Defining Complexity Factors for the Architecture Evaluation Framework

Oleksiy Mazhelis*, Jari A. Lehto †, Jouni Markkula* and Mirja Pulkkinen*
*Information Technology Research Institute,
University of Jyväskylä, P.O.Box35, FIN-40014, Jyväskylä, Finland,
Email: {mazhelis,markkula,pulkkinen}@titu.jyu.fi
†Nokia Networks, P.O.Box 407, FIN-00045 Nokia Group
Email: Jari.A.Lehto@nokia.com

***Abstract*— The design and implementation of telecommunication systems is an incremental and iterative process, and system architectures may need to be revised and refined several times during their lifetime. Formal evaluation facilitates the identification of the weak points, where improvements are due in these architectures. In the domain of telecommunications, such evaluation can be based on the Architecture Evaluation Framework (AEF).**

During the evaluation, a deep understanding of the processes within a system is needed. Meanwhile, the systems being designed are usually complex systems encompassing a large number of components with an intricate pattern of interaction between them. As a result, it is extremely difficult to understand, predict and control the behavior of such systems.

Theoretical studies in the field of complex systems describe potential reasons of system complexity, and explain its possible outcomes, as reflected in system structure and behavior. This knowledge may be utilized in architecture evaluation, in order to deepen the understanding of the interactions imposed by the architecture, as well as to extend the understanding of the involved architectural tradeoffs. For this, the complexity factors should be taken into account during the evaluation. However, no such factors are involved in the current version of the AEF.

In this paper, the attempt is made to identify how the knowledge about properties of complex systems could be utilized for the evaluation of information system architectures. Based on the theoretical advances in the field of complex systems, a list of the complexity factors to be included in the AEF is compiled. These factors are going to be further refined, as the AEF is employed for evaluating real-world architectures.

I. Introduction

From a structural viewpoint, a (computer) system can be defined as "a set of elements and relations among them, considered as a whole and for which there is a recognized purpose and capability" [1]. The system elements are referred to as components. A component may itself be a (sub)system consisting of a number of components, etc.; this recursion stops when a component is considered to be atomic, i.e. when its further decomposition into components is impossible, or is of no interest [2].

The components comprising a system can be software or hardware components, or a mixture of both. The system concept is illustrated in Figure 1, depicting the Mobile Internet as an example of a system. This system includes a number of components such as mobile terminals, the network part, and the content provision part. Each of these components represents a system of high complexity, interacting with other components in order to provide certain capabilities (e.g. voice communication). As such, these parts are comprised of a number of components. Network part, for example, is decomposed into a number of network elements. Each of the network elements is further decomposed into a number of software and hardware components (cf. Figure 1).

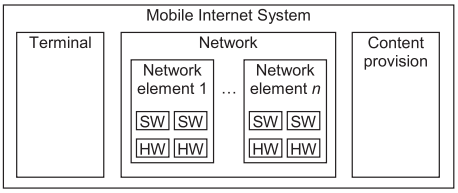

Fig. 1. An example of a system

The set of system components along with the relationships between these components represent a logical structure of this system. More than one type of relationships may exist between the components (e.g., hierarchical decomposition, data flow, control flow, interaction sequencing), and more than one systems structure may be used to capture these relationships. This logical structure or structures, together with the functional characteristics of the system, can be referred to as the *architecture of the system* [1].

The architecture of a system should match well with the underlying business-drivers (i.e. planned effects on the market), involve suitable external components, be coherent with other architectures in a product family, etc. In order to determine the appropriateness of an architecture with respect to these requirements, a formal architecture evaluation can be conducted. The process of evaluation can be based on a specific evaluation method, such as the Software Architecture Analysis Method (SAAM) and the Architecture Tradeoff Analysis Method (ATAM) [3]. Recently, building on the ideas of ATAM, the Architecture Evaluation Framework (AEF) [4] has been developed. The AEF defines the set of procedures and tools required for the evaluation of system architectures, and it is tailored to the domain of telecommunications.

In the evaluation process, a deep understanding is needed of the processes within the complex system whose architecture

is being evaluated. Meanwhile, the contemporary telecommunication systems usually encompass a large number of components with intricate patterns of interaction between them. Such systems can be considered as complex systems since [5]:

- The systems are "sufficiently" complex, i.e. they have a large number of components and interactions between them. As a result, it is extremely hard or impossible to know the details of all interactions taking place in the system.
- On the other hand, interactions between individual components may produce non-linear effects. Such non-linear interactions even in simple systems may be impossible to model analytically.

Thus, as the complexity of a system grows, it becomes increasingly difficult to understand and predict the system behavior.

Theoretical studies in the field of complex systems describe potential reasons for system complexity, stemming both from the structural and from the behavioral properties of the system components and interactions. These studies also describe possible outcomes of complexity, as reflected in the system structure and behavior. Such theoretical knowledge may be highly useful, when a deeper understanding of the processes within complex systems is needed. Therefore, it seems plausible that the awareness of complexity-related issues can be utilized in system architecture evaluation, in order to extend the understanding of the interactions imposed by the architecture, and the involved architectural tradeoffs (e.g. whether modifiability is preferred to performance). For this, it can be advisable to take the complexity factors into account in the AEF-based evaluation process. However, such factors are missing in the current version of the AEF.

This study aims at extending the AEF with additional factors, which would allow to take into account the complexity-related aspects in the course of architecture evaluation. Those properties of complex systems, whose analysis is deemed useful during the architecture evaluation process, will be considered. Based on the theoretical advances in the field of complex systems, a preliminary list of the complexity factors to be included in the AEF is produced. These factors are to be further refined, as the AEF is employed for evaluating real-world architectures.

