Collaborative Intensional Hypertext

John Plaice Blanca Mancilla
Computer Science and Engineering
UNSW SYDNEY NSW 2052, Australia
{plaice,mancilla}@cse.unsw.edu.au

2004

Abstract

We introduce a new approach for creating and viewing hypertexts in a collaborative manner. The hypertext is understood to be an *intension*, in the logical sense, i.e., a mapping from multidimensional contexts to simple texts. Creation and navigation then correspond to moving from one context to another, and links are made relative to the current context. Using an active context called an *æther*, collaborative hypertexts can be built, in which several users can view the same hypertext together, each with their own preferences. We give a summary of a Web-based infrastructure supporting the use of collaborative, intensional hypertexts.

1 Introduction

Hypertext documents naturally create the need for collaboration between viewers. Every view of a hypertext will result from the actions of a user, and can be summarized by some parameterisation of the hypertext, either implicit or explicit. When two or more users wish to share a particular view of the hypertext, then that parameterisation is all that needs to be shared among the users. Should these users wish to *navigate* together, then these users must share the changes to the parameterisation.

This paper presents an infrastructure that not only allows but encourages users to share a parameter space, which contributes to the dynamically generated hypertext document. In this infrastructure, every aspect of a view of a hypertext, including the contents, the look, the feel, and the available hyperlinks, will depend on the parameter settings. By sharing some or all of these settings, users can navigate together through a hypertext, and the actions of one user can renew the views of another.

2 Intensional Programming

Intensional programming [1] refers to programs that are sensitive to a global multidimensional runtime context, which the program itself can modify. In the general case, an intensional program contains a number of versioned entities, each of which has a single valid instance under a particular global context. The set of (*versioned entity*, *current context*) pairs is termed an *intension*, whereas a specific, resolved version of the entity is termed an *extension*. As the program executes, intensional entities are resolved to single versions against the global context. With intensional languages, a versioned entity consists of a set of pairs of context-tagged extensional values (E_x, C_x), termed *versions*. The process of determining which version of an entity is used is termed a *best-fit*, and uses a partial order over the set of all contexts called *refinement* [3] to compare the tag contexts of an entity with the global reference context C_r. The set of all tags in a versioned entity is termed a *context domain*.

The distinction between an intensional program and one that merely uses a context or pervasive data source as a polled reference for runtime configuration is that the semantics of an intensional

program varies implicitly, often with language-level support [9] for versioned control flow. In the extreme case, anything with an identifier can exist in multiple versions, including all functions and data structures, as with the Perl-like language ISE [6].

A context used by multiple intensional programs is called an *æther*. It is a reactive machine holding a context, with *participants* registered at some of the context's nodes. Each participant can modify the æther's context by sending a *context operator*; the æther then informs all other participants of relevant changes, also by sending context ops [5, 7].

3 Hypertext as Intension

A hypertext can be understood in two ways. The bottom-up approach is to consider the individual possible views of a hypertext; these we call the *extensions*. The top-down approach considers the complete set — possibly huge — of these views; this set we call the *intension*.

The intension is the semantics of a hypertext, and can be formalized as a mapping from a multidimensional context (the parameter space) to the individual extensions. Links from one view to the next can be structured to take advantage of the dimensions in the parameter space and formally become intensional operators.

Viewing the hypertext as an intension assists in the modularisation of the hypertext. Each component can be defined in multiple versions, and each of these versions can itself be made context sensitive. All variance need only refer to that part of the context that is relevant to it.

The idea of intensional hypertext is not new [4, 9]. However, previous such work did not focus so explicitly on the context. It turns out that this is the key to collaboration.

4 Collaborative Hypertext

By explicitly using an æther, collaborative construction and navigation of a hypertext is greatly facilitated. Because the æther automatically sends changes of context produced by the users' actions (hence changing the behaviour of the underlying software) to all other users, each user's hypertext server simply needs to update the current view upon reception of a context operator.

Context sharing need not be complete. Users may choose to share only parts of the context and retain control of others. Users might even decide to share one part of the context with one set of users, and another part with another.

