SELF-ADAPTIVE SYSTEM QUALITY MODELING

MUHAMMAD IMRAN HABIB*

AHSAN RAZA SATTAR*

MUHAMMAD AZAM ZIA*

ADNAN MUNIR*

HASNAT AHMAD HUSNI*

WASEEM BAIG*

Abstract

This paper investigates the effects of modeling dimensions in different State-of-the-art Frameworks, to mitigate the Impact of uncertainty in Self-Adaptive Systems. Self-Adaptive Software systems can modify their behavior according to environmental conditions without human interaction. Self-Adaptive property of a software system depends on a variety of aspects according to the system design and environmental conditions. These factors called modeling dimensions and system quality depends on these modeling dimensions. Software engineering for these systems deals with modifying behavior and uncertainty by using disciplines of software engineering; as a result different mapping techniques are recently introduced to tradeoff uncertainty in self-adaptive systems. This paper characterize these State-of-the-art mapping techniques according to set of challenges they are addressing and then compared these mapping techniques with modeling dimensions to find ignored factors and propose a systematic method to find out the best mapping technique to enhance the quality in Self-Adaptive Systems.

General Terms: Self-Adaptive Systems

* UNIVERSITY OF AGRICULTURE, FAISALABAD, PAKISTAN

Additional Key Words and Phrases: Quality in Self-Adaptive Systems, State-of-the-art Self-Adaptive Systems, modeling dimensions, Characterize Self-Adaptive Systems, automatic-computing.

1. INTRODUCTION

Advancement in computer technologies and software engineering results as explosive development in computing system and its application areas. However, as systems grows their complexity propagates as supplementary, as a result these system's capabilities are rapidly render and their development, configuration and management becomes breakthrough challenge in existing paradigms. New Systems become more interrelated and diverse architectural design are not so much able to anticipate communications among systems components even mostly systems are problem to configure, maintain, optimize and merge. This Leeds to consider alternative approaches which are successfully deals with challenges of complexity, heterogeneity, dynamism and uncertainty. The only option remains to make decisive response for these conflicting and changing demands is automatic-computing or self-adaptive-systems. Self-adaptive-system is a new strategic and holistic approach to design complex system; it stimulates the functionality of self-managing design. Automatic-system has the capability of self-managing, ubiquitous computing, autonomous, able to hide their complexity and has the ability to provide services as desired by user. Self-adaptive software change its own behavior in reaction to changes in its functioning environment and systems always decide on its own, they required only high-level guidance from user. They have the ability to check environmental constantly, optimize its status, and adopt changing conditions. But self-adaption is a great challenge itself.

1.1 Why We Need Self-Adaptive

Self-adaptive-Systems are design to control computing systems with self-managing mechanisms. There is an endless of list of self-managing mechanisms for example; self-administration, self-assessment of risks, self-configuration, self-correction, self-diagnosis of faults, self-evolution, self-governing, self-healing, self-learning, self-modeling, self-monitoring, self-adjusting, self-optimization, self-organization, self-planning, self-protection, self-recovery, self-scheduling, self-sensing/perceiving, self-tuning, etc. (Tianfield, 2003). In simple mean anything which is recognizable by system software, such as user input, managing hardware devices, view sensors and instruction are self-adaptive. In another point of view, in adaptation we map evolution. Evolution based on where, what, when and how (Buckley et al, 2005).

1.2 Self-Adaptation Properties

It is very important to identify adaptation properties that have been used for the analyzed spectrum of adaptive systems, from control theory to software engineering, to evaluate the adaptation process. The identified adaptation properties are stated as follows. The first four properties, called SASO properties, correspond to desired properties of controllers from a control theory perspective; note that the stability property has been widely applied in adaptation control from a software engineering perspective. The remaining properties in the list were identified from hybrid approaches. Citations attached to each property refer either to papers where the property was defined or to examples of adaptive systems where the property is observed in the adaptation process.

Stability: The degree in that the adaptation processes converge toward the control objective. Unstable adaptations indefinitely repeat the process with the risk of not improving or even degrade the managed system to unacceptable or dangerous levels. In a stable system responses to a bounded input are bounded to a desirable range (Parekh et al, 2002).

Accuracy: This property is essential to ensure that adaptation goals are met, within given tolerances. Accuracy must be measured in terms of how close the managed system approximates to the desired state (e.g., reference input values for quality attributes) (Solomon, 2010). Short settling time is that time which required for the adaptive system to achieve the desired state. Long settling times can bring the system to unstable states. This property is commonly referred to as recovery time, reaction time, or healing time (Parekh et al, 2002).

Small overshoot: The utilization of computational resources during the adaptation process. Managing resource overshoot is important to avoid the system instability. This property provides information about how well the adaptation performs under given conditions-the amount of excess resources required to perform the adaptation.

Robustness: The managed system must remain stable and guarantee accuracy, short settling time, and small overshoot even if the managed system state differs from the expected state in some measured way. The adaptation process is robust if the controller is able to operate within desired limits even under unforeseen conditions (Dowling and Cahill 2004).

Termination: (of the adaptation process). In software engineering approaches, the planner in the MAPE-K loop produces, for instance, discrete controlling actions to adapt the managed system, such as a list of component-based architecture operations. The termination property guarantees that this list is finite and its execution finished, even if the system does not reach the desired state. Termination is related to deadlock freeness, meaning that, for instance, a reconfigurable adaptation process must avoid adaptation rules with deadlocks among them.

