Making Doug's Dream Come True: Collaboratories in Context

Aldo de Moor (ademoor@kub.nl)

Infolab, Dept. of Information Systems and Management,

Tilburg University, The Netherlands

Abstract

Collaboratories consisting of systems of information tools are increasingly important as mediators of joint work in distributed groups. These systems should be constructed in a testbed development process. Such a process is far from trivial, and must be continuously improved. To aid in this improvement process, a tool context model is presented, in which the information system, work, design, and improvement contexts of the information tools making up a collaboratory can be represented. Using ontological, norm and rule definitions, the links between various context processes can be systematically defined and analyzed, promoting system integration.

1 Introduction

The Internet, having outgrown its infancy, connects ever more people. It gives them access to a wide range of information tools that allow them to better retrieve and process information. Most importantly, these tools much enhance the power to communicate. Professional communities distributed in time and space now have great potential for improved collaboration, increasing the efficacy of their work, such as the production of joint documents. However, for communities to work together, more is needed than just providing their members with access to a set of tools and wait for collaboration to emerge spontaneously. Instead, the focus should be on achieving *strong collaboration*, in which a group synergistically develops and improves a structured artifact much more efficiently than possible by the same group of people working independently [1]. To realize this synergy, these tools should be examined as part of an integrated socio-technical *system*.

Collaboratories are one interesting type of information tool–based technical systems supporting joint work in the research domain. A collaboratory consists of "various tools and technologies [...] integrated to provide an environment that enables scientists to make more efficient use of resources wherever they are located [2]." Although a few successful collaboratories are already operational, there are no standard solutions for these very complex and volatile socio-technical systems. As the mission statement of the Science of Collaboratories alliance states: "[T]he design, deployment, and adoption of new collaboratories remain difficult and uncertain processes. Each collaboratory has been built as an independent effort. Since these efforts involved complex responses to often idiosyncratic mixtures of social and technical factors, general lessons about collaboratory design remain elusive[1]."

The question now becomes: what design principles are needed? For these types of systems, an iterative socio-technical design approach is required. Such an approach should acknowledge that there are various contexts in which collaboration takes place in different ways by different scholarly populations [3]. In this paper, we examine the parameters of such an approach, and outline how it could be practically implemented in the PORT project.

2 The Testbed Development Process

Thirty years ago, Douglas Engelbart had a dream. In a "Knowledge Workshop 'mission-oriented communities' explore applications by explicit progression, beginning with tested techniques, whose

[1] http://www.scienceofcollaboratories.org/html/AboutSOC/Mission.html

'cultural shock' and financial investment are relatively low, and offering paced, open-ended evolution with time, experience, and perceived payoff [4]". In his view, the only feasible approach is a long-term, pragmatically guided whole-system evolution. It is crucial not just to create good socio-technical systems, consisting of co-evolving Human and Tool Systems. Also, one should continuously improve this design process, in other words "improve the improvement capability of the organization". Engelbart therefore asks the very important question of whether there is a set of basic capabilities whose improvement would significantly enhance both operational and self-improvement capabilities? He then goes on describing his CODIAK (**CO**ncurrent **D**evelopment, **I**ntegration and **A**pplication of **K**nowledge) process, which aims to address this question [5].

Engelbart's insight is a fundamental conceptual step on the way to effective, or pragmatic testbed development. Many collaboratory projects have focused on tool and infrastructure development, or at most the development of tailored systems for particular activities and domains, like computer simulation and healthcare [6]. However, so far only little systematic attention has been paid to the context in which these systems are used and developed.

Building on and generalizing from Engelbart's views on socio-technical systems design, we distinguish a hierarchy of tool contexts, presented in our tool context model.

2.1 A Tool Context Model

- It is clear that information tools should be seen as interdependent parts of an integrated technical system. For example, a mail is posted via a *mailer* to a *mailing list* and redistributed to the *mailers* of the members of the list. These links between tools in terms of data interchange formats, trigger events, shared resource prioritization schemes etc. we call their *information system context*. Together, tools plus links form the *information system*.
- The second layer concerns the *work context* of the information system. The integrated technical system does not operate in a vacuum, but is to serve the purpose of the community. In other words: how does the information system support the goals and activities of the community? Thus, clear definitions are needed of the workflows and organizational structures of the community, as well as mappings between these definitions and the functionality of the various tools in the information system. Together, information system and work context define the *work system*.
- Third, this socio-technical system needs to take into account the *design context* of the work and supporting technological systems. At this level, the development processes of the socio-technical system are modeled. The work system plus its design context together form the *design system*.
- The fourth layer concerns the *improvement context* of the design system: how do we ensure that the design process itself is improved and optimized according to the pragmatic standards of the community? Together, the design system plus improvement context form the *improvement system*.

