Metrics for the Evaluation of Adaptivity Aspects in Software Systems

Claudia Raibulet, Laura Masciadri
Dipartimento di Informatica Sistemistica e Comunicazione,
Universitá degli Studi di Milano-Bicocca
Milan, Italy
raibulet@disco.unimib.it, laura.masc@gmail.com

***Abstract*—Runtime adaptivity is related to the ability of the information systems to perform changes by themselves and on themselves during their execution. The engineering of runtime adaptivity is one of the most challenging issues to address in today's information systems. This is due to the fact that runtime adaptivity requires additional elements at the architectural or structural levels. Moreover, it increases the dimension and computation of a system. Its advantages are mostly related to the improvement of performances, enhancement of the functionalities' quality and automation of administrative tasks. In this paper we propose a set of metrics for the description and evaluation of adaptive properties of the information systems and of the frameworks which provide support for the development of adaptive systems. They aim to provide a concrete mechanism to analyze the quality of the design of adaptive systems, to determine the type of adaptivity of a system or to compare the adaptive features of different systems. Metrics are grouped into six categories: architectural, structural, performance, interaction, documentation, and miscellaneous. They have been identified and specified by analyzing several case studies which address runtime adaptivity issues through different approaches with different objectives in various application domains and several frameworks for the design and implementation of adaptive systems.**

Keywords-adaptivity, adaptive systems, evaluation, software metrics.

I. INTRODUCTION

Runtime adaptivity [3], [5], [7], [10], [15] indicates the ability of a system or software to perform changes by itself and on itself during its execution. The objectives of changes may be of various natures and may concern different issues which range from addressing unpredicted situations to ensuring the optimal working of a system's resources or to improving the performances of a system. Essentially, they result from the need to address the growing complexity of emerging systems and to improve productivity and performance, as well as to automate configuration, re-configuration, control and management tasks [15].

Due to the wide range of possible objectives, types and solutions related to runtime adaptivity, it would be very useful to have common mechanisms to evaluate and compare adaptive approaches in order to choose the most appropriate solution for the current needs, to integrate various solutions, or to make these solutions cooperate to achieve complex tasks.

We tried to address these issues by considering the available solutions described in the scientific literature [2], [5], [7], [8], [16] in order to determine how adaptivity is actually achieved, the main characteristics of the design of adaptive systems, as well as the advantages outlined by the authors of the adaptive systems.

The conclusions are summarized as follows. Adaptivity is a complex task. Independently of what it is changed at runtime, an adaptivity pattern consisting of four main steps (which should be implemented by any adaptive system) can be specified: monitoring (to retrieve information about the context or status of a system which is exploited at run-time in the adaptation process), analysis of the monitored information, decision (to determine whether changes should be made or not and, in the affirmative case, to choose the best solution for the current situation) and application of identified changes [3], [5], [10], [14].

Adaptivity requires additional elements at the architectural or structural levels in order to implement these steps. Even if it is considered a non-functional requirement, it influences the execution of a system, its interaction with the external world, and its performances. Therefore, its design and implementation are fundamental for a system's lifecycle.

Authors describe the advantages of adaptive systems in terms of performances, simplified and enhanced interaction with the users, and automation of administrative tasks. However, the evaluation of the described solutions is *adaptive* and *case study oriented*: the authors provide their point of view and outline the strong aspects of their solutions through a particular vocabulary/terminology. Therefore, it is difficult, if not impossible, to evaluate and compare adaptive systems.

Furthermore, the scientific literature presents also various frameworks [5], [7], [16], each introducing a different approach for the design and implementation of adaptive mechanisms. For example, the Rainbow framework [5] proposes a control loop which defines mechanisms to monitor the runtime properties of a system, to evaluate constraint violations, and to perform global and module level adaptations on a running system. All these mechanisms are provided at the architectural level. On the other hand, the Adaptive Server Framework (ASF) [7] describes an infrastructure of components and services which facilitates the construction of adaptation from a behavioral perspective. Hence, when developing an adaptive system, on which basis

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

there can be evaluated which of these frameworks is more appropriate for the current requirements?

In this context, we propose a set of metrics which may be adopted in the description, design, and evaluation of the adaptive properties of information systems and frameworks. The metrics are grouped into the following categories: architectural, structural, interaction, performance, documentation, and miscellaneous [12]. The architectural and structural metrics are mostly related to the design issues of adaptive systems; while the interaction and performance metrics reflect the advantages regarding the usability of the adaptive systems. Even if it may play a secondary role, the documentation category may be considered an indication on the usability and reusability, personalization, and the advantages of adaptive systems, as well as about their overall quality. Furthermore, it is a valuable indication on the usability and the overall quality of a framework for adaptive systems.

For the presentation of the metrics of adaptivity, this paper considers four of the available case studies, which in our opinion are representative for this topic: a Web-based client-server system and a video conferencing system which exploit the Rainbow framework [5], an adaptive image server which uses the Adaptive Server Framework [7], and the AHA! [2] system for adaptive e-learning.

The rest of the paper is organized as follows. Section II provides an overview on four representative adaptive case studies. Section III introduces the categories and subcategories of metrics defined in this paper. The application of the defined metrics to the four cases studies introduced in Section II is discussed within Section IV. The similar approaches for the evaluation of runtime adaptivity are addressed in Section V. Conclusions and future work are dealt with in Section VI.

II. CASE STUDIES ON RUNTIME ADAPTIVITY

This section introduces four of the case studies we have considered in the identification and specification of metrics. In Section X, an analysis of these case studies from the metrics point of view is presented.

A. Web-based Client-Server (WebCS)

In this case study Web-clients make requests of contents to various Web-server groups [5]. Adaptivity is related to the system's performances and more precisely to the response time the clients perceive for their requests to the Web servers. The two factors which influence the response time are the servers' load and the available bandwidth. To address this performance issue through adaptivity, an architectural level approach is adopted consisting in the definition of an architectural style enriched with adaptation operators and strategies for the dynamic aspects of a system [5]. Furthermore, each client has associated an invariant which verifies if the response time is less than a predefined value. If this is not true an adaptivity strategy is invoked. The results of the application of the adaptive strategies are reflected at the architectural level.

B. VideoConferencing (VConf)

This case study deals with the management of videoconferences in which participants may use various videoconferencing tools and communication protocols [5]. Adaptivity is related to the performance (determined essentially by the available bandwidth) and cost (determined essentially by the gateway costs) aspects. As in the previous example, adaptivity is addressed at the architectural level through an architectural style. Each handheld device and gateway has associated an invariant which establishes the range of the accepted values. Whenever an invariant is violated, an adaptive strategy is invoked. The results of the application of the adaptive strategies are reflected at the architectural level.

C. Adaptive Image Server (AIS)

In this case study, clients send to a server requests for images specifying the minimum and maximum resolution for the requested images [7]. In a non-adaptive scenario, the server provides the images with their current resolutions. In an adaptive scenario, the server scales the images resolution in order to optimize the overall performances of the system. In this case, performances are translated into the improvement of the throughput and the reducing of the response time which are determined by the resolution and the quality of an image. Furthermore, adaptivity takes into consideration also its overhead introduced in the system: the computation load (of the CPU) of the server because the time needed to process images may influence significantly the response time. This is compared to the latency determined by sending non-modified images.