Consistency: This property aims at ensuring the structural and behavioral integrity of the managed system after performing an adaptation process. For instance, when a controller bases the adaptation plan on dynamic reconfiguration of software architectures, consistency concerns are to guarantee sound interface bindings between components (e.g., component-based structural compliance) and to ensure that when a component is replaced dynamically by another one, the execution must continue without affecting the function of the removed component. These concerns help protect the application from reaching inconsistent states as a result of dynamic decomposition define this property to complete the atomicity, consistency, isolation and durability (ACID) properties found in transactional systems that guarantee transactions are processed reliably (Parekh et al, 2002).

Atomicity: Either the system is adapted and the adaptation process finishes successfully or it is not finished and the adaptation process aborts. If an adaptation process fails, the system is returned to a previous consistent state.

Isolation: Adaptation processes are executed as if they were independent. Results of unfinished adaptation processes are not visible to others until the process finishes. Results of aborted or failed adaptation processes are discarded.

Durability: The results of a finished adaptation process are permanent: once an adaptation process finishes successfully, the new system state is made persistent. In case of major failures (e.g. hardware failures), the system state can be recovered.

Scalability: The capability of a controller to support increasing demands of work with sustained performance using additional computing resources. For instance, scalability is an important property for the controller when it must evaluate an increased number of conditions in the analysis of context. As computational efficiency is relevant for guaranteeing performance properties in the controller, scalable controllers are required to avoid the degradation of any of the operations of the adaptive process in any situation.

Security: In a secure adaptation process, not only the target system but also the data and components shared with the controller are required to be protected from disclosure (confidentiality), modification (integrity), and destruction (availability).

1.3 Mapping Adaptation Properties and Quality Attributes

Are there any relation-ships between above said quality factors and their properties with system performance? To determine any relationship between factors and performance we discuss a quality model i.e. ISO 9126-1. According to this quality model self-adaptation depends on several quality factors, and Self-adaptiveness impact on system like system-evaluation, system-control, and system-governing. Other quality factors and their relationships are described in Table 1.

Once the adaptation goal and adaptation properties have been identified, the following step maps the properties of the controller, which are observable at the managed system, to the quality attributes of the managed system. It represents a general mapping between adaptation properties and quality attributes. These quality attributes refer to attributes of both the controller and the managed system depending on where the corresponding adaptation properties are observed. According to, SASO properties, including stability, can be verified at run-time by observing performance, dependability and security factors in the managed system. Quality attributes addressed is concern dependability (i.e. availability and maintainability), and performance (i.e. throughput and capacity-scalability).

Adaptation metrics provide the way of evaluating adaptive systems with respect to particular concerns of the adaptation process. Thus, metrics provide a measure to evaluate desirable properties. For instance, metrics to evaluate control systems measure aspects concerning the SASO properties (i.e. stability, accuracy, settling time, and small overshoot).

To characterize the evaluation of adaptive systems, we analyzed the variety of self-adaptive software systems to identify adaptation properties (i.e. at the managed system and the controller) that were evaluated in terms of quality attributes). Just as the evaluation of most properties is impossible by observing the controller itself, we propose the evaluation of these properties by means of observing quality attributes at the managed system. To identify relevant metrics, we characterized a set of factors that affect the evaluation of quality attributes such as speed, memory usage, response time, processing rate, mean time to failure, and mean time to repair.

These factors are an essential part of the metrics used to evaluate properties of both the controller and properties of the managed system (Reinecke et al, 2010).

As presented in Table 1, security of the controller should be evaluated independently of the managed system. This means that ensuring security at the managed system does not guarantee security with respect to the adaptation mechanism.

Scalability is also an adaptive property in K-Components, the agent-based self-managing system proposed by (Downing and Cahill, 2004). Scalability is addressed by evolving the self-management local rules of the agents. Another approach where scalability is addressed as an adaptation property is Madam, the middleware proposed by (Floch et al, 2006) for enabling model-based adaptation in mobile applications.

Scalability is a concern in Madam for several reasons. First, its reasoning approach might result in a combinatorial explosion if all possible variants are evaluated; second, the performance of the system might be affected when reasoning on a set of a concurrently running applications competing for the same set of resources. They proposed a controller where each component (e.g. the adaptation manager) can be replaced at run-time to experiment with different analysis approaches for managing scalability.

Adaptation Properties	**Quality Attributes**	
Stability	Performance	Latency
		Throughput
		Capacity
	Dependability	Safety
		Integrity
	Security	Integrity
Accuracy	Performance	Latency
		Throughput
		Capacity
Settling Time	Performance	Latency
		Throughput
Small Overshoot	Performance	Latency

		Throughput
		Capacity
Robustness	Dependability	Availability
		Reliability
	Safety	Interact. Complex.
		Coupling Strength
Termination	Dependability	Reliability
		Integrity
Consistency	Dependability	Maintainability
		Integrity
Scalability	Performance	Latency
		Throughput
		Capacity
Security	Security	Confidentiality
		Integrity
		Availability

Table 1: Quality Factors and Relationship

1.4 Self-Adaptation: state-of-the-art systems

Many approaches have already been tried to map self-adaptation uncertainty to avoid system failures. This this section we discuss the approaches and their ability to address Uncertainty for execution environment. These system are further called state-of-the-art systems.

Anticipatory Dynamic Configuration (ACD): Poladian and Sousa (2007) proposed an appropriate service of resource allocation for different services to fulfill user task is a technical challenge. This concept in Self-adapting is resolved by using ACD (Anticipatory Dynamic Configuration). ACD addressed three technical issues first how express resources availability or resource prediction, second how combine different predicted resources, third how these predicted resources are continually improve in self-adaption operation. These technical issues are address by using probability theory to maximize the expected value. In ACD decision process the cost of the adaptations is also considered. If the cost of switching is low then configuration switched and if cost is high then alternative configuration is selected.

