Towards a Pragmatic Web

Aldo de Moor[1], Mary Keeler[2], and Gary Richmond[3]

[1] Infolab, Dept. of Information Systems and Management, Tilburg University, Tilburg, The Netherlands
ademoor@kub.nl
[2] University of Washington, Seattle, USA
mkeeler@u.washington.edu
[3] City University of New York, New York, USA
garyrichmond@rcn.com †

Abstract. The Semantic Web is a promising step toward improving virtual community information systems. It gives information a clearer meaning, better enabling computers and people to cooperate. However, still lacking is the *purpose* of the information: how is it going to be used and evolve? In a Pragmatic Web, the context of the information would be defined as well, as the community examines goal-based conditional inferences in its work in progress. Scientific collaboratories could benefit substantially from such an approach. The PORT collaboratory was established to provide a model for pragmatic collaboratory evolution. In this paper, we outline a pragmatic community information systems development process by combining PORT with the Conceptual Graphs-based RENISYS method for the legitimate user-driven specification of community information systems. Peircean pragmatism provides a self-critical approach for tool selection in virtual communities.

1 Introduction

The Internet is changing the way the world works, literally and virtually. Originally developed for military use, it was quickly seized by the academic community. In the 1990s, its great breakthrough came with the World Wide Web, dramatically increasing the volume of both users and applications.

A critical mass having been reached, the Internet has begun to change the way people work together in learning, doing research and business, and managing healthcare. Such virtual professional communities can be viewed as complex adaptive sociotechnical systems, whose members collaborate towards accomplishing what they define as common goals.

The complex information systems that these communities require are often constructed out of many different *information tools*. Examples include mailing lists, chat tools, file management systems, and discussion boards. Often, such software can be

† Proc. of the 10th International Conference on Conceptual Structures, Borovets, Bulgaria, July 15-19, 2002. © Springer, Berlin, LNAI 2393, pp.235-249

tailored to meet the specific requirements of the community. Also, many tools provide partially overlapping functionality, having many shared and some unique functions. A virtual community's selection of tools, of the plethora available, considering their growing information needs and rapid technological advancement, is no trivial challenge. Essentially, the software selected must serve both sociability and usability. Sociability concerns social interaction: ensuring that the tools enable social policies that are understandable and acceptable to users and that support the community's purpose. Usability refers to human-computer interaction: ensuring that people can interact and perform their tasks intuitively and easily [7]. The complex and continuous process of sociotechnical change required is so costly, that natural community evolution tendencies are inhibited. Yet catalyzing, directing and even experimenting with change in a virtual community is essential to its continuous viability.

One major difficulty in community information systems development is deciding who should be involved in the development process [8]. The traditional method – of assigning a software engineer to make a model of the community, selecting some tools, and creating a suitable information system by "self-fulfilling prophesy" – is not sufficient anymore. System evolution is subtle and continuous. Much user experience and tacit knowledge is needed to interpret the requirements, and to produce the actual specifications [5]. Furthermore, such systems are never finished, but rather grow in complexity, as new requirements emerge, and more advanced technologies become available. The members of the communities must therefore play much more active roles in the systems development process than before. Somewhat as in stage direction, they must become self-aware of what is their role, how and when it relates to the roles of others, what is the nature of the tools they need, and how to specify those requirements. For example, an author of an article may report that the process for submitting a paper to an electronic journal is not efficient. Normally, however, decisions about whether and how to redesign the submission process are made by the editorial board (who view it in terms of workflow) and the system manager (who views it in terms of technical features), not necessarily taking into account the author's view (in terms of user-friendliness).

Another crucial issue in community IS development is how to direct development efforts: what should drive specification discourse? Goal-directedness is essential for productive virtual communities [7, 9]. This goal-orientation should extend from regular work processes to system evolution. Goals must be modifiable with increasing experience of those in the community, by a continuous process that we might call *goal reflection*. This goal reflection process must be integrated with the community's own information systems development effort, if that augmentation is to become more effective and efficient. In Peirce's terms, this form of improvement is pragmatic.

