Composable Context-Aware Architectural Connectors

Christine Julien and Dewayne Perry
The Department of Electrical and Computer Engineering
The University of Texas at Austin
c.julien@mail.utexas.edu, perry@ece.utexas.edu

ABSTRACT

In mobile and pervasive computing systems, the ability of an application to adapt its behavior in response to a changing environment is essential. In this paper we propose a new class of context-aware architectural connectors that enable software designers to incorporate context-aware aspects into a software architecture design. These context-aware connectors must also be composable to allow multiple types of context to be applied to a single architectural connection. We introduce our notion of context-aware connectors and describe some intended uses.

Categories and Subject Descriptors

D.2.11 [**Software Architectures**]: Data abstraction, Patterns (e.g., client/server, pipeline, blackboard) C.2.4 [**Distributed Systems**]: Distributed applications

General Terms

Design, Reliability, Theory.

Keywords

Software architecture, connectors, context-awareness

1. INTRODUCTION

As mobile computing systems become increasingly prevalent, it becomes increasingly important to be able to carefully and precisely model the architecture of these dynamic systems in an expressive manner. In addition, architectural models must be able to capture *context*, or the nature of the environment in which mobile applications operate. Some work has developed a methodology for expressing self-adaptive software architectures in a manner flexible enough to allow for runtime changes [6], identifying a number of important design challenges for self-adaptive systems. A bit closer to our intended approach, imposing context constraints on top of existing architectural components has been explored in conjunction with CommUnity [4]. This approach also motivates the need to explicitly separate context and context-awareness from other aspects of a software architecture. Along the same lines, the ability to express context acquisition in software architectures is also important, and should be separated from other architectural aspects [5].

In this paper, we put forth some first steps in defining the fundamental architectural constructs and composition approaches necessary to expressively modeling software architectures in dynamic mobile environments. We adopt the component and connector style of software architecture in which a software system is modeled as a set of components held together by connectors that define the interactive behavior [1, 8]. While we recognize that there is no fundamental *structural* difference between a connector and a component (i.e., both connectors and components are composed of process and data elements, have a sub-architecture, etc.), we also recognize the importance of the *logical* differences, similar to traditional coordination approaches which explicitly separate coordination tasks from computation [2]. In fact, explicitly separating *context* from interaction has previously been explored from a coordination perspective [9].

In moving from the more traditionally static software environments in which existing architectural modeling approaches have been applied to dynamic environments, one must be able to consider the *context* in which components' interactions occur. The nature of this context can significantly impact the nature of interaction specified by connectors, and, as the context changes in response to dynamics in the environment, the connectors must respond accordingly, perhaps changing the nature of the connections they specify, or, more radically, the specific endpoints that they connect.

Given that there is little or no structural difference between components and connectors and the fact that component composition is well-studied, it stands to reason that connector composition is also feasible. The general view of connectors as merely communication is too limited; they may be sources of mediation and coordination as well. To our knowledge, no one has ever before considered composing connectors; although different uses of connectors have been distinguished [7]. With the emergence of mobile software systems, the ability to represent the impact of context and context-awareness within an architectural model becomes essential, and we believe that connector composition offers a natural and expressive way to achieve this representation in a modular way.

In this paper, we discuss our research directions in defining a set of well-defined *context styles* that capture common notions of context-awareness and can be applied to architectural connectors. We also discuss the use of composition to compose the resulting context-aware connectors to allow a single connection to account for multiple types of context information and their impact on a single interaction.

SAM'08, May 10, 2008, Leipzig, Germany.

Table 1: Examples of Context Styles

Category of Context Style	Example (The context style applied to the printer connector connects…)
Functionality available in the environment	to the highest quality available printer.
Availability	to the available printer with the most up time in the last 24 hours.
Quality of Service (QoS)	to the available printer whose connection has the lowest latency.
Capability	using the communication type that requires the least amount of power.
Location	to the physically nearest printer.
Data	to a color printer if the document is in color; to a black and white printer otherwise.

2. DEFINING CONTEXT-AWARE ARCHITECTURAL STYLES

In this paper, we take the first steps necessary to define an abstract model of context-aware architectural connectors. In general, there are two places in which a context-aware connector can be adaptive:

1) The connector can internally adapt its behavior in response to changes in the operating context, e.g., to use a different type of communication or to provide more or less robust synchronization or end-to-end guarantees.
2) The connector can externally adapt the endpoints of its connection, e.g., to connect to a different component or connector based on the context.

