Managing Dynamic Context to Optimize Smart Interactions using Feedback Loops and SOA Governance Techniques

Hausi A. Müller
University of Victoria
hausi@cs.uvic.ca

Abstract
With the rapid growth of socio-technical ecosystems, smart interactions and services will proliferate and permeate every walk of life. Smart services are context-aware—are able to detect its current state or context and determine what interactions to perform or how to tailor services based on its context. In this endeavour, the capability of the system to adjust its behaviour in response to its perception of the environment and the system itself in the form of fully or semi-automatic self-adaptation has become one of the most promising research directions. Thus, smart interaction platforms need infrastructure to acquire, compose, and distribute context information to multiple execution endpoints. Moreover, context management must be controlled and governed to optimize end-to-end system properties. This paper posits that dynamic context acquisition and management can be orchestrated using feedback loops and SOA governance techniques.

1. Introduction
Smart Interactions deal with factors that impact the discovery, aggregation, and delivery of web resources and content related to users and their tasks [Ng2009]. Smart services provide the computing infrastructure to support Smart Interactions. The combination of smart interactions and smart services feed into the notion of a Smart Environment, Smart Internet, or even Web 2.0.

With the rapid growth of socio-technical ecosystems [ULS2006] including the many facets of the interactive, dynamic, semantic web, ubiquitous computing environments, sensor networks or computer-supported collaborative work (CSCW) systems, smart environments will proliferate and permeate all levels of society and every walk of life. Managing the complexity and controlling the operation of these smart, dynamic, decentralized, distributed computing systems presents formidable and exciting research challenges.

From a smart environment, we expect that it is able to detect its current state or context and determine what interactions to perform or how to tailor services based on its context. A classic definition of context is “any information that can be used to characterize the situation of an entity where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including users and applications themselves” [Dey2000b]. In this endeavour, the capability of the system to adjust its behaviour in response to its perception of the environment and the system itself in the form of fully or semi-automatic self-adaptation has become one of the most promising research directions.

In the web services domain, techniques that enable the exploitation of contextual information in services are typically referred to as context handling techniques [Kapitsaki2009]. Effective context handling provides dynamic service behaviour and content adaptation.

Context-awareness for interactions or services comes in many different forms and guises in smart environments. In the web service domain, context typically refers to all information about the client of a web service to adjust execution and output to provide customized and personalized behaviour (e.g., location, time, date, country, local customs, or personal preferences) [Keidl2004]. In general, context can refer and apply to multiple service execution points (e.g., Google remembers whether you are logged in or not, even when you move between offices connecting to different wireless networks). The 2007 Web Services Context Specification (WS-Context) standard provides a definition, a structuring mechanism, and service definitions for organizing and sharing context across multiple service execution endpoints [WSC2007].

However, there is more to context handling and management than hierarchical organization and XML communication protocols. In particular, how is context acquired (e.g., through sensors or profiles) or input (e.g., by end-users), distributed and applied to multiple execution endpoints, as well as controlled and
governed to optimize end-to-end system properties. These context manipulation tasks can be static or dynamic. We expect that smart interactions for web resource discovery, aggregation, and delivery will necessitate dynamic context manipulation.

This paper posits that dynamic context acquisition and management could be orchestrated using feedback loops and SOA governance techniques and technologies [IBMRedGov2007]. On the one hand, SOA governance ensures that the concepts and principles for service orientation and its distributed architecture are managed appropriately and are able to deliver on the stated business goals for services. On the other hand, SOA governance controls the evolution of these service-oriented systems [Müller2009b]. We advocate going one step further and using run-time governance techniques to manage context acquisition, aggregation, and delivery for smart interactions.

2. A motivating scenario

Imagine exploring a new or your own city using your mobile advisor. Depending on your location, time of day, as well as your dining, shopping, and sightseeing preferences, your mobile advisor will provide useful information tailored to you on shops, restaurants, and tourist attractions in close proximity. While you move about, the device continuously searches the web for shops, restaurants, and tourist attractions according to your GPS location. It then filters the retrieved information according to time of day, user preferences and weather. For example, pointing out nearby Italian and Chinese restaurants at lunch time—but no restaurant recommendations right after lunch; or highlighting low or high budget clothing stores or summer tourist attractions depending on preferences.

