ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

FUSION: Software System to Model Preferences of Multiple Users

K.Hussenvalli[1], Dr. S. Vasundra[2]

Dept. Of Computer Science & Engineering, JNTUACEA, Anantapuramu, Andhra Pradesh, India[1]

Professor, CSE Department, JNTUACEA, Anantapuramu, Andhra Pradesh, India[2]

Abstract: In present days software systems have turned to Self-adaptive software systems, which are accomplishing quality of service goals and uncertainty of their environments. An unexpected changes at runtime that disobey the assumptions made about the interior structure of the system could degrade the correctness of the adaptation decisions. In several real-world systems, to come pending a goal violation occurs and then reacting to it may be very costly. In addition, presently Self- adaptive software systems think an only stakeholder, with single utility function. In this paper, a mechanism called FeatUre-oriented SelfadaptatION (FUSION) framework to extend model multiple users preferences in terms of multiple utility function, and also we identify the vital challenges in self-adaptation decision must tackle to enable correctness of Learning adaptation decisions.

Keywords: Model preference, automatic computing, feature-orientation, Resource Adaptation, Self-adaptive Systems, Software Architecture, product line literature

1. INTRODUCTION

In the present software systems, increasing complexity and uncertainty and their environments, software engineers have turned too self-adaptive. The Self-adaptive systems [4] are expert up-and-coming requirements of continuously changing environment and that may be unidentified at design-time. Still different challenges remain even many researchers have made significant progress with frameworks and methodologies that intention the development of self-adaptive systems. The modern day in engineering self-adaptive software systems is to make use of a component-and connector view (architectural representation) of the adaptation decisions. It require internal arrangement of the managed system, this is refer as White box approach (architecture-based adaptation). It is faced with the following problems:

1. **Theory drifts**. In this technique, may not tolerate runtime changes, which means simplifying assumptions certain properties of the internal structure of the unexpected adaptation decisions inaccurate.
2. **Inter-Dependencies.** To design adaptive systems controllable, preponderance of the adaptation can be internal structural changes accepted independently.
3. **Effectiveness**. The effectiveness of study and preparation is principally expensive.

We present a black-box approach [1] for engineering self-adaptive systems. It means that the adaptation decisions are complete using abstractions that do not want knowledge of the internal structure of the software system. In this approach results in a clear separation of models used for goal management and those used for change management. The approach brings about three introduce for solving the aforementioned challenges: (1) The self adaptive software systems that includes bridges for management architectural mismatches and new method of modeling on the notion of feature-orientation from the product line literature [2]. 2) A new method of assessing and investigation about adaptation decisions through middleware that supports access to web services. 3) It domain skilled knowledge, represented in feature-models turn improves the exactness and helpfulness of adaptation decisions. The outcome of this approach, entitled FeatUre-oriented Self-adaptatION (FUSION) [3], which combines feature-models with web services.

The FUSION framework key assistances are as follows:

1. FUSION accommodates dynamics of the system, even those that are unexpected at Architecture level, through incremental inspection and orientation.
2. FUSION can be able of ensure stable performance and keep system goals in and after adaptation.
3. A FUSION uses features and inters feature dealings to considerably decrease the configuration space of a sizable system, building runtime study and learning practicable.

2. BACKGROUND

2.1 Fusion Overview

In below Figure1 shows the framework as it adapts a running system collected of a number of features. We suppose up-and-downs in the running systems are the intellect that features would be “select” and “deselect” on engage. FUSION modifies the feature selections to resolve Quality of Services tradeoffs and satisfy as much goals as achievable.

FUSION adaptation cycle makes adaptation decisions using a continuous loop, collects measurements (Metrics)

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

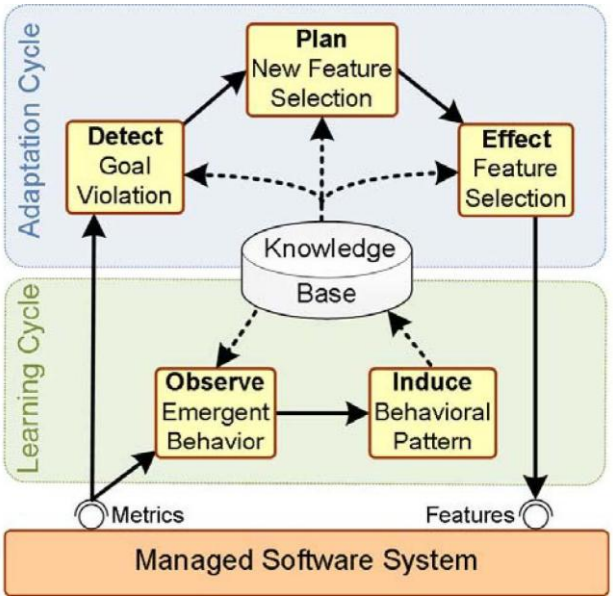

Figure1 overview of the FUSION framework.

and Supplementary the system via hanging three actions in the subsequent series:

❖ Identify achieved utility (i.e., measure of user's fulfillment), based going on the metrics derived from the running system, to elect if a goal violation has occurred.