2.2 Interfaces and Integration

The key units of analysis in each layer are *processes*. These we call *information and communication processes* (information system context), *workflows* (work context), *design processes* (design context), and *improvement processes* (improvement context).

In complex collaboratory development, distinguishing the various systems of users, tools, processes, and their contexts is key. A system is a collection of parts that interact with one another to function as a whole [7]. For each of the system processes, all interacting with parts of the same or other systems, interfaces and integration aspects need to be defined.

An *interface* is the place at which independent and often unrelated systems meet and act on or communicate with each other. Regarding process definition, we interpret the interface of a process to be those of its constituting entities that are visible to system entities outside the process. The internal workings of the process remain a black box to the outside world, however. In this way, the knowledge essential for *system* development is shared, while retaining as many degrees of freedom as possible for implementing or customizing the functionality of the individual tools. For instance, when defining the archival workflow, core interface entities are the actors involved (e.g. authors, system manager), the objects (e.g. archive directories, files), and functions (saving and indexing files).

One meaning of *integration* refers to blending into a unified whole[2]. We define integration as the existence of any *link* between two process entities in any of the systems. For example, an archival workflow may be linked to a design discussion process by separately storing and indexing any

[2] http://www.merriam-webster.com

discussion thread about which tool to use for archiving purposes on the PORT web site. The more system processes are linked, the higher the level of integration. In some cases, a high level of integration may be desirable, although it comes at a cost of higher complexity. An example is the linking of processes from different levels, such as connecting a workflow process to a design process, so that the workflow definition can evolve. In other cases, loosely coupled processes may be more useful. This could be the case with two workflow processes. One such process (e.g. an authoring process) can feed into another (e.g. a review process), while the inner workings of the first process only need to be a black box to the second process. Clearly, the abstract concept of 'link' needs to be further worked out. One interesting issue would be to develop a typology of links useful for collaboratory evolution modelling.

2.3 A Conceptual Lens

The complexity of socio-technical system evolution for collaboratories is such that methodological support is essential. Most methodologies, e.g. in knowledge and requirements engineering, focus on modeling the lowest levels (information system, work context). Pragmatic approaches at the higher meta-levels (design, improvement) that center on making efficient and effective community-controlled systems evolution possible, hardly exist. Engelbart's work is one of the few exceptions.

This paper introduced a (rudimentary) tool context model. It is not a methodology in itself, but rather a meta-model that allows methodologies to be positioned and compared. The model is a conceptual lens that allows one to focus on, literally, the missing links between IS models and methodologies. Thus, it becomes possible to identify the strengths of and gaps in available socio-technical methods for collaboratory development.

Using this contextual framework, testbed development methods for collaboratories can be constructed and tested more systematically. Context elements can be changed at any particular layer, while leaving other layers unchanged. This means that analyzing and comparing different approaches becomes much easier. Such a meta-methodological approach is essential, since still so little is known about what makes a collaboratory development process adequate: "Researchers themselves will build this New World largely from the bottom up, by following their curiosity down the various paths of investigation that the new tools have opened. It is unexplored territory. [6]" A contextual framework can help chart this terra incognita, making the process of exploration safer and more fruitful.

In the next section, we describe and illustrate the PORT collaboratory and its contexts, and show how the tool context model is well suited for meta-modeling the collaboratory development process.

3 PORT: A Contextual Collaboratory

The PORT (Peirce On-Line Resource Testbeds) project has two main foci: (1) developing a model collaboratory for distributed scholarly Peirce manuscript analysis and (2) meta-modeling the testbed development process, making use of Peirce's pragmatism to help guide collaboratory participants in the design process. Both objectives can and need to be developed simultaneously, but must also be clearly delineated. The contextual framework may help in keeping both development processes and their interactions in focus.

3.1 Tools

In PORT, various classes of tools can be distinguished.

- *Generic tools*: mailers, the PORT mailing list, the PORT web sites[3]
- *Specialized work tools*: Inote (a tool for collaboration that provides the capabilities to locate and attach notes to areas on images), Transview (a tool that provides two different views of the same material. This allows one to view the original of a document or image and a transformation of it), CORE (the Collaboration Online Research Environment. This is a tool that provides information management capabilities for collaborators)[4]

[3] http://peirce.monmouth.edu/~bill/, http://www.darmstadt.gmd.de/~dirsch/port/

[4] http://peirce.monmouth.edu/~bill/

- *Knowledge representation tools*: various CG and FCA tools.
- *Design tools*: RENISYS, AUGMENT etc.

