Collaborative Construction of Artifacts

Hannes Ebner
hebner@csc.kth.se

Matthias Palmér
matthias@csc.kth.se

Ambjörn Naeve
amb@csc.kth.se

January 25, 2007

School of Computer Science and Communication
Royal Institute of Technology (KTH), Sweden

Abstract

This paper describes an approach for collaborative construction of artifacts, such as e.g. graphical maps and annotatable text documents, without requiring write-access to a common single file. Our approach is applicable to any kind of artifacts that can be divided into separate contributions, where each one is authored and stored independently, and, on request, merged into the artifact. The goal is to move collaboration issues from the files where artifacts are expressed, to an information directory. This information directory manages information around artifacts and keeps track of existing contributions to artifacts. Our prototype of such an information directory, named Collaborilla, is designed to be a flexible service, which can be updated by anyone in a wiki-style manner. With this approach, viewing a collaboratively constructed artifact gives each viewer the control of including or excluding various contributions. Moreover, each viewer can easily choose to participate and provide a new contribution to the artifact without the other authors being aware of this. If information about this new contribution is published in the Collaborilla directory, the contribution will also be seen by others.

1 Introduction

Even though technology in many aspects has improved information management, collaboration is still rarely supported in practice. Even when there is technical support for collaboration, from a human perspective it is often either synchronous or turn-based. To work satisfactorily, these approaches to collaboration require careful control of concurrency [3] or a manual/supervised merging process. We will focus on situations where collaboration is mostly asynchronous and the risk of producing inconsistencies is small. In such situations, we argue that an important hinderance for technical support is a too tight connection between the *artifact* that is being developed in collaboration and the *expression* of it in a *container*. For example, when writing an article, the article is the artifact, the file where the article is located is the container, and the expression is Microsoft Office XML, Open Document Format, LaTeX, or whatever format is being used. Breaking this connection will allow multiple *contributions* to an artifact without requiring write access to the same container. The central questions are, how do you:

- Discover which artifacts and contributions exist?
- Decide which contributions to include when viewing an artifact?
- Publish artifacts and contributions to artifacts?

Part of the answer is to assign globally unique identifiers to artifacts [1], allowing them to be addressed independently of their expression in containers. This makes it possible to introduce an *information directory* where artifacts can be described. The focus of this paper is on the specific requirements of an information directory that keeps track of asynchronous constributions and to describe an implementation called Collaborilla [2]. Collaborilla has been developed for the purpose of supporting collaboration around a specific type of artifacts called Context-maps [10, 8]. However, we believe that the design of Collaborilla is generic enough to be useful for many other artifact types, provided that their expressions and the resulting artifacts fulfill the basic requirements for collaboration.

2 Existing Technologies and Related Work

Much of the work done within the field of CSCW focuses on the coordination of the communication process around collaboration. For example, in [9] an evaluation framework is introduced where the main variables are work-coupling, communication, and coordination. Communication and coordination refer to the human activities needed to perform the work. The work can be tightly or loosely coupled, which affects the context of the collaboration. Furthermore, in more technical discussions, such as in [3], the focus is on how to develop applications with concurrency-support via notification mechanisms, layers, modules etc., preferably supporting the principle of *What You See Is What I See* (WYSIWIS). In this paper we argue that, for asynchronous collaboration with small risk of inconsistencies, collaboration is better seen as centered around the artifact than around the application, community, or stated goal. A similar argument was made in [7] where metadata around documents is at the center of the collaboration process.

In this section we take a quick look on *artifact-centered* collaboration-related technologies - in contrast to previous work within CSCW.

2.1 Collaboration Technologies

Differential files [5] are a commonly used tool when developing software. Such files are used to update an already existing document to a newer version. A patch is a computer-readable summary of changes (*delta*) between two files or whole file sets. In theory, a patch can be used with any kind of document. However, the order of how the patches are applied to the original document is important. Furthermore, a patch is intended to be merged sooner or later, the longer you keep it separate the more likely it is that there will be problems merging it. In practice, you do not make a patch unless you foresee a merge which requires coordinated collaboration, such as in a revision control system (see below). With contributions to artifacts, such coordinated collaboration is not required. Instead, the merging is done every time an end user requests a view on an artifact. In short, artifacts have no "official" view, while differential files always have an official view, though it changes with time.

