REENGINEERING A SOFTWARE REVIEW PROCESS WITH THE FUNCTION-BASED PROCESS ANALYSIS (FPA) METHOD

Claude Stricker*
Jintae Lee**

Department of Decision Sciences
College of Business Administration
University of Hawai'i

ABSTRACT

With the current popularity and success of the World Wide Web, the question, How can the WWW technology improve existing processes, is in the forefront of many managers' mind. This paper examines this question in the context of the software review process and proposes a method that helps us systematically answer it. The method, Function-based Process Analysis (FPA[26]), represents a process as a function lattice, in which the functions that the process is to serve together with the subfunctions that implement these functions form a lattice. In FPA a technology like the WWW is also represented as a function lattice. FPA then exploits the uniform representation of both a process and a technology to help us systematically identify and examine the aspects of the process that can benefit most from the given technology. The paper illustrates these claims by applying the method and generating WWW-based alternatives to a software review process currently employed at a major company. More valuable than the specific alternatives generated, however, is the systematic way that the method provides for examining the relation between a process and a technology as well as the framework in which the relevant issues can be placed.

INTRODUCTION

With the current popularity and success of the Web technology, any manager responsible for running a given process has probably wondered about how this technology can improve the process. Some managers have explored this question and reengineered their processes. For example, Seafood Credit implemented a Web-based system that maintains financial information on more than 50,000 companies across North America and allows its analysts and customers to instantly and remotely access all the credit information (cf. Baum (1997)). Patients by Kaiser can request a non-urgent appointment for a procedure like a cholesterol check, ask confidential questions of a nurse or pharmacist, and search a health-care encyclopedia for anything, directly by connecting to Kaiser Permanente Online web site (cf. Kalin (1998)). This reengineering phenomenon is nothing new nor peculiar to the Web technology. It happens with the emergence of any new powerful technology like Lotus Notes, CASE, or TQM. With new technologies to come in the future, researchers as well as managers will continue to ask the question of how these technologies can help improve their processes.

Few methods exist, however, that help us systematically answer this question, and one has to rely on inspiration or serendipity for their answers. To be sure, exploring how a given technology can change and improve the existing process will always require good dosage of insights. However, we claim that it is possible to have a systematic method that, given descriptions of an existing process and of a technology, helps identify the list of areas where the technology can be used to improve the process. We claim further that such a method can help generate insights about how they can be improved, provide a checklist for completeness, and support reuse of the analysis process.

---

* Email: claude.stricker@epfl.ch

** Email: jl@hawaii.edu

[26] We are aware that sometimes FPA stands for Function Points Analysis (International Function Point Users Group '94). However, the acronym FPA will be still used for the method proposed here in order to preserve continuity with our earlier work.

This paper describes such a method called Function-based Process Analysis (FPA) and illustrates how it has helped us think about using the Web technology to improve a software review process. FPA represents a process as a *function lattice*, an extended function decomposition tree with multiple inheritance. FPA also represents a technology like the WWW as a function lattice. FPA then exploits the uniform representation of both a process and a technology to enable us to systematically identify and examine the aspects of the process that can benefit most or is easier to implement with the given technology. The paper illustrates these features by applying the method to a software review process currently employed at a major company and examine which aspects of the process can be improved by the Web technology (in particular, the Web Document Annotator technology).
The rest of the paper is organised as follows. The FPA method is described in the next section. Section 3 then briefly describes the software review process that has been examined. Section 4 illustrates the process of constructing the function lattice representation with the software review process as an example. Section 5 describes the WWW document annotator technology and its function lattice. Section 6 describes how the FPA method has been used on the two lattices, of the software review process and of the WWW document annotator technology, to explore the different ways in which the software review process can be improved with the technology. Section 7 concludes the paper with a few caveats in the use of this methodology as well as the current status and future directions of the FPA research.

FUNCTION-BASED PROCESS ANALYSIS

Definition

Function-based Process Analysis extends the function decomposition method[27] for the purpose of process modeling and redesign. In FPA, no distinction is made among processes, organisations, or technologies. They are all viewed and represented uniformly as a lattice of functions.
In the systems analysis context, Gibson & Hughes (1994) define a function as an "ongoing activity a business performs to further its mission" (e.g. Manufacturing). Note that "function" is used here not in the mathematical term (input/output) sense, but in the sense of objective or feature. A function can be decomposed into a set of subfunctions (e.g. Product Research and Development, Product Production, and Product Distribution), i.e. the functions that help achieve the original function (i.e. Manufacturing in our example). Each of these subfunctions can then be decomposed into its own subfunctions. A tree that results from this repeated decomposition is called a function decomposition tree (cf. Table 1).