RAINBOW framework: Garlan and Cheng (2007) proposed a framework for self-adaptive system which has ability to handle verities of architectural systems with dynamic modification to reduce the system cost. Garlan introduced a RAINBOW framework into three steps to handle uncertainty in self-adaptive systems there are problem state identification, strategy selection, and strategy out comes. In problem state identification monitoring and analysis is done through MAP loop. Mitigation uncertainty in done by monitoring the variability by observing the environment and then compared these outcomes with architectural. Once the problem is identified next step is to resolve the problem using best a strategy. Stitch language is used to select strategies and to modeling the uncertainty, RAINBOW has the ability to select strategy at runtime. In the last the before committing the changes the RAINBOW consider the uncertainty and strategy outcomes. All these steps determine the successive or failure effects in self-adaptive system and enhance its capabilities.

RELAX: Whittle and Sawyer (2009) introduced a new requirement specification language RELAX for DASs (dynamically adaptive systems) to explicit environmental uncertainty expression in requirements. They also discussed the way of translating requirements from traditional to RELAX requirements. RELAX is an organized natural language with operators that capture uncertainty. In RELAX language there are three types of operators and some keywords to handle uncertainty factor. In operator first temporal operator, second model operators and third standard operators. RELEX key benefits are: identify the source of uncertainty, monitor the system behavior, differentiate variant and invariant requirements, determine dimensions of uncertainty, and also identify shortcomings in monitoring infrastructure.

Extend RELAX (Goal Modeling): Cheng and Sawyer (2009) was extend RELAX to specify the requirement uncertainty by adding goal based modeling approach. This is a stepwise process to identify the basic requirements, modeling uncertainty factor in environment, integrate these requirement to RELAX goal modeling. These systematic analysis steps are described using four stepwise processes. First step identifies the top-level goal then refine the goal to leaf requirement to their respective agents. Third step is identification of uncertainty impact factors that potentially propagated up lattice goals. In last step these uncertainty factors are mitigated by using mitigation tactics. This process results a model explicitly capture needs of adaptation into environmental uncertainty for target system.

FLAGS: Baresi and Pasquale (2010) addressed different challenges posed by self-adaptive systems. How different requirements are satisfied at deployment time? They add the concept of adaptive goals by using KAOS model in FLAGS technique. Basics of FLAGS to find those requirements that effect on other requirements. These conflicting requirements are called adaptive goals. FLAGS counter measures that which specific requirements are not fulfilled due to predicted uncertainty. A worthwhile objective of FLAGS is that it also deals with goals' uncertainty (uncertainty in goal itself).

FUSION: Elkhodary and Esfahani (2010) presented method of engineering (FUSION framework), how to focus different characteristics like environment, requirements and system's operation, at runtime before deployment of system. FUSION uses MTL (Model Trees Learning) to solve the problems by learning their impact of adaptation decisions on the deployed system's goals. This framework reduces upfront effort by making run time analytical function efficiently and runtime fine tuning of requirement logic in unanticipated conditions. Learning is aided by feature selection space and inters feature relationships. There are two complementary cycles first is learning cycle that relates to different quality measurements of self-adaption actions. Learning cycle monitors environment and find errors in learned relations.

RESIST: Cooray and Kilgore (2011) proposed an approach which continuously furnishes reliability at runtime by integrating various sources of information. Software maintain optimal configuration in changing environment. RESIST framework proposed to address reliability concerns in critically dynamically changed software. RESIST uses reliability expectations in pre-emptive determination and find the optimal configuration. It predicts from several sources to extend the reliability of system. These predictions are further help to make decisions regarding software configuration changing. RESIST is used in mission critical systems due to its more dynamic configuration capability; it can use unknown operation profiles and fluctuation conditions. To estimate the reliability compositional approach in which component level of reliability is estimated by using Discrete-Time- Markov-Chain. When reliability of each component is estimated, the reliability of whole system can also be easily determined.

POISED: Esfahani and Kouroshfar (2011) assessed Impact of uncertainty both positively and negatively, they use possibility theory uncertainty underlying the adaptation decisions POISED. POISED's distinguish between external (arises from environment) and internal (software component system) uncertainty but focus only on internal uncertainty. Estimates of internal uncertainty in the elements problem are incorporated by possibility analysis. Possibility analysis uses the principals of Fuzzy logic and provide a strong basis for representing uncertainty. POISED applied in many types of self-adaptation problems aimed at improving a system's quality attributes by runtime reconfiguration in customizable components. POSIED is much different from convention approaches, Conventional approaches not analyze incorporate uncertainty while POISED cogitate a range of behavior. In some cases decision maker specify the aspect of uncertainty, which is more important and also have a low risk level.

2. MODELING SELF-ADAPTIVE SYSTEMS

Basic Goal of this paper is to compare and summarize recent state-of-the-art systems identifies their critical challenges and limitations. The comparison can be done with modeling techniques, which represent an ideal self-adaptive-system. Now in this section; we focus on those modeling techniques which were used to compare different state-of-the-art systems, which are described in last section. Now we discuss modeling dimensions for self-adaptive-systems. These dimensions describe a specific characteristic of a system that related for self-adaption.

2.1 Modeling self-adaptive properties

There are different things, which can be used to represent self-adaptive system or a self-adaptive system can be influenced by several aspects, like system or user needs, properties, environmental condition, etc. These aspects are used to understand the problem and choosing an appropriate solution give the pattern of system, define complexity of system, testing mechanism of system.