As a simple example to demonstrate our approach, consider a mobile user who wishes to be connected to a printer throughout his movement through his environment. From an architectural perspective, his printing application component is connected (via a context-aware connector) to a printer component. Throughout the remainder of this section, we use this example to demonstrate how context styles can be applied to this connector to enable its behavior to respond to a changing environment.

Our intent is to encapsulate aspects of the context mobile applications experience in architecture *styles*, or incomplete architectural prescriptions that, in the case of a *context style* specify constraints imposed on the relevant architectural connector. In our intended model, multiple such context styles can be composed on top of a single interaction connector to form a *context-aware connector*. The context styles define additional constraints that are placed on the communication and coordination activities implemented by the connector. Table 1 provides some categories of context styles and an example of each.

It is important that this approach explicitly separates the context style from the actual interaction behavior. This makes it easier to compose a single context style with an arbitrary connector that is *not* context-aware. In addition, we posit that this will make it possible to layer multiple (non-contradictory) context styles on a single connector. For example, a connector between a printing application component and a printer component could have both the data style and the capability style from Table 1 imposed at the same time; this connector would connect the application to the color printer whose connection consumes the least amount of power when printing color and to the black and white printer whose connection consumes the least amount of power otherwise. In this way, context-styles are simply additive with respect to a standard interaction-based connector.

3. DISCUSSION AND FUTURE WORK

While the model of context styles described in the previous section lays the ground work for enabling traditional connectors to become context-aware, several key questions remain to be resolved. In this section, we explore a few of these challenges.

First, while it is straightforward to compose a single context style with a standard interaction-based connector, imposing multiple context styles on a single connector may not be as simple. Traditionally, multiple compositions may induce "conflicts" which are commonly interpreted as erroneous. In this case, we would not like to view the conflicts as errors but merely as challenging environments that may require a degree of mediation to resolve the specific constraints imposed by the context styles with the dynamic environment. Future work will explore the nature of such mediation and its ability to resolve competing context styles in a manner that matches both the application components' expectations and the mobile computing environment.

Exploring and defining new context styles for mobile application environments has the potential to open up new research in defining architectural connectors that represent basic interactions in mobile networks. For example, one could envision connectors specifying different styles of routing and end-to-end guarantees that may apply in different mobile application scenarios. Composing these new connectors with context styles may generate new directions in implementing communication efficiently in mobile networks. In addition, this opens up the possibility of creating compositions that may not be efficiently implementable; future work will explore representations that expose these challenges and provide direction for their resolution.

4. REFERENCES

[1] R. Allen and D. Garlan. Formalizing architectural connection. In *Proceedings of the 16th International Conference on Software Engineering*, May 1994.

[2] D. Gelernter and N. Carriero. Coordination languages and their significance. *Communications of the ACM*, 35(2):97-107, 1992.

[3] M.M. Gorlick and R.R. Razouk. Using Weaves for software construction and analysis. In *Proceedings of the International Conference on Software Engineering*, pp 23-34, 1992.

[4] A. Lopes and J.L. Fiadeiro. Context-awareness in software architectures. In *Proceedings of the 2nd European Workshop on Software Architecture*, June 2005.

[5] J.J. Martinez and I.R. Salvert. A conceptual model for context-aware dynamic architectures. In *Proceedings of the 23rd International Conference on Distributed Computing Workshops*, pages 138-143, May 2003.

[6] P. Oreizy, M.M. Gorlick, R.N Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D.S. Rosenblum, and A.L. Wolf. An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems and their Applications*, 14(3):54-62, May/June 1999.

[7] D.E. Perry. Software architecture and its relevance to software engineering. Keynote. *2nd International Conference on Coordination Languages and Model*, September 1997.

[8] D.E. Perry and A.L. Wolf. Foundations for the study of software architecture. *Software Engineering Notes*, 17(4):40-52, October 1992.

[9] G.-C. Roman, C. Julien, and J. Payton. Modeling adaptive behavior in context unity. *International Journal of Theoretic Computer Science*, 376(3):185-204, May 2007.