Criteria for Self-$\star$ Systems Evaluation: a Unified Proposal

Jean-Pierre Georgé
IRIT - University Paul Sabatier
Toulouse - FRANCE
george@irit.fr

Marie-Pierre Gleizes
IRIT - University Paul Sabatier
Toulouse - FRANCE
gleizes@irit.fr

Elsy Kaddoum
IRIT - University Paul Sabatier
Toulouse - FRANCE
Kaddoum@irit.fr

Laura Masciadri
Dipartimento di Informatica
Sistemistica e Comunicazione
Università degli Studi di
Milano-Bicocca
Milan - Italy
laura.masc@gmail.com

Gauthier Picard
SMA@G2I - Ecole nationale
Supérieure des Mines
Saint Etienne - France
picard@emse.fr

Claudia Raibulet
Dipartimento di Informatica
Sistemistica e Comunicazione
Università degli Studi di
Milano-Bicocca
Milan - Italy
raibulet@disco.unimib.it

ABSTRACT

In the last few years, the growing complexity of current applications has led to design self-adapting systems presenting self-* properties. Those systems are composed of several autonomous interactive entities. They behave autonomously and present interesting characteristics allowing them to handle dynamics coming from exogenous and endogenous changes.

In this paper, we propose a set of criteria for the description and evaluation of the adaptive properties of those systems. They aim to provide a concrete mechanism to analyze the quality of the design of adaptive systems, to evaluate the effect of self-* properties on the perfomances and to compare the adaptive features of different systems. The criteria are grouped into different categories : methodologies, architectural, intrinsic, and run-time evaluation. They have been identified and specified by analyzing several case studies, which address self-adaptivity issues through different approaches with different objectives in various application contexts.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

Self-* systems, Characterization, Evaluation, Comparison

1. INTRODUCTION

In the last few years, the growing complexity of current applications has led to design self-adaptive systems presenting self-* properties. Self-adaptivity also called runtime adaptivity [16, 20] is related to the ability of systems to perform changes during their execution. Those systems are necessarily composed of several autonomous interactive entities (called agents in this paper) acting into an environment. In general, the global behavior of the system emerges from the local interactions between its agents. Those systems behave autonomously and must handle the dynamics coming from exogenous or endogenous changes so they are able to self-adapt [2, 23]. To clarify, endogenous changes are perturbations caused by entities belonging to the system. Exogenous changes solely occur from entities outside the system. Handling these perturbations can lead to change agents behavior or even the global objective of the system.

Each agent encapsulates different non-adaptive functionalities of the system. By adding self-* properties to the agent, the system becomes self-adaptive. Among these self-* properties, tje most common ones are: self-organization, which concerns the mechanism or the process enabling a system to change its organization without explicit external command during its execution time [6], self-stabilization or homeostasis where the system always reaches a stable state [26], self-healing which is the mechanism that allows a system to detect and recover from errors and self-tuning for the systems that are able to adjust their parameters.

The engineering of self-adaptivity is one of the most challenging issues to address in today's systems due to its complexity, as well as its impact on the entire system. Its advantages are mostly related to the improvement of performances, enhancement of the functionalities' quality and automation of administrative tasks. On the other hand, it increases the dimension and computation of a system.

The wide range of the objectives, facets and solutions concerning adaptivity leads to a difficult evaluation of adaptive systems. In this context it would be very useful to have common criteria to analyze 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 to cooperate to achieve complex tasks.

We aimed to identify and specify such common evaluation criteria by considering the available solutions described in the scientific literature [5, 8, 10, 12, 14, 16] in order to determine how adaptivity is actually achieved, the main

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...$10.00.

characteristics of the design of adaptive systems, as well as the advantages outlined by the authors of the adaptive systems. The conclusions point out that adaptivity requires additional elements at the architectural or structural levels. 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. The authors of the self-adaptive systems describe the advantages of adaptivity in terms of performances, robustness, simplified and enhanced interaction with the users, and automation of administrative tasks. However, the evaluation of the described solutions provided by their authors is personalized and focused on the current context: 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 self-adaptive systems.

