Jan **WEREWKA**

INVESTIGATION OF ENTERPRISE ARCHITECTURE AND SOFTWARE ARCHITECTURE IN RELATION TO QUALITY ATTRIBUTES IN MILITARY APPLICATIONS

Abstract. In this paper the relations between enterprise and software architecture are investigated. These relations may be different for a company which uses the software as part of their operation as opposed to a company which develops software systems. The relations are usually very complicated, therefore only the attribute of quality was chosen for this study. This paper shows the importance of finding mappings between quality attributes on enterprise architecture and software architecture levels. Some examples from the military domain are given. An interesting example used in the military is the survivability quality attribute, which has similar meaning on enterprise, system, and network levels. This paper proposes some ways in which the relations may be investigated.

Keywords: Enterprise architecture, software architecture, quality attributes, military, survivability.

1. INTRODUCTION

Organizations wish to improve their effectiveness and competitiveness in all their business lines. Any organization can benefit from understanding its structure, products, operations, technology and the relationships between them. To adopt IT solutions in the best possible way it is necessary to implement enterprise architecture and understand the relation between software and enterprise architectures.

One important problem is determining how to align the software to be developed (software level) to the business values of the organization (enterprise level) that will use that software in its operations.

An Enterprise Architecture (EA) can be regarded as a model describing the way in which an organization achieves its current and future business objectives, using IT. For an enterprise, it is important that by applying enterprise architecture it will align its business and IT, thus becoming as competitive as possible. Enterprise architecture influences software architecture and vice versa.

In this paper this relation is investigated on the basis of quality attributes, which have different meanings in enterprise and software contexts. From the perspective of software development, aligning software to enterprise architecture is an interesting challenge.

2. FROM ENTERPRISE TO SOFTWARE ARCHITECTURE

Before discussing the relation between enterprise and software architecture some clarification of terms is necessary.

Enterprise Architecture (EA) is [1] a coherent whole of principles, methods and models that are used in the design of an enterprise's organizational structure, business processes, information systems, and infrastructure. The importance of enterprise architecture stems from the need to capture the essence of the business, present a holistic view of the enterprise, provide visibility of various business aspects, provide means to better make decisions and balance

Dr sc. Prof. Jan **Werewka –** AGH University of Science and Technology, Department of Applied Computer Science, Krakow, Poland

requirements, facilitate the translation from corporate strategy to daily operations, apply architectural thinking, and act as an instrument for simplifying the business.

An enterprise architecture framework (EA framework) defines tools and methods for developing and using enterprise architecture. An architecture framework provides principles and practices for creating and using the architecture description of a system. It structures architects' thinking by dividing the architecture description into domains, layers or views, and offers models - typically matrices and diagrams - for documenting each view.

In the process of adapting enterprise architecture, a typical approach is to focus on the most popular frameworks. Examples of how to apply prescriptive enterprise methodologies, such as TOGAF [1] and the Zachman Framework, are known and described in the literature, e.g. [2]. TOGAF is a general tool for organizing or adapting the method of developing enterprise architecture and aims to provide a practical, easily-accessible, standardized and industrial method of designing enterprise architecture.

There are many ways of implementing enterprise architecture. TOGAF is one of the most popular general solutions and has as one of its biggest advantages the fact that it is a broadly accepted standard. Customizing TOGAF for the purposes of given business needs is the first step to be taken by any organization wanting to implement it. However, in practice TOGAF might be found to be too general to be adjusted to the needs of a given type of enterprise.

Another problem with implementing TOGAF or other general solutions is the lack of a clear relationship between the enterprise architecture built around TOGAF and IT systems architecture. This problem can be particularly apparent in software development companies, in which choosing the right architecture for developed systems is of critical importance to the organization.

One important task is the adaptation of enterprise architecture for different classes of companies and organizations. The presentation [3] discusses some ideas for adapting enterprise architecture in the military. An approach to developing enterprise architecture at a software development company is presented in [4]. The solution includes the following main set of activities: defining a motivation model, adapting architecture modeling tools, creating an IT product landscape, building architecture capabilities in the organization, implementing standards and guidelines, applying architecture governance, defining the architect’s roles, managing risks, and using architecture governance. The proposed solution was introduced in an iterative manner in the software development company.

