Declarative techniques in Distributed Media Center system

Jari Kleimola and Petri Vuorimaa

Helsinki University of Technology
Telecommunications Software and Multimedia Laboratory
(firstname.lastname@tml.hut.fi)

1. Introduction

In InHoNets project, we have been investigating home entertainment networks from home-to-home interconnectivity and internet content sharing perspective. The aim has been in research of a system where dynamic target infrastructure of networked devices and services can be controlled by heterogeneous remote UI devices. Special attention has been given to utilization of open standard interfaces and techniques. As a concrete result of the research, we have implemented Distributed Media Center (DMC) application, which acts as a convergence hub between controllable targets and remote UI devices (this paper focuses on XHTML-based browser UI, see figure 1). After one year's effort the work is still in-progress, but we have made some observations which might be of interest to the workshop.

Figure 1. Sample DMC -interfaced targets and some concretized remote UIs.

2. Architecture and Operation

High-level view of the system is presented in figure 2. Remote UI device(s) and DMC hub form an interaction loop, where inputs from the user trigger actions in the hub and its interfaced targets. Resulting changes are then reflected back into the remote UI in form of user interface composition or model state changes. State changes triggered outside the system are routed to the UI in a similar manner.

In the hub, input stream is parsed and routed to proper distributed model component, which replies with an XML fragment describing its partial state or result of the action. This is then transformed into an abstract UI page or fragment (e.g., selection list to be used as a menu), which is further adapted into a form suitable for remote UI device.

Currently, adaptation is done at the server end, but it could be performed at the client side as well. Exracted model fragment can also be transformed into a REX mutation event instead of UI fragment.
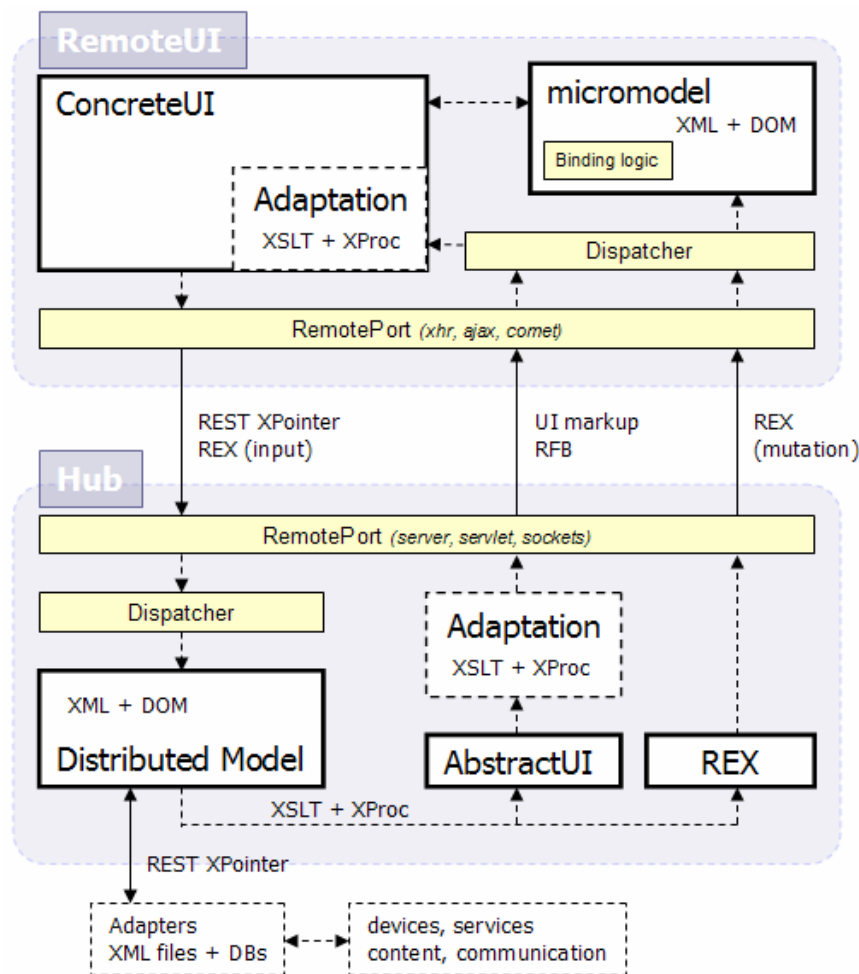

Figure 2. DMC system in its client-server –configuration.

Remote UI parses its input stream and either mutates its internal model, or updates the UI directly. Mutation triggers the binding logic in order to update bound widgets. As can be seen, remote UI and the hub are internally quite similar in construction, which suggests that some common conducting mechanism for hub-ui interplay exists.

Just like AbstractUI defines a common interface for the heterogeneous UI devices, controlled targets are normalized into a common declarative convergence layer. This layer is produced by Adapter plugins, which interface different middlewares (such as UPnP, zeroconf, oBIX and proprietary APIs). Server side model can be further distributed in generic XML files and databases.

White boxes of figure 2 represent declarative modules, while the yellow ones denote procedural portions. Fortunately, the procedural boxes of figure 2 are trivial units (such as simple HTTP server or URI parser) that are application independent. For example, it can be argued that the functionality in the scripted units of remote UI is integrated into the browser. The most complex logic of DMC is in aggregator functionality (allowing multiple components to handle the same request), and in session/subscription handling.