The paper is organized as follows. After a short introduction of the AEF in Section II, an overview of the theoretical advances in the field of complex systems is provided in Section III. After that, in Section IV, a list of the complexity factors to be included in the AEF is proposed. Finally, Section V gives conclusions to the paper.

II. Architecture Evaluation Framework

The Architecture Evaluation Framework [4] is intended for evaluating an architecture from the viewpoint of one or several business drivers. Besides, the purpose of the framework is to achieve a deeper, common understanding of the architecture, and the involved trade-offs, among the stakeholders. The AEF can be also used to facilitate the checking of the suitability of external components, evaluating the internal coherence of a product family, or prioritizing development efforts. The AEF represents an in-house evaluation framework building on the ideas of ATAM [3]. The development of the AEF was justified by two main arguments. First, the framework's scope of concerns had to be tailored to the company, and also the terminology was adapted to the company's internal vocabulary, in order to prevent the evaluators from the need for additional step of learning. Second, the time and efforts required for the evaluation had to be kept to a minimum, since the framework will be used in business environments in projects with a limited time frame. In response to this requirement, the AEF e.g. excluded an explicit consideration of scenarios. In this section, a brief summary of the framework is provided; for further details, the interested reader may consult [4].

According to the AEF, the evaluation starts with the selection of the evaluation team of experts representing the complete life-cycle of the system, and determining the relevant business drivers. After the business drivers have been identified, the evaluation proceeds as follows.

First, the hierarchy of the domain-specific factors which are potentially important from the architectural point of view is established. The top-level factors in the hierarchy are generic (e.g. operational efficiency), while leaf-level factors represent domain-specific factors (e.g. measures of operational efficiency for the target product). The framework provides the evaluators with a set of architectural factors; however, the evaluation team is supposed to consider these factors and adjust them to the particular domain of concerns. As a result of the discussions invoked during this step, a common understanding of factors among the evaluators should be achieved.

Having identified the factors, their relative importance, or factor weights, is determined by using the Analytic Hierarchy Process technique [6]. For each branch and for each level of the hierarchy, the factors are pair-wise compared with respect to a specific business driver, and their relative importance is defined (e.g. factor A is two times more important than the factor B, with respect to the business driver C). These values of pair-wise relative importance are subsequently used to calculate the relative importance coefficients for each factor. The use of these coefficients makes the framework tolerant of the presence of less relevant factors. As a result of the pair-wise comparison, the coefficients corresponding to the less relevant factors will be assigned low values, virtually excluding these factors from further use in the evaluation process. Consequently, the evaluation team may decide to exclude the factors, to which low importance coefficients were repeatedly assigned, from the factor hierarchy.

For each of the leaf-level factors, suitable metrics in the form of specific questions are defined. An example of such a question may be "what is the mean time between service interruptions?". In fact, the leaf-level factors are represented directly by these questions. The evaluation is performed by asking the members of the evaluation team to answer these questions and to evaluate, for each question/factor A, the effect

that the given answer has on the business driver C being considered. The effect is evaluated as a real number between 0 (negative effect) and 1 (positive effect), and this value is assigned to the corresponding metric.

The values of metrics and the values of relative importance coefficients are used to evaluate the overall appropriateness score of the architecture. Two alternative architectures can be compared quantitatively using the obtained scores, or the architecture being evaluated can be compared with an ideal one. Finally, a sensitivity analysis is run to evaluate how sensitive the appropriateness score is to the changes in the individual factors (importance coefficient).

While complexity-related factors may be of high value for architecture evaluation, such factors are missing in the current version of the AEF. Before defining the complexity-related factors, the theoretical advances in the domain of complex systems are overviewed in the following section.

III. Complex Systems

A system is considered as complex, when it displays complex behavior, i.e. when a large number of components and interactions exist in the system, or "when its elements interact in a non-linear fashion, such that it is impossible to predict the behavior of the system as a whole from knowledge of the elements themselves" [7]. For example, mobile telecommunication networks represent complex systems that are inherently difficult to understand, due to a number of factors including structural complexity, network evolution, connection diversity, dynamical complexity, and node diversity [8].

While individual events in complex systems have little or no predictability, taken together these events often form a mass phenomenon. As a result, at the system level, global patterns of behavior may appear (referred to as emergent properties). At the same time, due to high complexity, understanding the individual interactions and their causations is unlikely; equally, it is not usually practicable to predict the individual events. Appearance of these system-level properties referred to as the emergence phenomenon is an important feature of complex systems.

Traffic jam can be seen as an example of an emergent property, which may be induced by a complex interaction of a large number of motor vehicles. The appearing and dissolving of traffic jams cannot be explained by analyzing the properties of the individual vehicles, the internal structure and organization of their parts. Rather, the interactions between the vehicles and the environment of their action should be considered to model the traffic jams with a reasonable degree of accuracy. In the context of mobile networks, the emergent phenomenon of network congestion is analogous to the traffic jam phenomenon. In software systems, system safety is one of emergent properties arising in the interactions among system components. Therefore, even if each component's operation strictly complies with its specified behavior, the resulting system-level behavior may still fall into a hazardous state [9]. Similarly, the reuse of a software component may result in a less safe system as evidenced by the case of the medical linear accelerator Therac-25 [10] or by the case of the accident with the spacecraft Ariane 5 [11]. System-level performance and reliability are examples of emergent properties as well.

In this section, Holland's theory of complex adaptive systems, and the studies focused on the properties of complex networks are overviewed.

A. Holland's Complex Adaptive Systems theory

John Holland proposed his theory of complex adaptive systems (CAS) back in 1970s. Often, Holland is called the founder of the domain of genetic algorithms – in his seminal book "Adaptation in Natural and Artificial Systems" [12], he described the mechanisms of reproduction, mutation and selection, which represent a simplification of natural evolution, and are widely employed in the variety of genetic algorithms.