Alexey Rudkovskiy, University of Victoria, developed such an adaptive mobile advisor [Rudkovskiy2009]. Spasiba, which runs on Nokia S60 phones, delivers dining, shopping, and sightseeing suggestions when you walk down the street—tailored to your personal preferences, your location, current weather, and time of day.

The objective of this research was to formulate an architecture and model for an adaptive, user-oriented, context-aware mobile application. The model leverages context information to enhance the quality of the generated contents and its presentation. Architecture extension points facilitate integration of new, heterogeneous data sources seamlessly. Both the model and the architecture incorporate feedback loops to support autonomicity.

3. Related work

Context-aware systems have been investigated in many different fields including but not limited to interactive, dynamic web including Web 2.0 [O’Reilly2009], web services [Keidl2004], service composition [Fujii2009], context-aware service engineering [Kapitsaki], ubiquitous computing environments [Baldau2007], context-aware computing [Hong2001], sensor networks [Dey2009a], and CSCW systems [Dey2000b].

One of the leaders in this field is Anind Dey, Carnegie Mellon University, who wrote a dissertation a decade ago entitled “Providing Architectural Support for Building Context-Aware Applications” [Dey2000a]. Since then he has investigated various aspects of context-aware computing systems including feedback/intelligibility and control in ubiquitous computing, toolkits and end-user programming environments, and sensor-rich environments.

Kemper’s research team has investigated context-aware adaptable web services and defined context types and life cycles for context-aware web services [Keidl2004]. In a recent article, Georgia Kapitsaki surveys research efforts that attempt to decouple context handling from service logic [Kapitsaki2009]. Meijas and Vallejos proposed a model for context-aware systems using feedback loops [Meijas 2008].

4. Characterizing context

If we intend to control and govern context information in a smart environment, it is useful to characterize context information types and management categories. Context information can be organized along axes from static to dynamic, from nonvolatile to volatile, from nontransient to transient, and many other dimensions. Context management categories include source code level approaches, model driven approaches, architectural and message-oriented approaches, and control-oriented or feedback-loop approaches.

It is also important to define the life cycle of individual units of context information. At acquisition time, raw context information is retrieved through sensors from the environment, inferred from preferences, solicited from end-users, or acquired through web services. At processing time, context information is selected, filtered, composed, ignored, or communicated. Moreover, the processes that perform context information acquisition, composition, and application must be carefully orchestrated, controlled and governed. Keidl and Kemper define the following context types [Keidl2004]: location (e.g., GPS coordinates or time-zone of consumer), client, (e.g., type of operating system or web browser), consumer (e.g., name or e-mail of the person requesting a
service), or connection (e.g., quality preferences). The OASIS Web Services Context Specification characterizes further properties of web service context (e.g., context service and activity scope of a context).

Our running example, Spasiba, acquires context information from the end-user (e.g., preferences) as well as sensors (e.g., GPS location and local time) [Rudkovskiy2009]. The context information of Spasiba evolves over time—slowly when walking and quite rapidly when driving. Moreover, Spasiba is expected to perform with or without context information.

Search engines also use context information effectively and incrementally. Among the data sources for the ranking criteria is the number of user clicks on search results and our own personal search and browsing history (i.e., the feedback loop generated by the frequency of search terms) [O’Reilly2009].

The iPhone continues to impress the public with its innovative context information using built-in sensors (e.g., A-GPS, proximity sensor, ambient light sensor, or accelerometer) [Apple2009]. The Google Mobile Application for the iPhone detects the movement of the phone to our ear, and automatically goes into speech recognition mode. It uses its microphone to listen to our voice, and decodes what we are saying by referencing not only its speech recognition database and algorithms, but also the correlation to the most frequent search terms in its search database [O’Reilly2009].

Smart interaction technologies, as envisioned by the IBM CAS Smart Internet initiative [Ng2009], require innovative mechanisms of acquiring and using dynamic context information. In particular, we envision that feedback loops will play an increasing role in monitoring and leveraging highly dynamic context information to smarten up interaction.

5. Governing context-awareness

The collective context information of a smart environment also needs to be carefully managed and governed. For example, unresolved and uncontrolled privacy issues can easily derail ambitious projects. We envision an infrastructure level above the smart environment level which governs the smart interactions below.

The SOA community defined the SOA governance level above the basic SOA infrastructure to ensure that the concepts and principles for service orientation and its distributed architecture are managed appropriately and are able to deliver on the stated business objectives. Moreover, SOA governance controls the evolution of these service-oriented systems.