In this context, we propose a set of criteria which may be adopted in the description, design, and evaluation of self-* systems. The criteria are grouped into the following categories: methodologies characterization, architectural, intrinsic self-* properties criteria, and runtime evaluation criteria. The methodologies characterization and the architectural criteria are related to the design issues of self-* systems, while the intrinsic self-* properties criteria including structural, decentralisation and interaction criteria discusse the description of self-* systems. Finally, the runtime criteria reflect the advantages regarding the usability of the adaptive systems. An initial proposal for the definition of these criteria is presented in [21] and [13].

The paper is organized as follows. Sections 2,3,4 and 5 introduce the set of criteria proposed for self-* systems. Section 6 analyzes how those different criteria can be used to compare different self-* systems in addition of several case studies found in scientific literature. Related work is briefly discussed in Section 7. Conclusions and future work are presented in section 8.

2. METHODOLOGIES CHARACTERIZATION

To add adaptivity in a system, specific methodologies concerning the process to follow and the concepts to manipulate during the development phases are used. In fact, all the approaches one can consider to develop a self-* system are not equal from this point of view. However, even if we can try to define criteria concerning this perspective, highlighted points are more qualitative than quantitative. Our aim is to underline the criteria dedicated to the self-* properties such as: identification of autonomous agents, distribution, genericity and flexibility.

AAII: Adaptivity Agent Identification Index

One difficult task for the development of self-* system is to identify the functionalities to make adaptive. As this adaptivity is usually added by the introduction of autonomous agents, this task shift to the agents identification. In some methodologies, the problem specification lead straightforward to this identification. While in some others, some analysis phases must be done.

ADI: Adaptivity Distribution Index

All the self-* systems are not easily distributable depending on the paradigms they rely on such as communication. ADI regards the distribution of the adaptive elements on the physical nodes of an adaptive system. It provides information on the additional effort that must be done to deploy such systems. The evaluation of these efforts underline the fact that the methodology is easily used by any designer or requires specific knowledge.

GI: Genericity Index

Self-* approaches may also differ concerning their level of genericity, i.e. the provided effort to adapt a method to a given problem. The GI allow the disqualification of ad-hoc methodologies (conceived for a specific problem) compared with generic methodologies used for different types of problem.

3. ARCHITECTURAL CRITERIA

The architectural criteria aim to capture the main characteristics of adaptivity which emerge at the system level, hence which are visible and meaningful when considering a global perspective at the architectural level on a system. These criteria are focused on the separation of concerns, system's growth, distribution and replication of adaptivity tasks. Generally, the systems addressing runtime adaptivity through architectural mechanisms are composed of two parts: functional and adaptive. In the following, we use the term element 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 [9]). The defined metrics are valid and applicable to any architectural element.

3.1 Separation of Concerns

The separation of concerns has two facets: the separation between the functional logic and the adaptive logic of a system, and the separation among the main steps of the adaptivity process (consisting of the following four steps: monitoring, analyzing, deciding, and changing). The evaluation of the first facet of the separation of concerns provides useful information about the quality of the design of a system (e.g., a system should function also without its adaptive part), as well as of the reuse and maintainability of its various parts. However, there are cases in which this type of separation is not possible to be achieved due to the nature of the application domain that has a strong influence on the design of a system. In this case, a non-adaptive version of a system would provide different functionalities than an adaptive one. The evaluation of the second facet of the separation of concerns provides useful information on the quality of the design of the adaptive process, on its reuse and maintainability, as well as on its possible distribution among the elements of a system. One architectural element may implement one or more steps of the adaptivity process. These aspects are captured by the following metrics.

SCI: Separation of Concerns Index

$$SCI = \frac{Number\ of\ adaptive\ elements}{Number\ of\ functional\ elements\ dependenton\ adaptivity}$$

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 depend (e.g., interact) with the adaptive ones.

API: Adaptivity Pattern Index
API = Number of adaptive conceptual elements
This metric indicates the separation of concerns between the main conceptual (types of) elements implementing the four steps of adaptivity 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.

3.2 System's Growth

The growth of a system due to its adaptive part may be expressed in various terms, ranging from the minimum number of architectural elements required by adaptivity to the average number of elements needed by an adaptive functionality. These aspects are meaningful for the evaluation of the overall effort and costs required for the development of the adaptive part of a system, by providing a first quantitative perspective during the estimation or analysis of an adaptive system. These aspects are captured by the following metrics.