A military organization should be well organized, therefore it is interesting to see how enterprise architecture is adapted to their goals. DoDAF [5] and MODAF [6] are the two enterprise architecture frameworks used in the military domain which are most discussed in the literature. The Defense Architecture Framework (MODAF) is an Architecture Framework which defines a standardized way of developing Enterprise Architecture, originally developed by the UK Ministry of Defense. MODAF was developed from the US Department of Defense Architecture Framework (DoDAF) version 1.0, but has been extended and modified to meet MOD requirements by the addition of Strategic, Acquisition and Service Oriented Viewpoints.

DoDAF and MODAF have become popular [7] and their wide adoption has motivated the definition of UPDM, a standard interoperable UML-based profile for the exchange of DoDAF and MODAF models in XMI coding format. With the objective of reusing and integrating with available standards, UPDM also encapsulates UML/SysML specifications along with the SoaML profile.

Paper [8] shows a comparative analysis of different defense industry frameworks such as DODAF, MODAF, NAF and UPDM for defense information systems, and for C4I systems in particular. The paper concludes that the application of UPDM, DODAF, and MOADAF to

defense information system development such as C4I systems is more practicable than other defense industry frameworks.

Architecture governance [1] is the practice and orientation by which enterprise architectures and other architectures are managed and controlled at an enterprise-wide level. Governance essentially concerns ensuring that business is conducted properly. It is less about overt control and strict adherence to rules, and more about guidance and effective and equitable usage of resources to ensure the sustainability of an organization's strategic objectives.

Military organizations use different approaches to establishing enterprise architectures. One such approach which is based on cooperation between organizations distinguishes the following architecture structures:

- Federated – a family of coherent but distinct member architectures which conform to an overarching corporate architecture,
- Segmented – segments can be viewed as separate initiatives under the overall enterprise-level architecture,
- Service oriented – functions and applications are defined and designed as discrete and reusable capabilities or services that may be under the control of different organizational entities.

The term "System of Systems Architecture" (SoS) was introduced, which is closely related to Federated Enterprise Architecture. The Department of Defense (DoD) Defense Acquisition Guidebook defines a System of Systems as a "set or arrangement of systems that results from the integration of independent and useful systems into a larger system that delivers unique capabilities". SoS systems are large-scale concurrent and distributed systems that are comprised of autonomous constituent systems with operational and managerial independence. Due to the complexity of SoS, it is hard to model all the system aspects in a single model. Architecture frameworks were introduced to reduce SoS modeling complexity by using multiple layers called architecture views that depict the system from different perspectives (e.g. operational, functional, system). The US Department of Defense Architecture Framework (DoDAF) [5], the British Ministry of Defense Architecture Framework (MODAF) [6] and the NATO Architecture Framework (NAF) [7] are the most commonly used architecture frameworks for modeling SoS.

There are different definitions of Software Architecture. In this work the preferred definition is [9]: The software architecture of a system is the set of structures needed to deduce system behavior, comprising software elements, the relations between them, and the properties of both.

Software systems support enterprise operations, therefore it is important to investigate the relations between enterprise and software architectures.

3. RELATION INVESTIGATION BETWEEN ENTERPRISE AND SOFTWARE ARCHITECTURES

One of the largest problems which enterprises face is poor business alignment of increasingly complex and costly IT systems. This relation will be investigated in a simplified way using the ArchiMate [10] language. ArchiMate provides a complete set of relation description concepts for relationships between architecture fields, while producing a simple and uniform structure as a result.

The ArchiMate motivation layer is used because it is important to present enterprise motivations systematically. The ArchiMate [10] language provides a complete set of concepts

for describing relationships between architecture fields and producing a simple and uniform structure as a result.

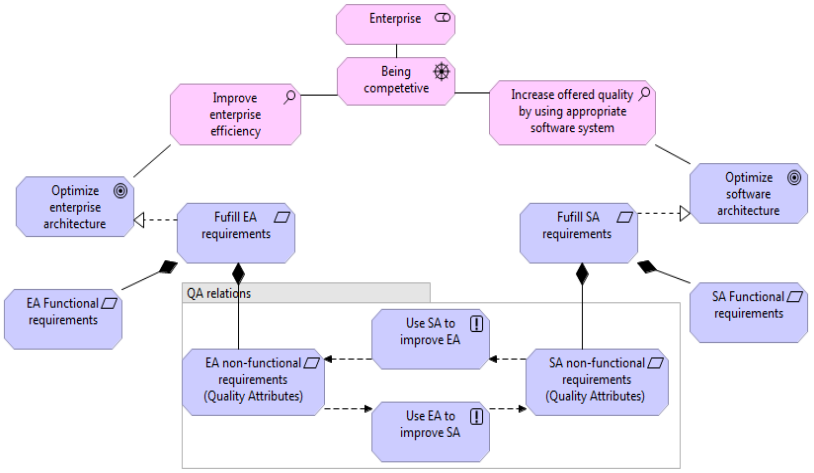

Fig. 1. Simplified motivation diagram for an enterprise based on a software system

