User adaptive system research paper samples

Engineering

Abstract

User adaptive system is everyday becoming an important activities, as users are always expecting intellectual services from them. What everyone should know and understand on the user adaptive system is the user model. Old devices, which are still learning how to create this user model, are mostly too rigid to understand the uncertainty of different people. Computing methods can be easily used to handle and even process human uncertainty as well as stimulating human decision-making. This context explains how soft computing methods, including neural networks, fuzzy clustering, fuzzy logic and genetic algorithms can be used. For different techniques, applications, future directions and limitations are presented. There is also guidelines which show soft computing techniques can be used following the task implemented by the app.

Introduction

Adaptive systems should adopt a user-centered approach, as the users are usually the main source of information, with the main target being that of the application. In other words, evaluation of a regular interactive system should take place in the entire design life cycle. It also provides a feedback for a design modification. For any evaluation of the system, it should be performed before any implementation of the system to avoid any expensive design mistakes. Moreover, in evaluation of an adaptive system, difficulties can be caused by the need to distinguish various adaptation aspects. Evaluating them differently avoids any complications. This report presents a comprehensive overview of techniques and methods that can be used for the evaluation of adaptive systems.

Majority of the techniques are usually used in evaluation for adaptive systems. Some of them are not properly designed such as controlled experiments hence they do not produce significant results. It is for this reason that methodologies are proposed in a detail manner. Other techniques are disregarded like the task analysis, systematic observation and ethnographic. They should be considered as they truly offer interesting results.

Privacy laws and self-regulatory principles

Privacy laws regulate the processing of personal data. These laws differ considerably and they are still flux. Any information about the current user is kept and processed in a modeling system falls within the scope of personal data. It is defined as privacy laws. For any data to be linked to identifiable person, it is necessary that the user to fully identify themselves. Self-regulatory privacy principles control the processing of personal data on the level of an industry sector. The platform for privacy preferences protocol (regale and cranor 1999) allows website to express and communicate their data collection practices.

Software engineering for adaptive and self-managing systems.

An important requirements for software intensive systems is its ability to self-manage. This is by adapting at the run time for handling such things as changing the user needs, resource variability and faults. Such system configure and reconfigure itself, optimize itself and continually tune, recover and protect itself. The topic of self-adaptive system has been studied in various application areas; software architectures, biological computing, programming languages and control systems.

Engineering self-adaptive system through a feedback loops.

A self-adaptation is a software intensive system that comes in many different guises. What all self-adaptive have in common is the design decisions that are moved towards runtime. This is to control dynamic behavior and that of individual systems, its state and of the environment. Keeping the web services running for a long time requires collection of information that reflects the current state of the system. Feedback loops usually provides generic mechanism for self-adaptation. A positive feedback takes place when the initial change is reinforced, leading to amplification of change. On the other hand, negative feedback triggers a response that counteracts a perturbation. Generic feedback loop typically involves activities like; collect, analyze, decide and act. Environment sensors and user context

Inform users and administrators

Game theory

Decision theory

Planning inference

Sensors or probes collect data from executive system with context on its current state. All accumulated data are cleaned, filtered and pruned. They are then stored for future reference in order to portray accurate model for current and past states. It is usually referred as the autonomic control that focuses on activities which recognizes the feedback. Firstly, the cycle starts with the collection of all relevant data from the environment and other sources that reflect the current state. Next, the system analyzes the collected data. There are many approaches used to collect data. The other step is decision, where a decision must be made about how to adapt the system to reach a desirable state. The final stage includes implementation of the decision. The system must act through available effectors.

Solutions, which are inspired by an explicit control

The autonomic element, which was introduced with IBM's architectural blueprint for an autonomic computing. It is the first architecture for self-adaptive system. In order to realize an autonomic system, a designer must compose arrangements of collaborating autonomic elements working towards a common goal. IBM uses autonomic element like fundamental building blocks for realizing self-healing, self-optimizing, self-configuration and self-protecting systems. An autonomic element consists of an autonomic manager and managed element with a feedback control loop. Therefore, the autonomic manager and the managed element correspond to the controller and to its process in a generic feedback loop.

An autonomic element itself can be a managed element. This means there will have to be an additional sensors and effectors at the top of the autonomic manager. They are usually used to manage the element by providing measurements of the sensors and receiving control input. A good example for a self-adaptive system following MIAC scheme is the robust feedback loop. It is used in a self-optimizing to become prevalent in a performance turning and resource provisions scenarios. Robust feedback control always tolerates all incomplete knowledge about the system model. It usually assumes that the system model has to be rebuilt frequently. This is accomplished through an estimator that estimates the state of variables, which cannot be observed directly. Another good example of an MIAC scheme is mechatronics system that consists of a system of autonomics shuttles. It operates on demand and in a decentralized manner using wireless network. To realize such mechatronics system is necessary to draw techniques offered by the domains of a control engineering and software engineering.

