Architectural Concerns and Viewpoints from an Architect's Perspective

Jianli Xu and Antti-Pekka Tuovinen
Software Technology Lab, Nokia Research Center, Helsinki, Finland
{Jianli.Xu, Antti-Pekka.Tuovinen }@nokia.com

***Abstract* – In this position paper we briefly discuss what are the most common architectural concerns we have to deal with in our practice of software architecture design and description, and what viewpoints are most needed to construct the architectural views address these architectural concerns from our own perspectives.**

I. INTRODUCTION

An architecture view, as defined in IEEE 1471, is a representation of a whole system from the perspective of a related set of concerns. In our practice we also follow the "4+1" model [1] of architecture views. We use views to address different areas of concern and to categorize architecture descriptions. Our understanding of viewpoint are basically from the work of Hilliard [2], and we just start to capture or mine some useful viewpoints from our own and other's experience. In this paper we give a list of architectural concerns that we, as system designers / architects, think should be considered as candidates for viewpoints.

II. IMPORTANT ARCHITECTURAL CONCERNS – CANDIDATES FOR VIEWPOINTS

Among the concerns listed in this chapter, *A – E* are the general architectural concerns that applicable almost every system, *F – J* are specific architectural concerns in our application domains – telecom systems like switch systems and mobile phones.

A. Most important system functions

The key functions are among the first important concern to be considered, they make the system or product unique, competitive and worth building. For example, making and receiving phone calls at anytime anywhere and maintaining the user contact information are the most important system functions of a mobile phone. Usually we use a logical view to classify the basic system components and their relationship for implementing the key functions.

B. System context

This concern is about the system boundary. The boundary or the environment of the system should be clearly defined and described, e.g. the cellular network and peripherals to a mobile phone. The descriptions of the system context are not view of the system itself, but they help to clarify all the external actors and events. We are not so sure about if we need a context viewpoint to construct the views for externals.

C. Development management concerns: change, reuse and work division

In the architecture descriptions of a specific system, module views are used to address these concerns. So viewpoints which support the ontology of module with properties such as substitutable, unit-testable, well-specified dependency on other modules and clear boundaries, are needed to help to construct the views dealing with these concerns.

D. Dynamic property concerns: performance, reliability and availability

These concerns are about the run-time properties of the system. Viewpoints address these architectural concerns should be able to handle the run-time concepts like execution thread, thread scheduling, resource sharing among threads, timing, etc.

E. Portability on different software/hardware platforms

Different layers or models of abstract machine are used to serve this concern. Viewpoints are needed for layers or abstract machines which provide rules and guidance to check the conformance between layers or different abstract machines.

F. Energy consumption

This is a very important architectural concern for the software system of a mobile device. We need a energy viewpoint to address this concern. Energy consumption is

Submmitted to WICSA2001 Workshop on Architectural Viewpoints, 31.08.2001, Amsterdam

also a run-time concern, so the energy viewpoint may share the concepts and methods with the viewpoints for C.

G. Memory usage

Memory usage is significant to mobile and/or embedded products, e.g. mobile phones, which are produced in large scale. Memory usage concerns both static usage (such as code size) and dynamic usage (like memory used by data and objects generated during run-time). In the architecture descriptions we need a view /views or models to analyze and monitoring the memory usage. A memory viewpoint is necessary to support this purpose.

H. On site upgrade / dynamic re-configuration

This is a very typical architectural concern of large telecom systems like telephone switches. The run-time or execution architecture should be able to fulfill this requirements. The execution architecture are usually described with views containing separately loadable and executable processes. We need a viewpoint to address this special concern.

I. Concerns on UI / user applications

The concerns on UI (User Interface)/ user applications are special for the software system of products like mobile phones and communicators. These concerns include: feature interactions, persistency of application state and application data, management of user data, and response time of user interaction. The UI subsystem of a mobile phone or communicator is the most complex part of the entire system and eats most of the design and development efforts. We certainly need the viewpoints that address these concerns and guide the design / description of UI architecture.

J. Interfacing with legacy software components

In our software development practice, for some reason, we often have to reuse many legacy code in the form of subsystems or components. Products developed totally from scratch are rare. Object-oriented techniques are now used in the development of new software, but the legacy software are usually developed some time ago and not follow the O-O paradigm. They are often very large components and have large and unstructured interfaces. In order to support the integration of the legacy components with the new O-O design / implementation, we need a view that can describe the interfaces of the legacy components in an object-oriented way. We have introduced the concepts of *interface object* and guidelines of how to structure the interface of large legacy components with *interface objects* [3]. We think this work can be further developed into an *interface object viewpoint.* In the following part of this section we just briefly show the main ideas and concepts of our contribution.

Our model for structuring and describing interfaces of large components is based on three main concepts: *interaction domains*, *interface objects*, and *interactions* (see Fig. 1). Large components often participate in several distinct types of interactions called *interaction domain*. In each interaction domain we use *interface objects* for groupings of interface elements related to specific function or role played by the component in its interactions with other components. An interface object has state that allow to specify and control conditional availability of its interface elements. *Interactions* between components are described as interactions between corresponding interface objects as an ordering of bi-directional service access, or a sequencing of sent and received messages.

III. OUR EXPECTATION TO THE WORKSHOP

There is a table in the Appendix of this paper (see next page), we intend to show the correspondence between the architectural concerns discussed in the previous chapter and the architectural view points which address those concerns. In the current table most of the content items are empty and even the filled items may not be appropriate. We do hope during or after the workshop we can work together with other participants to get a rather complete table.

IV. REFERENCES

[1] Kruchten , P.B. 1995. The 4+1 View Model of Architecture. IEEE Software 12(6): 42-50.

[2] Hilliard, R. Viewpoint Modeling. 1st ICSE Workshop on Describing Software Architecture with UML.

[3] Jazayeri, M., Ran, A. and van der Linden, F. (ed.). Software Architecture for Product Families – Principles and Practice. Chaper 6.3. Designing and Describing Interfaces. Addison-Wesley. 2000.

APPENDIX

	Architectural Viewpoints										
Architectural Concerns	Behavior	Module	Resource	Feature	?	?	?	?	?	Interface object	
Most important system functions	X	X		X						X	
System context	X			X							
Change		X									
Reuse		X								X	
Work division		X									
Performance	X		X								
Reliability	X										
Availability	X			X							
Portability											
Energy consumption	X		X	X							
Memory usage	X	X	X	X							
On-site upgrade	X										
Dynamic re-configuration	X		X								
UI feature interactions	X			X							
Persistency of app. state and data				X							
Management of user data				X							
User interaction response time	X			X							
Reuse of legacy components				X						X	