❖ Once a goal is violated, Plan discovery for an optimal feature selection that maximizes the in general system utility.

❖ Effect determines a set of enabling/disabling of features for given a new feature selection, to minimize destabilize that unhelpfully collision the system's goals.

In learning cycle (represent in Fig. 1) to study the collision of adaptation decisions in terms of feature collection on the system goals. The learning cycle occurs before the system's initial production. The system is moreover replicated in offline mode and metrics equivalent to each feature selection are together. This metrics is damaged to instruct FUSION to induce a preface model behavior of the systems. At runtime, the FUSION learning cycle constantly executes, and as the dynamics of the system and its environment modify, the framework tunes itself. The learning cycle collects such indicators and tunes itself by executing the following two activities in sequence:

❖ Based on the collected measurements commencing the system, Observe detects any up-and-coming patterns of behavior. An evolving pattern is detected when predictions set wrong expectations (i.e., inaccurate forecast of the impact of adaptation on utility).

❖ Induce learns the new behavior by applying web services recently collected data and stores a refined model of the behavior in the knowledge base, which is then used to make (more) informed adaptation decisions in future cycles.

The knowledge base provides input to the both adaptation and learning cycle. It is stores Quality of Services goals, all the models concerning to the managed system, including feature selection, and functions relating Metrics.

2.2 Fusion Model

This is modeling methodology centerpiece of this approach. FUSION enables feature-oriented models to learning effectiveness and identifies key factors in the self adaptive software system that affects the system goals.

2.2.1 Feature-Oriented Adaptation

In FUSION, a feature is a piece of adaptation.
A feature provided by the system abstraction of a capability. A feature is conventionally used for the period of the requirements phase to model a variation point in the software system [4]. The Figure2 shows how to select the features in a system.

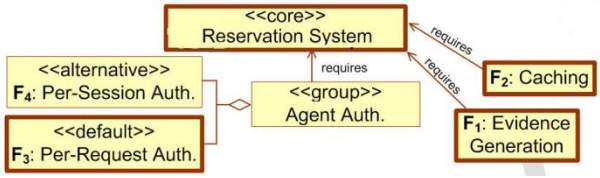

Figure2 the features enable, disable adaptation, where selected features thick borders are selected.

The above Figure2 represents in other manner, feature model is used to recognize the current system pattern in terms of a feature selection binary value. In a feature selection binary value enabled features are set to "1"; disabled features are set to "0".

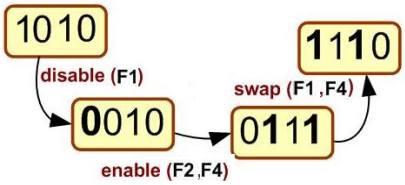

Figure3 the feature are enable, disable adaptation.

For example, In RS would be "1010", which way that all features from Figure3 are enabled except Per-Request Authentication. The adaptation of a system is modeled as a transition from one feature selection binary value

2.2.2 Goals

In FUSION, the functional or Quality of Services objectives for a particular execution scenario are set to be a goal. A goal is nothing but a metric and a utility. A measurable quantity (e.g. response time) is metric this is retrieve from running system. The user's preferences (satisfaction) express in terms of utility functions. FUSION seats one restriction on the variety of service functions: The metric value not acceptable the return utility value zero, when utility is less than or equal FUSION takes as violation of the initiates adaptation associated goal.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

2.2.3 Implications of Feature-Oriented Adaptation

In FUSION, the features offer as the edge between the adaptation logic and the managed system. In this approach reduces the adaptation space by using feature-oriented white-box approach, it operates on fully architecture independent manner. For example, the system with N different ways of authentication protocols, D components, which may be executing on P different production environments that is represent in terms of equation as follows

$(D^P \text{ ways deployments})^{N \text{ different ways of authentication}} = D^{NP}$ possible configurations

3. PROPOSED SYSTEM

3.1 Over view

The proposed system extends FUSION framework to accommodate with multiple users' preferences in terms of multidimensional utility functions. Multiple users make stateless requests [5] to the managed system, as Figure4 shows. In Managed system components are implemented in Java and provide remote method invocation interface for the effective adaption decision for all users. Each user connected to the managed system. The managed systems create an object for software system at the time of deployment, and then for every user request the managed system create a thread object for each user. We suppose up-and-downs in the running systems are the intellect that features would be "select" and "deselect" on engage for each user. FUSION modifies the feature selections to resolve Quality of Services tradeoffs and satisfy as much goals as achievable.