In the self-adaptive system, there is a lack of consensus, that how variation of these system measures. To build Self-adaptive systems they require, conceptual model of self-adaptation, leveraged tools for implementation and a conceptual model of adaptation. However, indeed, they only use engineering expertise and domain knowledge for implementation. They not follow a systematic self-adaption model, So It is very hard to compare or quantify different approaches systematically.

We refer to these points of distinctions as dimension or modeling dimension. These modeling dimension express different factors of self-adaptation and they are classified according to those dimensions. This classification allows specifying self-adaptive properties and finding suitable solution. This classification only covers important characteristics of the system. Infect, objective is that to provide a comprehensive evaluation method and fulfill the key aspects, which is how the different state-of-the-art system can easily evaluate by using these modeling dimensions we recognized and compared distinct systems, especially if they are from the separate domain.

Modeling dimensions are grouped four groups: first, the modeling dimensions related with self-adaptability organizations of the system called goals, second, the modeling dimensions related with reasons of self-adaptation called Change, third, the modeling dimensions related to the mechanisms to attain self-adaptability called Mechanisms, and fourth, the dimensions associated with effects of self-adaptability on a system called Effects. Now we conclude different facets of these four types of modeling dimensions.

2.1.1 Goals Dimension

Goals are the objectives which are under consideration throughout the system's lifetime. They are also called scenarios related to the system. Goals refer self-adaptability aspects of a system infrastructure that provide guidance to that application. A system has a basic goal or high level goal related to overall system and sub goals, which are related to only one or more than one attributes. An example of high level system goals is "Avoid collisions in an automotive vehicle." A high level goal is not sufficient for overall system, so system also contains some sub goals or sub attributes of this high level goal. Some sub attributes of these goals are:

Evolution: In a self-adaptive system, a goal can be changed within the lifetime of a system. A system contains a number of goals at a time. One or more than one goal may change at runtime, similarly some time high level goal themselves changes, which are not expected. In goal evolution, system can manage their goals during the lifetime of the system. These goals are changed due to make the system more protected and safe. So, system can create goals; change their gorals to safe the system and goal evolution is the ability of the system to change their goals at runtime.

Flexibility: Means the system's goals are elastic; they can be change, or they are fixed. Flexibility associated to goal specification, there are three conditions in flexibility, firstly, system is rigid, and its goals never change during the life time of the system. For example, we fix goal that "system shell did this." In this condition, goal is rigid; it never changed throughout the system lifetime. Second condition in goal change is constrained; this refers some conditions are applied to the system if the system goal is changed, as an illustration, if exceptional condition happens, then the system can do a specific task, otherwise it can run smoothly. Constrained is an intermediate ground because it may be rigid or flexible for long time. Third condition is unconstrained in which flexibility in system goal dimension. E.g., in some specific environments, “system may do this, or it can do this."

Duration: Refers the authority of a goal throughout the system’s lifetime. Duration ranges from persistent to temporary. Persistent goals duration are valid for throughout the lifetime of the system while temporary goal valid in a specific period of time they are identified as short term, Medium term, and Long term. Persistent goals duration is more restricted in self-adaptability of the system because they bound the system flexibility in change adaptation. On the other hand, the goals that relate to the temporary goal, it is more illustrative and consider persistent to fulfill the purpose of the system.

Multiplicity: Refers how many goals associated with self-adaptability aspects in self-adaptive-system. A self-adaptive-system may ensure only one goal, which is called single goal or system contained more than one goal, which is called multiple goals. In these goals single or multiple, single goal can easily realize then multiple goals.

Dependency: This dimension captures how different goals are related to each other. This dimension works if the system has more the one goal or multiple goals. If there is more than one goal then, there are two conditions either goals are dependent to each other, or they are independent. In some cases, a system may have some dependent goals; they don’t affect each other, in this case we called them independent goals. In the other case if system’s goals affect other goals, they may have a conflict; one goal may depend to other, i.e. one objective should be achieved to complete a specific goal. These configurations are identified by analyzing the tradeoffs goals. If system goals have no dependencies, then it is called a single goal.

2.1.2 Change Dimension

Most important, adaption cause is a change. When the system’s environment changes, systems have to decide either it needs to adopt change or not. Environment is the context which can change the behavior of the system. Environment is called outward with which the systems interrelate, and also affect the activities of smooth system running. Those changes which accrue due to environment change are called environment-dependent variations. Sometime system itself influences the behavior of the own system this change is called system-dependent variation. These changes are classified as context-dependable changes in self-adaptive-systems, when change accrued either system-dependent or environment-dependent, its type and its rate of recurrence is more important. These values are determined by either it changes can be anticipated or not. All these foundations are important to identify the system reaction of system at runtime (Jackson, 1997).

Source: Means the source of change or origin of change. There are two types of change origin external and internal. External changes refer to change in environment, and internal changes are those changes which the system dependent. To address the change it is important that cause of change recognizes more specifically where change occurred.

Type: This dimension refers nature of change. There are three types of changes functional changes, nonfunctional changes and technological changes. Functional changes are those changes which are specific to functionality of the system, e.g. technical details, processing etc. on the other side nonfunctional changes are related to system quality these changes are further classified in different categories, i.e. performance, maintainability, safety, etc.
(Alain, et al. 2004). Technological changes refer to both aspects related to software and hardware to support the delivery of services.

Frequency: refers the rate of change occurred. Rate of particular change affects the responsive of adaptation. Since rate of occurrence of hindrances is uncommon throughout the system's lifetime.