D. Adaptive Hypermedia Architecture (AHA!)

AHA! [2] is an adaptive e-learning system. Adaptivity regards the content (visualized to a user as pages) and the navigation in the content (implemented through links) based on the knowledge level of a user. The information offered by AHA! is organized hierarchically (consisting in fragments-pages-courses) through a domain model. Each element in this domain model may have associated one or more concepts.

When a user requires a page, based on (1) the user's model (consisting in concepts which have associated a set of attributes among which his knowledge level) and (2) the adaptivity rules defined by the adaptation model, an adaptivity engine (I) builds the requested page (inserting the content and the navigation elements) accordingly to the user's current knowledge level, (II) updates the user model (through the rules defined by the adaptation model by considering the concepts inserted in the requested page) and (III) visualizes the page.

III. METRICS FOR RUNTIME ADAPTIVITY EVALUATION

In the definition of the metrics we have assumed that the functional part of a system is designed first, (or more generally a system should provide a version of its functionalities which does not exploit adaptivity), and further it is enriched with adaptive mechanisms.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Each metric is presented through its name, description and further explanations wherever necessary.

We have defined five categories of metrics each of them may be further divided into subcategories. Furthermore, we have defined additional aspects which may be considered to evaluate adaptivity and which are grouped in a miscellaneous category due to the fact that they capture different facets of adaptivity. An overview on these metrics is presented in Figure 1.

The architectural category of metrics aims to capture the main features of adaptivity which emerge at the system level. These characteristics are visible and meaningful when considering a global perspective on the architecture of a system. The metrics in this category are further divided in two subcategories: architectural growth and architectural separation of concerns.

The structural metrics are collocated at a lower abstraction level then the metrics in the previous category. They concern the actual implementation aspects of an adaptive system. The metrics in this category are further divided in three subcategories: structural growth, structural separation of concerns, and personalization.

The interaction category focuses on the advantages provided by runtime adaptivity in terms of the automation of the human tasks. Hence, it considers both the interactions of the administrators and the final users with an adaptive system.

The performance metrics aim to evaluate the quality of the functionalities provided by the adaptive systems. Hence, the aspects they consider are visible and meaningful for the final users of a system.

The documentation category provides information on the quality of a design of an adaptive system or a framework for the implementation of adaptive systems. It defines meaningful metrics for the usability and understandability of the adaptive properties.

The miscellaneous indexes propose further evaluation mechanisms which may be exploited to analyze the overall effort necessary to implement adaptive functionalities.

These categories of metrics are described in detail in the following sections.

A. Architectural Metrics

During the presentation of the architectural metrics we use the term *elements* as defined by [1] for software architecture: "The software architecture of a program or computing system is a structure or structures of a system, which comprise software *elements*, the external visible properties of those elements and the relationships among them". Hence, the term element expresses an architectural unit (e.g., components and connectors [6]). The metrics are

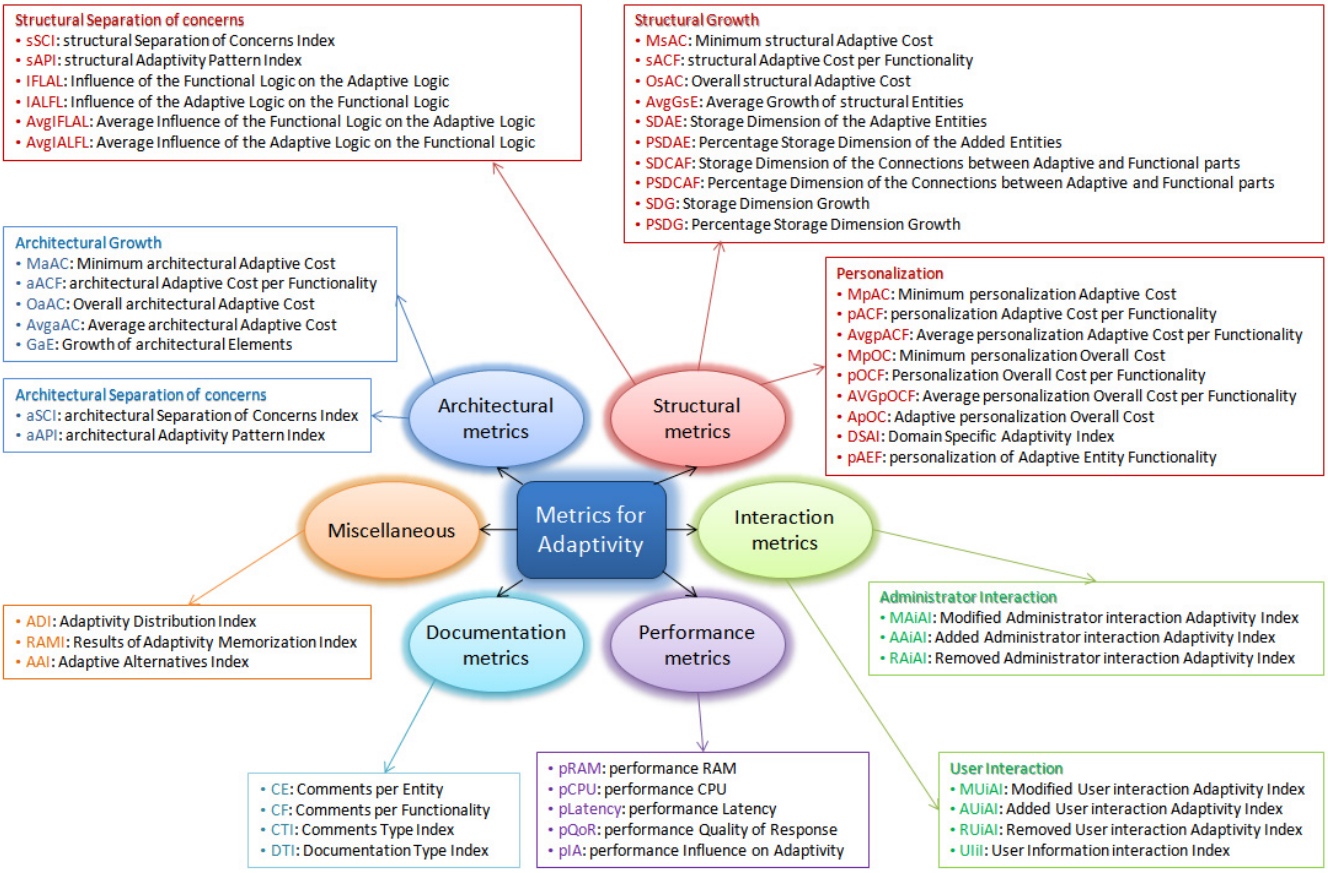

Figure 1. Overview on the metrics for runtime adaptivity

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

valid and applicable to any architectural element.