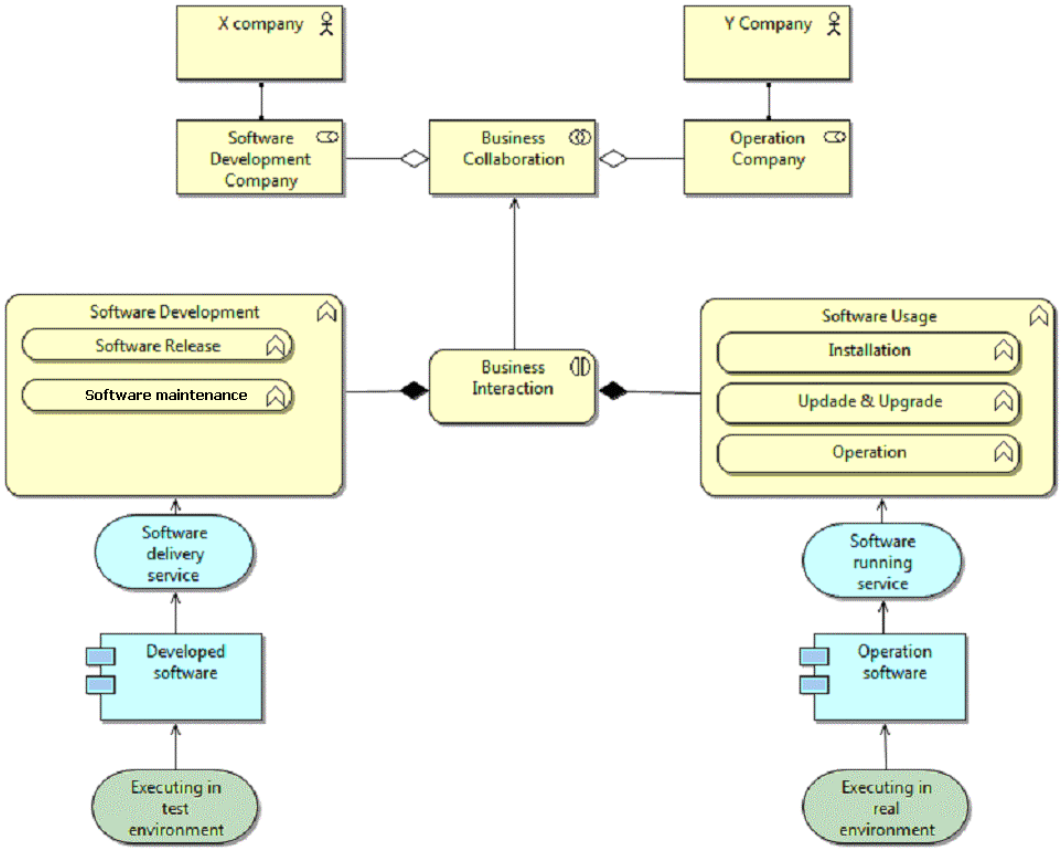

Fig. 2. Relations between a software development company and an operation company

The simplified motivation model (Fig. 1) assumes that the main enterprise drivers for using appropriate software systems are increased competitiveness through improved efficiency and quality. These improvements and increases should be assessed. The goal defined is the optimization of enterprise and software architectures, achieved by fulfilling enterprise and software architecture requirements, which can be functional or non-functional. For non-functional requirements (quality attributes) a relation is investigated on enterprise and

software systems levels. This relation is based on two principles of Enterprise Architecture (EA) which can influence software architecture, and vice versa.

Additionally, the relation between two organizations is interesting. The first organization develops software for a market segment and the second uses this software in its line of business operation. These two organizations may be completely separate companies, or one organization which has a great influence on another organization (Fig. 2). This paper only investigates the relation of quality attributes between enterprise and software levels. To ensure that these organizations cooperate effectively it is necessary to build broader and deeper relationships which go beyond the simple rules of cooperation between the client and the contractor. In [11] a SMESDaD (Synergetic Methodology for Enterprise Software Development and Deployment) methodology is presented which concerns the operation and cooperation of two organizations (enterprises). The first of these organizations is an IT company supplying the software (SDE - Software Delivering Enterprise) for the main business line of the second company, which operates on the market (MOE -Market Operating Enterprise).