MAG: Minimum Adaptivity Growth
MAG = Minimum number of elements for adaptivity
This metric indicates the minimum number of elements needed to make a system adaptive independently of the number of functionalities it provides. Essentially, this metric expresses the fix growth of adaptivity at the architectural level. It considers the adaptive elements necessary for the first adaptive functionality of a system.

AGF: Adaptivity Growth per Functionality
AGF = Number of elements for the ith functionality
This metric indicates the number of elements needed for the i-th adaptive functionality. It may be seen as a variable growth for introducing adaptivity per functionality at the architectural level.

OAG: Overall Adaptivity Growth

$$OAG = MAG + \sum_{i=2}^{n} AGF$$

The sum between the last two metrics expresses the architectural growth in number of elements needed by adaptivity.

AvgAG: Average Adaptive Growth

$$AvgAG = \frac{OAG}{n}$$

This metric expresses the average growth per functionality at the architectural level due to adaptivity. It indicates the average number of elements needed for each adaptive functionality.

GaE: Growth of Architectural Elements

$$GAE = \frac{OAG}{Number\ of\ functional\ elements} * 100$$

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

3.3 Distribution and Replication

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

4. INTRINSIC Self-$\star$ PROPERTIES CRITERIA

The intrinsic characteristics considered in this paper are related to the separation of concerns, structural growth, influences on the adaptive logic, computational complexity, decentralization and local algorithms, and automation of the human tasks. The separation of concerns and system's growth criteria presented in the previous section may be adapted and applied also at the implementation level as shown in [21]. Hence, this section is focused on the presentation of the further evaluation critera.

4.1 Influence on the Adaptive Logic

The adaptive logic may be influenced by factors external to the system, which are usually domain-dependent, and internal to the system, which are coming from the functional logic. The evaluation of these aspects provides information on the domain- and system-dependence of the adaptive logic, and implicitly of its reusability property. These aspects are captured by the following metrics.

DFIA: Domain Factors Influencing Adaptivity
DFIA = Number of input domain factors in the adaptive logic
This metric counts the number of domain specific factors which are taken in input by the adaptive logic. Higher is this value, stronger is the influence of the application domain or contextual aspects on the adaptive logic.

SFIA: System Factors Influencing Adaptivity
SFIA = Number of input system factors in the adaptive logic
This metric counts the number of system specific factors which are taken in input by the adaptive logic. Higher is this value, stronger is the dependence between the functional and the adaptive logic.

FIA: Factors Influencing Adaptivity
FIA = DFIA + SFIA
This metric counts the total number of factors influencing adaptivity.

4.2 Decentralization and Local Algorithms

In our experience, nearly all systems exhibiting self-$\star$ properties on real-world complexity scale problems rely in one way or another on distribution of the computation, decentralization of the control and *local* mechanisms. When one local, even simply calculated (in terms of complexity) change can have non-linear repercussion throughout the whole system, how can we calculate the computational complexity of the whole system? There exists work [19] on complexity theories for local search algorithms, with for instance the definition of the *PLS* (Polynomial Time Local Search) class, but they fix very strict boundaries.

LCC: Local Computational Complexity
The LCC concerns the analyzis of the local algorithm of each agent. It disqualifies certain local algorithms which require too much computational power.

DD: Decentralization Degree
DD = control distribution between agents
This metrics concerns the deciding step of the adaptivity process. It indicates how the decision process is ditributed among agents and how many agents is involoved in. The DD metrics combined with the LCC one gives unformation on how simple are the agents in the system and how many other agents does it need to know or interact with.

ALI: Action Locality Index
ALI = repercussions of a local action on the rest of the system
Since self-$\star$ systems are complex systems, they are prone to the well known *butterfly effect*. But we can already evaluate how probable this will be by studying the strategies of the agents in regard to this. For instance, does an agent systematically broadcasts its informations and needs? Or does it proceeds by selecting its targets based on an evaluation of their relevance?

ANI: Agent Number Influence

$$ANI = \frac{efficiency\ of\ the\ system\ with\ n\ agents}{efficiency\ of\ the\ system\ with\ p\ agents}$$

p can be higer or less than n. This metrics informs on how the efficiency can drastically increase or decrease depending on the number agents. A ratio between ANI and the costs of an agent helps to evaluate the benefit of introducing/removing agents from the system. More importantly for self-$\star$ systems, ANI enables to judge the scalability of a system as it evolves during its life.