Anticipation: This dimension is used to capture whether change was predicted before time or not. There are three types of techniques used to measure the degree of anticipation first is foreseen in which is used for caring the anticipation, second is foreseeable, which are planned for anticipation, third and last are unforeseen, which are not planned for anticipation (Laprie, 2008).

2.1.3 Mechanisms Dimension

Mechanisms imprisonments the reaction of the system towards changes, it is as a group of dimensions, which are associated with refer the self-adaption autonomy. It explains how control self-adaptation, self-adaptation influence in terms of time and space, what are the responsive, if the change proceeds, and then how the system is react in response of this change? What is the level of this sovereignty in self-adaptation? (Laprie, 2008).

Type: This dimension related to the structural parameters of self-adaptive-system. There are two types of Type dimension, which are based on structure of the system as well as its components. These are parametric type and structural type. In parametric type depends on system's components, and structural type depends on structure of the system, i.e. how different components are integrated to each other. More important is a parametric type because the structural type cannot change runtime while structural type can change runtime, e.g. speed is controlled to cover more distances in shorter time, etc.

Autonomy: In this dimension, system recognizes the degree of external intervention throughout adaptation. There are two possible ranges of external Autonomy, assisted and autonomous. In assisted autonomy the system has some external influence on the system. This external influence assisted by other systems or human participation. These external influence bodies are considering as other systems. On the other hand, autonomous autonomy there is no external influence on the system, which system should adopt.

Organization: Determines whether self-adaption procedure is to be performed by which specific component of system. If the Organizational procedure performed by single component, it is called centralized. On other sides if organizational procedures are performed by more than one component, then it is called decentralized. In decentralized procedure, no single component can handle overall system.

Scope: Identifies whether adaptation effect encompasses entire system, or it only involves only one component. If the system affects the entire system or involves more than one component, then it is termed as global scope. In global type of change, the entire system required to commit the adaptation, involvement of the entire system required to commit hence impact of change is mitigated to be reduce on the entire system.

Duration: Refers a time period in which the system is self-adapting, as well as the time duration of adaptation carried on. There are three types of adaptation process duration, Short, Medium and Long. This time a characteristic depends on application domain. The application domain defines the exact time duration, application domain also define a minimal time associated with specific change or its dependencies on system life. Normally short duration is measured in seconds. Medium time duration is measured in minutes, and longtime duration is measured in hours.

Timeliness: This dimension makes surefire the time period to accomplish self-adaptation. The timeline dimension makes detentions to be best-effort in time period range. If change accrues, it is quite-often, it must be sure that adaptation change are take place within the best time period, other it is possible that another change may be accrued before it. It provide guaranteed the finest effort for the timeliness connected with self-adaptation.

Triggering: This dimension recognizes how and when adaptation change occurs, and also initiates the adaptation triggers according to change, there is two adaptation change triggers for initiation adaptation-change first is the event –trigger and second one are the time trigger. It is more difficult to provide a mechanism of how or when change accrues, but it is possible that provide a mechanism of when change occurs it reaction should be reacted.

2.1.4 Effects Dimension

Effect is also a set of dimension in which we can capture the adaptation impact upon a specific system; it deals adaptation mechanisms, adaptation properties. This group of dimension in addition deals with seriousness of adaptation, as well as hew much a system is predictable, and what are the overheads connected with adaptation; either system is unaffected by anticipate changes or not.

Criticality: In this dimension captures the system failure impact. If the system fails in self-adaptation, then what are the impacts on the system? They are three possible ranges of system failure impact first harmless, means there is no serious impact on the system if self-adaptation fails. Second possibility is mission-critical, objective or goals not achieved but the system is safe. Third possibility is a very critical system may have loss of life, which is called safety-critical. Safety criticality level in a self-adaptive-system may lead to an accident.

Predictability: In this dimension, recognizes environmental consequences of self-adaptation are predictable or not. These consequences are time or value. Time related predictability defines the timelines to the adaptation contrivances, as well as also predicate the timeline association of the system. Predictability devises two sides either it is deterministic, or it is non-deterministic. Good adaptation requires deterministic behavior.

Overhear: Only negative impact adaptation is captured and calculate its effect on system performance. In adaptation failure, environmental consequences, there are two types of overheads first is insignificant, that there is no signifying impact on the system. System remained normal if failure accrued, this may reduce system performance, and system is not able to deliver its services only. However, second condition is threshing of the system in case of self-adaptation failure. In this case system failure and life, threat can occur. This type of overheads should be insignificant, and must have the capacity to avoid an obstacle. Resilience: Determines the importunity of service delivery, it must be justified and trusted, when adaptive system facing changes, there are two possible issues which resilience is under consideration. First issue when the system can service delivery, this is called resilient behavior. In second system can substantiate the provided resilience, this ability is called vulnerable resilience (Laprie, 2008).

2.2 Dimension Degree Modeling

Adaptation dimension provide the way of evaluating adaptive systems with respect to particular concerns of the adaptation process. Thus, dimension provides a measure to evaluate desirable properties. For instance, dimension to evaluate control systems measure aspects concerning the SASO properties (i.e. stability, accuracy, settling time, and small overshoot). To characterize the evaluation of adaptive systems, we analyzed the variety of self-adaptive software systems to identify adaptation properties (i.e. at the managed system and the controller) that were evaluated in terms of modeling dimension. Just as the evaluation of most properties is impossible by observing the controller itself, we propose the evaluation of these properties by means of observing quality attributes at the managed system. To identify relevant dimension, we characterized a set of factors that affect the evaluation of quality attributes. These factors are an essential part of the modeling dimension used to evaluate properties of both the controller and properties of the managed system Although these modeling dimensions are directly related to the measurement of quality factors, we expect that these dimensions are be useful for evaluating adaptation properties based on our proposed mapping between quality attributes and adaptation properties.