Manufacture Product
    Maintain Operation
        Maintain Parts
        Production Operations
    Research and Development
    . .
Distribute Product
    . .
Market Product
    Market Analysis
    Advertising
    Research and Development

Table 1. Example of function decomposition tree

A function decomposition tree provides a good conceptual overview of the organisational activities in support of the overall objectives. Creating a function decomposition tree also forces one to be explicit about how a function is currently implemented (descriptive) or to be implemented (prescriptive).

---

[27]Description of function decomposition method are given in many software engineering methods, e.g. Martin, (1990)

FPA extends the function decomposition method in several ways. The extensions include: generalized notion of function, function sharing via multiple parents, introduction of AND and OR constructs that resolve expressive semantic ambiguities, and additional relations such as *specialisation* and *conflicts* that complement the traditional *decomposition* relation. This paper will describe only the first two extensions, which were used extensively in the project being reported.

In FPA, "function" is generalized to include not only activities but also any feature whether it is a desirable feature or an existing feature. In particular, not only *Market product* or *Advertising* is a function, but also is *Make profit* (desired feature) as well as *Electronic mail* (existing feature). In other words, FPA views organisations, processes, technologies indiscriminately as collections of functions because at some level of abstraction they all exist to serve functions even though some of them may be intangible (e.g. *Promote the feeling of loyality among the employees*) or outdated (e.g. *Circulate physical copies of the monthly reports.*). A consequence of this uniform representation is that one can create a function lattice which consists of goals, which are implemented by activities, which in turn are implemented by technological features without worrying about where their boundaries are. Also it means that this function lattice could be generated by synthesizing the lattices that represent objectives, processes, and technologies. For example, the activity *Advertise* can be decomposed into *Internet Ad*, *Magazine Ad*, *TV Ad* that implement this activity. *Internet Ad* can in turn be decomposed of the technological features that make it up. Table 2 shows an example of a function tree with extended notion of function.

Market Product
- Market Analysis
- . .
- Advertising
  - Internet Ad
    - WWW
      - WWW Server
      - WWW Client
      - WWW Protocol
    - Electronic Mail
    - Internet
  - Magazine Ad
  - TV Ad
  - . .
- . .
- Research & Development

Table 2. An Example of function tree with extended notion of function

FPA also allows multiple functions to be implemented by a given function. That is, the same function can be a subfunction of more than one function. This multiple "parents" allows us to express the fact that one function can help achieve multiple objectives. For example, if *Market Analysis* helps not only *Market Product* but also *Distribute Product*, then *Market Analysis* will be a subfunction of both. This is called "function sharing" in design literature. This function sharing is represented in the lattice by a function linked to multiple parent functions. (cf. Figure 1). The lattice representation makes clear any occurrence of function sharing by having multiple links coming out of the node being shared, whereas in the outline representation (cf. Tables 1 and 2) function sharing is more difficult to detect because a node being shared is multiply copied where appropriate in the outline.

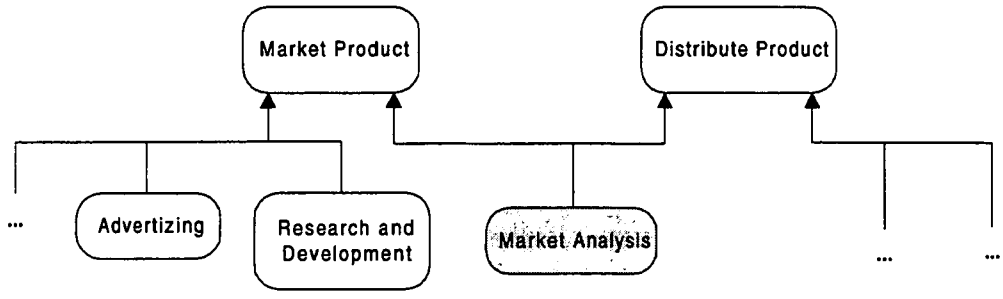

Figure 5. Function sharing

With this extension, FPA helps us:

- model a process as a function lattice
- model a technology as a function lattice
- identify the areas of the process that can be redesigned and improved with the technology.