Revision control is popular among software developers, both to keep track of changes in source code and documentation, and in situations where software is developed in a distributed and collaborative way. It is common to use revision control in connection with "diffing" and patching tools. Advanced features like branching and merging make revision control superior to just storing snapshots of a project. However, like differential files, revision control is not restricted to any kind of document or project. Popular revision control implementations are the *Concurrent Versioning System* (CVS) and its successor *Subversion*.

Annotea [6] is an extensible technology developed by the W3C that provides a mechanism for adding comments inside existing web documents such as HTML and XML. The approach makes use of xpointers for pointing into existing web pages and a tailored repository for storing the annotations. Annotations are typically small chunks of text that are identified and described with metadata expressed in RDF. Annotea requires specific support in the web browser for supporting editing, publishing and fetching relevant annotations from a given repository and include them in the right place. Of the mentioned technologies, Annotea is the one that is closest to Collaborilla. The major distinction is that Annotea focuses on the annotations rather than on the documents where the annotations apply. With its focus on artifacts, Collaborilla can be seen as complementary to Annotea.

Context-maps [10] are visual representations of concepts and concept relations. The major characteristics of Context-maps are that concepts and concept-relations can be shared between maps, content can be tied to concepts and concept-relations, metadata can be attached to nearly everything, and everything is expressed in RDF. That context-maps are expressed in RDF has consequences such as allowing formalized knowledge modeling in ontologies as well as interoperability with metadata standards in general. It also makes it possible to merge independent contributions by simply joining their corresponding RDF graphs. Furthermore, a context-map is not bound to a specific modeling style. Instead it can be configured to present maps in a manner which resembles concept-maps, UML, argumentative maps etc.

2.2 Information Directory Technologies

There are a number of technologies that do not explicitly support collaboration, but still deserve to be mentioned, since their capabilities may be part of the design that we are attempting. Database Management System is a commonly used technology to hold a large amount of data. A single database usually consists of several tables, which are associated with each other using key fields. Depending on the structure of the database and its data, queries can be complex and thus make the desired design complicated. Versioning of data is not natively supported.

Lightweight Directory Access Protocol [11] (LDAP) is widely used as a backend for information directories. It consists of a tree built out of directory entries, where each entry can hold one or more attributes that contain the information. The design of LDAP allows the creation of entries as children of already existing nodes, making it possible to map an already existing information structure without structural modifications. Versioning of entries is not natively supported.

Domain Name System (DNS) and its extension Resource Records [4] offer a variety of additional fields, which allow more than a simple Hostname-to-IP translation. However, the identifiers in DNS are hostnames, and DNS is not designed to work with more complex identifiers like e.g., full paths.

3 Collaborilla Architecture

Viewing an artifact as a union of separate contributions forces a range of decisions. For Collaborilla these decisions have been made largely based on the needs of context-maps. First, contributions have no separate identity but are assumed to be uniquely identified by the container they are expressed in and the artifact's identifier. Second, there are no dependencies between contributions, only between artifacts and containers where contributions are expressed. Third, the artifact's contributions - indicated via containers - are ordered. Fourth, metadata is availabe on artifacts and containers.