In this paper, we explore how to operationalize community information systems development by viewing it as a process of pragmatic tool selection in a testbed environment. The RENISYS method for legitimate user-driven system specification is one example of a testbed development methodology. It facilitates virtual communities in the formal specification of changes to their socio-technical systems [2, 3].

Peirce identified pragmatism as the logic of abduction. We can make RENISYS a more pragmatic method by using Peirce's insights to establish criteria for abductive operations, giving users the capability to formulate hypotheses. We rely on Peirce's in-

sight to formulate some basic pragmatic criteria. The PORT (Peirce On-line Resource Testbeds) collaboratory serves as the case study virtual community in which we describe and develop our approach. PORT is a collaboratory based on Peirce's archived manuscripts, and is dedicated to apply his principles of inquiry, experimentally, in its collaboratory development. In both process modeling and tool development, PORT has ties with the Conceptual Graphs community. We use conceptual graphs as the knowledge formalism, because they are well suited to model evolving knowledge structures of different levels of detail.

2 Towards a Pragmatic Web

Much valuable work is currently being done on the Semantic Web[1]. This is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. Technologies like XML enable the structured description of meta-information of web page elements. On top of that, the Resource Description Framework (RDF) allows for making specifications that provide a lightweight ontology system to support the exchange of knowledge on the Web[2]. This addition of semantics to web data prepares the way for software agents that collect Web content from diverse sources, process the information and exchange the results with other programs [1].

Enthusiasts think that "The Semantic Web, in naming every concept simply by a URI, lets anyone express new concepts that they invent with minimal effort. Its unifying logical language *will enable* these concepts to be progressively linked into a universal Web." [1] (our italics). This view seems to take for granted that a semantic language by itself will somehow take care of knowledge and community evolution. Admittedly, the Semantic Web is a necessary step from the syntax (HTML) level to the semantics (meaning) level. However, still one crucial level is lacking: that of pragmatics: what is the *purpose* of the information? How do we use it, and change it, as we use it?

To determine the context of use of the information is not trivial. For example, let us take a look at the scientific publication review process. At the semantic level, the review process can be precisely defined: there are draft documents, reviewers, review reports, editorial decisions, notifications to authors, etc. However, many differences exist between review processes. Some are open, in the sense that reviewers know the authors' names, others are blind. In some there is a discussion among reviewers, in others only the editor sees the review reports. With a new journal, deciding correctly on these details is essential. These specification knowledge decisions cannot be left to software agents. They do not belong at the semantic level, but at the pragmatic level.

In most community information systems development, these choices are left to informal decision making and change processes. We are entering an ever more rapidly changing world, with a continuous introduction of new information technologies. Ensuring that the purpose of communities is reflected in the design of their socio-technical systems cannot be left to chance. To put it more strongly, significant improvement requires a long-term, pragmatically guided process of whole-system evolution, in which

[1] http://www.w3.org/2001/sw/
[2] http://www.w3.org/RDF/

human and technological systems are calibrated together [4]. If such pragmatic aspects are systematically addressed, web-based community information systems should be much more useful.

Summarizing, we think that the Semantic Web is a necessary, but not a sufficient condition for satisfying the needs of today's virtual communities. We therefore also propose the development of a Pragmatic Web. In this web, essential pragmatic processes are carefully defined and automated where possible. In this way, human beings can focus on their unique qualities of creative thinking, balancing options, and wisely using their unlimited supplies of tacit knowledge.

How to operationalize pragmatic aspects? How to use them in more effective community information systems evolution? In this paper, we present an approach that may help to pave the way to a Pragmatic Web.

3 PORT: Peirce On-Line Resource Testbeds as a Model Collaboratory

William Wulf first conceived collaboratories "to accelerate the pace and quality of discourse and broaden the awareness of discovery"[3]. In collaboratory operation, user-oriented rapid-prototyping testbeds support partnerships between users and technologists to explore the utility of any technical approaches by which the scientist user community might take advantage of emerging technologies in support of the growing need for effective collaboration. With the creation of digital archives worldwide, and the subsequent development of intellectual resources based on these artifact sources, testbed partnerships should find a critical role in collaboratories for such digital resource development. The Peirce On-line Resource Testbeds (PORT) collaboratory is conceived to be a model digital resource collaboratory, in which participants can jointly study an archive of digitally imaged artifacts, while they study their own needs for technology to augment that collaborative research.