Dimensions		Degree		
Goals	Evolution	Static	Dynamic	
	Flexibility	Rigid	Constrained	Unconstrained
	Duration	Temporary	Persistent	
	Multiplicity	Single	Multiple	
	Dependency	Independent	Dependent	
Change	Source	External / Environmental	Internal / Application	Middleware/ Infrastructure
	Type	Functional	Non-Functional	Technological
	Frequency	Rare	Frequent	
	Anticipation	Foreseen	Foreseeable	Unforeseen

Mechanisms	Type	Parametric	Structural	
	Autonomy	Autonomous / System	Assisted / Human	
	Organization	Centralized	Decentralized	
	Scope	Local	Global	
	Duration	Short	Medium	Long
	Timeliness	Best Effort To Guaranteed		
	Triggering	Event-Trigger	Time-Trigger	
Effects	Criticality	Harmless	Mission-Critical	Safety-Critical
	Predictability	Non-Deterministic	Deterministic	
	Overhead	Insignificant	Failure	
	Resilience	Resilient	Vulnerable	

Table 2: Dimensions Degree Definition

These modeling dimensions measure the ability of a self-adaptive system to adapt. And argue that adaptively can be evaluated using a metameric named payoff which is defined in terms of performance metrics to measure the effectiveness of the adaptation process. That is, the optimal adaptive system is characterized by the fact that its adaptation decisions are always optimal (i.e. always yield the optimal payoff).

To apply their metric it is necessary to Identify the adaptation tasks, define one or more performance metrics on these tasks (i.e. these metrics should reflect the contribution of these tasks toward the adaptation goal), define a payoff metric in terms of the performance metrics, and to apply the metric by observing the performance of the system.

DISCUSSION

We map the state-of-the-arts techniques discussed in last sections and uncertainty dimensions. Now we find some state-of-the-arts of self-adaptive-systems and finds how they address our modeling techniques. To find out strong rest we also compare uncertainty addressing techniques with application either they fulfill the requirements or not.

Modeling Dimensions Vs State-of-the-art-techniques

We have identified and grouped most important modeling dimension to for self-adaptive-systems. This classification is used to check the system quality in existing techniques which address uncertainty in self-adaptive-systems (State-of-the-art-systems), this comparison provided a check list for physical properties covered by any existing techniques. After determining the scope of a technique we can identifies its application domains and system behavior where they are used. An acute comparison is given in Figure 1 & 2.

As shown in Figure 1 Rainbow can only express goal and mechanism uncertainty in self-adaptive system, RELAX and FLAGS are used to cover goal uncertainty but its level is limited in goal dependency, FUSION technique is used to address goals, changes and efforts. ADC have ability to address only for mechanism uncertainty, RESIST is valid for Goal, Change, Mechanisms, and POISED address goal and mechanism.

	Goal					Change				Mechanisms							Effects			
	Evolution	Flexibility	Duration	Multiplicity	Dependency	Source	Type	Frequency	Anticipation	Type	Autonomy	g n	Scope	Duration	Timeliness	Triggering	Criticality	v	Overhear	Resilience
Rainbow	■	■	■	■	■					■	■	■	■	■						
RELAX	■	■	■	■																
FLAGS	■	■	■	■																
FUSION	■	■	■	■	■	■	■	■	■								■	■	■	■
ADC										■	■	■	■	■	■	■				
RESIST	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■				
POISED	■	■	■	■						■	■	■	■	■	■	■				

Figure 1: Filled box shows that ability to address this particular dimension and blank box shows that they are not able to address this particulate dimension.

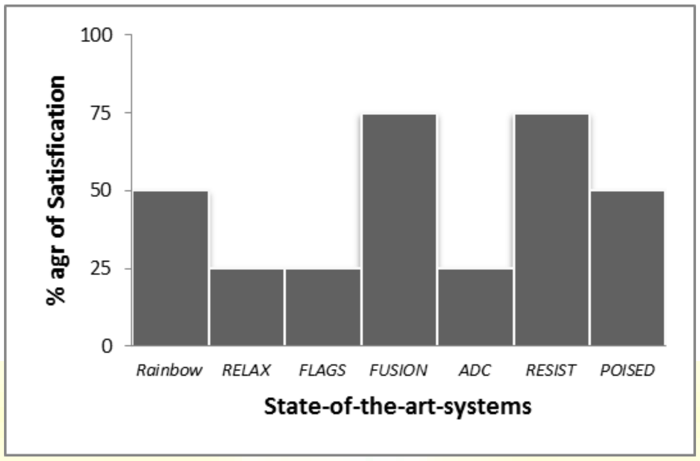

Figure 2: State-of-the-art techniques and their dimension handling capabilities. RAINBOW and POISED covers 50% uncertainty, in self-adaptive systems RELAX, FLAGS, ACD covers 25% of uncertainty. FUSION and RESIST covers 75% uncertainty in self-adaptive-system.

State-of-the-art Technique	Modeling Dimensions
Rainbow	Goal, Mechanisms
RELAX	Goal
FLAGS	Goal
FUSION	Goal, Change, Effects
ADC	Mechanisms
RESIST	Goal, Change, Mechanisms
POISED	Goal, Mechanisms

Table 3 State-of-the-art systems and their respective dimension which they able to cover to fulfill requirements.