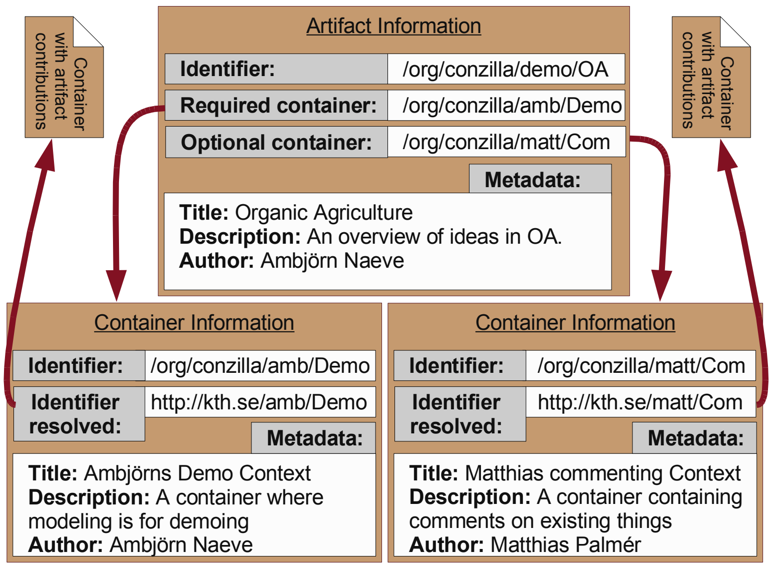
Container with artifact contributions
Artifact Information
Identifier: /org/conzilla/demo/OA
Required container: /org/conzilla/amb/Demo
Optional container: /org/conzilla/matt/Com
Metadata:
Title: Organic Agriculture
Description: An overview of ideas in OA.
Author: Ambjörn Naeve
Container with artifact contributions
Container Information
Identifier: /org/conzilla/amb/Demo
Identifier resolved: http://kth.se/amb/Demo
Metadata:
Title: Ambjörns Demo Context
Description: A container where modeling is for demoing
Author: Ambjörn Naeve
Container Information
Identifier: /org/conzilla/matt/Com
Identifier resolved: http://kth.se/matt/Com
Metadata:
Title: Matthias commenting Context
Description: A container containing comments on existing things
Author: Matthias Palmér

Figure 1: Information and dependencies managed by Collaborilla

Collaborilla is designed as a centrally-managed information directory containing:

- Artifact information: an artifact's persistent unique identifier, descriptive metadata, and dependencies to containers from where contributions can be loaded.

- Container information: a container's persistent unique identifier, descriptive metadata, and a resolving to an locatable address for the container, i.e. a URL.

The container files that provide the actual contributions to a published artifact need to be available at a publicly accessible place. Collaborilla does not, in itself, store the physical files. See figure 1 for an overview of the most important characteristics of Collaborilla.

In the following will we take a closer look at metadata, dependencies, and supporting services of Collaborilla. We will also look at how versioning is done and the technologies used when implementing Collaborilla.

3.1 Metadata

When searching or browsing for artifacts, or contributions to artifacts, it is useful to get access to descriptive information in advance, i.e. metadata. This allows viewers to make informed decisions on which artifacts to view and which contributions to artifacts that should be included. Since the contributions have no separate identity, the contributions cannot have metadata directly. As a compensation, Collaborilla provides metadata on the level of containers where the contributions are expressed. Since every container may contain many contributions (for different artifacts) the metadata describes the contribution context rather than individual contributions. This is an intentional design to minimize both the burdens of editing metadata and the administrative hurdles if the need to update metadata arises. The exact nature of the metadata is not decided but is foreseen to include a short description, who the author(s) is, a purpose, date of modification etc.

3.2 Building a Dependency Tree

In order to be able to work with artifacts and related contributions, a centrally managed information directory has to take care of dependencies between the artifacts and its contributions. Since contributions have no separate identity, the dependencies will be expressed between artifacts and containers. The dependencies of an artifact are composed of two different kinds: required and optional containers. If a container is referred to as required, the referring artifact should not be loaded without it, since it contains essential information. For optional containers it is in the sphere of control of the user to decide wether to include them or not.

3.3 Services

Collaborilla consists of two services: a resolving and a referring service. Resolving a persistent identifier into a location from where a container can be loaded requires information about the location of the requested component. Storing and managing this is the task of the resolving service. The dependency tree and the metadata with information about artifacts and containers are handled by the referring service. Both services rely on the same information directory. It would be possible to use them separately, but in reality the resolving service will always be utilized in connection with the referring service. As both services are part of the collaboration process, both services are implemented within the same application. Currently, you connect to the services through a simple text based protocol inpired by HTTP. A more standardized alternative, such as a Web Service, will be developed in the near future.