According to this theory, system components represent autonomous agents that interact with each other [13]. These agents are able to respond locally to stimuli and thereby adapt to a dynamically changing environment. The agents, through interaction with other agents, are capable of transmitting information; consequently, the system as a whole may be able to learn (adapt or evolve). Below, the properties and mechanisms commonly found in CAS are described, and the processes of learning or adaptation are briefly considered.

1) Properties and mechanisms of CAS: Holland postulated that common to all CAS are four properties (aggregation, nonlinearity, flows, and diversity) and three mechanisms (tagging, building blocks, and internal models) [14].

a) Aggregation property: The property of aggregation is manifested in CAS in two ways: as a technique for constructing models, and as an integration of less complex behaviors into a more complex behavior [14]. In the first sense, aggregation refers to the process of simplifying complex systems by aggregating the properties of system components into categories. In the second sense, aggregation refers to the fact that, through interaction, less complex systems aggregate into more complex systems. As a result of such interaction, complex large-scale behavior emerges. This effect of an appearing more complex behavior due to the interaction of less complex agents is known as the emergence phenomenon.

The emerging more complex systems can be seen as agents at a higher level, or meta-agents. The properties of meta-agents (which are the aggregate properties in the first sense above) are referred to as the emergent properties. An example of such a meta-agent is an economy, emerging because of the interaction of a number of participating firms. An example of an emergent property that the economy has is the gross domestic product.

b) Tagging mechanism: For an agent, the process of selecting another agent to interact with is not completely random. Whether a particular (type of) agent to interact with is selected depends on experiences obtained in past interactions with agents of this type.

Therefore, a mechanism for differentiating agents of different types is needed. According to [14], agents distinguish each other by using tags, such as flags/banners in the warfare, or trademarks in commercial interactions.

By allowing agents to distinguish each other, tags also facilitate filtering, specialization, and cooperation between agents, and by doing so, they facilitate the formation of aggregates.

c) Nonlinearity property: A numerical property is linear w.r.t. the values of its constituents, if the property's value is a linear function of the constituents' values. Linearity is often assumed, as it simplifies the analytical solutions. However, for CAS, this assumption does not hold, i.e. system properties are nonlinear functions of the constituents' values. Therefore, CAS are said to possess the nonlinearity property. As stated in [14, p. 23], "nonlinear interactions almost always make the behavior of the aggregate more complex than would be predicted by summing or averaging".

The nonlinearity has two implications. First, nonlinear interactions are likely to result in highly complex system behavior even when the number of interacting agents in the system is small. Consequently, often no analytical solution can be found that would relate the system level emergent properties with the properties of system components/agents. An example of such a complex behavior is the chaotic behavior of three celestial bodies under their mutual gravitational attraction known as the Newtonian three-body problem, or the fluctuation of the execution time of a program in a multi-task environment due to the influence of other concurrently running tasks.

Another consequence of nonlinearity is the difficulty and often impossibility of accurate prediction of the system behavior. Unpredictable consequences may occur, because only an incomplete picture of the system is available, and a tiny event may trigger a major shift in the system-level behavior.

d) Flows property: The flows represent various interactions between components/agents of a system. A flow could be thought of as an exchange of resources over a network of nodes and connectors. For instance, in the Internet, messages (resources) are exchanged over a network of computer workstations (nodes) connected with cables (connectors). Here, nodes designate the processors/agents, while connectors reflect the possible interactions between these agents.

The flows in CAS are not fixed; they can vary over time. Similarly, nodes and connectors can appear and disappear as agents adapt to their environment.

The flows have a multiplier effect and a recycling effect. The multiplier effect refers to the effect of propagating changes through the network. This effect means that an interaction (an exchange of resources) between two nodes often induces further interactions with other nodes, resulting in a chain of changes within the system. As a result, the effect of the initial change is multiplied during its propagation within the network. The recycling effect is a special case of multiplier effect, wherein the change is propagated through cycles in the network.

e) Diversity property: The existence of an agent depends on the other agents, i.e. on the agent's context. Agents of the same type fill a "niche" of inter-agents interactions; should this type of agents disappear, the system adapts by generating a new agent to fill that niche. As a result of such adaptations, different types of agents appear.

Thus, the diversity of agents is a product of progressive adaptation – a niche is an opportunity for new interactions, and agents of various types fill different niches.

f) Internal models mechanism: The internal models correspond to the mechanisms of prediction and anticipation, which actively determine an agent's behavior.

Two types of internal models should be distinguished according to [14]. The tacit internal models prescribe current action of an agent under implicit anticipation of a future desired state. A bacterium e.g. has the tacit internal model that suggests the movements in the direction of chemical gradient – it is implicitly assumed that food is located in that direction.

On the other hand, an overt internal model presumes the existence of a lookahead process, i.e. explicit internal exploration of possible alternatives, similar to the mental exploration of possible consequences before making a move in chess.

The model is called effective if it is capable of predicting future useful consequences. Variants of the internal models undergo progressive adaptations – less effective internal models are eliminated in the evolutionary process.

g) Building blocks mechanism: The problems, which people face daily, usually represent complex scenes, and often a complex scene is faced that has not been encountered before. To find a solution for a new complex scene, this scene is decomposed into a number of "reusable" elements, each of which are already separately tested (e.g. through natural selection and evolution). These reusable elements represent building blocks that can be employed in a variety of situations. Thus, a solution to a new problem can be found by combining relevant building blocks.

Combinations of building blocks generate the agents' internal models considered in the previous subsection [14]. The process of combining building blocks (and hence the internal model) advances orders of magnitude faster when the internal model is overt, as compared against the tacit internal model.