We propose the same architecture to optimize context information objectives, control the dynamic evolution of context infrastructure, or govern the context sharing among applications. While not readily apparent, most SOA infrastructure implementations embody classic software evolution concepts, such as levels of indirection and feedback loops. These classic concepts can also be applied to govern context awareness of smart interactions.

To optimize the functions, smart interaction systems should be extensively instrumented to keep track of useful dynamic context information figures (e.g., user clicks on search results, dynamic service attributes, location, time of day, throughput, latency, or resource consumption). Processes that govern context-awareness then use these figures to identify trends, adjust policies and processes, and manage service levels accordingly [Bianco2008]. In particular, such processes can monitor and control the effects and outcomes of policies and processes using feedback loops as depicted in Figure 1 [Müller2008, Giese2009, Müller2009a, Salehie2009]. Selected results of context-awareness policies and processes are fed back to a context-awareness controller which then decides whether there is a need to adapt policies and/or processes (e.g., variables) to optimize outcomes.

Dynamism is both a curse and a blessing. On the one hand, it allows for the orchestration of dynamic value sets as advocated by Mills [Mills2007]. On the other hand, it mandates carefully crafted rules, regulations, and policies for working with these modern, decentralized, distributed socio-technical ecosystems. With the proliferation of user context awareness and dynamic service attributes, replacement of context that is part of a bigger context becomes an important maintenance task [Lewis2008]. Feedback loops can be used to determine when a better or more appropriate context becomes available by monitoring dynamic context variables. For example, a cost variable controller can monitor the dynamic cost attribute of several equivalent services and then adapt the managed process to select the most economical service as depicted in Figure 1.
IBM researchers introduced the notion of an autonomic element as a fundamental building block for designing self-adaptive and self-managing systems such as governance controllers [ACRA2006].

An autonomic element consists of an autonomic manager (i.e., controller), a managed element (i.e., process), and two manageability interfaces. The core of an autonomic manager constitutes a feedback loop, referred to as monitor-analyze-plan-execute-knowledge (MAPE-K) loop, as depicted in Figure 2. The manager gathers measurements from the managed element as well as information from the current and past states from various knowledge sources via a service bus and then adjusts the managed element if necessary through a manageability interface (i.e., the sensors and effectors at the bottom of this figure) according to the control objective. Note that an autonomic element itself can be a managed element the sensors and effectors are used to manage the element (i.e., provide measurements through its sensors and receive control input (e.g., rules or policies) through its effectors). If there are no such effectors, then the rules or policies are hard-wired into the MAPE loop. Even if there are no effectors at the top of the element, the state of the element is still exposed through its top sensors.

The autonomic element as an architectural pattern is now well established and has been applied in many different application domains. For example, Britenham et al. [Brittenham2007] have distilled common patterns for IT service management (ITSM) based on the best practices in the IT Infrastructure Library as depicted in Figure 3.

Collections of related feedback loops are often orchestrated by higher level feedback loops. The IBM ACRA [ACRA2006], as depicted in Figure 4, is an elegant architectural solution particularly suitable for controlling the evolution of governance policies. It organizes the policies into layers where the higher level policies orchestrate lower level policies.

5. Conclusions

Context-aware engineering is one of the most promising research directions for developing smart interaction platforms and highly relevant for the IBM CAS Smart Internet initiative [Ng2009].

A system facilitating smart interactions will be able to adjust its behaviour in response to its perception of the environment and the system itself in the form of fully or semi-automatic self-adaptation based on dynamic context information. This paper posited that to take smart service technologies to the next level we need to take context-awareness to the next level. In particular, acquisition, leveraging, and management of context have to be highly dynamic and self-adaptive to be able to develop smart interactions, as envisioned by the IBM CAS Smart Internet initiative. Feedback loops will play an increasingly important role in monitoring and leveraging highly dynamic context information to smarten up interaction.
To manage the collective context information of a smart environment carefully, we can borrow architectures, policies and processes from the SOA governance domain. We envision an infrastructure level of above the smart environment level which governs the smart interactions below.

While much progress has been made over the past decade, many open research problems and challenges remain in this exciting field of engineering context-aware systems, including constructing context models [Strang2004], managing and leveraging uncertainty due to transient and volatile context [Müller2009a], categorizing control-centric architectural patterns for context-aware systems [Müller2008], or characterizing analysis context frameworks for smart interactions.

References