PORT, as an effective operating model, can demonstrate the advantages of testbed-based resource development in fundamentally improving the efficiency of human-human (user-technologist) interaction to make possible more effective development of both technology augmentation and human uses of that technology. Technological advancements will not be effective without evolution in the conduct of testbed participants, by self-critical awareness and habit change.

PORT has a number of interrelated objectives:

– to integrate knowledge processing tools and demonstrate how they can improve the establishment of effective testbed partnerships between users and developers in collaboratory operation.

[3] For Wulf's "collaboratory", see National Collaboratories: Applying Information Technology for Scientific Research–Committee of a National Collaboratory: Establishing the User-Developer Partnership, Computer Science and Telecommunications Board, Commission on Physical Sciences, Mathematics, and Applications, National Research Council (Washington, D.C.: National Academy Press, 1993); the quotation is from Joshua Lederberg, and Keith Uncapher, Towards a National Collaboratory: [NSF] Report of an Invitational Workshop (Rockefeller University, New York City, 13-15 March 1989), p. 3.

– to demonstrate the effectiveness of knowledge processing tools as interfaces for efficiently creating digital resources, by developing a collaboratory model based on content (Peirce's philosophical manuscripts archived at Harvard's Houghton Library).
– by increasing efficient access to Peirce's work in knowledge representation and theory of inquiry, to improve the theories and methods of knowledge science, in a continuing program of research to augment the operation of collaboratories as interdisciplinary "communities of inquiry" for international resource development, learning, and research.
– to apply the testbed method in monitoring the ever-advancing limits of knowledge processing technology, which must be watchfully instituted in a manner that truly augments, not simply tries to replicate and replace, human inference by knowledge processing.

Collaboratory operation requires: (1) system architecture and integration to explore ways that people and machines can use component technologies most effectively, (2) a research program to study the conditions required for collaboration, and (3) user-oriented rapid-prototyping testbeds, to understand the impact of technologies used. Testbeds must give users the interface by which to monitor how the integration of new functions in their system of operation might improve their work, enabling them to take more critical control. Knowledge science research in conceptual graph theory has begun to establish the formal basis for such pragmatic integration, but now requires the testbed method to carry out a pragmatic program of continuing analysis, testing, and development of tools.

Knowledge science's continuing challenge is to distinguish which inference processes in knowledge representation require human intelligence and which are better served by the computer's automation capabilities. Beyond effective partnerships between human and machine intelligence in any particular context of operation, the ultimate challenge–of better human-to-human partnerships–will require human-computer interfaces by which to observe and collaboratively contribute to conceptual evolution as it progresses. Because the operation of effective partnerships must respond to the inevitably continuous change in technology and user needs, knowledge science research can employ testbeds as the pragmatic method in a semiotic research program to investigate the conditions required for effective technological augmentation to occur.

According to Peirce's pragmatism, our natural cognitive urge to conceptualize, form habits of thought, or "automate behavior" in routines and tools must be checked by our discriminating sensory capability, through pragmatic conduct that continually conceives and tests these ideas for validity and reliability by observing their implications in experience. To establish self-critical control in human-computer "partnerships," knowledge scientists must represent in relational detail any functions to be automated, making it possible to observe conditional dependencies that define goal-directedness. As explained in [6], Peirce's general theory of knowledge representation, communication, and learning (or semiotic) explains inquiry as a continuing collaborative argument with premises, conclusions, and an account of the interpretational procedure to reach judgments from the evidence. His pragmatic (or methodological) caution asserts that judgment should proceed heuristically–not algorithmically, by unexamined authority

or habit of mind. Any judgment established by a community of inquirers may well be mistaken. A critical editorial function must track the conceptual relations among individual interpretive reports, to identify possible emerging patterns of thought as hypotheses to be tested in the community by re-examining evidence. Facts may be considered more-or-less confirmed judgments, but their meaning or implication must be regarded as always in the future. In testbed operation, even application tools and systems can be treated as hypothetical conjectures (in terms of conditional statements).

