.

The vocabulary of actions Act used in the evaluation is as follows:

- AC: adding a dedicated component,
- UF: using a function of a component,
- $MStr$: modifying a component structure,
- MF: modifying a component function,
- RC: remove component,

– CC: cloning a component,
– PC: partitioning component,
– MS: modify component state,
– CD: creating data managed by a component,
– RD: reading component data,
– MD: modifying data,
– DD: deleting data.

For Inf (influences) three values were used: IRU: increase resource utilization, ICP: increase capacity and ICL: influence component's logic.

For evaluation purposes, the SIM matrix was represented by a spreadsheet with scenarios placed at rows and components (including interfaces) at columns. The vocabulary of possible actions and influences was selected to fit the particular task of Dynamic Map evaluation. For other applications, it can be extended according to particular needs.

4.2 Architectural Decision Matrix

The second artifact prepared as an input for evaluation is the Architectural Decision Matrix (ADM). Its goal is to assign design approaches or decisions to each component. Filling the ADM is an important step of evaluation, as it helps to ask questions about certain decisions that can be perceived as important to reach expected scenario responses.

The Architectural Decision Matrix is defined as a partial function that assigns sets of possible decision values to pairs of decisions and components under consideration. The mapping is defined in (2) as follows. For each design decision $d \in D$, the function $ADM(d)$ assigns a set of possible decisions to a component. The whole mapping is a sum of $ADM(d)$ mappings for individual decisions corresponding to rows of an ADM matrix.

$$ADM = \bigcup_{d \in D} ADM(d), \text{ where } ADM(d) : C \to 2^{V(d)} \tag{2}$$

Examples of design decisions from the set D and their values are:

1. Platform (PostgreSQL, Linux, IIS, Glassfish,..., choice<*list of platforms*>)
2. Component logic (fixed, customizable)
3. Query granularity (bulk, low, medium, high, flexible)
4. Transaction distribution (no, over<*list of components*>)
5. Use of web services (REST, SOAP, RPC, document, encoded, literal)
6. Communication type (synchronous, asynchronous)
7. Buffering (yes, no)
8. Security method (https, WS Security, XML Encryption, XML Signature...)
9. Separation of concerns (coupled with <*component name*>)
10. Use of ontologies (structure definition, integration, semantic querying)

We do not claim that the above list is either complete or general enough to be applied to any architectural design. However, it reflects some salient questions that emerged during the assessment of the Dynamic Map architecture. We treat the lists of decisions and their values as a flexible mean to collect and describe those properties of the design which are missing in architectural views.

As ATAM-based evaluation should be conducted at an early stage of a software lifecycle, it may frequently occur that particular design decisions have not been made or some specification is missing. In this case, we use the values *not decided* or *not specified* (e.g. a platform is not decided, the logic of a component is not specified).

We consider ADM a useful tool, which provides an overview of the map of architectural decisions and helps detect white spots (e.g. decisions that have not been made), which can introduce a substantial risk during future development. Beyond the assessment task, ADM provides a useful background reference supplementing the architectural views, therefore we decided to maintain it during system development.

5 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 system is being developed by several distributed teams and it was extremely difficult to gather all stakeholders to participate in the brainstorming sessions. Direct communication was replaced by teleconference and wiki-based voting.

5.1 Architecture

The architecture of the Dynamic map is functionally decomposed into three subsets of components: the Static Map, the Traffic Repository and the Event Repository (the top layer). These subsets are collections of active components and interfaces grouped around databases.

1. The Static Map is responsible for storing information about road connections and other objects appearing on the map. These data originate from Open Street Map project [2] and have similar representation. The Static Map also contains information about traffic organization and uses additional structures: Lanes, Turns and Crossroads.
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. The design of the Traffic Repository is general enough, to integrate various types of sensors. This approach was positively verified while integrating video detectors [6].

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, certain queries are distributed between databases; that may introduce performance risks. It should also be mentioned that all data structures, including dictionaries representing types of monitoring parameters and events, are defined using ontologies which were used at the development stage to model the domain [15] and provide further semantic reference to support integration [19, 16].

The Static Map architecture is given in Fig. 1. 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.

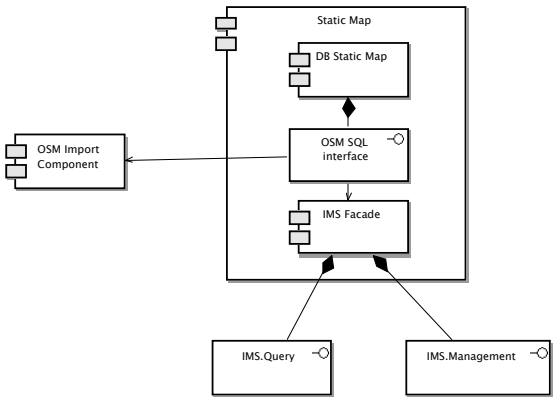

Fig. 1: Static Map architecture.