3.4 Versioning of Artifacts

In Collaborilla, the most recent collaboration information is always held along with previous revisions. This is done to avoid breaking eventually existing dependencies. The information is archived as a revision before it is actually modified. The information about such revisions cannot be modified in order to keep an authentic history of artifacts and its contributions. Collaborilla may be configured to require authorization, although an open and unrestrictive wiki-style approach to modifications are envisioned and explicitly supported in the design. Furthermore, the metadata is always coupled to a specific revision of a container, so it is advisable to keep revisions of every container. The reason is simple: somebody may have linked to - or reused - something that might not exist anymore in the most recent revision of a container.

3.5 Technology to be built upon

The collaboration technologies discussed in subsection 2.1 do not contribute directly to our approach of collaboration. Conventional patching is too linear, and Annotea follows a different approach. Only revision control can be used to complement our backend. The same applies to the available information directories. No perfect match for our requirements exists, but we can still build upon one of the backends. LDAP [11] is a good choice, since it supports tree structures and the data structures can be easily modified if necessary. Aside from that, the design of Collaborilla gives us the flexibility to switch the backend at a later point of development without architectural modifications. Hence, the decision to use LDAP can easily be changed at a later point.

4 Conclusions and Future Work

We have found a solution to three of the fundamental problems posed as questions in the introduction, i.e. how to discover, decide and publish artifacts. The solution allows contributions to be expressed without any requirements on either common storage or specific coordinated collaboration processes of the authors involved. The implementation, Collaborilla, is an information directory where needed dependencies and metadata are kept. Moreover, the dependencies and metadata are updated in a wiki-style manner. Hence, there is no requirements for authentication. Possibly erronous or intentionally bad information will be approached in a wiki-style, where e.g., a community takes responsibility and does rollbacks or creates new revisions. There is currently two fundamental limitations to Collaborilla that would hinder it from being useful for other artifact-types than Context-maps. First, it does not handle contributions as separate entities, only the containers where the contributions are expressed are dealt with in Collaborilla. This does not allow contribution-specific metadata. Second, Collaborilla does not allow dependencies between collaborations. This would be neccessary for other artifact-types when contributions cannot be merged without requiring 'previous' contributions to be loaded already.

Acknowledgements

This work has been carried out with financial support from the EU-FP6 projects Prolearn and LUISA, which the authors gratefully acknowledge.

References

[1] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifiers: Generic syntax. Request for Comments 3986, Internet Engineering Task Force, 2005.

[2] H. Ebner. Collaborilla - an enhancement to the conzilla concept browser for enabling collaboration. Master's thesis, Department of Computer and Systems Sciences, Royal Institute of Technology, Stockholm, Sweden, 2006. `http://collaborilla.sf.net`.

[3] S. Greenberg and D. Marwood. Real-time groupware as a distributed system: Concurrency control and its effect on the interface. *Proceedings of the ACM Conference on Computer-Supported Cooperative Work CSCW'94*, pages 207–218, 1994.

[4] A. Gulbrandsen, P. Vixie, and L. Esibov. A dns rr for specifying the location of services (dns srv). Request for Comments 2782, Internet Engineering Task Force, 2000.

[5] P. Heckel. A technique for isolating differences between files. *Commun. ACM*, 21(4), 1978.

[6] M. Koivunen. Annotea and semantic web supported collaboration. *ESWC, UserSWeb workshop*, 2005.

[7] A. LaMarca, W. K. Edwards, P. Dourish, J. Lamping, I. Smith, and T. Thornton. Taking the work out of workflow: mechanisms for document-centered collaboration. *Proceedings of the Sixth European conference on Computer supported cooperative work*, pages 1–20, 1999.

[8] A. Naeve. The concept browser - a new form of knowledge management tool. *Proceedings of the 2nd European Web-based Learning Environments Conference (WBLE 2001)*, October 2001.

[9] D. Neale, M. Carrol, and M. Rosson. Evaluating computer-supported cooperative work: models and frameworks. *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, 2004.

[10] M. Palmér and A. Naeve. Conzilla - a conceptual interface to the semantic web. *Invited paper at the 13th International Conference on Conceptual Structures*, 2005.

[11] M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (v3). Request for Comments 2251, Internet Engineering Task Force, December 1997.