4.3 Automation of the Human Tasks

One of the main objectives of exploiting adaptivity in self-$\star$ systems is to automate the human tasks [20]. We are interested in the automation of the administrators' tasks because an adaptive system should be as self-managing as possible. Furthermore, we are interested also in the interaction of the final users with an adaptive system, interaction which should be simple and straightforward. These aspects are captured by the following metrics.

AAT: Automated Administration Tasks

$$AAT = \frac{Number\ of\ AAT}{Number\ of\ Total\ Administration\ Tasks} * 100$$

This metric indicates the percentage of the automated administration tasks achieved through adaptivity with respect to the total number of administration tasks required by the system.

AvgUIF: Average User Interaction per Functionality (F)

$$AvgUIF = \frac{\sum_{i=0}^{n} Number\ of\ Interactions\ for\ the\ i_{th}\ F}{n}$$

This metric indicates the average number of interactions users should perform with a system to require a functionality. n is the total number of functionalities offered by the system.

5. RUNTIME EVALAUTION CRITERIA

Different questions are to be considered by designers to validate the well-functioning of the system and of the self-$\star$ mechanism, such as: is the system able to solve the problem for which it is conceived? Is the system able to self-adapt in an efficient way? Are the self-$\star$ properties implemented in the system sufficient to insure the robustness? etc. Evaluating the system at run-time concerns the results obtained at the end of its execution and its behavior during it. The former part gives information on the system performance. While the latter part enables the study of how the system handles dynamics and points up the self-$\star$ characteristics. Note that the end of the execution of a self-adaptive system is not so easy to determine. This kind of systems has to continuously self-adapt to its environment and has not to stop. If it has to reach a given state or a solution, because of the distribution, the end cannot be known by the system itself. Designers must add a global observer of the system which can decide if the system has reached its goal and by that has to stop.

5.1 Performance metrics

The evaluation criteria presented in this section concerns the results of different system executions for a given scenario. We consider different metrics that can be studied to evaluate the perfomance of a system. Once these measures are done, they can be used by the evaluator to compare its system with others.

Latency

$$Latency = \frac{Response\ time\ in\ a\ \text{self-}\star\ \ situation}{Respnse\ time\ in\ an\ nominal\ situation}$$

This metric indicates the variation of the response time in the presence of self-$\star$ conditions when adaptive mechanisms are exploited with respect to the response time in normal conditions for the same functionality. To measure the response time, different definition can be utilized such as the number of performed atomic operation [18], the computing time (CPU) [4, 11, 17] or the number of steps needed by agents to reach the solution [4, 15]. We consider that this last measure is the most appropriate and easiest way to measure time. In fact, considering the measurement of CPU needs to take into account machines and platform characteristics, while counting the number of verified constraints in distributed context must consider the asynchrony between the agents [18] which is also the case of considering the time needed by the most costly operations.

WAT: Working Vs. Adaptivity time

$$WAT = \frac{Working\ time}{Adaptivity\ time}$$

The working time concerns the time needed to perform the usual function of the system, while the adaptivity time concerns the time needed to self-adapt to changes in the environment. WAT indicates if the system spends too much time communicationg (adapting to changes) than working. If this ratio is less than one, this means that a lot of time is given to the adaptation which gives indication on the robustness of the system. For some systems, this distinction is not possible, we propose to evaluate the mean of the time differences MTD between the execution of the system in a scenario without perturbation and its execution in a scenario with perturbations.

CL : Communication Load

$$CL = \frac{Communication\ Load\ in\ a\ \text{self-}^{\star}\ \ situation}{Communication\ Load\ in\ a\ nominal\ situation}$$

In self-* systems, agents usually need to communicate in order to reach the solution or the organization needed to attain the solution. CL enables the study of network load evolution with adding the adaptivity. A way to measure this parameter is the amount of exchanged messages. But, as the number of exchanged message is highly dependent on centralization/decentralization degree (cf. section 4.2) of the algorithms, it would be interesting to analyze the communication load in regard to the decentralization degree. In decentralized algorithms more messages are needed among agents while in centralized approach agents doesn't have to exchange a lot of information. Unfortunately, rare are the studies that combine them. Different other parameters are interesting to examine when studying the communication load [3]. We distinguish between them the message *size* and *complexity* which are relevant indicators of the time needed to understand and process messages. The network load is mainly calculated using this communication load in addition of transit and latency time.