The Traffic Repository architecture (Fig. 2) introduces several new components. The External Subscriber Façade allows to subscribe to chosen traffic parameter data feeds and therefore 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 Event Repository architecture (Fig. 3) 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. Since events are geo-localized the Query Facade and the Event Reporting Facade utilize the Static Map interfaces (OSM SQL interface, IMS.Query).

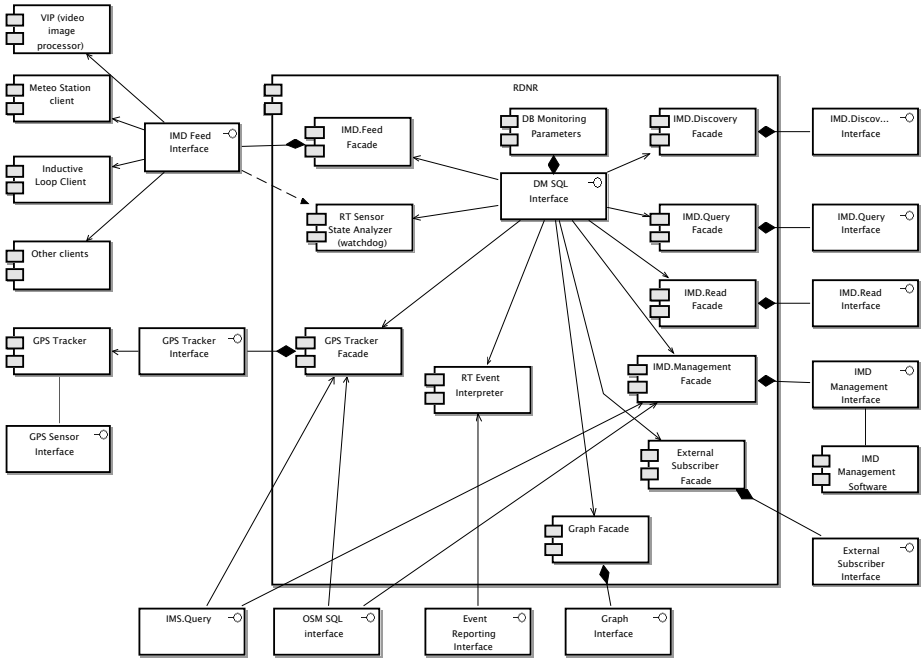

Fig. 2: Traffic Repository architecture.

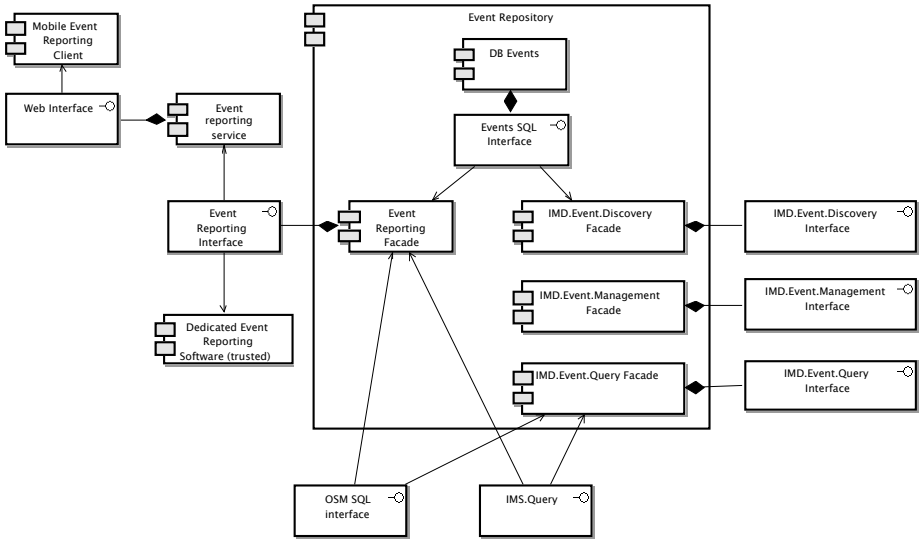

Fig. 3: Event Repository architecture.