Generally, the systems addressing runtime adaptivity through architectural mechanisms are composed of two parts: functional and adaptive. The adaptive part is usually composed of four main conceptual elements corresponding to the adaptation steps: monitoring, analyzing, deciding and changing (see Figure 3-A).

The architectural metrics concern two main aspects: separation of concerns and architectural growth.

The separation of concerns regards two aspects: (1) the separation between the functional logic and the elements ensuring adaptivity, and (2) the separation among the elements implementing the four steps of adaptivity. It is expressed through two metrics.

aSCI: architectural Separation of Concerns Index

$$aSCI = \frac{Number\ of\ adaptive\ elements}{Number\ of\ functional\ elements\ modified}$$

This metric indicates the degree of dependence between the functional logic and the adaptive elements of a system at the architectural level. It enables the evaluation of the separation of concerns at the architectural level by comparing the number of elements inserted in the adaptive part (which provide exclusively adaptive functionalities) and the number of the functional elements which have been modified (to interact with the adaptive ones).

aAPI: architectural Adaptivity Pattern Index

$$aAPI = Number\ of\ adaptive\ conceptual\ elements$$

This metric indicates the separation of concerns between the main conceptual (types of) elements implementing the four steps defined by the adaptivity pattern at the architectural level. If the value of this metric is four, then the adaptive part of a system defines at least a conceptual element for each of these four steps. If it is zero, the adaptive part of the architecture is totally integrated with the functional one. The values between zero and four suggest that two or more adaptivity steps are provided by the same conceptual architectural element.

These two metrics provide useful information related to the modularity, reusability and maintainability of the adaptive part of a system.

The architectural growth regards the number of elements introduced by the adaptive part of an architecture. It is express through five metrics.

MaAC: Minimum architectural Adaptive Cost

$$MaAC = Minimum\ number\ of\ elements\ for\ adaptivity$$

This metric indicates the minimum number of elements which should be added to make a system adaptive independently of the number of functionalities it provides. Essentially, this metric expresses the fix cost of adaptivity at the architectural level. It considers the adaptive elements necessary to make the first functionality adaptive.

aACF: architectural Adaptive Cost per Functionality

$$aACF = Number\ of\ elements\ for\ the\ i^{th}\ functionality$$

This metric indicates the number of elements which should be added to make the i-th functionality adaptive. It may be seen as a variable cost for introducing adaptivity per functionality at the architectural level.

OaAC: Overall architectural Adaptive Cost

$$OaAC = MaAC + \sum_{i=2}^{n} aACF$$

The sum between the last two metrics expresses the architectural growth in number of elements needed to add adaptivity (see Figure 2). The results obtained through this metric are dependent on the order in which the adaptive functionalities of a system are actually implemented. If the function has a linear evolution, then the adaptive functionalities may be considered independent of each other, hence they need independent components (see Figure 3-A). Otherwise, if the function has a logarithmic evolution, then the adaptive functionalities may share common components achieving implicitly reusability issues (see Figure 3-B).

AvgaAC: Average architectural Adaptive Cost

$$AvgaAC = \frac{OaAC}{n}$$

This metric expresses the average growth per functionality at the architectural level due to the introduction of adaptivity. It indicates the average number of elements which have been added for each functionality. Hence, for each functionality it is added 1/n (where n is the total number of functionalities) of the fix costs related to the introduction of adaptive mechanisms in a system.

As for the OaAC metric, if this function has a linear behavior then we can assume that the adaptive functionalities are independent of each other (see Figure 3-A).

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

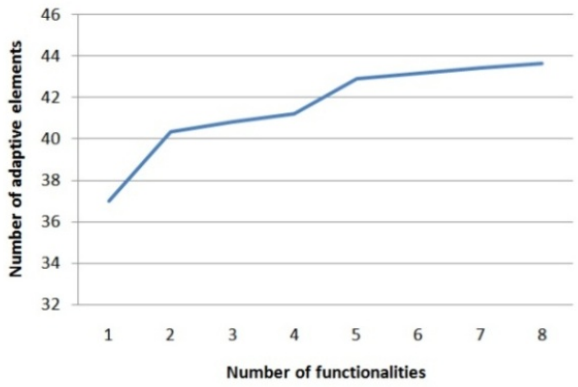

Figure 2. Overall architectural Adaptive Cost

Otherwise, if the function has a logarithmic behavior, there is a reuse of several of the already inserted elements in the adaptive part (see Figure 3-B). In this case the behavior of the function may be influenced by the order in which functionalities are made adaptive (because some of them share common elements). Figure 3 shows the generic components implementing the four steps of the adaptivity pattern: 1a for the monitoring, 2a for the analyzing, 3a for the deciding, and 4a for the changing.

GaE: Growth of architectural Elements

$$GaE = \frac{OaAC}{Number\ of\ functional\ elements} * 100$$

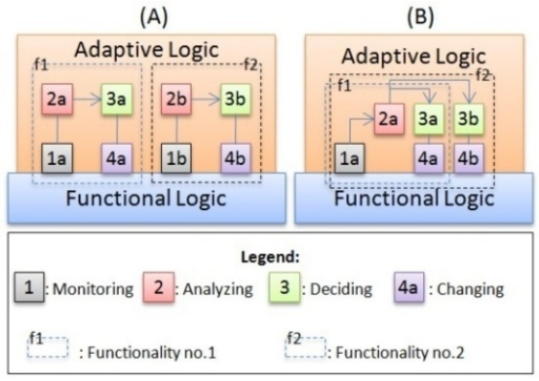

Figure 3. Adaptive functionalities and elements

This metric expresses the percentage growth at the architectural level due to the introduction of adaptivity.

B. Structural Metrics

During the presentation of the structural metrics we use the term *entity* to denote the software units. For example, in an object-oriented system an entity is a class.

The systems addressing runtime adaptivity at the structural level are composed of two types of entities: functional and adaptive. The structural metrics concern three main aspects: separation of concerns, structural growth and personalization.

As for the architectural metrics, the separation of concerns regards two aspects: (1) the separation between the functional entities and the entities ensuring adaptivity, and (2) the separation among the conceptual (types of) entities implementing the four steps of adaptivity. It is expressed through six metrics.

sSCI: structural Separation of Concerns Index

$$sSCI = \frac{Number\ of\ adaptive\ entities}{Number\ of\ functional\ entities\ modified}$$

This metric indicates the degree of dependence between the functional and adaptive entities at the implementation level. It enables the evaluation of the separation of concerns by comparing the number of entities inserted in the adaptive part (which provide exclusively adaptive functionalities) and the number of the functional entities which have been modified (to interact with the adaptive ones). Theoretically, the obtained value should be similar to the one resulted for the aSCI metric. Actually, the two values may be significantly different being determined by the adopted implementation strategy: an approach based on few entities (each implementing more functionalities) or a highly modular one (each entity implementing few functionalities). Hence, the two values may differ from each other for different design methodologies at the architectural and the implementation levels.

sAPI: structural Adaptivity Pattern Index

$$sAPI = Number\ of\ adaptive\ conceptual\ entities$$

This metric indicates the separation of concerns between the main conceptual (types of) entities implementing the four steps of adaptivity at the implementation level. If the value of this metric is four, then the adaptive part of a system defines at least a conceptual entity for each of these four steps. If it is zero, the adaptive entities are totally integrated with the functional one. The values between zero and four suggest that two or more adaptivity steps are provided by the same conceptual entity.