For example Rainbow covers Goal and mechanisms but in mechanism dimension there are two dimensions which are not covered by Rainbow. In RELAX dimension goal can be defined but not fully one dimension i.e. goal Dependency is not cover by RELEX technique. RESIST technology coves all dimensions fully which are goal, change and mechanism.

It is noted that there is no one technique which covers all dimensions, when we propose these models for different applications. Then there may be some applications which are fully covered by these dimensions, but there are some applications which require more dimensions but one state-of-the-art-system cannot fully fulfill that application. So, in that case we can use combination of two or more technique to achieve full system goals in particular application.

SUMMARY AND FEATURE WORK

Uncertainty is a well-known challenge in the construction of dependable self-adaptive software. Deficiency of a coherent understanding of uncertainty creates hindered the development of suitable techniques to mitigate it. This paper we planned the address these issue by shedding light on the role of uncertainty in self-adaptive software and its distilling characteristics. In this paper we have, analyzed different state-of-the-art self-adaptive systems and their ability to address uncertainty in process of making adaptation decisions. In this paper we finds some common sources of uncertainty in self-adaptive software, illustrate these sources of uncertainty using modeling dimensions. This process provides the intuition behind the challenges posed by uncertainty in particular domain. Finding these sources of uncertainty provide a more elaborate definition of uncertainty by enumerating its characteristics in the context of prior literature. At the end of this phase we present modeling dimensions for better understanding the impact of uncertainty on self-adaptive software.

This research also present an overview of state-of-the-art techniques commonly used for representing uncertainty, as well as provide reasoning about it, and then categories the techniques as they target to address the different faces of challenges, impersonated by uncertainty.

As a result it is found that all sources of uncertainty have not the same characteristics. The modeling dimension classification presented in this research can also apply to any self-adaptive software system. This classification is useful in several different development situations. It can be used as a driver for traditional forward engineering, but also useful in a reverse engineering context where engineers comprehend the existing solutions.

In future to prove our dimension-modeling we fit these dimensions for different self-adaptive system's application and intend to leverage the proposed classification model, which allows for systematically identifying the variations among different self-adaptive software systems applications.

REFERENCES

Alain. A, W. James, P. Moore, Bourque, and R. Dupuis. 2004. Guide to the software engineering body of knowledge. Technical report, IEEE Computer Society: 1(1): 1-15.

Andersson, J. and R. Lemos. 2009. Towards a classification of self-adaptive software system. International Journal,Software Engineering for Self-Adaptive Systems.LNCS:1(1):55-25.

Autili, M. Baclawski, and Y. A. Eracar. 2011. Towards self-evolving context-aware services. In Proceeding of the DisCoTec CAMPUS'11,:1(1):1-25.

Baresi, L. and L. Pasquale. 2010. Fuzzy goals for Requirements-Driven adaptation. Proceeding International Requirements Engineering Conference, held in Sydney, Australia : 1(1): 125-134.

Bigus, J. P., D. A. Schlosnagle, J. R. Pilgrim, W. N .Mills, and Y. Diao, 2002. Able: A toolkit for building multivalent autonomic systems. IBM Systems Journal:41(3): 350-371.

Buckley, J., T. Mens, Zenger, M. Rashid, and G. Kniesel, 2005. Towards a taxonomy of software change. Journal on Software Maintenance and Evolution: Research and Practice : 1(1):309-332.

Bruni, R., A. Corradini, A. Lluch, F. Gadducci and A. Vandin. 2012. A conceptual framework for adaptation. In: Proceedings of 15th the International Conference on Fundamental Aspects of Software Engineering (FASE'12). LNCS, Springer. Available at http://eprints.imtlucca.it/1011/

Cheng, B.H. and D. L. Lemos. 2009. Software engineering for Self-Adaptive systems: A research roadmap. Proceeding of, conference in Software Engineering for Self-Adaptive Systems : 1(1):15-26.

Cheng, B.H. and P. Sawyer. 2009. Fuzzy goals for Requirements-Driven adaptation. Proceeding International Requirements Engineering Conference , held in Sydney, Australia. : 1(1): 125-135.

Chou, P., P. Li, K. Chen and M. J. Wu. 2010. Integrating web mining and neural network for personalized e-commerce automatic service. International Journal of Expert Systems with Applications, 37(4). 2898-2910.

Cooray, D. and D. Kilgore. 2011. RESISTing reliability degradation through proactive reconfiguration. Proceeding of International Conference on Automated Software Engineering, held in Antwerp, Belgium : 1(1): 24-54.

Coulson, G., G. Blair, P. Grace, A. Joolia , K. Lee, and J. Ueyama. 2009. A generic component model for building systems software. ACM Transactions on Computer Systems: 45(1): 33-38.

Dowling. J and V. Cahill. 2004. Self-managed decentralized systems using k-components and collaborative reinforcement learning. In Proceedings 1st ACM SIGSOFT Workshop on Self-

Managed Systems, (WOSS ’04), New York, NY, USA, ACM: 1(1):39–43.

Elkhodary, A. and N. Esfahani. 2010. FUSION: a framework for engineering Self-Tuning Self-Adaptive software systems. Proceeding International Conference on the Foundations of Software Engineering, held in New Mexic: 1(1):6-7.

Esfahani, N. and E. Kouroshfar. 2011. Taming uncertainty in Self-Adaptive software. Presented in joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, held in Hungary.

Esfahani, N. and E. Kouroshfar. 2011. Uncertainty in Self-Adaptive Software Systems. Published by Department of Computer Science George Mason University.

Floch, J., S. Hallsteinsen, E. Stav, F. Eliassen, K. Lund, and E. Gjorven. 2006. Using architecture models for runtime adaptability. IEEE Software, 23 March 2006: 1(1):62–70.