Modeling a Process

FPA models a process by identifying the functions of the process, the subfunctions that implement these functions, and the subfunctions that implement these subfunctions, and so on recursively. This generates a function tree or lattice, whose root consists of high level functions that the process is designed to serve (e.g. producing good quality software for software review process) and whose leaves consists of the most specific subprocesses such as the individual speech of a meeting's participant.
A function lattice can be built from top-down and from bottom-up. The top-down construction proceeds by identifying the high level objectives (e.g. detect defects) that the process is to serve and examining how they are implemented with more and more concrete activities. The bottom-up construction proceeds by examining the observable, individual activities (e.g. moderator asks opinion of author) and articulating the higher level functions that they serve. The concurrent use of the top-down and the bottom-up approaches help us not only build a model of the process but also help us to identify the areas in the current process that can be improved, as discussed further in Lee (1993).

Modeling Technology

A function lattice can also be used to represent a technology because a technology such as WWW can be viewed as a collection of functions, implemented in terms of subfunctions, which in turn are implemented as subfunctions, and so on. As in the case of process modeling, a technology can be modelled top-down by identifying the high level functions that the technology provides (e.g. access documents anywhere anytime) or bottom-up by identifying the purpose of an existing feature (e.g. http protocol).

Identifying the areas of the process for improvements

In order to identify the areas that can be redesigned to benefit from the given technology, the two function lattices representing the process and the technology can be matched systematically. That is, for each function in one lattice, one asks whether one can find a similar function in the other lattice. If so, then one can examine whether the

subfunctions that implement this similar function can be used to implement the function in the first lattice. If the alternative implementation offers any advantage, then one marks the current subprocess as a potential area for redesign. Section 6 illustrates this step with examples. Although a typical procedure is to start with the subprocesses (i.e. subfunctions) that are known to be problematic or inefficient (e.g. too much searching time of relevant documents), this step can be in principle performed for each of the functions in the lattices.

SOFTWARE FORMAL REVIEW PROCESS

In the organisation we have studied, the quality assurance standard is implemented through a software formal review process adapted from Fagan (1976) and Freedman & Weinberg (1990). The goal of the review process is to prevent models, concepts, design and codes from major problems by detecting and correcting defects before the product tests. It consists of a methodical inspection by reviewers of different types of artifacts (code sources, specifications, data models, etc.). The reviewers are usually experts in the domain of the artifact and are selected outside the project team.

In the first stage, the reviewers try to detect potential defects by reviewing privately the artifact usually with a checklist. Then they review the artifacts again, but this time in a group meeting. The review process organizer plays the role of a moderator, guiding the discussion between the reviewers. The goal of the meeting is to discuss the potential defects, agree on their seriousness, and produce an official report stating which defects should be corrected. The process terminates with the corrections produced by the author and approved by one of the reviewers. Other important goals of this formal review process are to develop a better understanding of the work products and to improve the skill of all the review participants.

The main steps of the software review process are shown in Figure 2. The first step is preparing the resources for the next steps. The main input resources to the reviewer's preparation, besides the reviewers, are the artifacts to be reviewed and the list of checkpoints. An essential input resource to the group meeting is the potential defects, detected in the preceding step by the reviewers. These potential defects are discussed during the review meeting.

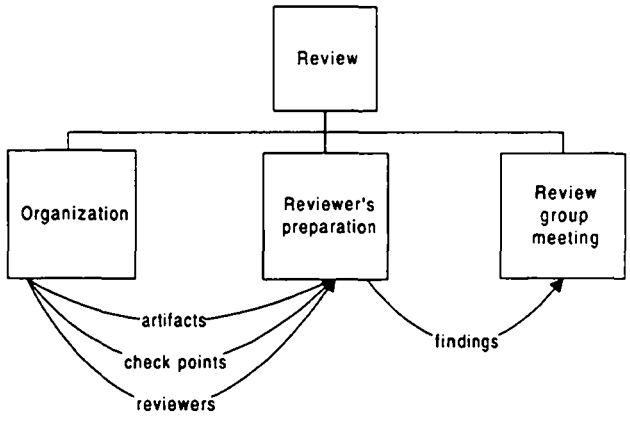

Figure 6. Main functions or chronological steps of the formal review process and their input/ouptut.