2) Adaptation or Learning: For modeling purposes, an agent can be described as a set of message-processing rules in a form of IF-THEN statements. In the process of learning/adaptation these rules are being changed and replaced with new ones, by using the procedures of credit assignment and rule discovery that are outlined below.

a) Adaptation by credit assignment: Credit assignment is the procedure of assigning and modifying "strength" to each rule so that the strength would reflect the degree of the rule's usefulness from the system point of view.

The strength of a rule is modified according to the past experiences, obtained after the rule was applied. For each interaction in which the agent is involved, the agent is assigned a "credit". In a chain of interactions, the credit is distributed along the chain according to the feedback obtained on later stages of interaction (with the first feedback coming from the environment). Thus, if a chain of interactions results in a successful interaction with the system's environment (whatever the environment means), the positive feedback from the environment is distributed backward along the interaction chain, increasing the strength of the involved rules.

When several rules simultaneously satisfy the conditions for interaction, only one of them needs to be selected for the interaction. For selecting the rule, bidding process is used, wherein a rule with the biggest bid is selected. The rule's bid, in turn, is proportional not only to the strength, but also to the specificity of the rule, i.e. more specific rules are preferred, everything else being equal. As a result of the credit assignment process, a default hierarchy of rules is developed, where the rules are organized according to their specificity.

b) Adaptation by rule discovery: While the credit assignment process allows prioritization of more useful rules (i.e. more specific rules with higher strength), the process of rule discovery is aimed at changing the agents' capabilities by replacing less useful rules (or parts thereof) with new ones.

The rules are seen as consisting of sets of building blocks. To modify a rule, two (parent) rules are selected with the probability proportional to the usefulness of the rule (called as fitness). The selected rules are crossed over (crossover operation), i.e. the building blocks, of which parent rules are composed, are recombined to produce offspring rules. The offspring rules then undergo a mutation, i.e. their building blocks are randomly modified to produce new ones. Finally, the produced offsprings substitute randomly chosen rules in the agent's rule set.

The above process of selection, crossing over, and mutation models the process of natural evolution. Repeated over and over again, it results in improved fitness of the rules of the agents comprising the system.

B. Network complexity

While Holland's theory of complex adaptive systems addresses in detail the adaptive character of CAS components/agents, a number of studies on CAS focus on the properties of the complex networks of agents' interactions, while paying less attention to the properties of the individual system components/agents.

1) Complex behavior at the "Edge of chaos": Stuart Kauffman, when studying evolutionary processes, attempted to find an answer to the question of what is the relationship between the average connectedness of genes and the ability of organisms to evolve. To answer that question, he studied the behavior of networks of varying degree of connectedness K defined as the (maximum) number of connections that a network node may have with other nodes. Thus, in the networks being studied, the behavior (state) of a node depended on the behavior (state) of up to K connected nodes. Kauffman found that the behavior of the network is largely determined by the degree of connectedness rather than by the rules governing the behavior of individual nodes.

Kauffman also recognized two types of CAS behavior: ordered (frozen or periodic) and disordered (chaotic). Later, Langton extended this classification by including the third type of CAS behavior – complex behavior residing at the transition point between ordered and disordered behaviors [15].

Thus, according to Langton, the order in the behavior of CAS can be categorized into three categories: chaotic, frozen or periodic, and complex; furthermore, the order in the system behavior depends on the degree of connectedness, i.e. on the number of connections/interactions in the system [5]. At least three types of connectedness can be distinguished corresponding to distinct types of system behavior:

- *Low connectedness*. The systems of this type exhibit fixed or periodic behavior.
- *Medium connectedness*. The systems with this type of connectedness are said to reside on an "edge of chaos". They exhibit complex behavior, which is illustrated in Figure 2 in a form of "twinkling islands of stability, changing shape at their borders".
- *High connectedness*. For the systems having high degree of connectedness, chaotic behavior is normally observed.

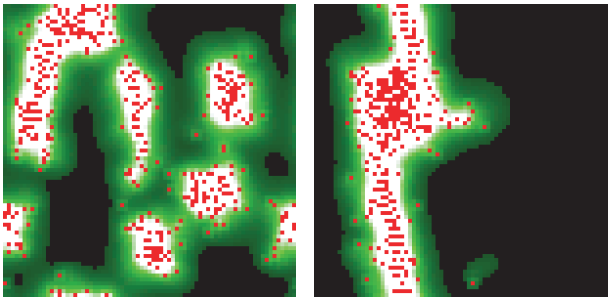

Fig. 2. Islands of stability produced by the complex model of slime in StarLogo simulator (`http://education.mit.edu/starlogo`)

Systems with complex behavior have different properties as compared with those having fixed, periodic, or chaotic behaviors. The systems with complex behavior can store and transmit information and therefore can adapt to a changing environment, while neither the systems with fixed/static behavior nor the chaotic systems are capable of doing this [15]. This is due to the fact that complex networks at the edge of complexity have a relatively small number of stable configurations – "basins of attractions" [5], and, having been disturbed, they quickly converge to one of these basins of attraction. A system at the "edge of complexity", thus, exhibits a self-organized order.

2) Small-world networks: During the last decade, increasingly much attention is paid to the structure and behavior of various networks. However, a significant breakthrough in this area can be attributed to the research of sociologist Stanley Milgram who discovered "Six degrees of separation" in 1960s [16]. According to his study, personal contacts that people have form a global social network such that arbitrary two individuals in this network are in average detached from each other just by a few degrees of separation (six according to the study). In 1983, Granovetter discovered the so-called weak links inside social networks, and described the role of these weak links [17].