IFLAL: Influence of the Functional Logic on the Adaptive Logic

$$IFLAL = Number\ of\ inputs\ in\ the\ adaptive\ logic$$

IALFL: Influence of the Adaptive Logic on the Functional Logic

$$IALFL = Number\ \ of\ outputs\ from\ the\ adaptive\ logic$$

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

These two metrics provide information about the role of the adaptive part of a system in its overall functionality. Higher is the value for the IFLAL, stronger is the influence of the application domain or contextual aspects on the adaptive part of a system (see Figure 4-B). Vice-versa, the IALFL metric indicates the degree of influence of the adaptive logic on the functionalities provided by a system (see Figure 4-A). Comparing the values denoted by these two metrics it is possible to determine the degree of influence of one part on the other.

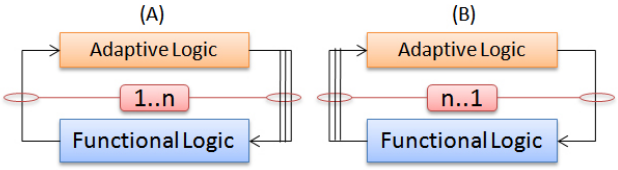

Figure 4. Influence of the adaptive part on the functional one (A) and vice-versa (B)

AvgIFLAL: Average Influence of the Functional Logic on the Adaptive Logic

$$AvgIFLAL = \frac{IFLAL}{Number\ of\ functionalities}$$

AvgIALFL: Average Influence of the Adaptive Logic on the Functional Logic

$$AvgIALFL = \frac{IALFL}{Number\ of\ functionalities}$$

The last two metrics provide information about the average number of inputs (respectively outputs) in the adaptive part for each of the functionalities of a system. The information they provide can be seen as a complexity degree of the provided functionalities based on the influence of the adaptive part of a system.

When AvgIFLAL is significantly greater than AvgIALFL the adaptive logic is strongly related to the application domain and the strategies of the adaptive part consider more factors in their logic than those they can influence in the functional part of a system. Vice-versa, when AvgIALFL is significantly greater than AvgIFLAL we expect that the functional part of a system has different behaviors in the presence of the adaptive part than in its absence due to the strong influence it has from the adaptive entities.

The structural growth regards the number of entities introduced by the adaptive part of a system. It is express through ten metrics.

MsAC: Minimum structural Adaptive Cost

$$MsAC = Minimum\ number\ of\ entities\ for\ adaptivity$$

This metric indicates the minimum number of entities to be added to a system to become adaptive independently of the number of the functionalities it provides. Essentially, this metric expresses the fix cost of adaptivity at the implementation level.

sACF: structural Adaptive Cost per Functionality

$$sACF = Number\ of\ entities\ for\ the\ i^{th}\ functionality$$

This metric indicates the number of entities which should be added to make the i-th functionality adaptive. It may be seen as a variable cost for introducing adaptivity per functionality at the implementation level. The results of this metric may influence the interpretation of the AvgIFLAL and AvgIALFL: minor is the reusability of the entities implementing the adaptive steps, more precisely are the considerations derived from these metrics.

OsAC: Overall structural Adaptive Cost

$$OsAC = MsAC + \sum_{i=2}^{n} sACF$$

The sum between the last two metrics expresses the structural growth in number of entities needed to add adaptivity at the implementation level. The observations made for the OaAC are valid also for OsAC.

AvgGsE: Average Growth of structural Entities

$$AvgGsE = \frac{OsAC}{n}$$

This metric expresses the average growth per functionality at the implementation level due to the introduction of adaptivity. It indicates the average number of entities which have been added for each functionality. Hence, for each functionality it is added 1/n (where n is the total number of functionalities) of the fix costs related to the introduction of adaptive mechanisms in a system.

SDG: Storage Dimension Growth

$$SDG = KB_{withAdaptivity} - KB_{withoutAdaptivity}$$

PSDG: Percentage Storage Dimension Growth

$$PSDG = \frac{SDG}{KB_{withoutAdaptivity}} * 100$$

These two metrics indicate the physical storage growth in kilo bytes, and respectively in percentage, due to the presence of the adaptive mechanisms in a system. Adaptive mechanisms include both adaptive entities and their link with the functional entities.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

SDAE: Storage Dimension of the Adaptive Entities

$$SDAE = KB_{AllAdaptiveEntities}$$

PSDAE: Percentage Storage Dimension of the Adaptive Entities

$$PSDAE = \frac{SDAE}{SDG} * 100$$

These two metrics indicate the physical storage growth in kilo bytes, and respectively in percentage, needed to store the adaptive entities.

SDCAF: Storage Dimension of the Connections between Adaptive and Functional parts

$$SDCAF = KB_{withAdaptivity} - (KB_{withoutAdaptivity} + SDAE)$$

PSDCAF: Percentage Dimension of the Connections between Adaptive and Functional parts

$$PSDCAF = \frac{SDCAF}{SDG} * 100$$

These two metrics indicate the physical storage growth in kilo bytes and percentage, due to the entities which have been defined for the link and communication between the functional and adaptive part of a system.

The personalization category of metrics regards the changes which are made on a framework for runtime adaptivity or an adaptive system in order to apply it to an actual case study. These metrics are defined from the developers' point of view and try to capture the effort needed to adapt the framework or a system to other solutions.

It consists of nine metrics.

MpAC: Minimum personalization Adaptive Cost

$$MpAC = \frac{\begin{array}{c}Number\ of\ personalized\ entities\ in\ the\\ adaptive\ part\ for\ the\ 1^{st}\ functionality\end{array}}{\begin{array}{c}Number\ of\ entities\ in\ the\ adaptive\\ part\ for\ the\ 1^{st}\ functionality\end{array}} * 100$$

This metric indicates the percentage of entities which are personalized considering only the minimum number of entities necessary to make a system adaptive. Hence, it is calculated through the number of personalized entities for making one (e.g., the first) functionality adaptive.

If the result is 100%, then it may be assumed that the adaptive entities are totally generic or that the factors which influence the adaptive logic are totally domain dependent due to the fact that all the necessary adaptive entities have been personalized.

If the result is sensibly less than 100%, then it can be considered that the adaptive entities are dependent on general factors which may be considered generic enough to be reused without modifications in many application domains or case studies.

pACF: personalization Adaptive Cost per Functionality

$$pACF = \frac{\begin{array}{c}Number\ of\ personalized\ entities\ in\\ the\ adaptive\ part\ for\ each\ functionality\end{array}}{\begin{array}{c}Number\ of\ entities\ added\ in\ the\\ adaptive\ part\ for\ each\ functionality\end{array}} * 100$$

This metric indicates the percentage of entities which are personalized for each adaptive functionality considering the number of adaptive entities added for this particular functionality in the adaptive part. This metric may be influenced by the order in which adaptive functionalities are added (two or more functionalities may exploit common adaptive entities, and hence these entities are added only for the first inserted functionality).