5.2 High priority scenarios

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

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> sec (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. Updating of data defining the road network can cause a loss of at most 10 values of a single monitoring parameter (Reliability).
6. Sensor failures will be detected in a configurable period ranging from 3 to 20 minutes (Reliability).
7. New event types and corresponding customizable detection routines can be introduced in 1 day (Modifiability)
8. System will be able to accommodate and interpret location data from 5000 GPS tracker units updated every 5 sec (Performance).
9. A Dynamic Map component can be ported to another platform within 20 days (Portability)

5.3 Results of evaluation and mitigation strategies

The input for the architecture evaluation were two matrices: the SIM and ADM discussed in Section 4.

Table 1 representing SIM specifies influence of scenarios (listed in rows) on architecture components (table columns). Cells of the table are filled with actions defined in Section 4.1.

For ADM shown in Table 2 column headers contain architecture components, row headers types of architectural decisions. Cells define assignments of decision values to particular components.

The contents of the presented matrices are excerpts from the specifications that are a few times larger. For better readability, just a subset of the defined scenarios (see Section 5.2) and decisions (see Section 4.2) is chosen.

Further part of the section presents results of evaluation for two selected scenarios.

Scenario #1: New sensor types and sensor processing software will be integrated in <6 days> The goal of the scenario is to integrate a new sensor, e.g. a videodetector, within the Dynamic Map, with developed and tested processing software capable of calculating several types of traffic parameters. It is

Scenario	DB Static Map	OSM import component	IMS facade	DB Monitoring Parameters	Monitoring Parameters Ontology	IMD.Feed facade	RT Sensor State Analyzer	RT Event Interpreter	IMD.Discovery facade	IMD.Query facade	DB Events	Event Reporting Facade	IMD.Event Discovery Facade	Event Subscribe	Graph Interface
1 New sensor types and sensor processing software will be integrated in <6 days>				A: MD	M: MD	IRU	IRU	M: MF	ICP	ICP					ICP
2 Dynamic Map will be capable of accepting feeds from up to 1000 sensors occurring every <60> sec				A: CD		A: UF	A: UF	A: UF			A: MD A: CD	A: UF		A: UF	ICP
4 New map features can be introduced and made discoverable with built-in functionality	A: CD	A: UF	A: UF		A: RD										A: UF
5 Updating of data defining the road network can cause a loss of at most 10 values of a single monitoring parameter	A: MD	A: UF		M: MD				ICL							ICL

Table 1: Scenario Influence Matrix. Used acronyms: A – automatic, M – manual, MD – modifying data, MF – modifying a component function, ICP – increase capacity, IRU – increase resource utilization, CD – creating data managed by a component, UF – using a function of a component, RD – reading component data, ICL – influence component's logic.

Decision	SM database	OSM Import Component	IMS Facade	DB Monitoring Parameters	Monitoring Parameters Ontology	IMD Feed Facade	RT Sensor State Analyzer	RT Event Interpreter	IMD Discovery Facade	IMD Query Facade	DB Events	Event Reporting Facade	IMD.Event Discovery Facade	Event Subscribe	Graph Interface
Data objects	OSM Lane Turn Crsd.	OSM	OSM Lane Turn Crsd.	Snsr. Type Inst. Value	All	Inst. Value	Snsr. Inst. Value	Value Event	Type Inst.	Inst. Value	Event Occr.	Occr.	Event Occr.	Event	OSM Lane Turn Crsd.
Use of ontologies	SD	ND	yes				RD	RD	RD		SD	RD	RD	RD	
Use of web services		ND	yes SOAP		yes SOAP	yes SOAP			yes SOAP	yes SOAP		yes SOAP	yes SOAP	ND	ND
Communication type			syn		syn	syn ND			syn	syn		asyn ND	syn	syn or poll	
Platform	Pg	Lin Sc	Nd IIS Gs		Gs	Nd IIS Gs	Nd IIS Gs SP	Nd IIS Gs RB	Nd IIS Gs	Nd IIS Gs		Nd IIS Gs	Nd IIS Gs	Nd IIS Gs	
Query granularity		B	F		H F	H F			H F						
Component logic		Fi			Fi		F	F	ND	ND	Fi	ND	ND	ND	

Table 2: Architectural Decision Matrix. Used acronyms: ND – not decided, Nd – not defined, SD – structure definition, RD – reference data, Pg – PostgreSQL, Lin – linux, Sc – script, Gs – Glassfish, IIS – Internet Information Services, SP – stored procedure, RB – rule-based (rule expression language), B – bulk, F – flexible, H – high, Fi – fixed.

assumed that the sensor software will be augmented by a ready-to-use communication component using a web service interface over a SSL-secured network.