These tools provide support for processes in different context layers. Generic tools, such as web browsers, can be used in every layer, acting as gateways to all activities of the virtual community. Specialized work tools are especially useful to support processes in the information system and work context layer. RENISYS and Engelbart's AUGMENT[5] cover the higher layers. RENISYS supports legitimate user-driven community information systems development, but so far has not been able to self-improve its methodology. Incorporating the mentioned pragmatic inquiry approach is one step on the way to a more adequate improvement context. AUGMENT provides many tailorable functionalities that could play a role in systematically improving the socio-technical design process, although improvement processes still seem to be quite implicit.

3.2 Processes

Instead of extensively describing the (continuously changing) PORT improvement system, we illustrate the use of the framework by presenting key example processes and tools supporting these processes, taken from existing web site or papers on PORT. In this way, we roughly indicate how the various initiatives and project elements are related. Note that we do not describe the interface and integration issues here. This framework, however, would provide a good basis for such a more detailed analysis.

- **Information/communication processes:** text editing, mailing, file management, annotating.
- **Workflows:** image capturing, catalogue development, archival processes, report editing.
- **Design processes:** ad hoc discussions on PORT development (as conducted on the PORT mailing list), structured conversations for specification (in the RENISYS method of legitimate user-driven specification, selected users discuss requested changes to the socio-technical system. In a series of well-defined conversational moves, they produce only acceptable changes to system knowledge definitions [8]).
- **Improvement processes:** in [8], we propose to use a pragmatic inquiry process, using Peirce's abduction, deduction, and induction of hypotheses on effective systems development. This is a good example of improving the development process, by making conversations for specification more efficient over time.

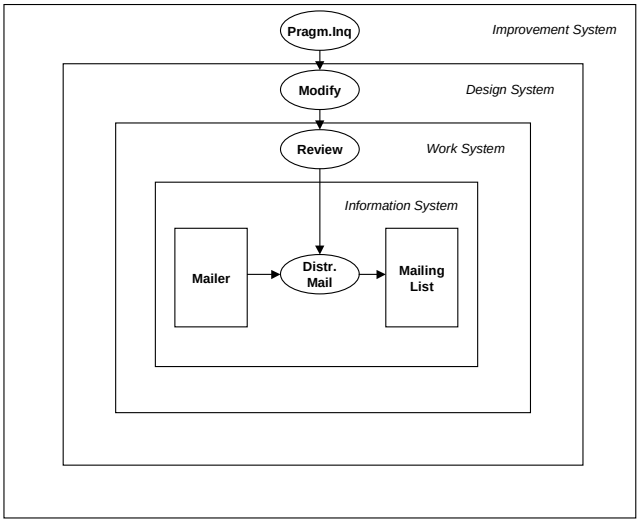

Fig.1 An Application of the Tool Context Model

In Fig.1, we give a (simplified) example of an application of the context model. The problem here is an improvement issue. Currently, a mailer is used to distribute mails via a mailing list, as part of the review process. A modification process of this workflow has been defined, using a manual version of

[5] http://www.bootstrap.org/augment/OAD/2221.html

the RENISYS methodology. However, users have complained that they would like to test several alternatives before making a final decision on which tool to use for the review process. To structure this process, a simple pragmatic inquiry process is introduced, in which users always test two alternatives.

4 Formally Modeling Tool Contexts

Given that there is such a variety of tools, serving complex processes in different contextual layers, formal models can be of great help in organizing and reasoning about collaboratory development. Still, it should be clear that formal context knowledge cannot, in general, be automatically generated. Agreeing on those (continuously changing) knowledge definitions that best suit the particular community to which they apply, requires hard human negotiation. However, once agreed upon, these formal definitions can be useful, for instance, in selecting the right people to informally discuss a particular design problem. In other words, the formal models are not particularly used to model the various worlds in detail, but to structure the discourse *about* these worlds [9]

For formalization purposes, conceptual graph theory is particularly useful, because it is well suited for generalization/specialization operations and context modelling. These properties are important for developing diverse views on, and links between collaboratory systems and contextual entities.