3. Open Interfaces

REST [1] is used in interfacing the hub from remote UI and adapter plugins. UI requests pages or widget fragments using HTTP GETs, and uses PUTs to issue commands. On the other side, adapters notify infrastructure changes using POSTs and DELETEs, but they may emit also PUTs when state changes in individual devices are observed.

When using REST, addressing becomes an important aspect. Because the model is declarative, a natural choice is to use XPointer [2,3]. In this way URI is divided into three parts: first to address the DMC itself (host:port), then to select one of the distributed models (/zones), and finally to address the individual target device or service (/bedroom/TV/channel). Parts two and three are expressed in XPath [5]. Using these XPath-endian URIs facilitate also generic declarative model hosting, because a component can read in an XML file describing an arbitrary model, and execute appropriate retrieval and mutation actions based on the received REST verbs.

Remote UI micromodel is modified using REX mutation events [7]. Simple javascript is used to parse the incoming events and inject the changes into the micromodel DOM, which is uDOM compliant [8]. UI widgets can be bound to the micromodel using XPath, which are evaluated after mutation. DMC hub generates REX events either as request replies, or notifications that are sent to the subscribing clients. Subscription is currently handled at server side, which unfortunately requires a session between client and server.

AbstractUI description is generated from extracted model XML fragment by using XSLT [4]. In adaptation phase, abstract definition is routed into an XSLT pipeline which (after any number of stages for personalization, localization, branding etc.) produces the concrete UI markup that can be rendered in client device. Currently, pipeline chaining is handled procedurally, but XProc [6] is considered as a more flexible solution.

In addition to being remoted, UI can be further distributed by decoupling its input and output devices so that for example, the output is rendered on TV screen, while input events are generated with a mobile phone keypad. In this case, remote UI sends input key or mouse events using REX (which are then passed to adapters).

4. Home-to-home and internet scenarios

RESTful HTTP has a further side benefit in that it can be used to access devices and services that are located behind firewalls. When used to address targets in other homes, only the DMC host address and port portion of the URI needs to be changed, the XPath endian part remains similar in syntax and semantics. Also, as many internet media sources/feeds offer RESTful APIs to their content, the web service adapter coding is simplified.

For home-to-home connectivity, DMC advertises itself using Zeroconf [9]. Thus, unicast DNS-SD can be used to locate DMCs at selected homes, and after resolving their addresses and ports, remote DMCs can be easily integrated into the UI already containing the local targets and local content.

5. Conclusion

DMC's main task is to provide unified access to resources behind middlewares and other APIs, and to support their control and monitoring using heterogeneous remote UI devices. Its role as a dispatcher and aggregator does not require complex logic, which is actually located outside of the system. However, it shows that there are already a wealth of techniques which can be utilized in distributed application and remote user interface realizations.

XSLT is in central position in DMC. It is used to transform the fragments extracted from virtual distributed model into abstract UI description, which is still further transformed into the rendered concrete UI representation. XSLT is used also to transform portions of the virtual model into UI's micromodel, and into the REX mutation events that are used to update the micromodel. XProc would provide flexible declarative glue to orchestrate the transformations.

REST is another key paradigm used in DMC. The earlier versions of DMC used RPC interfaces, which seemed to result stateful constructs that were difficult to manage even in single user scenarios. All DMC operations can be realized with four CRUD verbs. The use of XPath –endian URIs as described in XPointer draft offers general and scalable way to address resources.

References

[1] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, Ph.D. Thesis, University of California, Irvine, California, 2000.

[2] P. Grosso, E. Maler, J. Marsh, and N. Walsh (eds.), *XPointer Framework*, W3C Recommendation, World Wide Web Consortium (W3C), 25 March 2003.
http://www.w3.org/TR/xptr-framework

[3] S. DeRose, E. Maler, and R. Daniel (eds.), *XPointer xpointer() Scheme*, W3C Working Draft, World Wide Web Consortium (W3C), 19 December 2002.
http://www.w3.org/TR/xptr-xpointer

[4] J. Clark (ed.), *XSL Transformations (XSLT)*, Version 1.0, W3C Recommendation, World Wide Web Consortium (W3C), 16 November 1999.
http://www.w3.org/TR/xslt

[5] J. Clark and S. DeRose (eds.), *XML Path Language (XPath)*, Version 1.0, W3C Recommendation, World Wide Web Consortium (W3C), 16 November 1999.
http://www.w3.org/TR/xpath

[6] N. Walsh and A. Milowski (eds.), *XProc: An XML Pipeline Language*, W3C Working Draft, World Wide Web Consortium (W3C), 17 November 2006.
http://www.w3.org/TR/2006/WD-xproc-20061117

[7] R. Berjon (ed.), *Remote Events for XML (REX) 1.0*, W3C Working Draft, World Wide Web Consortium (W3C), 13 October 2006.
http://www.w3.org/TR/rex

[8] World Wide Web Consortium (W3C), *SVG uDOM IDL Definitions*, appendix B to SVG Tiny 1.2 Specification, Candidate Recommendation, 10 August 2006.
http://www.w3.org/TR/SVGMobile12/svgudomidl.html

[9] S. Cheshire and D. H. Steinberg, *Zero Configuration Networking*. O' Reilly Media, Inc., 2005.