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 the TM database. Adding a new sensor indirectly increases the required capacity of components that store or allow access to instances of measured parameters and their values.

- *Sensitivity point*: definitions of monitoring parameter types appear in the ontology and dictionaries within the TM database.
- *Tradeoff*: coupling of directory and storage functions within the 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> sec 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)

6 Conclusions

Experience gathered during theanalysis 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 applying ATAM.

A significant contribution of this paper is development and application of two supporting tools for ATAM: the Scenario Influence Matrix (SIM) and the Architectural Decision Matrix (ADM).

SIM is used to collect information on the impact of a scenario on particular components. Obviously, for use case scenarios, e.g. related to performance, message sequence charts are a more natural way to represent the communication flow. However, for growth scenarios, there is no appropriate language to represent their impact on components.

ADM provides a centralized view on properties and decisions related to components (and in some cases, connections). Maintaining such a view is of high importance for a large project being developed by independent teams working in different locations, because many implicit decisions are made locally and are difficult to track.

References

1. INSIGMA project. `http://insigma.kt.agh.edu.pl` (Last accessed: Jan 2013)
2. OpenStreetMap wiki. `http://wiki.openstreetmap.org/wiki` (Last accessed: Jan 2013)
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, pp. 180–198. Springer (2006)
4. Clements, P., Kazman, R., Klein, M.: Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley Professional (2001)
5. Ferber Stefan, Heidl Peter, L.P.: Reviewing product line architectures: Experience report of ATAM in an automotive context, vol. 2290, pp. 364–382. Springer (2001)
6. Glowacz, A., Mikrut, Z., Pawlik, P.: Video detection algorithm using an optical flow calculation method. In: Dziech, A., Czyżewski, A. (eds.) Multimedia Communications, Services and Security, Communications in Computer and Information Science, vol. 287, pp. 118–129. Springer Berlin Heidelberg (2012), `http://dx.doi.org/10.1007/978-3-642-30721-8_12`
7. ISO/IEC: ISO/IEC 9126. Software engineering – Product quality. ISO/IEC (2001)
8. 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)
9. Jones, L.G., Lattanze, A.J.: Using the architecture tradeoff analysis method to evaluate a wargame simulation system: A case study. Technical Report CMU-SEI2001TN022 Software Engineering Institute Carnegie Mellon University Pittsburgh PA (December), 33 (2001)

10. Kazman, R., Barbacci, M., Klein, M., Carriere, S.J., Woods, S.G.: Experience with performing architecture tradeoff analysis. Proceedings of the 21st international conference on Software engineering ICSE 99 pp. 54–63 (1999)
11. Kazman, R., Bass, L., Klein, M.: The essential components of software architecture design and analysis. Journal of Systems and Software 79(8), 1207–1216 (2006)
12. Kazman, R., Klein, M., Clements, P.: ATAM: Method for architecture evaluation. Tech. rep., Carnegie Mellon University, Software Engineering Institute (2000)
13. 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. pp. 358–363. SNPD '09, IEEE Computer Society, Washington, DC, USA (2009)
14. Roy, B., Graham, T.C.N.: Methods for evaluating software architecture : A survey. Computing 545(2008-545), 82 (2008)
15. Sliwa, J., Gleba, K., Chmiel, W., Szwed, P., Glowacz, A.: IOEM - ontology engineering methodology for large systems. In: Jędrzejowicz, P., Nguyen, N., Hoang, K. (eds.) Computational Collective Intelligence. Technologies and Applications, Lecture Notes in Computer Science, vol. 6922, pp. 602–611. Springer Berlin Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-23935-9_59
16. Szwed, P., Kadluczka, P., Chmiel, W., Glowacz, A., Sliwa, J.: Ontology based integration and decision support in the insigma route planning subsystem. In: Ganzha, M., Maciaszek, L.A., Paprzycki, M. (eds.) FedCSIS. pp. 141–148 (2012)
17. Van Den Berg, H., Bosma, H., Dijk, G., Van Drunen, H., Van Gijsen, J., Langeveld, F., Luijpers, J., Nguyen, T., Oosting, Gerand Slagter, R., et al.: Archimate made practical. Work (2007)
18. 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 pp. 5–5 (2007)
19. Wojnicki, I., Szwed, P., Chmiel, W., Ernst, S.: Ontology oriented storage, retrieval and interpretation for a dynamic map system. In: Dziech, A., Czyżewski, A. (eds.) Multimedia Communications, Services and Security, Communications in Computer and Information Science, vol. 287, pp. 380–391. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-30721-8_37