The scope of Peirce's theory explains the continuity of inference from its most algorithmic form (machine) to its most analogic form (human). His pragmatism, as the conduct of inquiry implied by his semiotic, describes the self-critical practice or procedure required for successful collaboratory operation. As the conduct implied by his theory, pragmatism is the exercise of self-critical control–or learning by continuing to test representations for their effectiveness, never considering them final or complete. His philosophical perspective encourages us to investigate the conditions necessary for meaning to grow as knowledge, and his pragmatism instructs us to continue this investigation indefinitely. Testbeds can be developed as the pragmatic method for observing, comparing, and judging competitive efforts in network application technology development, in an evolving Pragmatic Web.

Peirce's logical analysis of the conduct of inquiry as creating, testing, and validating representations has three stages (abduction, deduction, and induction) which account for the effective formation of intellectual concepts as theoretically explained by semiotic and conducted according to pragmatism. Zeman traces the process:

"[Abduction is] educated hypothesis-formation which proposes initial organizations of figure in the problematic field. Deduction enters in a mediating way, drawing out the consequences of the abductive hypotheses. And induction consists in the return to experience which aims at confirming or refuting those hypotheses by seeing whether the deduced consequences hold or not [10]."

Peirce proposed pragmatism as the logic of abduction. In PORT development we plan to integrate RENISYS and develop it for our collaborative tool selection operations, beginning at the stage of abduction.

Our ultimate concern in developing a pragmatic-testbed method is not just to establish consensus that would simply resolve diverse opinions, but to reach provisional agreement about interpretations that could then continue to be tested and modified in further experience. In testbeds, we can cultivate the habit of maintaining provisional views of our judgments by self-critically examining the actual and possible outcomes of implementing them, integrating as many means of representing those implications as we can create "tools" to do so.

PORT participants must learn to operate with a dynamic set of modular, user-initiated processes, including those that modify other processes. In the testbed context, integrated communication facilities must enable them (1) to report (and demonstrate) their project experience and results efficiently, (2) to track similarities and differences among requirements and techniques offered and their test results, and (3) to conceptually-map the progress of their work, with respect to other testbed members. To operationalize the testbed development process, we first summarize our view on the pragmatic inquiry process.

3.1 The Pragmatic Inquiry Process

Inquiry becomes science when self- and hetero-criticism finally results in a methodeutic for reaching consensus that is congruent with reality. Experimental science remains the prototype of this manner of what Peirce called "fixing our beliefs," in the present context, arriving at consensus in a matter of importance to the collaboratory. While a personal judgment is not criticizable yet, given a personal history of the matter it blends subtly into "guesses", hypotheses, abductions about what some phenomena might mean so that finally a compelling hypothesis is formulated to be considered by the community. Deduction then would be what would necessarily follow if this hypothesis is correct, the concern being to devise experiments for testing the hypothesis. Finally, induction is the actual experimental testing to determine to what extent the hypothesis conforms to reality. The pragmatic inquiry process therefore goes as follows:

- *Abduction:* Proposing hypotheses in regard to PORT's goals, tools, and possible sub-projects. Decisions as to what hypotheses ought to be entertained in PORT's testbeds should conform to Peirce's "economy of research", by asking such questions as: What appears to be most natural, efficiently developed, and capable of supporting habits of value to the community? Such economy would keep in general view the optimal use of all resources, including, and especially, human resources. Since individuals (rather than groups) will usually propose hypotheses, they must be formulated to maintain consistent relations among the goals, values, purposes, desiderata, projects, tools or anything having to do with the operation of the entire community.
- *Deduction:* Formulating these complex interactions and relations among elements and operations of the community testbed experiments, to maintain a coherent view of the implications of all abductions, in terms of their value to the community.
- *Induction:* (following experiments and their deductive validation) Testing to determine to what extent the validated experiments actually conform to the goals, values, tool-interactions, etc. in contexts of operation, and what further abductions are required for better performance in the collaboratory.