The existence of weak links was later employed by Watts & Strogatz to explain the phenomenon of the small degree of separation in networks – the so-called small-world phenomenon [18]. According to [18], social networks can be quite

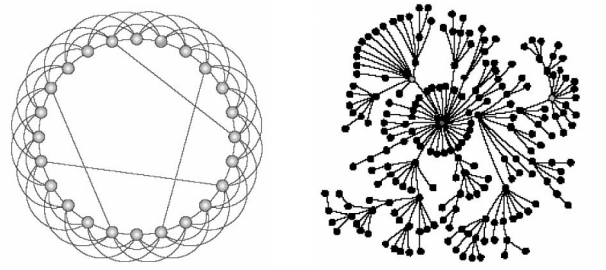

Fig. 3. Examples of egalitarian (left) and aristocratic (right) small-world networks (adopted from [8])

accurately modeled by the so-called small-world networks, which are neither regular nor completely random.

A small-world network, i.e. the networks with small degree of separation, can be created by randomly rewiring a small portion of vertices in a regular network. As more and more vertices are randomly rewired, the random type of network is approached. The bigger the proportion of the rewired vertices, the closer the network to the random network. Formally, a network is called as a small-world network, if the value of the mean geodesic (shortest) distance between vertex pairs scales logarithmically or slower with network size for fixed mean degree [19].

Within small-world networks, two types of networks can be identified [20]:

- *Egalitarian networks* (illustrated in Figure 3 on the left) have nodes with roughly same number of links;
- *Aristocratic networks* (a.k.a. scale-free networks illustrated in Figure 3 on the right) differ from the egalitarian networks in that the number of connections the nodes of a network have follows power low (fat-tail) distribution. In other words, these networks have hubs with a large number of connections. Examples of such networks are the Internet [21] and the World Wide Web [22].

The main properties of small-world networks can be summarized as follows [23], [20], [19]:

Fast information transfer. One of the most important properties of small-world networks is their high level of connectedness (or, equally, low degree of separation), which enables very high speed of information transfer across the network to be achieved. This, in turn, allows easy synchronization between network nodes.

Transitivity. The property of transitivity reflects the proportion of triangles of nodes in the network. In small-world networks, the value of transitivity is greater than in random networks (i.e. "the friend of your friend is likely also to be your friend").

Clustering. The nodes in these networks are highly clustered (their local cliquishness is high), i.e. nodes within a cluster are highly interlinked with each other, while their connectedness with the nodes of other clusters is lower.

Mixing patterns. Social networks are assortative (the vertices are more likely to be connected if they are of the same type), while other networks, including technological, information and biological networks, are disassortative (the vertices of distinct types are more likely to be connected).

Type of the network to a large degree determines its tipping point, i.e. the point at which any seemingly insignificant changes in a single system component are likely to propagate far throughout the system. Epidemics are an example of such far reaching changes that can be triggered, if the average number of other individuals an infected person infects is greater or equal than one (i.e. the average number of infected individuals equal one is the tipping point in this example). For egalitarian small-world networks, the tipping point depends on the number of random links; the bigger the proportion of random links, the lower the tipping point. For aristocratic networks, no tipping point exists, i.e. changes (such as worms in Internet) always propagate through the network.

Network resilience. Networks are robust to the distraction of random nodes. In other words, a relatively high proportion of randomly selected network nodes being excluded from the network have little effect on the overall connectedness of the network. At the same time, small-world networks are sensitive to the (targeted) distraction of weak links. In egalitarian networks, weak links are the links outside of a cluster of nodes; in aristocratic networks, weak links are represented by the hubs with a large number of links. In general, egalitarian networks are less sensitive to targeted distraction of weak links.

In the evolution process, networks appear as aristocratic networks. After reaching the saturation point, they are becoming egalitarian networks [20]. The following regularities have been observed in the process of (self-organized) network evolution [23]:

- Preferred attachment – a new node is connected to the nodes with probabilities proportional to the number of connections the nodes have (possibly multiplied with the fitness of these nodes);
- New links are added between existing nodes, and existing links are rewired;
- Duplication – parts of the network (building blocks) are duplicated and subsequently rewired.

Above, various properties of complex systems and networks have been summarised. In the next section, they are employed in order to extend the AEF with the complexity-related factors (questions).

IV. Including Complexity Aspects into the AEF Factor Hierarchy

Formal evaluation methods, such as AEF, facilitate the identification of the weak points in the architecture, for which improvements are due. In addition to the system components, the evaluation process should pay attention to the properties of the system as a whole. Often, analyzing these system-level properties is challenging as they are a product of interaction of a number of system components. Complexity theories, as considering emergent properties of complex systems, may assist in identifying and analyzing the above system-level properties during the architecture evaluation.

Below, we introduce a tentative list of questions, pinpointing various complexity-related properties of systems, which may be potentially important for the evaluation of a system architecture. These questions are intended to serve as leaf-level factors in the AEF factor hierarchy. The properties of complex systems discussed in the previous section are taken as the basis for the questions. The list in its current form has been produced by two of the authors, and underwent several iterations of refinement based on the discussions among the authors. Further refinement is to be made by merging the proposed questions with the AEF factor hierarchy and using the produced hierarchy for evaluating architectures.

As was described in previous sections, system-level properties are the result of complex interactions between numerous system components. Therefore, we consider separately the properties at the component level and at the system level. The system-level properties are further divided into behavioral properties (behavioral patterns), which reflect regularities in behavior at the system level, and the structural properties (structural patterns) that describe the organization of components in the system.

In the remainder of the paper, let $X_1, X_2, \ldots, X_N$ denote properties of system components that can be measured quantitatively or qualitatively, and let $Y_1, Y_2, \ldots, Y_M$ denote system-level properties that can be measured quantitatively or qualitatively. For example, X_1 and X_2 may denote the reliability (MTBF) of two components, of which the system is composed, and Y_1 may denote the reliability of the system. The terms "agent" and "node" are used interchangeably with the term "component"; similarly, "connection" and "link" is used interchangeably with "interaction".