Garlan, D. and B. Schmerl, 2002. Model-based adaptation for self-healing systems. . Proceeding conference of Workshop on Self-healing Systems: 1(1): 27-32.

Garlan, D. and S. W. Cheng. 2004. Rainbow: Architecture -Based Self-Adaptation with Reusable Infrastructure. International Journal, IEEE Computer Systems: 3(1):46-54.

Hellerstein J. L., Y. Diao, S. Parekh, and D. M. Tilbury. 2004. Feedback Control of Computing Systems. John Wiley & Sons.: 1(1):25-50.

Hinchey, M. G. and R. Sterritt, 2006. Self-managing software. IEEE Computer:39(1): 107-109.

Horn, P. 2001. Autonomic computing: Journal, IBM's perspective on the state of information technology. Available at www.research.ibm.com/journals.html

IBM-AC 2001. Autonomic computing 8 elements. Available athttp://www.research.ibm.com/ autonomic/overview/elements.html.

Jackson, M. 1997. The meaning of requirements. Annals of Software Engineering: 3(1):5-21.

Kephart, J. O. and W. Walsh, 2004. An artificial intelligence perspective on autonomic computing policies. In Proceeding conference of IEEE int. workshop on Policies for Dist. Systems and Networks:1(1): 3-13.

Kokar, M. , Baclawski, and Y. A. Eracar, 1999. Control theory-based foundations of self-controlling software. IEEE Intelligent Systems :14(3): 37-45.

Kramer, J. and J. Magee. 2007. Self-managed systems: an architectural challenge. In: Future of Software Engineering:1(1): 259–268.

Laprie, J.C. 2008. From dependability to resilience. In: International Conference on Dependable

Systems and Networks (DSN 2008), Anchorage, AK, USA :1(1): G8–G9.

Liu, H., Parashar, M., and S. Hariri, 2004. A component-based programming model for autonomic applications. In Proceeding conference on Autonomic Computing :1(1): 10-17.

Marzo S., G. Karageorgos, A. Rana, and O.F. Zambonelli. 2004. Engineering Self-Organising Systems: Nature-Inspired Approaches to Software Engineering. Lecture Notes in Computer Science, Springer :2977(1).

McCann, J. A., Lemos, R. D., Huebscher, M., Rana, O. F., and Wombacher, A. 2006. Can self-managed systems be trusted? some views and trends. Knowledge Eng. Review:21(3): 239-248.

Murch, R. 2004. Autonomic Computing. Prentice Hall press: 1(1): 234-245.

Nagone, S., B. Kapse and M. Bhagwat 2011. E-Commerce Application using Web API and Apriori Algorithm of Data Mining. Second National Conference on Information and Communication Technology. Proceedings published in International Journal of Computer Applications:1(1): 8-10.

Parashar, M. and S. Hariri, 2005. Autonomic computing: An overview. Hot Topics, Lecture Notes in Computer Science:3566(1):247-259.

Pawlak, R., L. Seinturier, L.Duchien, and G. Florin. 2001. JAC: A flexible solution for aspect-oriented programming in Java. In Proceeding of Metalevel Architectures and Separation of Crosscutting Concerns:1(1):1-24.

Parekh, S., N. Gandhi, J. Hellerstein, D. Tilbury, T. Jayram, and J. Bigus. (2002) Using control theory to achieve service level objectives in performance management. Real-Time Systems, 23 July 2002:127–141.

Poladian, V. and J. P. Sousa. 2007. Dynamic Configuration of Resource-Aware Services. Proceeding International Conference on Software Engineering, held in Scotland:1(1):604-613.

Reinecke, P. K. Wolter, and A. van Moorsel. 2010. Evaluating the adaptively of computing systems Special Issue on Software and Performance. Performance Evaluation: 67(8): 676–693.

Salehie, M. and R. Asadollahi. 2009. Starmx: A framework for developing self-managing java-based systems. Presented in ICSE workshop on Software Engineering for Adaptive and Self-Managing Systems :1(1):24-56.

Sharma, A., S. Kumar and M. Singh. 2011. Semantic Web Mining for Intelligent Web Personalization. Journal of Global Research in Computer Science :2(6):77-81.

Salehie, M. and T. Ladan. 2009. Self-Adaptive Software: Landscape and Research Challenges.

Published in ACM Transactions on Autonomous and Adaptive Systems: V(1): 11-15.

Shang, S. W. and D. Garlan. 2007. Handling uncertainty in autonomic systems. Presented in International Workshop on Living with Uncertainty, held in Atlanta, Georgia. :1(1).

Sterritt, R., M. Parashar , H.Tianfield, and R. Unland, 2005. A concise introduction to autonomic computing. Advanced Eng. Informatics:19(1): 181-187.

Solomon , A , M. Litoiu, J. Benayon, and A. Lau. (2010) Business process adaptation on a tracked simulation model. In Proceedings 2010 Conference of the Center for Advanced Studies on Collaborative Research, (CASCON '10). ACM, 2010 :10(1):24-56.

Tianfield, H., 2003. Multi-agent autonomic architecture and its application in e-medicine. Proceedings of the 2003 IEEE/ WIC International Conference on Intelligent Agent Technology (IAT'2003), Halifax, Canada, 13–16 October 2003 :1(1):601–604.

Whittle, J. and P. Sawyer. 2009. A Goal-Based modeling approach to develop requirements of an adaptive system with environmental un-certainty. Proceeding of International Conference on Model Driven Engineering Languages and Systems, held in Denver, Colorado:1(1):468-483.