QoR: Quality of Response

$$QoR = \frac{Quality\ of\ response\ in\ a\ \text{self-}^{\star}\ \ situation}{Quality\ of\ response\ in\ an\ nominal\ situation}$$

This metric indicates the variation of the quality of response in the presence of self-* properties when adaptive mechanisms are exploited with respect to the quality of response in normal conditions for the same functionality. The accuracy or quality of solution refers to functional adequacy of the designed system.

ND: Non Determinism

ND = Distance betwenn different solutions

In self-* systems, information are distributed among agents which behave asynchronously. These characteristics normally make the resolution not deterministic. As a consequence, for one considered scenario, the solutions obtained from one execution to another may be different [7]. Studying the distances between those obtained solutions give information about the stability of the system in comparison with exact approach.

Progress

Progress = percentage of the main goal reached at each step

Evaluating the progress of the system refers to the study of how the system progressively reaches a complete solution using self-* properties. It is just the measure of the activity. During time, this can be observed by looking at the curves representing this progress as they will have specific forms (linear, exponential, logarithmic) (see section 6). More interestingly for self-* systems, we can analyze how self-adaptation influences this progress: does it maintain or even increase the rate (which is an ideal situation), does the progress slow down (how much?), or even worse, is there a decrease in the progress (the worst being a reset to zero of the progress).

RAMP: RAM Performance

$$RAMP = \frac{RAM\ performance\ in\ a\ \text{self-}^{\star}\ \ situation}{RAM\ performance\ in\ an\ nominal\ situation}$$

This metric indicates the variation of the RAM performance in the presence of self-* conditions when adaptive mechanisms are exploited with respect to the RAM performance in nominal conditions for the same functionality.

CPUP: CPU Performance

$$CPUP = \frac{CPU\ performance\ in\ a\ \text{self-}^{\star}\ \ situation}{CPU\ performance\ in\ an\ nominal\ situation}$$

This metric indicates the variation of the CPU performance in the presence of self-* conditions when adaptive mechanisms are exploited with respect to the CPU performance in normal conditions for the same functionality.

SDG: Storage Dimension Growth

This metric indicates the physical storage growth in kilo bytes due to the presence of the adaptive mechanisms in a system. Adaptive mechanisms include both agents and their link with the functional entities.

SDA: Storage Dimension of the Agents

This metric indicates the physical storage growth in kilo bytes needed to store the agents.

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

This metric indicates the physical storage growth in kilo bytes due to the entities which have been defined for the link and communication between the functional and adaptive part of a system. Those last three metrics give information on the cost and memory usage when adding the adapatation mechanisms.

5.2 Homeostasis & Robustness

As stated above, the main property of self-$\star$ systems is their ability to maintain their functioning in an environment presenting a high level of dynamics. In other words, we can say that self-$\star$ systems possess robustness, and homeostasis/ self-stabilization abilities. The robustness ability is presented by their capacity to maintain their behavior when perturbations occur [23]. The homeostasis ability is what enables the system to regain its normal behavior after being perturbed. In [26], authors defined the homeostasis as the capacity of regaining an ideal state in which the system is operating in a maximally efficient way. Studying those two abilities enables to understand in depth how the system handles the dynamic changes of the environment, and by that validate its adequacy to solve the problem.

RI: Robustness Index

Considering dynamics is a very interesting point in complex problem solving. Events occur at different steps and are usually unpredictable. A robust system is a system which is able to maintain a stable behavior even under perturbations. To measure RI, we can consider:

- The system maintains a functional adequacy even during perturbation. In other terms, the variation of the solution quality is weak under perturbations.
- The new solution (state) reached by the system is closed to the previous one.
- The amount of changes inside the system between the state when the perturbation occurs and the new stable state, is minimal.