4 Operationalizing Pragmatic Testbed Development

In this section, we describe our approach for pragmatic testbed development. Sect. 4.1 outlines the RENISYS method for the legitimate user-driven specification of community information systems. Sect. 4.2 explains how this pragmatic testbed development process can be operationalized in the RENISYS context.

4.1 Making RENISYS More Pragmatic

Starting point for the pragmatic development of community information systems development is to model their evolution. RENISYS (**RE**search **N**etwork **I**nformation **SY**stem **S**pecification) is a method for legitimate user-driven system specification, which allows members of goal-oriented virtual communities to model and support the evolution of their socio-technical system

RENISYS currently contains four main components: (1) an *ontological framework* describes the entities necessary to describe evolution in virtual professional communities; (2) *conversations for specification* allow specification changes to be made by users playing well-defined specification roles; (3) a system of *composition norms* is used to calculate which users can legitimately initiate, execute, and evaluate specification changes; (4) a *functionality matching* metamodel and process can be used to describe tool selection in virtual communities.

In this paper, we do not focus on these components of the RENISYS method. They have been explained in considerable detail previously, e.g. [2, 3]. Instead, we focus on developing an extension with a pragmatic inquiry process, as it could be implemented in this, or similar community information systems development methods.

The virtual community's structure, operations, and evolution are modelled as knowledge definitions, based on the RENISYS ontologies. The structure of these ontologies and the possible definitions is not relevant here, as we focus on the *meta*-level in which they are reflected upon. For the interested reader, the ontologies and possible knowledge definitions are explained in [2].

Despite its capabilities, RENISYS cannot operate truly pragmatically for improving a community's capabilies to define, learn, and organize their work. While knowledge definitions are acceptable, they are not optimal. The driver of change is still an "individual user facing a breakdown" mechanism. Most significantly, no explicit process provides for users to experiment with technologies, which is essential because the operation of tools in realistic work settings cannot be fully predicted, and must be identified by actual use in those settings. A systematic meta-improvement process needs to be added to make community evolution more effective and efficient. One theoretical approach that offers guidance in designing that improvement process is Peirce's pragmatism, in particular his pragmatic inquiry process.

There are several ways in which RENISYS could benefit from the pragmatic inquiry process. First, the main driver of change in RENISYS was individual users becoming aware of breakdowns. Now, specification processes can also be triggered because of other, more sophisticated pragmatic reasons. Second, in the old version of RENISYS, only a primitive definition change process was supported. Once a definition has been accepted, there is no follow-up. With the pragmatic inquiry process, definitions can be monitored over time and alternative definitions for the same problem can be tested and compared.

In this paper, we add a new type of knowledge definition, *hypothesis definitions*, and show how to use them to select which knowledge definitions to investigate in the RENISYS conversation for specification, using a Peircean pragmatic inquiry process. Benefits are that specification processes can be triggered for more sophisticated pragmatic reasons than work breakdowns and that monitoring definition and implementation processes becomes more manageable.

4.2 Case: Link Classification in PORT

To illustrate how to operationalize the pragmatic testbed development process, we use real events from the PORT case.

In July 2001, work started on a community information system for PORT. CommunityZero is a provider of free platforms for virtual communities, and the PORT@Home web site was established[4]. One initial activity to support was the classification of links related to the various discussion topics in PORT: the idea was to have users submit links, after which they could be classified and stored, developing a dynamic link archive accessible to all members. Besides the PORT@Home web server, a workspace was created in the BSCW file management tool[5], for the purpose of storing PORT material.

Initially, there were three users, the authors of this paper. Once established, the following evolution of the link classification system took place.

1. The initial idea was to develop a link classification system. All users agreed on this broad goal.
2. User #1 (the system manager) then implemented this system partially on the BSCW server (where the actual links plus their descriptions were stored), while the classification/indexing was done on the PORT@Home server, forcing users to move back and forth between both sites.
3. User #2 did not think that placing the links on a different server than PORT@Home was right and proposed to put the links where their indices were, abandoning the BSCW tool for this purpose.
4. User #1 agreed with his objections of it being user-unfriendly, but in his role as system manager objected to changing the status quo: BSCW, being on a university server, is securely backed up, while PORT@home is hosted by a potentially unreliable commercial provider.
5. The issue was not really settled yet, and temporarily put on hold.