4. QUALITY ATTRIBUTES OVERVIEW

Quality may comprise several aspects, as presented in Fig. 3. It is assumed that the quality is composed of process, data, product, service, usage and business qualities.

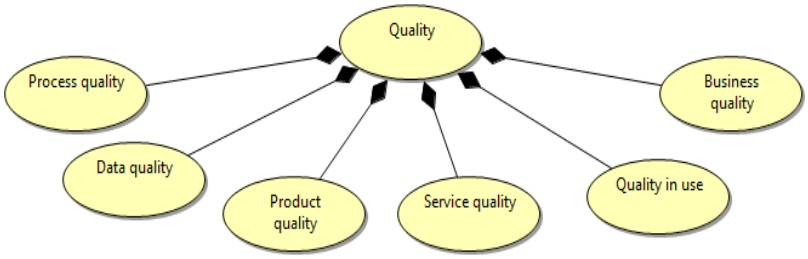

Fig. 3. Composition of quality

Process quality. In COBIT [12] a process is defined as ‘a collection of practices influenced by an enterprise’s policies and procedures that take and manipulate inputs from a number of sources (including other processes), and produce outputs (e.g., products, services)’.

Paper [13] demonstrates the results of the application of the Process Quality Measurement Model (PQMM) that can be acquired by analyzing the process quality for software organizations. The following characteristics are distinguished: maintainability, reliability, fault tolerance, functionality, and usability. For each characteristic (quality attribute) sub characteristics and measures are defined.

Data Quality. Data or information quality is interpreted in different ways. Some examples are given below. Paper [14] identifies and explains the requirements of collaborative Business Processes (BPs) on the quality of a company’s data resources. The paper shows which combinations of data classes and DQ dimensions are crucial for the different collaborative BPs in business networks. In [15] the following data quality dimensions are proposed:

- Accuracy - the extent to which data correctly represents an action or a real-world object.
- Completeness - the extent to which values are present in a data collection.
- Timeliness - the extent to which data represents the real world at a given point in time.

- Consistency - the extent to which data in one database corresponds to data in a redundant or distributed database.
- Relevancy - the extent to which data is applicable and helpful for the task at hand.
- Accessibility - the extent to which data is available at a given point in time.

In COBIT [12] the following information criteria (qualities) are proposed:

- Effectiveness should meet the needs of the information consumer who uses the information for a specific task. This corresponds to information quality goals: appropriate amount, relevance, understandability, interpretability, and objectivity.
- Efficiency relates more to the process of obtaining and using (i.e. requiring few resources, such as physical effort, cognitive effort, time, and money). This corresponds to the following information quality goals: believability, accessibility, ease of operation, and reputation.
- Integrity means the information is free of errors and complete. This corresponds to the following information quality goals: completeness and accuracy.
- Reliability means the information is true and credible. Corresponding quality goals: believability, reputation, and objectivity.
- Availability of the information quality goals means accessibility and security.
- Confidentiality corresponds to the restricted access information quality goal.
- Compliance, in the sense that information must conform to specifications, is covered by any of the information quality goals, depending on the requirements.