ULS system might be included in user adaptive mechanisms developed independently by various working teams. This enables users to solve different classes of problems at different abstraction levels. From complexity of both systems and development, processes may result in the impossibility of coordinating the user adaptive mechanism by design. It may also result in unexpected interactions with negative effects on the overall system behavior. One of the essentialstep is by making feedback loops visible toward the design of distribution coordination mechanisms. This can prevent undesirable system characteristics like the various forms of instability and divergence. All of this is because of competing self-adaptive systems.

Solutions that are inspired by natural systems

In this section, we present the existing nature inspired solutions for self-adaptive software systems. As some work, deal with building biologically inspired self-adapting software system. It is a challenge to employ a biological knowledge to get an understanding of how to build software system. In nature, the process of crystal growth can result in a well-formed regular crystals or even irregular crystals. One of the key aspects that determine which type of crystal will forms, is the speed. The tile architectural style, its software inspired by crystal growth. Shows the feedback exhibited by crystal growth allows a fault and adversary tolerance. The tile style allows distribution computation of NP-complete problems on a larger network in a dependable, secure and scalable manner. Loops within tile style systems are very difficult to classify as positive or negative. However, they do fit nicely into the feedback loop. The individual components attach to one another, collecting information on what other components may attach to one another.

Mammalian immune system provides a defense mechanism by detecting foreign materials. This is by coordinating a collective decentralized response to destroy them. These distributes, decentralizes system balance detecting and removing all malicious agents against interference. This is with the normal cell processes as well as employing learning techniques. Software intrusion with detection research already leverages immune system ideas. This offers much more insight into engineering self-adaptive systems. Engineering self-organizing systems encompass the use of autonomous components.

Challenges

Understanding and reasoning on how to control loop is a key for advancing construction of self-adaptive systems. It is from trial and error endeavor towards a more disciplined approach. To achieve all goals some issues have to be addressed.

Architecture and design; a control theory found that a system with a control loop are easier to reason about than a system with multiple loops. Even after this the latter are far more common. Because of this good engineering practice uses simple design. It is the responsibility of every engineer to minimize the number of control loops or decoupling control loops from each other. In case decoupling is not possible, then the designer must make control loop interactions. Designs containing multiple control loops must select, consider and analyze different designs. It was also identified that hierarchical organization of control loops reduces the design complexity. This implies that loops influence each other top-down, operating at different time scales. Reference architectures for adaptive systems should highlight key aspects of feedback loops.

Modeling; there should be modeling support to make the control loop explicit exposing user adaptive properties. A good model has to capture what can be observed and be influenced easily. It is desirable to have a widely agreed upon standard for user adaptive systems that includes the control loops. Nature of self-adaptive system, which requires to reify properties that would be encoded implicitly. Reified properties need to be modeled appropriately that can be queried and is modified during runtime.

Maintenance; it constitute a significant portion of a software system’s life cycle. To understand maintainability concerns specific to a self-adaptive system poses a very important challenges. Issues that should be tackled are how maintainability concerns of self-adaptive system. Reengineering, issues for self-adaptive systems are approach from perspective of Greenfield development. A big number of legacy applications can benefit from user adaptive features. Reengineering of existing systems with goal of making them more self-adaptive in a cost effective and principled manner poses an important challenge.

Conclusions

It has been outlined that feedback loops are a key factor in software engineering in a user adaptive system. In case a system is inspired by biological and natural system, identifying the feedback loops and understanding their impact is very important. It is very important that all approach for the models, design, architectures, maintenance and implementation to be executed.

References

1. Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Thomas, F., Knight, J.,

Nagpal, R., Rauch, E., Sussman, G. J., Weiss, R.: Amorphous computing. Commu-

nications of the ACM 43(5), 74–82 (2000)

2. Diao, Y., Hellerstein, J. L., Parekh, S., Griffith, R., Kaiser, G., Phung, D.: Control

theory foundation for self-managing computing systems. IEEE Journal on Selected

Areas in Communications 23(12), 2213–2222 (2005)

3. Di Marzo-Serugendo, G., Gleizes, M. P., Karageorgos, A.: Self-organisation in MAS.

Knowledge Engineering Review 20(2), 165–189 (2005)

4. Brun, Y., Medvidovic, N.: Fault and adver

sary tolerance as an emergent property of

distributed systems' software architectures. In: 2nd ACM International Workshop on Engineering Fault Tolerant Systems (EFTS 2007), Dubrovnik, Croatia, pp.

38–

43 (2007)