It should be understood that the æther can be used to get many applications working together implicitly, not simply a hypertext system. Also, use of the æther for collaboration in no way precludes other means of collaboration (chat rooms, physical proximity, cameras, etc.)

5 Implementation

Examples of such hypertexts exist in the form of intensional Web sites, in particular the author Plaice's Web site, and author Mancilla's context-dependent mapping infrastructure allowing collaboration between map developers [2]. The underlying infrastructure combines Web programming with intensional context sharing.

The infrastructure created uses the Tomcat server and JSP to allow sharing of the global context between users and the Web server components. The infrastructure provides a means to access the Java libintense implementation from JSP pages [5, 7]. The Tomcat servlet container in which the JSP translator resides is multi-threaded. Each individual request is serviced by a different thread from the pool managed by the container. As such, æthers cannot be used directly from JSP pages, as each instance of the page might attempt to update the æther simultaneously, bringing it to inconsistency. This is solved by implementing a thread-safe variant of the æther, such that only one thread may access any node of the æther at any time. These classes are put in a package called IJSP (for Intensional JSP).

The IJSP package is designed to allow many users to share the same context using JSP pages. When one page is following the changes of the context, made by another user, it is notified with only the portion modified. Using the HTTP protocol, one option is a listening applet that updates the hypertext document when it is notified of a context change. In fact, any piece of code can listen at a specific node of the context, and when a change occurs at that node or below, that piece of code is run. A context change is sent as input parameter to the notification code.

6 Conclusion

The general approach of considering the hypertext as an intension is very promising. Simply by making the context explicit and allowing direct access to it — via tools for editing and viewing hypertexts — provides immediate means for collaborative work.

Using intensional programming and the æther data structure greatly simplifies the development of more complex and efficient Web servers to produce hypertexts. The existence of a context ensures that different server components can easily communicate while easily allowing the addition of new components, with very little change to the whole structure; simply changes to the context provoked by the presence of these new components. This statement is actually true for any server or system using the context as backbone.

The approach is also not in any way tied to the Web, nor to programming Web pages. We will soon be using the AFID (Active Intensional Functional Intensional Database [8]) for building stand-alone hypertexts in a declarative manner, complete with multiple authors and version control.

References

[1] A.A. Faustini and W.W. Wadge. Intensional programming. In J.C. Boudreaux, B.W. Hamil and R. Jenigan, eds., *The Role of Languages in Problem Solving 2*, Elsevier North-Holland, 1987.

[2] Blanca Mancilla. *Intensional Infrastructure for Collaborative Mapping.* Ph.D. Thesis, The University of New South Wales, Sydney, Australia, 2004.

[3] John Plaice and William W. Wadge. A new approach to version control. *IEEE Transactions on Software Engineering* 19(3):368–276, March 1993.

[4] m.c.schraefel, B. Mancilla, and J. Plaice. Intensional hypertext. In M. Gergatsoulis and P. Rondogiannis, eds. *Intensional Programming II*, pages 40–54. World Scientific, 2000.

[5] Paul Swoboda. *A Formalisation and Implementation of Distributed Intensional Programming.* Ph.D. Thesis, The University of New South Wales, Sydney, Australia, 2004.

[6] Paul Swoboda and William W. Wadge. Vmake, ISE and IRCS: General tools for the intensionalization of software systems. In M. Gergatsoulis and P. Rondogiannis, eds., *Intensional Programming II*, World-Scientific, 2000.

[7] Paul Swoboda and John Plaice. A new approach to distributed context-aware computing. In A. Ferscha, H. Hoertner and G. Kotsis, eds., *Advances in Pervasive Computing.* Austrian Computer Society, 2004. ISBN 3-85403-176-9.

[8] Paul Swoboda and John Plaice. An active functional intensional database. In *Database and Expert Systems Applications. LNCS*, 2004. In press.

[9] William W. Wadge, Gordon D. Brown, m.c. schraefel, and Taner Yildirim. Intensional HTML, In E.V. Munson, C. Nicholas and D. Wood, eds., *Principles of Digital Document Production. LNCS* 1481:128–139, 1998.