Hypotheses In RENISYS, two main categories of concepts are distinguished: *entities*, modelling the socio-technical system, and *definitions*, used to describe and reason about this system [2]. We now add a subtype of the definition concept: the *hypothesis*. Two subtypes of hypothesis are *proposed hypothesis* and *tested hypothesis.* A tested hypothesis either has *failed* or has been *successful*.

Note that in the following the definition referents are informal to save space. The real underlying graph representation of, for instance, the "Use some tool to support link storage process" referent in h1 would be the following formal definition of a *required implementation-definition* [3]:

```
[State: [Req_Impl: #165] -
  (Inst) -> [Tool]
  (Obj) -> [Workflow_Mapping: #123] -
     (Part) -> [Store_Link]].
```

So, how to define hypotheses? An initial attempt at definition is presented next.

```
[Prop_Hyp: "Use some tool to support link storage process"] (h1)
[Succ_Hyp: "Use BSCW to support link storage process"] (h2)
[Failed_Hyp: "Use PORT@home to support link storage process"] (h3)
```

[4] http://www.communityzero.com/port

[5] http://bscw.gmd.de

However, h2 is only successful, and h3 only a failure from a security perspective as well as User #1's (system manager) point of view. From a user-friendliness perspective and both User #1 and User #2's point of view, however, h2 is a failure, and h3 a possible success, which could be tested. Thus, the representation pattern used for h1-3 is too primitive. At least, a user's point of view and some effectiveness criterion must be added to it. The general definition of a hypothesis becomes:

```
[Hypothesis: [Definition] -
   (Agnt) -> [User]
   (Chrc) -> [Criterion]].
```

The hypothesis-related part of the ontology for this community thus is:

```
T >
  Definition >
    Hypothesis >
      Prop_Hyp
      Tested_Hyp >
        Failed_Hyp
        Succ_Hyp
    Criterion >
      Secure
      User-Friendly
      ...
```

Example:

The hypotheses in the case evolved as follows:

• *Stage 1*: Everybody agreed to implement the link storage process, no specific criteria were defined:

```
[Prop_Hyp: "Use some tool to support link storage process"]. (h1)
```

• *Stage 2*: User #1 (the system manager) implemented (=tested) h1, using BSCW, in his view successfully, with no specific criteria in mind:

```
[Succ_Hyp: "Use BSCW to support link storage process" -  (h2)
   (Agnt) -> [User: #1]].
```

• *Stage 3*: According to User #2 and the user-friendliness criterion, using BSCW for the link storage purpose failed. Instead, he proposed to replace BSCW by PORT@Home, thus putting it on the agenda to be tested:

```
[Failed_Hyp: "Use BSCW to support link storage process" -
   (Agnt) -> [User: #2]
   (Chrc) -> [User-Friendly]].  (h3)

[Prop_Hyp: "Use PORT@Home to support link storage process" -
   (Agnt) -> [User: #2]
   (Chrc) -> [User-Friendly]].  (h4)
```

• *Stage 4*: User #1 (the system manager) clarified his reasons for choosing BSCW instead of PORT@Home. He agreed that BSCW failed from a user-friendliness point of view. However, PORT@Home failed from a security perspective, whereas BSCW, in his view, is successful there:

```
[Failed_Hyp: "Use BSCW to support link storage process" -
   (Agnt) -> [User: #1]
   (Chrc) -> [User-Friendly]]. (h5)

[Failed_Hyp: "Use PORT@Home to support link storage process" -
   (Agnt) -> [User: #1]
   (Chrc) -> [Secure]]. (h6)

[Succ_Hyp: "Use BSCW to support link storage process" -
   (Agnt) -> [User: #1]
   (Chrc) -> [Secure]]. (h7)
```

The Pragmatic Inquiry Process in RENISYS The pragmatic inquiry process could be implemented in RENISYS as follows:

1. Abduction

– *Hypothesis generation*
This stage can have different triggers: an individual user facing a breakdown, regular intervals, or the follow up of other inquiry processes that have reached their inductive (testing) stage. Hypotheses can be generated automatically (using some forms of graph expansion, for instance) or manually in informal discussion, as in the case example.