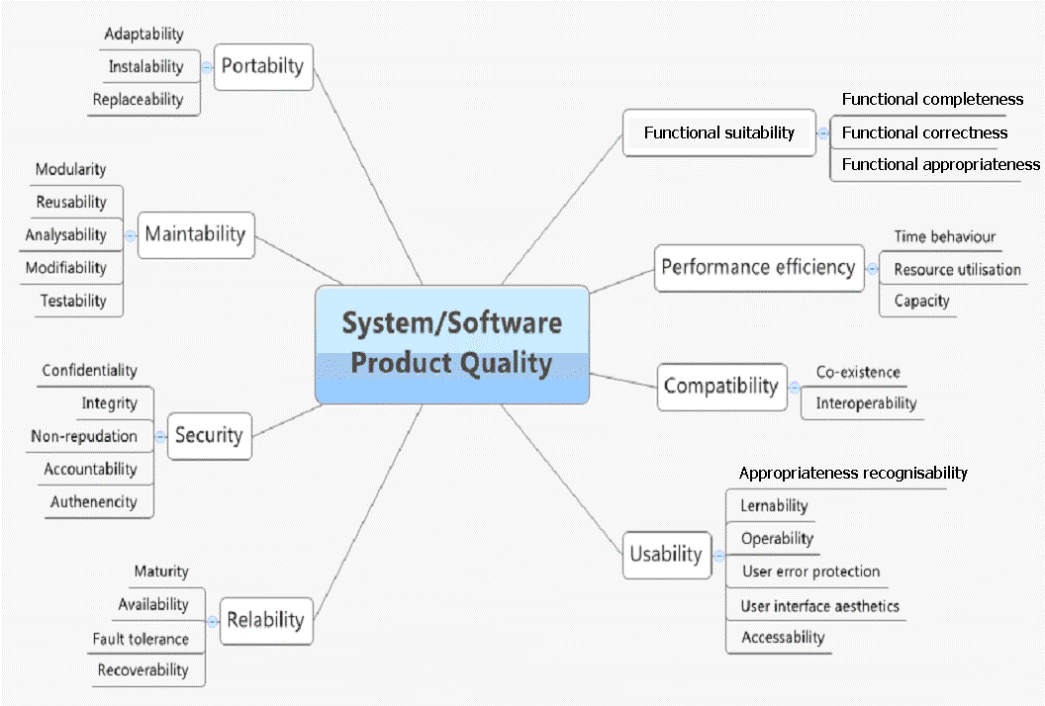

Fig. 4. The ISO/IEC 25010 classification of product quality

Product Quality. The ISO/IEC 25010 standard defines the Systems and software quality models [16] entitled SQuaRE (Systems and software Quality Requirements and Evaluation). This edition of ISO/IEC 25010 makes obsolete and replaces ISO/IEC 9126-1:2001. The standard (SQuaRE) defines system quality attributes as presented in Fig. 4.

Service Quality. There are different concepts for the qualification of services that are common to various domains. General QoS categories include: Performance, Dependability (comprising reliability, availability, safety and integrity), Security, Integrity, and Coherence. Characteristics defined within these categories are described in detail in [17]. In the paper [18] an initial business-level service quality model is proposed, aimed at qualifying services for construction projects.

Quality in use. Quality-in-use (QinU) and Product Quality specifies characteristics from the user perspective. QinU defined in ISO/IEC 25010 standard [16] means "the capability of a software product to influence users' effectiveness, productivity, safety and satisfaction to satisfy their actual needs when using the software product to achieve their goals in a specified context of use". The QinU model defines classification for quality in use presented in Fig. 5.

Fig. 5. The ISO/IEC 25010 classification of quality in use

Business quality. COBIT [12] provides a framework that assists enterprises in achieving their objectives for the governance and management of enterprise IT. Enterprises exist to create value for their stakeholders in terms of benefit realization and risk and resource optimization.

In the paper [19] the following IT business values are distinguished: (1) customer value creation, i.e. how the company creates value for the customer, (2) earnings logic, i.e. how the company yields a profit from its operations, (3) value network, i.e. external relationships for value creation, (4) resources and capabilities, (5) strategic decisions about competitive positioning, and (6) target markets or customers.

The paper [20] examines how system dynamics can be used in evaluating IT business value on a company level. System dynamics were utilized to construct a value creation model for an existing Gaming Management System. This value creation modeling covered two dimensions: (1) structural evaluation of IT impacts with cause and effect models, (2) dynamic evaluation and simulation of value realization over time. The examined approach proved its potential for providing a common language for technology and business parties, thus improving IT business alignment. From the point of view of economic worth, we consider company earnings logic as a reference point for reflecting and linking IT business value.

5. BUSINESS, DATA, APPLICATION, TECHNOLOGY LEVEL QUALITY ATTRIBUTES

For an enterprise it is important to gain business value, which can be achieved in different ways. It is appropriate to define the quality attributes of enterprise architecture in order to obtain business value.

In the exploratory case study [9] ten quality attributes are identified for EA products and services, utilizing data collected from 14 EA practitioner interviews. EA quality attributes are

used to describe the non-functional characteristics of EA products, services and processes that comprise the overall quality of EA. EA product quality has the following six characteristics [9]:

- Clarity and conciseness. EA descriptions should compress a fairly large amount of information into a set of models, while still maintaining clarity
- Granularity. Produce architectures that can provide both a holistic view and a sufficient level of detail.
- Uniformity and cohesion. Lower level architectures should conform to the upper level architectures. Cohesion should enable developing a set of models from different viewpoints.
- Availability. Availability should consider all types of EA products: documents, models and reports.
- Correctness. Three potential reasons for erroneous products were identified: the data sources used in the modeling of the architecture, outdated architecture, and incomplete architecture.
- Usefulness. EA products should be relevant and potentially beneficial to their users.