SOFTWARE REVIEW PROCESS AS A FUNCTION LATTICE

Before we began building the function lattice, information about the goals, steps and activities of the process were gathered by looking at the document standards, by interviewing the process participants, by collecting questionnaires, and by observing a meeting recorded on video tape. The lattice has been built by using concurrently the following top-down and bottom-up strategies.

Top-down strategy

The most general goals of the process are placed at the top levels of the lattice. In Figure 3, the most general goal *Improve quality of software products* is decomposed into *Prevent models, concepts, design and codes from major problem, Develop better understanding of the work products* and *Improve developers' skill.* These goals have been directly extracted from the gathered information or from discussion with the people. Each of them is then decomposed down further to the level of activities implementing them. In some cases, a subgoal or a subfunction serves more than one function. For instance *Review artifacts formally* shown in the diagram of Figure 3 serves the three goals *Detect defects before testing, Develop better understanding of the work products* and *Improve developers' skill.* As discussed above, the lattice allows the representation of the function sharing as multiple parents. Curved lines indicate that the subfunction is an alternative way to implement its parent. Dots indicate that the decomposition includes more subfunctions, but they are not represented in the diagram for space reason.

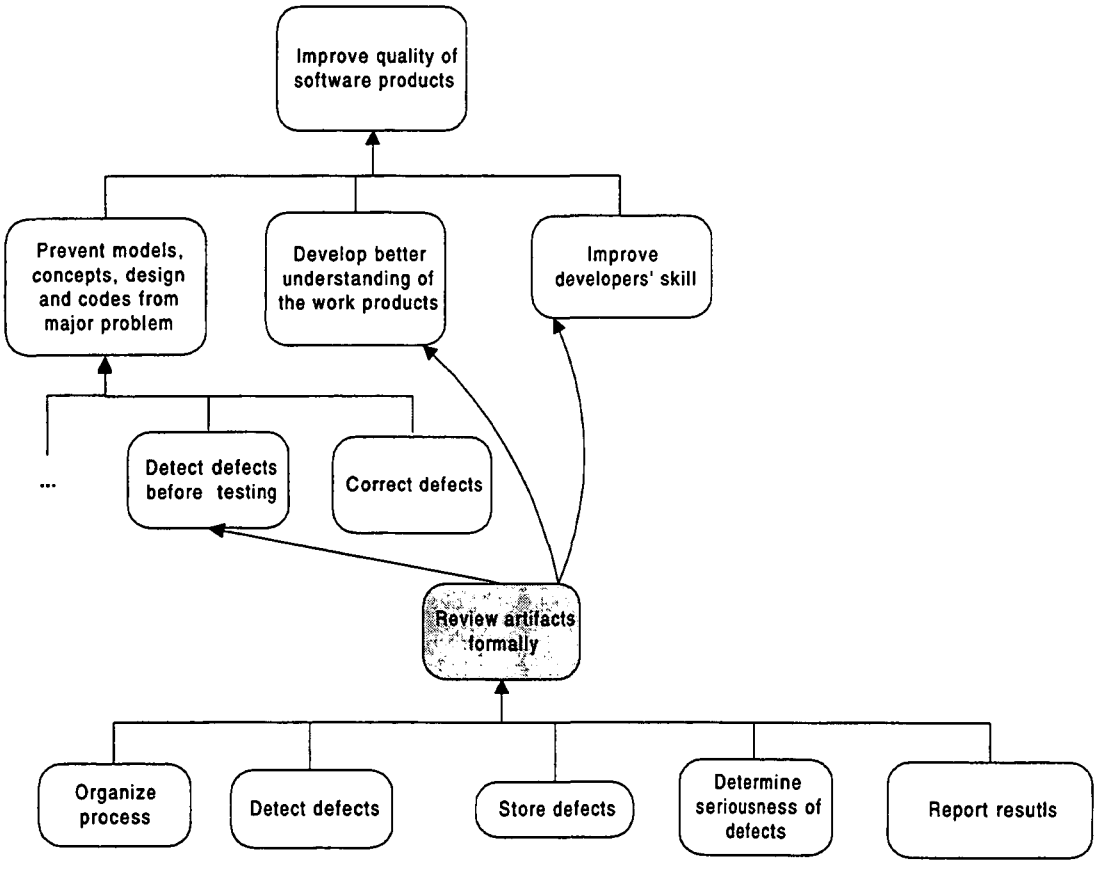

Figure 7. Top goals and major functions of the software review process

At the lowest level, activities directly observed during the review meeting are placed. These activities can be as detailed as individual speeches of the participants. Figure 4 shows how the function *Determine seriousness of defects* is decomposed down to the level of activities performed during a review meeting

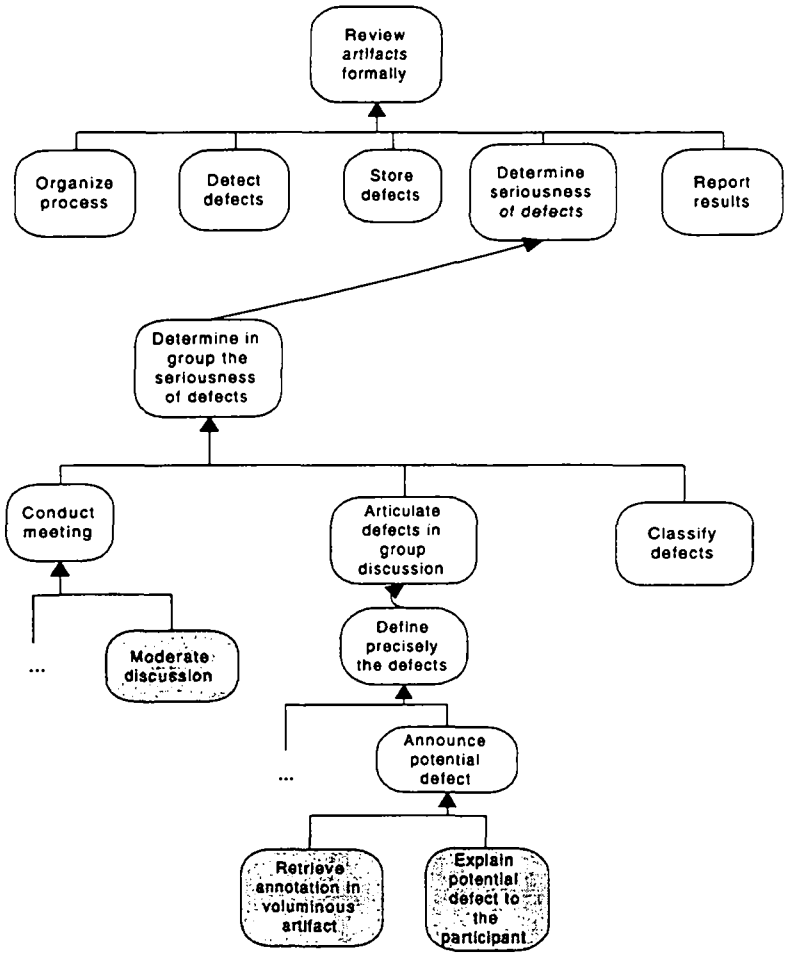

Figure 8. Detailed activities' decomposition of the meeting

Bottom-up strategy

In conjunction with the top down strategy, a bottom up strategy has been applied in the construction of the lattice. In some cases, observed activities could not be attached anywhere in the function lattice created during the top-down strategy. For example, one of the activities observed is that the *Moderator asks opinion of author.* However this new activity can be placed in the existing lattice only if two intermediary nodes, *Moderator checks author's understanding* and *Check validity of findings,* are introduced to make clear the function of this activity (cf. Figure 5).