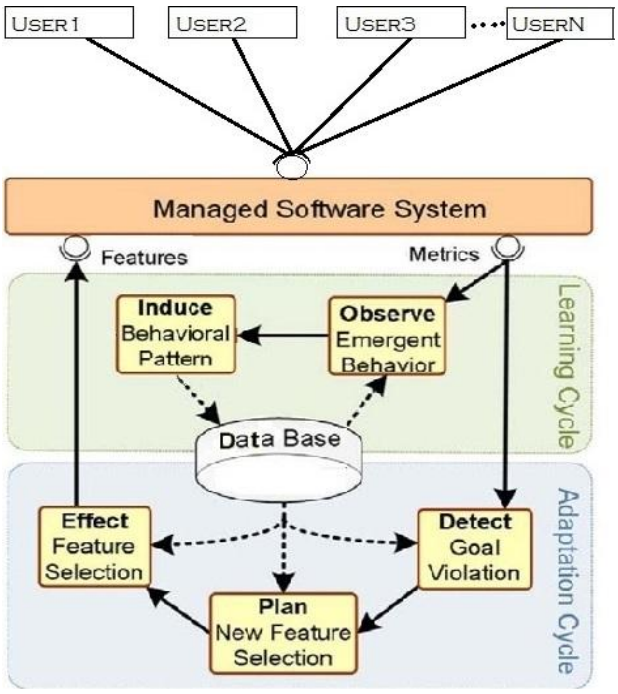

Figure 4 Overview of Extend FUSION framework for multiple users

3.2 FUSION Adaptation Cycle

FUSION extended adaptation cycle makes adaptation decisions using a continuous loop, collects measurements for each user (Metrics).

FUSION adopts a core view, which we consider to be the most reasonable, and achieves the subsequent objectives are decrease interruption, capable analysis and secure.

3.2.1 Detect

The adaptation cycle is initiated when detect determines goal violations to achieve utility functions for each user. A utility function serves for : (1) when the metric values are unacceptable, returns zero to indicate a violated goal, and (2) when the metrics satisfy the minimum value i.e. , returns a positive value less than one to indicate the user's preference for improvement.

3.2.2 Plan

FUSION relies on the Generic procedure to reach the multiple users' adaptation objectives:

❖ Here we use the knowledge base to eliminate all of the features with no significant impact on the goal. We consider the list of features that affect a given goal Shared Features

❖ Shared Features represents the adaptation parameters and they can even affects other goals, called to be the Conflicting Goals, Here we use the knowledge base to detect the conflicts using backtracking the learned functions

3.2.3 Effect

Once the Plan activity had been found a new feature selection, it is passed to Effect for placing the system in the target configuration. Effect is responsible for choosing a path containing several adaptation steps whether enables/disables the features, toward the new feature selection. We present a novel algorithm based on A* search algorithm, that uses the learned knowledge to find a path that altogether eliminates, and if not possible minimizes the extent of, goal violations during the adaptation process. Effect a heuristics based search algorithm that finds a suitable adaptation path.

3.3 Fusion Learning Cycle

FUSION copes with the changing dynamics of the system through learning process. Learning process discovers relationships between features and metrics. Each relationship is represented as a function that quantifies the impact of features, along with any other relevant contextual variables, on a metric.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

3.3.1 Observe

Observe starts the learning cycle. Observe is a continuous execution of two activities: 1) normalizing raw metric values to make them suitable for learning, and, 2) test the accuracy of learned functions. We describe each of these activities.

3.3.2 Induce

Induce constructs several functions that estimate the impact of making a feature selection on the corresponding metrics at a given execution context. Induce executes two steps. The first step is a significance test that determines the features with the most significant impact on each metric. This allows us to reduce the number of independent variables that learning needs to consider for each metric (also known as feature extraction). After the significance test, we apply the learning, which derives relationships between metrics and features using the normalized observations.

The Database provides input to the both adaptation and learning cycle. It stores each user details in tables in the database, and also provide
quality of Services goals, all the models concerning to the managed system, including feature selection, and functions relating Metrics.