EA service quality has four characteristics [9]:

- Availability and timing. EA services should be available when they are requested and at the right time to yield the greatest benefits to the recipients.
- Awareness. Service customers should be aware of the services available, the conditions on which they are offered, and their best practices and potential benefits.
- Activeness. As with any service, activeness and a can-do service attitude from the service organization was perceived to be an important EA service quality factor.
- Usefulness. EA services should be practical and useful for their recipients. The services should be motivating and beneficial to use.

The enterprise architecture is built according to TOGAF with business, data, application, and technology layers.

This paper [21] proposes the idea of EA quality attributes and their description using EA quality attribute general scenarios. Finally, as a sample of EA quality attributes, EA maintainability has been defined and characterized. According to the IEEE definition, system maintainability is defined as the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment [22]. EA maintainability is defined as the ease with which EA components can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment. EA maintainability has different meanings in different layers of EA:

- In the business layer, EA maintainability is the ease with which strategic components and business functions and processes can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment.
- In the data layer, EA maintainability is the ease with which data components including data entities, physical data structures, data dictionaries, documents etc. can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment.
- In the application layer, EA maintainability is the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment. EA maintainability in the application layer is broader than software maintainability.
- In the technology layer, EA maintainability is the ease with which technology components including network structures, network protocols, and software and hardware infrastructures, etc. can be modified to correct faults, or improve performance.

The quality of an architecture can be investigated in different ways, but investigation based on quality attributes seems the most interesting. In [23] SOAROAD methodology was proposed for assessment of software architectures developed according to SOA principles. During system development several stages and corresponding architecture evaluation goals can be identified. The first stage is related to formulation of a strategy for a new system development or the integration of existing software. The next stage consists of proposing competitive architectural approaches and assessing them with respect to selected quality attributes. The third stage has as an input the assumed system architecture and aims to identify requirements (usually expressed as scenarios) and determine risks and costs for achieving the assumed scenario responses. This step can be referred to as early architecture evaluation. The last stage that can be considered as late architecture evaluation is related to software verification and validation resulting in test case specification used for TDD (Test Driven Development) or BDD (Behavior Driven Development) approach. In this paper, application of the Architecture Tradeoff Analysis Method (ATAM) [24] is proposed, as this is a mature, scenario-based, early method for architecture assessment. ATAM defines a quality model, an organizational framework for evaluation process and expected results: sensitivity points, tradeoffs and risks. The first stage of the evaluation process is the collecting of information on expected system qualities, architectural approaches and decisions from architecture documentation and interviews with stakeholders. Subsequently, a team of experts analyzes selected properties of components to identify sensitivity points and evaluate risks.

When developing a software system, quality attributes play an important role. In the paper [25] construction methods are proposed for Supervisory Control and Diagnostic Systems (SCDS) based on a CAN bus. Fulfilling real time constraints has become a major aspect of CAN bus time behavior. The studies [25, 26] enumerate the possibilities of using various communication mechanisms of CAN, applicable for real-time systems. The proposed solutions are especially suitable for general-purpose SCDS.

6. SURVIVABILITY QUALITY ATTRIBUTE ANALYSIS

The survivability quality attribute is quite interesting because it has similar meaning on enterprise, system, and network levels. In military environments survivability is an important quality attribute which refers to the ability to remain alive or continue to exist. There are several types of survivability in the military, including aero system survivability, naval survivability, and weapon system survivability, etc.

In [27] for survivability several architecture tactics are proposed based on solutions: multiple layers of protection, redundancy and static diversity, physical constraints in the architecture to impose modular isolation and containment, detection and correlation, adaptive response, and design based on weak assumptions.

An example of using survivability on an enterprise level is given in the paper [28], in which an analytical model-based approach is presented to assess the survivability of global software development projects. In this paper, the survivability metric under study is APD, meaning average project downtime after a specific failure event until full project recovery. Examples of software project disasters in global software development environments are: breakdown in communication with customer, breakdown in communication with a remote site due to lack of cultural sensitivity, loss of critical software due to process issues, loss of software due to improper maintenance, loss of software due to improper security procedures, etc. In the paper [22] survivability implementation techniques and examples of survivability architectures are introduced and discussed.

In the paper [29] a unified definition for reliability, survivability, and resilience is introduced. Resilience is defined as the time taken by a system to return to equilibrium (or

acceptable resistance or reliability after a failure). In other words, this is the period starting from the moment when the system's resistance level drops below the persistence level and subsequently rises (due to built-in recovery or repair mechanisms) above the persistence level.