– *Hypothesis selection*
The selection of hypotheses is a key subprocess. The Peircean approach might include such notions as: select those hypotheses that are most natural, that are doable, that seem likely to create habits of value to the project, to individuals involved in it, to the community, etc. All this ought to conform to an "economy of research", thus to selection criteria.
The selection process goes as follows:

- Select the *personal viewpoints*, *criteria*, and *status* (e.g. proposed or tested) of the hypothesis that are of interest for the selection.
- Create one or more *selection graphs*.
- *Project* these graphs onto the set of all hypotheses.
- *Interpret* projection results.

For example: suppose that in stage (5) a new system manager is hired, whose priority is user-friendliness. We therefore should select all hypotheses that match with the user-friendliness criterion and have not been successfully implemented yet: either proposed ones (still to be tested) or failed hypotheses (those that were tested, but failed, but, with extra effort might now be successfully implemented).

We construct the following hypothesis *selection graphs*[6].

```
[Prop_Hyp: [Definition] -
   (Chrc) -> [User-friendly]]. (s1)

[Failed_Hyp: [Definition] -
   (Chrc) -> [User-friendly]]. (s2)
```

We now project both s1 and s2 on the set of hypotheses H, resulting in set H', containing only the specializations of either selection graph: H = {h3,h4,h5}. When interpreting these specializations, Users #1 and #2 both agree that BSCW is not satisfactory from the user-friendliness point of view (h3,h5), and that the alternative proposed by User #2, namely only to use PORT@home and not BSCW for link storage purposes (h4) could work. User #1 (the system manager) therefore agrees that, in the current situation with extra development capacity, it may be useful to investigate if replacing BSCW by PORT@home for the purpose of link storage is more user-friendly. Therefore, hypothesis h4 is amended by adding the system manager User #1 as a supporter, and selected for testing:

```
[Prop_Hyp: "Use PORT@Home to support link storage process" -
      (Agnt) -> [User: {#1,#2}]
      (Chrc) -> [User-Friendly]]. (h4')
```

2. Deduction

In the deduction stage, the selected hypotheses are extended by automatically joining them with *testing condition graphs*. These graphs describe properties to be investigated in the inductive (testing) process.

The formal notation of the selected hypothesis h4' is an example of a required implementation definition. Such a definition links a workflow mapping to an enabling tool (the PORT@home server). A workflow mapping defines how an activity (link storage) is enabled by an information or communication process. In this case, we assume such a process to be URL-management.[7]:

```
[Prop_Hyp: [State: [Req_Impl: #124] -
              (Inst) -> [Web_Server: #PORT@home]
              (Obj) -> [Workflow_Mapping: #67] -
                 (Part) -> [Link_Storage]
                 (Part) -> [Interaction]
                 (Part) -> [URL_Management]] -
   (Agnt) -> [User: {#1,#2}]
   (Chrc) -> [User-Friendly]]]. (h4')
```

[6] Note that the graphs mentioned throughout this paper are not presented to or created by the user in raw conceptual graph format. Many systems, like WebKB, allow for (pseudo)-natural language translation, for instance.

[7] see [3] for details of the structure of these *required implementation* and *workflow mapping*-definitions

This means that, from a user-friendliness point of view and in the eyes of both User #1 and #2, PORT@home is the required implementation for all link storage processes that make use of URL management facilities.

These definitions can be extended in the deduction stage, by joining them with matching *testing condition graphs*[8]. These conditions are to be joined with the selected hypothesis, so that key aspects are not overlooked in testing.