If the result is greater than 100% it means that the new functionality required the modification of already available adaptive entities which have been added previously for other functionalities.

If the result is 100% or less, then it is considered that a number of adaptive entities equal or less then the number of the added functionalities have been modified (without being able to specify if only new added entities have been personalized).

AvgpACF: Average personalization Adaptive Cost per Functionality

$$AvgpACF = \frac{MpAC + \sum_{i=2}^{n} pACF}{n}$$

This metric indicates the average cost for introduction of adaptive mechanisms per functionality.

MpOC: Minimum personalization Overall Cost

$$MpOC = \frac{\begin{array}{c}Number\ of\ personalized\ entities\ in\ the\\ entire\ system\ for\ the\ 1^{st}\ functionality\end{array}}{\begin{array}{c}Number\ of\ entities\ in\ the\\ adaptive\ part\ for\ the\ 1^{st}\ functionality\end{array}} * 100$$

This metric indicates the percentage of entities which are personalized in the entire system (functional and adaptive) with respect to the minimum number of entities in the adaptive part necessary to make a system adaptive. As in the case of MpAC, it is calculated for the first functionality chosen to be made adaptive.

If the result is equal to the one obtained for the MpAC, then no entity in the functional part has been modified. If these two results differ, then functional entities have been

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

modified too. Greater is the difference between the values of the two metrics, more significant are the modifications in the functional part of the system.

This metric may be useful during the analysis of the separation between the functional and adaptive parts, as well as of the integration of the adaptive part in a system.

pOCF: personalization Overall Cost per Functionality

$$pOCF = \frac{\begin{matrix}Number\ of\ personalized\ entities \\ in\ the\ entire\ system\end{matrix}}{\begin{matrix}Number\ of\ entities\ in\ the\ adaptive\ part \\ for\ each\ functionality\end{matrix}} * 100$$

This metric indicates the percentage of entities which are personalized in the entire system (functional and adaptive) with respect to the total number of adaptive entities necessary to provide a functionality.

If the result is equal to the one obtained for the pACF, then no entity in the functional part has been modified (besides the modifications made for one functionality). If these two results differ, then functional entities have been modified too. Greater is the difference between the values of the two metrics, more significant are the modifications in the functional part of the system.

Comparing this metric to the previous one MpOC, usually it can be observed that the modifications performed for the first functionality made adaptive may be greater than those performed for each of the other functionalities (because there may be entities which are used further by all functionalities).

AvgpOCF: Average personalization Overall Cost per Functionality

$$AvgpOCF = \frac{MpOC + \sum_{i=2}^{n} pOCF}{n}$$

This metric indicates the average overall cost for introduction of adaptive mechanisms per functionality. If the result is equal to the one obtained for the AvgpACF, then the functional part is not modified. If these two results differ, then functional entities have been modified too. Greater is the difference between the values of the two metrics, more significant are the modifications in the functional part of the system.

ApOC: Adaptive personalization Overall Cost

$$ApOC = \frac{\begin{matrix}Number\ of\ personalized\ entities\ in \\ the\ adaptive\ part\end{matrix}}{\begin{matrix}Total\ number\ of\ personalized \\ entities\ in\ the\ entire\ system\end{matrix}} * 100$$

This metric indicates the percentage of the personalization of the adaptive part with respect to the personalization of the entire system (functional and adaptive) to make it adaptive.

DSAI: Domain Specific Adaptivity Index

$$DSAI = \frac{\begin{matrix}Number\ of\ domain\ influence \\ factors\end{matrix}}{Number\ of\ total\ influence\ factors} * 100$$

This metric indicates the percentage of the factors specific to the application domain which influence the adaptive part of a system. Higher is this value, higher is the number of personalized entities in the entire system.

pAEF: personalization of Adaptive Entity Functionality

$$pAEF = \frac{\begin{matrix}Number\ of\ personalized\ functionality \\ of\ an\ adaptive\ entity * 100\end{matrix}}{\begin{matrix}Total\ functionalities\ of\ an \\ adaptive\ entity\end{matrix}}$$

This metric indicates the percentage of functionalities personalized for an adaptive entity. For example, in an object-oriented system this regards the methods signature. If this value is low, then modifications are made mostly inside the functionalities (in the definition of methods and not in their declarations).

C. Interaction Metrics

The purpose of the interaction metrics is to evaluate the variations in the interaction between administrators or users and the adaptive and non-adaptive versions of a system.

MAiAI: Modified Administrator interaction Adaptivity Index

$$MAiAI = \sum \begin{cases} 1\ if\ an\ task\ is\ modified \\ 0\ otherwise \end{cases}$$

This metric indicates the modified tasks which should be performed by the administrator. These tasks are necessary both in the non-adaptive and adaptive versions of the system. The introduction of the adaptive part may have made them more or less complex.

AAiAI: Added Administrator interaction Adaptivity Index

$$AAiAI = \sum \begin{cases} 1\ if\ an\ task\ is\ added \\ 0\ otherwise \end{cases}$$

This metric indicates the new added tasks which should be performed by the administrator after the introduction of the adaptive part. These tasks were not necessary in the non-adaptive version of the system.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

RAiAI: Removed Administrator interaction Adaptivity Index

$$RAiAI = \sum \begin{cases} 1 \; if \; an \; task \; is \; deleted \\ 0 \; otherwise \end{cases}$$

This metric indicates the removed tasks which should not be further performed by the administrator after the introduction of the adaptive part. These actions were necessary in the non-adaptive version of the system.

Greater is the RAiAI and/or lower is the AAiAI, more efficient is the introduction of the adaptivity mechanisms from the administration of the system point of view. However, these three metrics do not provide information on the complexity of the administration tasks. This would be very useful to complement the MAiAI metric in order to check whether adaptivity has simplified or not the administration tasks (for those which have not been removed).

The user interaction metrics concern two aspects: the variations in the interaction between users and a system in the absence and presence of adaptivity, as well as the provisioning of the parameters needed for adaptivity.

MUiAI: Modified User interaction Adaptivity Index

$$MUiAI = \sum \begin{cases} 1 \; if \; an \; task \; is \; modified \\ 0 \; otherwise \end{cases}$$

This metric indicates the modified tasks which should be performed by the users. These tasks are necessary both in the non-adaptive and adaptive versions of the system. The introduction of the adaptive part may have made them more or less complex.

AUiAI: Added User interaction Adaptivity Index

$$AUiAI = \sum \begin{cases} 1 \; if \; an \; task \; is \; added \\ 0 \; otherwise \end{cases}$$

This metric indicates the new added tasks which should be performed by the users after the introduction of the adaptive part. These actions were not necessary in the non-adaptive version of the system.

RUiAI: Removed User interaction Adaptivity Index

$$RUiAI = \sum \begin{cases} 1 \; if \; an \; task \; is \; deleted \\ 0 \; otherwise \end{cases}$$

This metric indicates the removed tasks which should not be further performed by the users after the introduction of the adaptive part. These tasks were necessary in the non-adaptive version of the system.