Persistence is the minimum level of resistance that a system must maintain to be operational. For example, a persistent system may be specified as: $S(t) > ST$, where $S(t)$ is the resistance (reliability) measured with survivor function, and ST is the persistence threshold that the system must maintain to avoid being 'destroyed' by catastrophic failures.

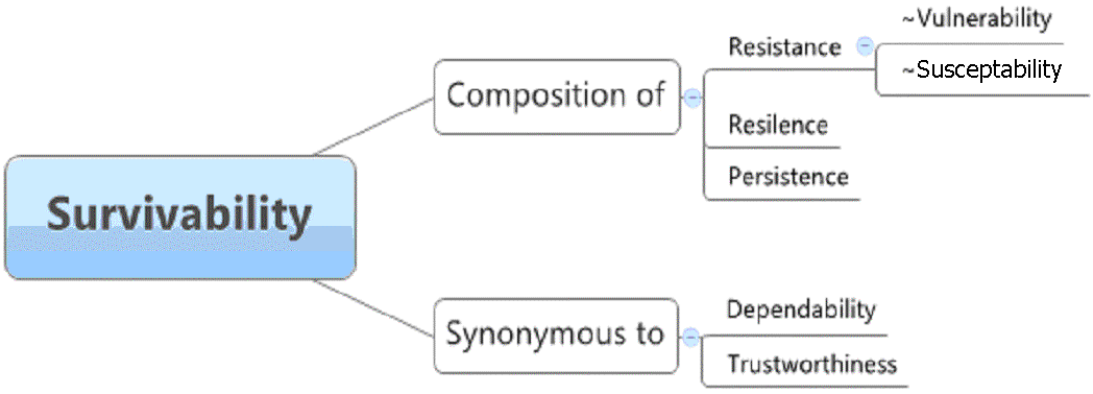

Fig. 6. Survivability characteristics

The Fig. 6 shows that survivability is composed of resistance, resilience, and persistence. The resistance consists of inverse values of vulnerability (inability to withstand a hit) and susceptibility (the inability to avoid being hit).

These three concepts of survivability, dependability and trustworthiness are essentially equivalent to their goals and address similar threats [30]. Dependability is the ability of the system to deliver a service that can be justifiably trusted. Survivability is the capability of a system to fulfill its mission in timely manner. Trustworthiness gives assurance that a system will perform as expected.

8. CONCLUSIONS

It is important to investigate the complex and multidimensional relations between enterprise and software architecture. The investigation can be performed in different ways. In this paper it is assumed that development of enterprise and software architecture can be done in a similar way using development based on the SEI approach [31]. In this approach one of the first steps is determining NFR (Non Functional Requirement), using, for example, QAW (Quality Attribute Workshop). In the developing of both types of architectures we should start with the determination of quality attributes. It is then necessary to investigate the relations between attribute meaning on enterprise and software architecture levels.

The presented proposal can be used for different application domains. In the paper a military application was considered using a survivability quality attribute, which can be used in military scenarios and has a similar meaning on enterprise, system, and network levels.

9. REFERENCES

[1] The Open Group: TOGAF Version 9.1 (2009-2011), p. 692.

[2] Lankhorst M.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Enterprise engineering series, Springer (2009).

[3] Werewka J.: From Enterprise Architecture to Software Architecture, Military Approach. Presentation on 6th International Scientific – Technical Conference Application of Data Exchange Networks in Military and Civilian Technology, OBRUM, Chorzów 2014, Poland,
https://www.researchgate.net/profile/Jan_Werewka/contributions.

[4] Werewka J., Jamróz K., Pitulej D.: Developing Lean Architecture Governance in a Software Developing Company Applying ArchiMate Motivation and Business Layers, In book: Beyond Databases, Architectures, and Structures, Vol. 424, pp. 492-503, Springer International Publishing (2014).

[5] The DoDAF Architecture Framework Version 2.02, US Department of Defense. http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf

[6] MOD EA & MODAF Documents, British Ministry of Defence, http://www.modaf.org.uk.

[7] NATO Architecture Framework V.4.0., http://nafdocs.org.

[8] Alghamdi A. S., Ahmad I.: Comparative Analysis of Defense Industry Frameworks for C4I System, Second International Conference on Computer Engineering and Applications, 2010, pp. 443-447.

[9] Niemi E., Pekkola S.: Enterprise Architecture Quality Attributes: A Case Study, 46th Hawaii International Conference on System Sciences, 2013, pp. 3878-3887.