A. Properties of components and interactions

The properties of individual components and interactions include the general properties of CAS described by Holland, and the properties of the network growth process. The questions concerning these properties are listed below.

1) Aggregation:

a) Into how many components the system can be decomposed?: System-level properties are produced as a result of the interaction of a number of components. How many components are involved in such interactions? E.g., the interaction of how many components needs to be taken into account to evaluate the grade of service (at the level of the network system), or the reliability of a base station (at the level of network elements)?

b) Is it useful to model the network as a system of interacting agents?: Here, the agent model implies that no "commanding officer" is needed in the system. The agents can act independently, and their behavior depends on the behavior of other agents as well as on the internal goals and rules of the agents themselves.

c) In how many interactions are the components involved (degree of connectedness)?: Interactions between system components can be modeled as a network of nodes (components) and connectors (interactions). A node (component) may be connected (i.e. may interact) to one or several other nodes. Can the nodes' degree of connectedness be described as low, medium, or high?

d) What is the number of types of interactions?: A component may interact with a variety of types of components; furthermore, more than one interface may be employed for interacting with the same component. In how many types of interactions (i.e. types of components to interact with and types of interfaces) are the system components involved?

e) How intensive the inter-component interactions are?: How often do the components interact? How is the time between interactions distributed (bursts of interactions vs. uniform time between interactions)?

2) Nonlinearity:

a) Linear vs. non-linear interactions: Let a system-level property Y be produced as a result of the interaction of several components with properties $X_1, X_2, \ldots, X_N$. Can the value of Y be expressed or accurately approximated as a linear function of $X_1, X_2, \ldots, X_N$?

3) Diversity:

a) Among the components with an identical purpose, how many types of components are available?: There may be components of different types, which have an identical function. For example, in a mobile network, different types of base stations, from cheaper and less reliable to more expensive and more reliable, may be deployed (or available for deployment) at the same time. How many types of components are encountered among those having the same purpose?

4) Flows:

a) Are there chains of interactions (flows)?: The interaction between components may trigger further interactions. Thus, a chain of interactions may result from a single interaction; this chain of interactions can be seen as a flow of a specific resource (e.g. messages in computer networks). At the level of a computer network, establishing an end-to-end communication channel and transferring information over this channel can be seen as an interaction flow.

b) Does the multiplier effect occur in the flows of interactions?: The multiplier effect means that the cumulative effect of an initial change (interaction) is increased (multiplied) as the change is propagating through the network. For example, if a message of length l (in bites), sent over the network, passes through m hosts before reaching the destination, in total lm bites are transferred through the network.

c) Does the cycling effect occur in the flows of interactions?: Are there cycles in the chains of interaction? If so, the recycling effect occurs – the effect of interactions at each node in the cycle is increased as these interactions are triggered multiple times.

5) Internal models:

a) Does a component have an internal model, which i) determines the component's behavior; and ii) can be changed according to a feed-back?: The behavior of an agent (component) can be represented by a set of IF-THEN rules. Can these rules be automatically modified to better fit to the agent's environment (special rules are needed for this

purpose)? For example, can the rules/policies of call admission control (CAC) be modified, if the resulting grade of service is not satisfactory? Are the rules of CAC adapted by taking into account the typical behavior of a customer group or a specific customer? Does the CAC take into account the load of the neighboring base-stations?

6) The way the network of system components evolves:

a) Is the network creation planned (in advance) by the enterprise, or does it emerge randomly (as a response to the actions of external actors, e.g. customers): Example of a planned network is a company LAN, whose structure is planned and implemented by the company. On the other hand, the network of WWW pages connected by hyperlinks is an example of a randomly created network, i.e. the network created as a result of actions of numerous external agents (web-masters, visitors of web-pages, etc.).

b) Does the process of network evolution/development involve preferred attachment?: According to the preferred attachment principle, a new node is connected to an existing node with probability proportional i) to the number of connections the existing nodes already have, and ii) to the fitness of the existing node (definition of fitness depends on the application domain).

c) Does the process of network evolution/development involve adding new links between existing nodes, or rewiring of existing links?: E.g., in a web-page, a new link may be added to refer to yet another (existing) favorite web-page. In turn, rewiring refers to substituting one existing link with an alternative one, referring to a different web-page.

d) Does the process of network evolution/development involve duplication of parts of the network with subsequent rewiring?: An example of duplication is the creation of a new web-page by using an existing web-page as a template. Hyperlinks in the new webpage may need to be subsequently rewired to meet the purposes of the page.

e) Does the process of network evolution/development involve saturation, i.e. does a limit exist on the number of connections a node may have?: Base-stations in telecommunications networks have a limited number of radio-channels they can serve; therefore, there is a maximum number of channels (saturation point) in these base stations. On the other hand, a web-page in the WWW network may have practically unlimited number of connections (hyperlinks) to other web-pages; thus, no saturation point exists in the WWW network.

B. System-level structural properties

Only the type of the network has been considered as a system-level structural property.

1) Network type:

a) What is the category, to which the network structure belongs?: Can the network structure be characterized as:

- Regular – network nodes have the same number of links, the regular pattern of inter-node links is usually present;
- Random – network nodes are connected randomly;
- Small-world egalitarian – can be seen as regular networks with a small proportion of connections randomly rewired; network elements have roughly the same number of links;
- Small-world aristocratic – number of links, which the nodes have, follows power low; as a result, a few hubs with a large number of links exist.

b) Can the system be considered as consisting of hierarchically organized modules, i.e. having a (hierarchical) clustering?: The modular structure of the network implies that nodes within a module have a high density of links between them, while a lower density of links exists between the modules. For example, according to the principles of high-quality software design [24, p. 218], the system-component decomposition should result in components with high cohesion (the component's parts are closely inter-linked) and loose coupling (there should be minimum inter-dependencies between components). Thus, the decomposition of high quality represents a modular structure.