Greater is the RUiAI and/or lower is the AUiAI, more efficient is the introduction of the adaptivity mechanisms from the users' point of view. However, these three metrics do not provide information on the complexity of the user interaction tasks. This would be very useful to complement the MUiAI metric in order to check whether adaptivity has simplified or not these tasks (for those which have not been removed).

UIiI: User Information interaction Index

$$UIiI = \begin{cases} 0 \; if \; all \; parameters \; are \; system \; side \\ 1 \; otherwise \end{cases}$$

This metric indicates if the monitored parameters are available on the system side or they should be gathered through the interaction with the users.

D. Performance Metrics

The performance metrics concern five main aspects related to the usage of the system resources (in terms of RAM and CPU), the response time, the improvement of the response quality in the presence of adaptivity and influence of the performance factors on the adaptive strategies. Usually, these metrics reflect the goals of the adaptive systems.

pRAM: performance RAM

$$pRAM = \frac{RAM \; usage \; in \; presence \; of \; adaptivity}{RAM \; usage \; in \; absence \; of \; adaptivity} * 100$$

This metric indicates the variation of the RAM usage due to the computational overhead introduced by the adaptive part of a system.

pCPU: performance CPU

$$pCPU = \frac{CPU \; usage \; in \; presence \; of \; adaptivity}{CPU \; usage \; in \; absence \; of \; adaptivity} * 100$$

This metric indicates the variation of the CPU usage due to the computational overhead introduced by the adaptive part of a system.

pLatency: performance Latency

$$pLatency = \frac{Response \; time \; in \; presence \; of \; adaptivity}{Response \; time \; in \; absence \; of \; adaptivity} * 100$$

This metric indicates the variation of the system's responses in the presence of adaptivity with respect to the response in the absence of adaptivity.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

pQoR: performance Quality of Response

$$pQoR = \frac{Quality\ of\ response\ in\ presence\ of\ adaptivity}{Quality\ of\ response\ in\ the\ absence\ of\ adaptivity} * 100$$

This metric indicates the variation of the quality of the system's responses in the presence of adaptivity. Generally, the obtained value for this metric should be greater than 100% in order to overcome the increments introduced by one or more of the previous three metrics.

pIA: performance Influence on Adaptivity

$$pIA = \begin{cases} 0\ if\ there\ is\ no\ influence \\ 1\ otherwise \end{cases}$$

This metric indicates if the adaptive strategies are influenced by the first three performance metrics. For example, the adaptive part may decide to apply a strategy which uses less RAM or CPU, or which provides a response in less time by paying in the quality of the response (offering a medium, rather than a high quality).

E. Documentation Metrics

The documentation metrics concern two aspects: the comments and the available documentation related to an adaptive system. Their roots are in the general software metrics and they have been interpreted and adapted to provide useful information related to the design of adaptive issues.

CE: Comments per Entity

$$CE = \frac{Number\ of\ comment\ lines}{Number\ of\ entity\ lines} * 100$$

This metric indicates the percentage of the number of comment lines for each adaptive entity. It may be computed for the entities which do not need personalization and those which need personalization (but before their actual personalization). If these two values differ significantly we may suppose that the last category has been predisposed to be personalized and due to the available comments, the personalization may be performed easier.

CF: Comments per Functionality

$$CF = \frac{Number\ of\ comment\ lines}{Number\ of\ functionality\ lines} * 100$$

This metric indicates the percentage of the number of comment lines for each functionality offered by an adaptive entity. It may be computed for the functionalities which do not need personalization and those which need personalization (but before their actual personalization). The considerations for the previous metric hold also for the present one.

CTI: Comments Type Index

$$CTI = \frac{Number\ of\ comment\ lines\ per\ category}{Number\ of\ comment\ lines\ in\ an\ adaptive\ entity} * 100$$

This metric indicates the percentage of the comments of a given type (e.g., auto-generated, formal language, natural language, commented code) considering all the comments in an adaptive entity.

DTI: Documentation Type Index

$$DTI = \frac{Documentation\ quantity\ per\ category}{Overall\ available\ documentation} * 100$$

This metric indicates the percentage of the documentation of a given type (e.g., descriptive, samples, personalization examples for adaptive entities, auto-generated) considering all the available documentation.

F. Miscellaneous Metrics

We have identified three more aspects which should be considered for evaluation of adaptive systems.

ADI: Adaptivity Distribution Index

It regards the distribution of the adaptive elements and entities on the physical nodes of an adaptive system. ADI provides information on the replication of adaptive elements and entities inside a system.

RAMI: Results of Adaptivity Memorization Index

It indicates if the results of each adaptive step are stored temporarily or persistently in the system in order to provide or optimize the adaptive functionalities.

AAI: Adaptive Alternatives Index

It is related to the way in which the various alternatives of adaptivity are provided. Alternatives may be of two types: horizontal and vertical. In the horizontal mode, adaptivity optimizes the usage of the resources to provide the required functionality; while, in the vertical mode, adaptivity optimizes the quality of information to provide the required functionality.

IV. ANALYSIS OF CASE STUDIES THROUGH METRICS

This section analyzes the four case studies introduced in Section II from the adaptivity metrics point of view. Two premises should be made here. First, the case studies are well-defined and thought to outline their adaptivity features. However, several observations have been extracted from

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE I. ADAPTIVE ASPECTS RELATED TO FOUR CASE STUDIES

Case Study	Goal of Adaptivity	Adaptivity Type	Main Steps of Adaptivity				Models	Applicable Categories of Metrics
			Monitoring	Analyzing	Deciding	Changing		
Web-based Client Server	Automate administration tasks Performance optimization	Architectural	Number of servers Bandwidth Response time	Constraints evaluation through invariants	Adaptive strategy	Mapping / reflecting the changes at the functional level	Rainbow – definition of specific architectural styles	Architectural separation of concern Architectural growth Administrator interaction Performance Miscellaneous
Video-conference	Performance optimization	Architectural	Bandwidth Gateway cost	Constraints evaluation through invariants	Adaptive strategy	Mapping / reflecting the changes at the functional level	Rainbow - definition of specific architectural styles	Architectural separation of concern Architectural growth Performance Miscellaneous
Adaptive Image Server	Performance optimization Resource usage optimization	Behavioral	Image resolution Bandwidth CPU load Throughput	Adaptive policies guide the analysis of the monitored parameters and determine the actions to be applied The results are stored in a temporary repository to be used directly in identical situations		Modify image	Adaptive Server Framework – personalization of the framework components	Architectural separation of concern Architectural growth Structural separation of concerns Structural growth Personalization User interaction Performance Miscellaneous
AHA!	Enhance functionalities	Content Navigation	User knowledge level	Rule based users' update profiles Rule based adaptation of content Rule based adaptation of navigation		Modify content Modify navigation		Structural separation of concerns Structural growth Administrator interaction User interaction Performance Miscellaneous

their study. Second, the information we used to analyze these case studies consisted exclusively in their description in various articles. Hence, only qualitative observations have been drawn.