To build formal models, three core knowledge categories are needed:

- *Ontological (and State) Definitions*

Ontologies can be used to make definitions of concept types, defining the type hierarchies and meaning of concepts. Each contextual layer has its own ontologies. For example, part of a type hierarchy of the work context layer could be:

```
Workflow >
  Archive >
    Archive_Images
    Archive_Transcripts
```

Whereas part of the pragmatic inquiry process ontology of the improvement context is (see [8]):

```
Hypothesis >
  Proposed_Hyp
  Tested_Hyp >
    Failed_Hyp
    Succ_Hyp
```

Ontological concepts are the basis for relevant knowledge definitions, such as norms and states-of-affairs (“state”) definitions. One example of a state definition that can be used to define links (integration aspects) between entities from the system and work context is that of the *support*-relation [10]:

```
[State: [Support:#145] –
   (Poss) <- [User: #John]
   (Inst) -> [Inote: #1]
   (Obj) -> [Workflow_Mapping:#123]].
```

This definition explains that John uses the INote tool for supporting his workflow #123, which could be a certain kind of review process-by-annotation.

- *Norms*

A norm is a principle of right action binding upon the members of a group, and serving to guide, control, or regulate proper and acceptable behaviour. Norms can thus be used to direct processes at the various context levels. A RENISYS composition norm at the design context level could be:

```
[Req_Comp: [Editor] <- (Agnt)-
  [Eval] -> (Obj) -> [Specify] -> (Rslt) –
     [Type: [Archive]]].
```

This norm states that the editor must evaluate (i.e. approve of) any change in the specification of the archival process properties.

- *Rules*

Using the ontological (and state knowledge) and norms, the complete collaboratory improvement system can be defined. In order to activate this declarative knowledge, CG tools that provide a procedural instead of merely declarative knowledge support are needed. Examples are PROLOG+CG (which allows PROLOG rules to reason about CG knowledge definitions) or tools that enable conceptual graph actors to fire (like CharGer). Using rules, design and improvement events can be automatically triggered, resulting in more adequate systems development processes. For example, if a tool is introduced that enables a new type of communication processes (information system context), a discussion at the design level could be automatically started, in which the support for the workflows provided by the current tools is reconsidered. An outcome of this process might be to assign the new tool as the software of choice for an existing workflow, such as the review process.

5 Conclusions

Collaboratories are composed of systems of information tools. In this paper, we stressed the importance of systematically describing and analyzing the information system, work, design, and improvement contexts of these tools, leading to better interfaces and integration. In this way, complex collaboratory development processes can be made more systematic and pragmatic. Douglas Engelbart already foresaw the importance of such a structured approach a long time ago, summarized by his motto of "improving the improvement process". With the maturation of Web-based information tools, and sophisticated knowledge representation and reasoning formalisms like conceptual graph theory, the time has come to make Doug's dream come true.

5.1 References

1. Johnson, P. and C. Moore, *Investigating Strong Collaboration with the Annotated Egret Navigator*. 1994, University of Hawaii.
2. Committee Toward a National Collaboratory: Establishing the User-Developer Partnership, N.R.C., *National Collaboratories: Applying Information Technology for Scientific Research*. 1993, Computer Science and Telecommunications Board.
3. Sumner, T. and S.B. Shum. *From Documents to Discourse: Shifting Conceptions of Scholarly Publishing*. in *Proc. of CHI'98: Human Factors in Computing Systems, Los Angeles, 18-23 April 1998*. 1998: ACM Press.
4. Engelbart, D.C. *Coordinated Information Services for a Discipline- or Mission-Oriented Community*. in *Proc. of the 2nd Annual Computer Communications Conference, San Jose, California, January 24*. 1973.
5. Engelbart, D.C., *Toward High-Performance Organizations: a Strategic Role for Groupware*. 1992, Bootstrap Institute.
6. National Research Council, *Issues for Science and Engineering Researchers in the Digital Age*. 2001, National Academy of Sciences.
7. Maani, K.E. and R.Y. Cavana, *Systems Thinking and Modelling: Understanding Change and Complexity*. 2000, Auckland: Pearson.
8. de Moor, A., M. Keeler, and G. Richmond. *Towards a Pragmatic Web*. in *Proc. of the 10th International Conference On Conceptual Structures (ICCS 2002), Borovets, Bulgaria, 15 - 19 July,*. 2002.
9. Shum, S.B. and A.M. Selvin. *Structuring Discourse for Collective Interpretation*. in *Proc. of Distributed Collective Practices 2000: Conference on Collective Cognition and Memory Practices, Paris, 19-20 September*. 2000.
10. de Moor, A. and W.J. van den Heuvel. *Making Virtual Communities Work: Matching their Functionalities*. in *Proc. of the 9th International Conference on Conceptual Structures, Stanford, July 30-August 3*. 2001.