C. System-level behavioral properties

The system-level behavioral properties encompass understandability, predictability, adaptability, scalability, information transfer delays, and fault tolerance and survivability.

1) Understandability:

a) Can one designer comprehend all details of the interfaces?: How difficult is it for the designer to keep in mind the purposes and likely outcomes of invoking the functionality declared by interfaces?

b) Can one designer understand the causation of inputs?: An input results in a chain of processing, proceeding through a number of interfaces, and involving numerous conditional statements. Can the designer understand how a particular input will be processed, and/or identify what input(s) may have produced a particular result?

c) Is the analytical solution describing the relationships between system-level properties and components' properties available?: Let a system-level property Y be produced as a result of interaction of several components with properties $X_1, X_2, \ldots, X_N$. Can the value of Y be expressed or accurately approximated as a function of $X_1, X_2, \ldots, X_N$?

2) Predictability (as reflected in specific system-level parameters):

a) How well the (system level) response to events can be predicted?: An input results in a chain of processing, proceeding through a number of interfaces, and involving numerous conditional statements. Can the designer predict what effect a particular input will have on the system?

b) Does the system exhibit simple (fixed or periodic) behavior?: In a system with simple behavior, external events do not induce far-reaching effects on the system. Usually, only the state of the nodes receiving/processing the events are changing. Having processed the events, the system is in a (new) equilibrium state.

c) Is system behavior chaotic?: Chaotic behavior means that any external event induces endless chains of processing and interaction between nodes. An equilibrium state is achieved after extremely long time, or not achieved at all.

d) Does the system exhibit complex behavior at the "Edge of chaos"?: A system at the "Edge of chaos" responds to external influences with a series of adaptations, including processing within nodes and interaction between nodes. As a result, one of few equilibrium states, referred to as basins of attractions, is (re-)achieved. Contrary to the simple behavior, the complex behavior implies that a relatively small number of equilibrium states exist.

e) Is the network continuously changing (from the service point of view) in its structure? Is the network preserving its structure to some extent?: Ad-hoc networks are an example of the networks wherein the structure is not preserved but it changes continuously. On the other hand, the network of routes in Internet has a relatively stable structure; however, it changes as routers update their routing tables.

f) Is the network continuously changing (from the service point of view) in its behavior? Is the network preserving its behavior in some respect?: Many parameters of networks, such as the network load in telecommunications networks, are continuously changing. Still, some regularities in behavior can be identified – e.g., peaks in network load during rush hours.

g) Can global patterns of behavior be identified?: The above regularity in the load of the telecommunications network represents one example of the global patterns of behavior (emergent property). Network stagnation and network disintegration into a set of isolated islands are other examples of global behavioral patterns. The existence of such patterns cannot be understood by analyzing system components only, without paying attention to interactions between them.

h) Are there "islands of stability" in the system?: Can regions with relatively stable values of specific properties be identified in the system? For example, might subsets of routers with relatively stable routing tables be found in the network of Internet routers?

3) Adaptability:

a) Has the system the property of learning or adapting?: Can the system adapt, in order to better fit to a changing environment, as a result of i) modifying internal models of system components/agents and ii) interaction between these components. E.g. can the rate of interrupted calls be preserved low, even when the behavior of customers changes dramatically, by making the CAC rules modifiable according to the load of serving a base station and the loads of the neighboring base stations?

b) Who fulfils the requirements of adaptation?: The adaptation can be done automatically, or it may require a human operator to make the necessary changes. In the context of CAC, automatic adaptation would mean that new rules/policies for CAC are generated, tested, and applied automatically. Human-operator-facilitated adaptation, on the other hand, implies that the creation of new rules/policies, and enforcing them, is done by a human.

4) Scalability:

a) Does the system extension process involve a major redesign? Or is the possibility for extension built-in into the design?: Is it necessary to make changes into an existing system before it can be extended?

b) Can the extension be done automatically based on the "changing environment"? Or does it require human intervention for planning and implementing?: E.g. can auxiliary server be added automatically (transparently and instantly) to the server pool on demand, in order to balance out the load during peaks of service requests? Similarly, may an additional base-station be added in the area with increased load?

c) Is the modularity (hierarchical clustering) of the system preserved in the process of growth?: In software systems, for example, the high cohesion and loose coupling of system components (which reflect system modularity) should be preserved during the extension of the system.

5) Information transfer delays:

a) Is there a long delay in the information transfer due to a high degree of separation (hops) between nodes?: Delays depend on the network structure, amongst other things. Among networks of the same size, small-world networks have a smaller degree of separation, i.e. smaller average path length between nodes, than the networks with regular or random structure. As a result, shorter delays are usually observed in small-world networks than in regular or random networks.

6) Fault tolerance and survivability:

a) Is the network tolerant to the destruction (unavailability) of randomly selected nodes?: Does the network remain connected after a significant part of its nodes (randomly chosen) are excluded, or does it disintegrate into a set of unconnected "islands"? Similarly, does the network's degree of separation regress rapidly, as the number of (randomly chosen) nodes excluded from the network grows?

b) Is the network tolerant to the destruction (unavailability) of highly-connected nodes (hubs)?: In case a network has nodes with higher number of connections (hubs), does this network remain connected after a part of hubs are excluded, or does it disintegrate into a set of unconnected "islands"? Similarly, does the network's degree of separation regress rapidly, as the number of hubs excluded from the network grows?