TA: Time for Adaptation

TA = needed time to regain a normal behavior after a perturbation

Its the time needed to take into account one type of dynamic changes. The number of steps needed by agents to return to the normal functioning is an appropriate measure of this criterion. Two levels are to be considered to measure adaptation time: local level and global level. In the first one, the TA needed by each agent is examined. Measuring this time allows the designer to examine the self-adaptivity of the agents. In the second one, the self-adaptivity of the global behavior of the system is studied by measuring the TA needed by the system to regain its normal behavior.

6. DISCUSSION

7. RELATED WORKS

Several approaches for the evaluation of adaptivity properties have been proposed in the scientific literature. They are characterized by providing one perspective on adaptivity and/or to be application domain-specific.

A methodology for empirical evaluation of adaptive systems is presented in [27]. In this case, the objective of adaptivity is to reduce the complexity of the interaction between users and information systems. Hence, this work analyzes adaptivity from the user's point of view.

A set of primary features based on which adaptive hypermedia systems may be evaluated is described in [24]. These features are categorized as follows: adaptation, software quality, software engineering, and technology. This approach considers only a specific type of systems.

A more detailed set of evaluation criteria is presented in [13]. 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 if this approach is strongly related to the multi-agent domain, it may be adopted for other application domains.

Metrics for the evaluation of adaptability in information systems are introduced in [25], which identifies three generic indexes applicable only 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 software metrics for adaptivity presented in [22, 21] are categorized in four groups architectural, structural, interaction, and performance issues. They are applicable to the systems for which there are available both a non-adaptive and an adaptive version of the same system.

8. CONCLUSION

In this paper, we have presented a set of criteria meaningful for the analysis of various aspects concerning self-$\star$ systems. The main advantage of these criteria is the specification of a common vocabulary for the various design, implementation, and performance issues of adaptivity. They provide a common means for the evaluation of adaptivity, as well as for the comparison of self-adaptive systems from the adaptivity point of view.

The methodologies and architectural criteria are best suited for the analysis and design phase when building adaptive systems. The intrinsic criteria provide information on the characteristics of these systems and the advantages of daptivity from the administrators and users points of view which is significantly important in various application domains such as e-learning, finance or healthcare. Finally, the runtime evaluation outlines the advantages of exploiting self-adaptivity from the overall quality of the system. They are of a determinant significance for all the actors of adaptive systems.

This work is a first step for evaluating self-$\star$ systems and highlighting the consequences of their specific behaviors devoted to their self-adaptativity abilities. The criteria proposed in this paper have been identified through a process similar to the reverse engineering by considering the available relevant case studies addressing self-adaptivity issues. Hence, they concern those aspects which are outlined as advantages of the design and exploitation of self-$\star$ properties by the authors of these case studies. Further work will be related to the validation and revision of these criteria by applying them to more case studies. Nevertheless, we are aware that formal validation are fundamental for the evaluation of these systems. Currently, because we can't formally prove all the behavior of these systems, we think that partial proofs can be done and a lot of work must be done mixing simulation and formal classical proofs.

9. REFERENCES

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

[2] C. Bernon, M.-P. Gleizes, S. Peyruqueou, and G. Picard. Adelfe: a methodology for adaptive multi-agent systems engineering. In P. Petta, R. Tolksdorf, and F. Zambonelli, editors, *Third International Workshop on Engineering Societies in the Agents World (ESAW-2002)*, volume 2577, pages 156–169, Madrid, Spain, September 2002. Springer-Verlag (LNAI).

[3] I. Brito, F. Herrero, and P. Meseguer. On the evaluation of discsp algorithms. *The Fifth International Workshop on Distributed Constraint Reasoning (DCR04) Toronto, Canada September 27, 2004 held in conjunction with Tenth International Conference on Principles and Practice of Constraint Programming (CP 2004)*, (5):142–151, 2004.

[4] G. Clair, E. Kaddoum, M.-P. Gleizes, and G. Picard. Self-Regulation in Self-Organising Multi-Agent Systems for Adaptive and Intelligent Manufacturing Control. In *IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), Venice Italy, 20/10/08-24/10/08*, page (on line), http://www.computer.org, 2008. IEEE Computer Society.

[5] P. De Bra, A. Aerts, B. Berden, B. de Lange, B. Rousseau, T. Santic, D. Smits, and N. Stash. Aha! the adaptive hypermedia architecture. In *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 81–84, New York, NY, USA, 2003. ACM.