[10] ArchiMate® 2.0 Specification, Open Group Standard, 2009-2012, pp. 183, http://pubs.opengroup.org/architecture/archimate2-doc/toc.html.

[11] Rogus G., P. Skrzyński, P. Szwed, M. Turek, J. Werewka: SMESDaD – a Synergetic Methodology for Enterprise Software Development and Deployment. In: Aspects of production engineering and management. ed. Piotr Łebkowski, AGH, 2011.

[12] COBIT® 5: A Business Framework for the Governance and Management of Enterprise IT, ISACA, ISBN 978-1-60420-237-3, United States of America (2012).

[13] A. S. Guceglioglu, O. Demirors: The Application of a New Process Quality Measurement Model for Software Process Improvement Initiatives, 11th International Conference On Quality Software, 2011, pp. 112- 120.

[14] C. Falge, B. Otto, H. Österle: Data Quality Requirements of Collaborative Business Processes, 2012 45th Hawaii International Conference on System Sciences, pp. 4316 – 4325.

[15] B. Otto and V. Ebner: Measuring Master Data Quality: Findings from an Expert Survey, in M. Schumann, L. M. Kolbe, M. H. Breitner and A. Frerichs, eds., Multikonferenz Wirtschaftsinformatik 2010, Göttingen, 2010.

[16] ISO/IEC, "ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models," International Organization for Standardization, Tech. Rep., 2010.

[17] UML Profile for modeling quality of service and fault tolerance characteristics and mechnisms v1.1, formal/2008-04-05, http://www.omg.org/spec/QFTP/1.1/PDF/.

[18] M. Bjeković, S. Kubicki: Service quality description – a business perspective, Proceedings of the Federated Conference on Computer Science and Information Systems, 2011, pp. 513–520.

[19] S. Nenonen and K. Storbacka: Business model design: conceptualizing networked value co-creation, Int. Journal of Quality and Service Sciences, vol. 2, no. 1, 2010, pp. 43-59.

[20] H. Töhönen, M. Kauppinen, T. Männistö: Evaluating the Business Value of Information Technology. Case Study on Game Management System. 22nd International Requirements Engineering Conference, IEEE, 2014, pp. 283-292.

[21] P. Närman, P. Johnson, L. Nordström: Enterprise Architecture: A Framework Supporting System Quality Analysis, 11th IEEE International Enterprise Distributed Object Computing Conference, 2007, pp. 130-141.

[22] P. Travainen: Survey of Survivability of IT systems, Proceedings of the 9th Nordic Workshop on Secure IT-Systems, Helsinki, 2004.

[23] P. Szwed, P. Skrzynski, G. Rogus, J. Werewka: SOAROAD: An Ontology of Architectural Decisions Supporting Assessment of Service Oriented Architectures, Informatica, Special Issue: Advances in Semantic Information Retrieval Guest Editors: Vitaly Klyuev, Vol. 38, 2014; 38, 31-42.

[24] P. Clements, R. Kazman, M. Klein: Evaluating Software Architectures: Methods and Case Studies. 2001.

[25] M. Dach, J. Werewka: Accelerator's supervisory control system based on CAN bus. Archives of Control Sciences, Vol. 18 (LIV), No. 3, 2008 (pp. 357-383).

[26] J. Werewka: The design of distributed reactive event systems based on the CAN bus on the example of the supervisory control-diagnostic system. (In Polish) Journal: Szybkobieżne Pojazdy Gąsienicowe (23), No. 1, Gliwice, 2008 (pp. 31-42).

[27] J. Chong, P. Pal, M. Atigetchi, et al: Survivability Architecture of a Mission Critical System: The DPASA Example. In: the 21st Annual Computer Security Applications Conference. Tucson, Arizona, USA, December 5-9, 2005.

[28] A. Avritzer, S. Beecham, J. Kroll, D. S. Menasche, J. Noll, M. Paasivaara: Survivability Models forGlobal Software Engineering. 2014 IEEE 9th International Conference on Global Software Engineering, pp. 100-109.

[29] Zhanshan (Sam) Ma: Towards a Unified Definition for Reliability, Survivability and Resilience (I): the Conceptual Framework Inspired by the Handicap Principle and Ecological Stability, Aerospace Conference, 2010 IEEE, pp.1 – 12.

[30] J. – C. Laprie: Dependability vs Survivability vs Trustworthiness. 42nd 10.4 meeting. Workshop on Dependability and Survivability.

[31] L. Bass, P. Clements, R. Kazman: Software Architecture in Practice (SEI Series in Software Engineering), Third Edition, 2012.