Table I summarizes the main adaptive aspects of the four case studies in terms of the goals of runtime adaptivity, the type of the addressed issues, the main conceptual steps of adaptivity, the models or frameworks used to achieve adaptivity, and the categories of metrics which provide meaningful information for their evaluation.

The analysis of the case studies through the adaptivity metrics has lead to the following considerations.

The goals of exploiting runtime adaptivity are of various natures: automate administration tasks, resource usage optimization, or enhancement of functionalities. A recurring goal is related to the performance optimization. The question is what to measure to evaluate performance aspects? The authors of three case studies indicate as one of the performance aspects the response time, which is expressed through the pLatency metric. Only WebCS uses this terminology, while the other two focuses on the bandwidth (which influences the response time). WebCS and VConf are proposed by the same authors. These are simple examples, however they point out the importance of having the same metrics for the same performance issues and moreover, for their evaluation and comparison in different case studies.

Adaptivity may regard various aspects (architectural, behavioral, content and navigation in these cases). Independent of these aspects, it is fundamental to fulfill the separation of concerns metrics at the architectural and/or structural levels. The last two case studies merge the analysis and decision steps in one single step. This is mostly related to the fact that the adaptivity issues are strongly related to the application domain (e.g., specific information as images or learning content) and the factors which are considered in input of the adaptive process.

The separation of concerns and growth metrics at the architectural or structural levels provide information on the modularity, reusability, flexibility, extensibility and scalability of an adaptive system.

Furthermore, in three of the case studies changes are visible at the functional level. The metrics providing information about this aspect are the structural and performance ones. AIS updates also the adaptive knowledge by storing the information resulted from the various adaptive steps in a repository in order to use it in similar cases without performing the same computation again. This characteristic is described through the RAMI index. AIS considers also the computational overhead introduced by the adaptivity part and expressed through the pCPU performance metric.

There are various changes which may be applied to address the same performance issues. For example, in WebCS, to improve the response time an architectural

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

approach is used: a server is added to the system. This implies that horizontal adaptive alternatives are available (see Figure 5-A). In AIS, the same performance issue is addressed by modifying images, which have different dimensions and hence, may be changed less or more depending on the current request and the status of the system. This implies the availability of vertical adaptive alternatives (see Figure 5-B). As in the case of the AAI index, in the first case, the usage of the resources which provide adaptive functionalities is optimized, while in the second case, the quality of the information needed to provide adaptive functionalities is optimized. In WebCS the administrator tasks have been automated in that the adaptive mechanisms decide whenever a server should be connected or disconnected from the system in order to ensure its performances. In this context, interaction metrics provide information on the efficiency of the adaptivity modules from the point of view of the reduction of the overhead introduced by the interaction with the administrators.

The first three case studies exploit the adaptive concepts (e.g., elements and entities) defined by a general model (e.g., the Rainbow and ASF framework). Moreover, WebCS and VConf are based on a common approach: Rainbow. In these cases the personalization metrics are useful to evaluate the adaptation and usability of the models' concepts in various contexts.

The AHA! approach does not exploit or personalize any model. One of the reasons motivating this characteristics is that adaptivity is specific to the application domain and furthermore, only domain-specific content and its provisioning are adapted. The functioning of AHA! in the absence and in the presence of adaptivity differs significantly: in a non-adaptive scenario the same content is presented in the same way to all users, while in an adaptive scenario users have to be identified and the content representation and provisioning modified. These modifications are properly described through the structural growth metrics, user interaction, and pQoR metric.

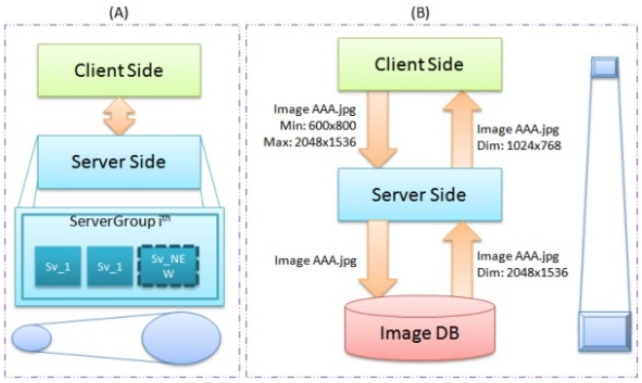

Figure 5. Adaptive alternatives in WebCS and AIS

The last column in Table I lists the categories and subcategories (as shown in Figure 1) of the metrics which are meaningful to evaluate the adaptive features of each case study. The documentation category has been not mentioned because the only documentation we have considered is composed of the articles describing these case studies.

V. Related Work

There are various approaches for the evaluation of runtime adaptivity properties. Typically, they are defined for specific application domains and/or consider a particular perspective on adaptivity.

For example, a methodology for empirical evaluation of adaptive systems is presented in [21]. It considers the use of adaptivity to reduce the complexity of the interaction between users and information systems. Hence, it addresses adaptivity from the users' point of view. The methodology defines six steps to achieve this goal: evaluation of reliability and external validity of input data acquisition, evaluation of the inference mechanism and accuracy of user properties, appropriateness of adaptation decisions, change of system behaviour when the system adapts, change of user behaviour when the system adapts, and change and quality of total interaction. Each of these steps is addressed independently in the context of a framework which enables the evaluation of reliability and external validity of input data acquisition, inference mechanisms and accuracy of user properties, adaptation decisions, and overall interaction (including system and user behaviour and usability).

Or, [19] proposes a set of primary features based on which adaptive hypermedia systems may be evaluated. These features are categorized as follows: adaptation, software quality, software engineering, and technology. This approach considers only a specific type of systems.

Metrics for the evaluation of adaptivity in information systems are introduced in [20], which identifies three generic indexes applicable at the architectural level: element adaptability index (which is 1 for adaptable elements, and 0 otherwise), architecture adaptability index (defined as the sum of all element adaptability indexes divided per total number of elements), and software adaptability index (defined as the sum of the architecture adaptability indexes for all the architecture of a software divided per the total number of architectures of that software). The same authors propose a framework called the Process-Oriented Metrics for Software Architecture Adaptability (POMSAA) [4] to calculate scores for the adaptability of software architectures. The quantitative scores are computed based on the satisficing degree [4] of a non-functional requirement, which in this case regards adaptivity. Both these works consider adaptability only at the architectural level.

A more detailed set of evaluation mechanisms is presented in [11]. This work proposes the evaluation of self-* systems from three points of view: (1) the methodology adopted for their development, (2) the performances offered at runtime, and (3) the intrinsic characteristics of such systems. More specifically, this paper focuses on aspects related to performance, robustness, computational complexity, and decentralization and local algorithms. Even

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

if this approach is strongly related to the multi-agent domain, it may be adopted for other application domains.

To our knowledge there is no work in the scientific literature addressing the evaluation of the frameworks for adaptivity through metrics.