[6] M.-P. Di Marzo Serugendo, Giovanna and and A. Karageorgos. Self-Organisation and Emergence in Multi-Agent Systems: An Overview. *Informatica*, 30(1):45–54, janvier 2006. ISSN 0350-5596.

[7] F. Gaillard, Y. Kubera, P. Mathieu, and S. Picault. A reverse engineering form for multi agent systems. In A. Artikis, G. Picard, and L. Vercouter, editors, *Proceedings of the 9th International Workshop Engineering Societies in the Agents World (ESAW'2008)*, 2008. Actes ?lectroniques uniquement.

[8] D. Garlan, S.-W. Cheng, A.-C. Huang, B.-R. Schmerl, and P. Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *IEEE Computer*, 37(10):46–54, 2004.

[9] D. Garlan and M. Shaw. An introduction to software architecture. Technical Report CMU-CS-94-166, Carnegie Mellon University, January 1994.

[10] I. Gorton, Y. Liu, and N. Trivedi. An extensible and lightweight architecture for adaptive server applications. *Software - Practice and Experience Journal*, 38(8):853–883, 2008.

[11] K. M. Hansen, W. Zhang, and M. Ingstrup. Towards self-managed executable petri nets. In *SASO '08: Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, pages 287–296, Washington, DC, USA, 2008. IEEE Computer Society.

[12] J. He, T. Gao, W. Hao, I.-L. Yen, and B. F. A flexible content adaptation system using a rule-based approach. *IEEE Transactions on Knowledge and Data Engineering*, 19:127–140, 2007.

[13] E. Kaddoum, M.-P. Gleizes, J.-P. Georgé, and G. Picard. Characterizing and evaluating problem solving self-* systems. In *Proceedings of the ADAPTIVE 2009 Conference, IEEE Press.*

[14] G. Karsai, A. Lédeczi, J. Sztipanovits, G. Peceli, G. Simon, and T. Kovácsházy. An approach to self-adaptive software based on supervisory control. In *LNCS 2614*, pages 77–92, 2003.

[15] N. Lynch. *Distributed Algorithms*. Morgan-Kaufmann, 1996.

[16] P. McKinley, S. Sadjadi, E. Kasten, and B. Cheng. Composing adaptive software. *Computer*, 37:56–64, 2004.

[17] D. Meignan, J.-C. Creput, and A. Koukam. A coalition-based metaheuristic for the vehicle routing problem. pages 1176–1182, June 2008.

[18] A. Meisels, E. Kaplansky, I. Razgon, and R. Zivan. Comparing performance of distributed constraints processing algorithms. In *AAMAS-02 Workshop on Distributed Constraint Reasoning*, number 5, pages 86–93, 2002.

[19] C. H. Papadimitriou, A. A. Schäffer, and M. Yannakakis. On the complexity of local search. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 438–445, New York, NY, USA, 1990. ACM.

[20] C. Raibulet. Facets of adaptivity. In *Proceedings of the 2nd European Conference on Software Architecture, LNCS 5292*, pages 342–345, 2008.

[21] C. Raibulet and L. Masciadri. Towards evaluation mechanisms for runtime adaptivity:from case studies to metrics. In *Proceedings of the ADAPTIVE 2009 Conference, IEEE Press.*

[22] C. Raibulet and L. Masciadri. Evaluation of dynamic adaptivity through metrics: an achievable target? In *WICSA/ECSA*, pages 341–344, 2009.

[23] P. Robertson, R. Laddaga, and H. Shrobe. Introduction: the first international workshop on self-adaptive software. In R. L. . H. S. P. Robertson, editor, *Proceedings of the 1st IWSAS*, volume 1936 of *LNCS*, pages 1–10. Springer, 2000.

[24] H. Sadat and G. A.A. On the evaluation of adaptive web systems. In *Workshop on Web-based Support Systems*, pages 127–136, 2004.

[25] N. Subramanian and L. Chung. Metrics for adaptability. In *Journal of Applied Technology Division*, pages 95–108, 1999.

[26] R. P. Würtz. *Organic Computing*. Springer, 2008.

[27] S. Weibelzahl. Evaluation of adaptive systems. In *User Modeling*, pages 292–294, 2001.