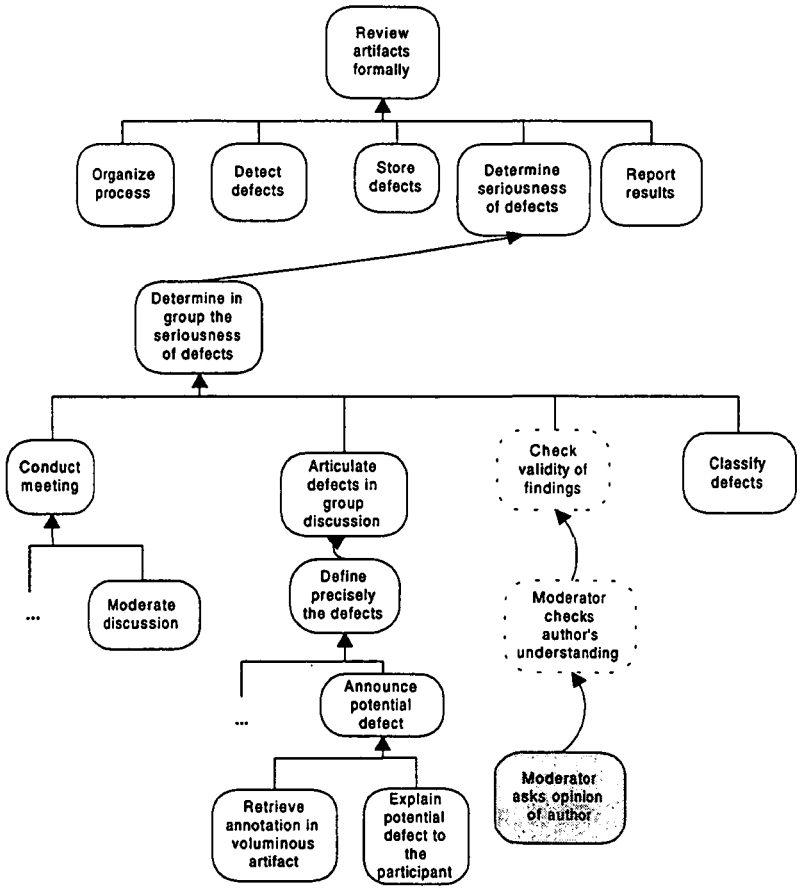

Figure 9. Bottom-up construction of the lattice

The final lattice of functions contains more than 300 nodes, which have been validated and refined based on the feedback provided by the people involved in the process itself.

In the course of building the function lattice, FPA helped us identify potential problem areas by forcing us to articulate functions at appropriate levels of abstraction. For instance, it was noticed in the recorded meeting that reviewers had difficulty retrieving findings. They spent much time browsing through their voluminous artifacts in order to find the location of written notes and then trying to remember what their notes said. This observation was captured in the lattice as the function *Reviewers retrieve annotation in voluminous artifact* as shown in Figure 6. At first, no parent function seemed to be served by this function. However, because we knew that this function exists to serve the reviewers remembering the defects found earlier, the parent function *Reviewers remember findings* was added under the pre-existing function *Announce potential defects.* This newly identified function then served later as a starting point for exploring alternative ways of remembering. (cf. Figure 6).