4 PERFORMANCE ANALYSIS

We simulate extended FUSION framework for multiple users in Struts MVC framework on top of Java platform on IDE My Eclipse5.0. We estimate system performance by using system characteristics like workload known at run time it supports for larger scale adaption and support for multi user scheme, it also uses feature-oriented representations to model variability in the system and its context. Extended FUSION adopts the similar modeling methodology. However, in addition to this, features are units of runtime learning and reasoning in Extended FUSION also. FUSION itself fits in the goal management layer. The change management [9] layer is realized on top of XTEAM [7] an extensible architectural description and analysis environment. The component control layer is realized on top of Prism-MW [8]—a middleware environment aimed at architecture-based software development.

5 RESULTS

Tables are used to measure the induced function parameters and their performance improvement. These approaches improve Normalization process to multiple users' capture the observation records using studentized residual [6] as follows in the Table1, it represent the improvement in leaning metrics functions for the self adaptative systems.

Table1 Learning Metric Factions for users

Significant Variable	Induced Functions				
	UM_{G1}	UM_{G2}	UM_{G3}	UM_{G4}	..
Core	0.124	0.161	1.432	0	..
F1	1.654	1.145		2	..
F2	…	..	..	..	..
F3		0.672			
F4				1	
F5				4	
F6				0.244	
F7	0.163				
F1F3			0.534		
…	…	…	…	…	…

This Figure 5 represents performance of FUSION's path search algorithm in terms of the execution time. FUSION also takes into consideration the objective of minimizing utility loss during adaptation, which is ignored in the FC and K+FC approaches.

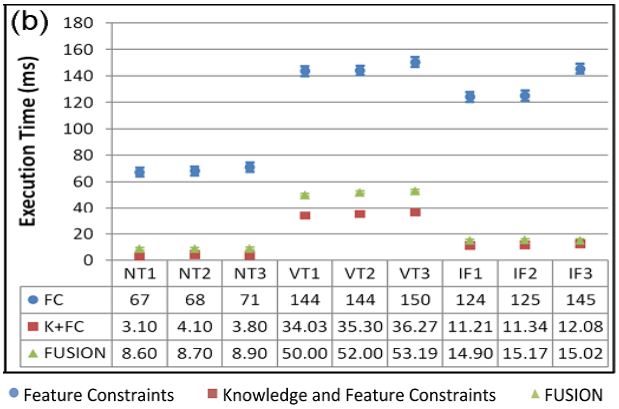

Figure5 the result of path searches for different execution time.

6. CONCLUSION

We presented approach for engineering self adaptive systems that brings about two innovations for solving the aforementioned challenges a new method of modeling and representing a self-adaptive software systems that builds on the notions of feature-orientation, assessing and reasoning about adaptation decisions.

Extended FUSION framework to extend model multiple users preferences in terms of multiple utility function, and also we identify the vital challenges in self-adaptation decision must tackle to enable correctness of Learning adaptation decisions.

REFERENCES

[1] "FUSION: A Framework for Engineering Self-Tuning Self-Adaptive Software Systems," by A. Elkhodary, N. Esfahani, and S. Malek, Proc. Int'l Symp. The Foundations of Software Eng., pp. 7-16, 2010.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

[2] H. Gomaa, "Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures", illustrated ed. Addison- Wesley Professional, 2004.
[3] Naeem Esfahani, Ahmed Elkhodary, and Sam Malek "A Learning-based Framework for Engineering Feature-Oriented Self-Adaptive Software Systems" IEEE Transactions On Software Engineering, VOL. 39, NO. 11, November 2013.
[4] M. Salehie and L. Tahvildari, "Self-Adaptive Software: Landscape and Research Challenges," ACM Trans. Autonomous Adaptive Systems, vol. 4, no. 2, pp. 1-42, May 2009.
[5] D. Garlan, S.W. Cheng, A.C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure," Computer, vol. 37, no. 10, pp. 46-54, Oct. 2004.
[6] C. Chatfield, The Analysis of Time Series: An Introduction, sixth ed. Chapman and Hall/CRC, 2003.
[7] G. Edwards, S. Malek, and N. Medvidovic, "Scenario-Driven Dynamic Analysis of Distributed Architectures," Proc. Int'l Conf. Fundamental Approaches to Software Eng., pp. 125-139, 2007.
[8] S. Malek, M. Mikic-Rakic, and N. Medvidovic, "A Style-Aware Architectural Middleware for Resource-Constrained, Distributed Systems," IEEE Trans. Software Eng., vol. 31, no. 3, pp. 256-272, Mar. 2005.
[9] J. Kramer and J. Magee, "Self-Managed Systems: An Architectural Challenge," Proc. Int'l Conf. Software Eng., pp. 259-268, 2007.