D. Further refinement of the list

The above list of questions (factors) needs to be refined by excluding less relevant factors and, possibly, by augmenting the list with additional ones. In order to assess the relevance of a factor, the experience of the experts involved in the process of architecture evaluation can be used. During the evaluation, irrelevant factors will be filtered out automatically, since small values of relative importance are likely to be assigned to them. Eventually, those factors, to which small importance is repeatedly assigned by the experts, can be excluded from the list, or substituted with other ones.

V. Conclusion

An architecture of a complex system is seen appropriate when it meets a set of requirements such as fitting the underlying business-drivers and coherence with other architectures

in a product family. In order to determine the appropriateness of such architecture, an evaluation of it may be performed. As a result of the evaluation, the tradeoffs involved in the architecture are revealed, and the match between the architecture and its business drivers is assessed.

For the evaluation process to succeed, it is crucial to attain a deep understanding of the processes within the system among the stakeholders involved in the evaluation. However, a large number of components and interactions between them, which are usually present in telecommunication system architectures, results in complex systems that are highly difficult to conceive, making the evaluation of these architectures extremely difficult.

A formal evaluation method, such as AEF, can be employed in order to support the evaluation process to some extent. The AEF supplies the evaluation team with a number of questions concerning properties of the architecture and their relative importance. By answering these questions, the appropriateness score of the architecture can be evaluated. However, in the current version of the AEF, complexity-related issues are not directly taken into account.

In this study, an attempt has been made to augment the list of questions available in the AEF with additional questions addressing various aspects of system complexity. The paper suggests a list of complexity-related questions that can be potentially useful for architecture evaluation. The selection of the questions was based on the analysis of available studies in the domain of complex systems. These studies were juxtaposed with the system domain the AEF is created for. However, assessing whether these questions are indeed relevant from the architectural point of view is assigned to the experts. While using the AEF extended with complexity-related questions, they can judge the relevance and the relative importance of these questions, which thus remains a subject for future research.

ACKNOWLEDGMENT

This research work presented in this paper was done in MODPA research project (`http://www.titu.jyu.fi/modpa`) at the Information Technology Research Institute, University of Jyväskylä. MODPA project was financially supported by the National Technology Agency of Finland (TEKES) and industrial partners Nokia, SysOpen Digia, SESCA Technologies, Tieturi, Metso Paper, and Trusteq.

REFERENCES

[1] L. A. Difford, E. M. Dwyer, E. Gray, J. J. Logan, R. P. Mullinax, H. E. Thiess, J. D. Weigel, and S. A. Zaveler, "American national standard dictionary of information technology (ANSDIT)," National Committee for Information Technology Standards (NCITS), American National Standards Institute (ANSI)," Working draft document, 1999, available at http://www.incits.org/tc_home/k5htm/WD.htm, read 12.06.2005.

[2] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[3] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*. Addison Wesley Longman, 2001.

[4] J. A. Lehto and P. Marttiin, "Experiences in system architecture evaluation: A communication view for architectural design," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences 2005 (HICSS '05)*, 2005.

[5] J. S. Lansing, "Complex adaptive systems," *Annual Review of Anthropology*, vol. 32, pp. 183–204, 2003.

[6] E. Forman and M. A. Selly, *Decision By Objectives – How to Convince Others that You are Right*. World Scientific Press, 2001.

[7] A. C. Gatrell, "Complexity theory and geographies of health: a critical assessment," *Social Science & Medicine*, vol. 60, pp. 2661–2671, 2005.

[8] S. H. Strogatz, "Exploring complex networks," *Nature*, vol. 410, pp. 268–276, 2001.

[9] N. G. Leveson, "Safety as a system property," *Communications of the ACM*, vol. 38, no. 11, p. 146, November 1995.

[10] N. Leveson and C. Turner, "An investigation of the therac-25 accidents," *IEEE Computer*, vol. 26, no. 7, pp. 18–41, July 1993.

[11] N. G. Leveson, "Role of software in spacecraft accidents," *Journal of Spacecraft and Rockets*, vol. 41, no. 4, pp. 564–575, July-August 2004.

[12] J. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Univ. of Michigan Press, 1975.

[13] M. Lyons, I. Adjali, D. Collings, and K. Jensen, "Complex systsems models for strategic decision making," *BT Technology Journal*, vol. 21, no. 2, pp. 11–27, 2003.

[14] J. Holland, *Hidden order: how adaptation builds complexity*. Addison-Wesley Publishing Company, Inc., 1995.

[15] C. G. Langton, "Computation at the edge of chaos," *Physica D*, vol. 42, pp. 12–37, 1990.

[16] S. Milgram, "The small-world problem," *Psychology Today*, vol. 1, pp. 60–67, 1967.

[17] M. Granovetter, "The strength of weak ties: A network theory revisited," *Sociological theory*, vol. 1, pp. 203–233, 1983.

[18] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, pp. 440–442, 1998.

[19] M. E. J. Newman, "The structure and function of complex networks," *SIAM REVIEW*, vol. 45, no. 2, pp. 167–256, May 2003.

[20] M. Buchanan, *Nexus: Small words and groundbreaking science of networks*. W. W. Norton & Company, Inc., 2002.

[21] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, 1999, pp. 251–262.

[22] R. Albert and A.-L. Barabasi, "Statistical mechanics of complex networks," *Review of modern physics*, vol. 74, pp. 47–96, 2002.

[23] A.-L. Barabasi, *Linked*. Plume, Penguin Group (USA) Inc., 2003.

[24] I. Sommerville, *Software engineering*, 5th ed. Harlow: Addison-Wesley, 1998.