For example, the following testing condition graph says that any URL-management implementation must be tested for proper password management:

```
[State: [Testing_Cond: #234] -
  (Obj) -> [URL_Management]
  (Chrc) -> [Password_Management]] (T1)
```

In the deduction stage, RENISYS tries to automatically join (the referents of) all testing condition graphs with the selected hypothesis. The join is tried on the object (defined by the (Obj)-relation) of the testing condition graph, in this case *URL-management*. Here, the join succeeds with T1 on h4', leading to the following definition to be tested in the induction stage[9]:

```
[Prop_Hyp: [State: [Req_Impl: #124] -
               (Inst) -> [Web_Server: #PORT@home]
               (Obj) -> [Workflow_Mapping: #67] -
                 (Part) -> [Link_Storage]
                 (Part) -> [Interaction]
                 (Part) -> [URL_Mgt] -
                             (Obj) <- [Testing_Cond: #234] -
                               (Chrc) -> [Password_Mgt]
  (Agnt) -> [User: {#1,#2}]
  (Chrc) -> [User-Friendly]]. (h4'')
```

3. Induction

The extended selected hypothesis is empirically tested in the inductive stage by implementing the tool change.

In the case of h4", the developers know that PORT@home should be the new tool for link storage management, that the purpose of this change is user-friendliness, and that they must check that password management is taken care of sufficiently. Once the tool change has taken effect, the proposed hypothesis h4" can be evaluated by Users #1 and #2 and possibly other users in a conversation for specification [2]. If they agree that the change has been successful, then the status of Prop_Hyp is changed into Succ_Hyp. A new inquiry cycle can then start.

---

[8] *Testing conditions* are another subtype of *Definitions* in the hypothesis ontology

[9] Of course, many more criteria and testing conditions would be defined in a realistic, complex development setting.

5 Conclusions

In this paper, we presented a pragmatic method for community information systems development. We combined the existing RENISYS method for legitimate user-driven specification with pragmatic principles from Peircean theory. The PORT (Peirce Online Resource Testbeds) project describes the link between theory and practice.

We put Peirce's theory of pragmatism into practice in two ways. First, by using his insights in self-organizing, purposeful communities. Second, by operationalizing, testing, and implementing his theories on pragmatic inquiry in an actual setting, the RENISYS method.

We demonstrated only a hint of the richness of Peirce's ideas here: a robust meta-method in which more refined insights can be integrated in the future, along with many Conceptual Structures tools, of which WebKB[10] is a prime example as it combines web presentation capabilities with powerful conceptual graph operations. Self-critical, evolutionary capabilities are essential for successful virtual communities. As Doug Engelbart says: we need to "improve the improvement process" [4]. The framework presented here might help focus methodological efforts towards a more Pragmatic Web.

References

1. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 2001. May 1.
2. A. De Moor. Composition norm dynamics calculation with conceptual graphs. In *Proceedings of the Eighth International Conference on Conceptual Structures, ICCS2000, Darmstadt, Germany, August 14–18, 2000*, 2000.
3. A. De Moor and W.J. Van den Heuvel. Making virtual communities work: Matching their functionalities. In *Proceedings of the 9th International Conference on Conceptual Structures, Stanford, July 30-August 3, 2001*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2001.
4. D. Engelbart. Toward high-performance organizations: A strategic role for groupware. Technical report, Bootstrap Institute, 1992.
5. P. Jarvis, J. Stader, A. Macintosh, J. Moore, and P. Chung. Representing and exploiting organisational structure and authority knowledge within workflow systems. In D. Bustard, P. Kawalek, and M. Norris, editors, *Systems Modelling for Business Process Improvement*, pages 81–94. Artech House, 2000.
6. M. Keeler. The philosophical context of Peirce's existential graphs. In *Third International Conference on Conceptual Structures: Applications, Implementation and Theory, Proceedings Supplement, Dept. of Computer Science, University of California, Santa Cruz*, pages 94–107, 1995.
7. J. Preece. *Online Communities: Designing Usability, Supporting Sociability*. John Wiley & Sons, New York, 2000.
8. R. Scheepers and J. Damsgaard. Using Internet technology within the organization: A structurational analysis of intranets. In *GROUP'97, Arizona, USA*, pages 9–18, 1997.
9. M. Surman and D. Wershler-Henry. *Commonspace: Beyond Virtual Community*. FT.Com Books, Pearson, 2001.
10. J.J. Zeman. Peirce's philosophy of logic. *Transactions of the Charles S. Peirce Society*, 22(1):12, 1986.

[10] http://meganesia.int.gu.edu.au/ phmartin/WebKB/