VI. Conclusions and Further Work

This paper has proposed a set of metrics for runtime adaptivity. These metrics should be considered as a starting point towards the identification and specification of what should it be evaluated and how should it be evaluated in adaptive system.

Our initial work on this topic has been previously presented in [13], [17], [18]. In [17] we have focused our attention on the feasibility of the definition of measurable evaluation mechanisms for adaptive systems, and on the usability of these metrics as design hints in the development process of new adaptive systems and as formal approaches for the evaluation of the existing adaptive systems. A first set of metrics for the evaluation of adaptive systems has been presented in [18]. The aim of this work was to provide a concrete mechanism for the evaluation of the adaptive features of information systems. In [13] we have extended the evaluation of adaptive properties also to the frameworks which provide support for the development of adaptive systems. It is fundamental to understand their personalization and documentation aspects in order to evaluate the effort necessary to develop an adaptive system based on such a framework.

The advantage of such metrics is the specification of a common vocabulary for different design, implementation, and performance issues of adaptivity. They provide a common means for the evaluation of adaptive systems, as well as for the comparison of information systems from the adaptivity point of view.

From the software engineering point of view, the architectural and structural metrics suit best as hints in the analysis and design phases of adaptive systems, as well as concrete mechanisms to evaluate the design of adaptive systems. They provide valuable information about the modularity, maintainability, re-usability, or scalability of adaptive systems. Structural metrics are particularly relevant for the implementation and its evaluation in adaptive systems. The personalization sub-category defines common mechanisms to evaluate and choose an appropriate solution for the issues of the current system in the case a framework or a previous solution should be exploited. The documentation metrics may complement the architectural and structural categories in order to offer additional information on the design of runtime adaptivity. These metrics provide additional information on the quality of the design of adaptive systems and frameworks.

The interaction metrics provide information on the advantages of runtime adaptivity from the administrators and users points of view. This is significantly important in various application domains such as e-learning, finance or healthcare.

On the other hand, the performance metrics provide information on the advantages of exploiting runtime adaptivity from the resource usage and overall systems' quality points of view. They are of a determinant significance for all the actors of adaptive systems.

The miscellaneous indexes capture conceptual and distributed aspects of adaptive systems. They are more related to the deployment and efficiency of such systems.

The metrics proposed in this paper have been identified through a process similar to the reverse engineering by considering the available relevant case studies addressing runtime adaptivity issues. Hence, they regard those aspects which are outlined as advantages of the design and exploitation of adaptivity by the authors of these case studies.

Further work will concern the validation and revision of these metrics by applying them to more case studies. Moreover, we will consider the extensibility of this set of metrics also towards standard software engineering metrics for non-functional properties [9] which may be adopted and adapted for the evaluation of runtime adaptivity.

A future development is related to the application of these metrics from the initial phases of the development of adaptive systems. Hence, they should be considered during the identification and specification of the non-functional requirements regarding the runtime adaptivity properties. In this way, it will be possible to indicate a range of acceptable values which should be satisfied by the final system in order to be successfully deployed and exploited.

References

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, Addison Wesley, USA, 2003

[2] P. De Bra, A. Aerts, B. Berden, D. de Lange, B. Rousseau, T. Santic, D. Smits, and N. Stash, "AHA! The Adaptive Hypermedia Architecture", *Proceedings of the 14th ACM Conference on Hypertext and hypermedia*, 2003, pp. 81-84

[3] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, "*Software Engineering for Self-Adaptive Systems"*, LNCS 5525, Springer, 2009

[4] L. Chung and N. Subramanian, "Process-Oriented Metrics for Software Architecture Adaptability", *Proceedings of the 5th International Symposium on Requirements Engineering*, 2001, pp. 310-311

[5] D. Garlan, S. W. Cheng, A. C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-based Self-Adaptation with Reusable Infrastructure", *IEEE Computer*, Vol. 37, No. 10, 2004, pp. 46-54

[6] D. Garlan and M. Shaw, "An Introduction to Software Architecture", Technical Report CMU/SEI-94-TR-21, 1994

[7] I. Gorton, Y. Liu, and N. Trivedi, "An extensible and lightweight architecture for adaptive server applications", *Software – Practice and Experience Journal*, Vol. 38, No. 8, 2007, pp. 853-883

[8] J. He, T. Gao, W. Hao, I.-L. Yen, and F. Bastani, "A Flexible Content Adaptation System Using a Rule-Based Approach", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 1, 2007, pp. 127-140

[9] ISO IEC 9126-1 Standard, http://www.iso.org, 2001, June 2010.

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

[10] G. Karsai, A. Ledeczi, J. Sztipanovits, G. Peceli, G. Simon, and T. Kovacshzy, "An Approach to Self-Adaptive Software Based on Supervisory Control", LNCS 2614, 2003, pp. 77-92

[11] E. Kaddoum, M-.P. Gleizes, J.-P. Georgé, and G. Picard, "Characterizing and Evaluating Problem Solving Self-* Systems", *Proceedings of the ADAPTIVE 2009 Conference*, IEEE Press, 2009, pp.137-145.

[12] L. Masciadri, "A Design and Evaluation Framework for Adaptive Systems", MsC Thesis, University of Milano-Bicocca, Italy, 2009

[13] L. Masciadri and C. Raibulet, "Frameworks for the Development of Adaptive Systems: Evaluation of Their Adaptability Feature Software Metrics", *Proceedings of the 4th International Conference on Software Engineering Advances (ICSEA 2010)*, 2009, pp. 309-321

[14] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. C. Cheng, "Composing Adaptive Software. Computer", *IEEE Computer Society*, Vol. 37, No. 7, 2004, pp. 56-64

[15] C. Raibulet, "Facets of Adaptivity", *Proceedings of the 2nd European Conference on Software Architecture*, LNCS 5292, 2008, pp. 342-345

[16] C. Raibulet, F. Arcelli, S. Mussino, M. Riva, F. Tisato and L. Ubezio, "Components in an Adaptive and QoS-based Architecture", *Proceedings of the ICSE 2006 Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pp. 65-71

[17] C. Raibulet and L. Masciadri, "Evaluation of Dynamic Adaptivity through Metrics: an Achievable Target?", *Proceedings of the Joint IEEE/IFIP Conference on Software Architecture 2009 & European Conference on Software Architecture* 2009, pp.341-344

[18] C. Raibulet and L. Masciadri, "Towards Evaluation Mechanisms for Runtime Adaptivity: from Case Studies to Metrics", *Proceedings of the ADAPTIVE 2009 Conference*, IEEE Press, 2009, pp. 146-152

[19] H. Sadat and A. A. Ghorbani, "On the Evaluation of Adaptive Web Systems", *Proceedings of the Workshop on Web-based Support Systems*, 2004, pp. 127-136.

[20] N. Subramanian and L. Chung, "Metrics for Adaptability", *Journal of Applied Technology Division*, 1999, pp. 95-108.

[21] S. Weibelzahl, "Evaluation of Adaptive Systems", *Lecture Notes Computer Science LNCS 2109*, 2001, pp. 292-29

2010, © Copyright by authors, Published under agreement with IARIA - www.iaria.org