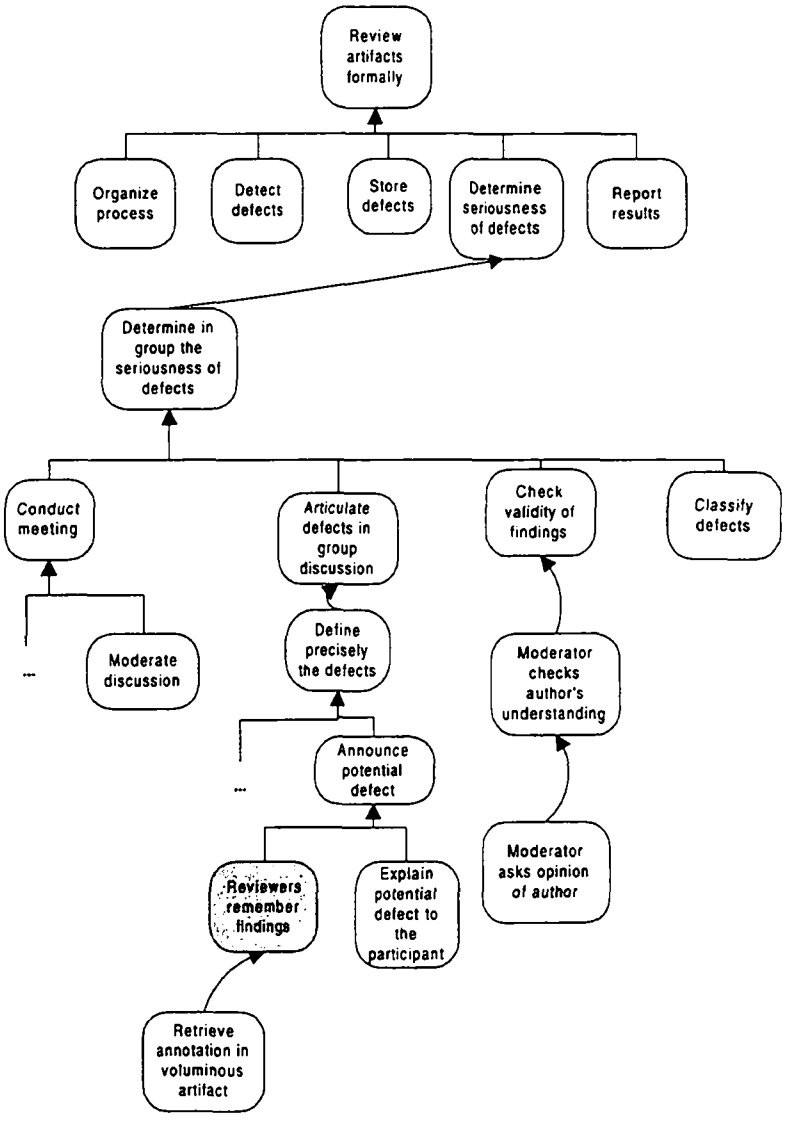

Figure 10. Placement of function at the appropriate level of abstraction

FPA also helped us to define the desired level at which metrics could be computed. For example, a metric for the meeting management overhead could be computed by summing over the time duration associated with the following functions: *Articulate defects in group discussion*, *Check validity of finding* and *Classify defects* (cf. Figure 7.). The total time associated with the time function, it turns out, was 75 % of the total duration of the 2 hours meeting. It means that a quarter of the time of the meeting was devoted to the function *Conduct meeting*, which contained the activities of meeting management and moderation.

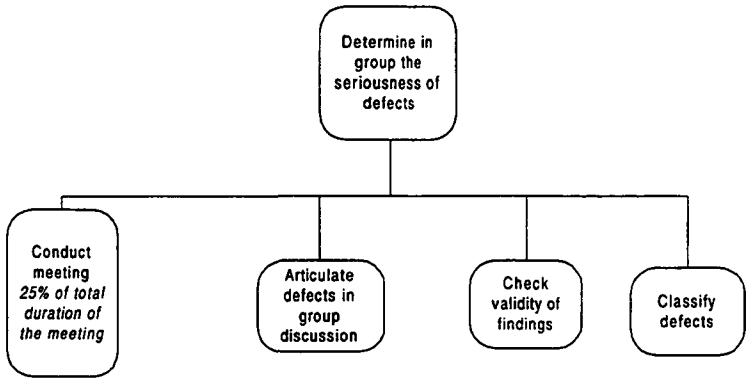

Figure 11. A sublattice can help define a process metric by revealing the appropriate level at which individual submetrics could be aggregated.

W3 DOCUMENT ANNOTATOR (WDA) TECHNOLOGY AS A FUNCTION LATTICE

In order to illustrate how the FPA method can be used to model a technology, W3 Document Annotator groupware system built by Schenk (1995) has been modelled, It also provides the annotation technology that is necessary for the Web-based software review process. However, other group annotation technologies[28] could have been chosen for illustration here equally as well.

WDA is a system that supports group annotations on the World Wide Web documents. Annotations may be linked to any specific part of a document and may themselves be further annotated. Based on the "argumentation model" gIBIS of Conklin and Yakemovic (1988) and SYBIL of Lee (1990), it allows a group of people to enter and retrieve annotations linked to any pre-defined part of a WWW document. In particular, a user can propose a topic of discussion by issuing proposals. The participants can respond to each proposal with arguments that either support or refute the proposal. Arguments can be supported or countered with another argument and so on. The user can browse through the list of annotations and their corresponding arguments, and access their content displayed in a separate WWW document.

Figure 8 portrays a part of the lattice of functions constructed for WDA, following the strategies described earlier in Section 2 and 4.

[28] See for instance: WIT (W3 Interactive Talk) of Ari Luotonen at http://www.cern.ch/wit

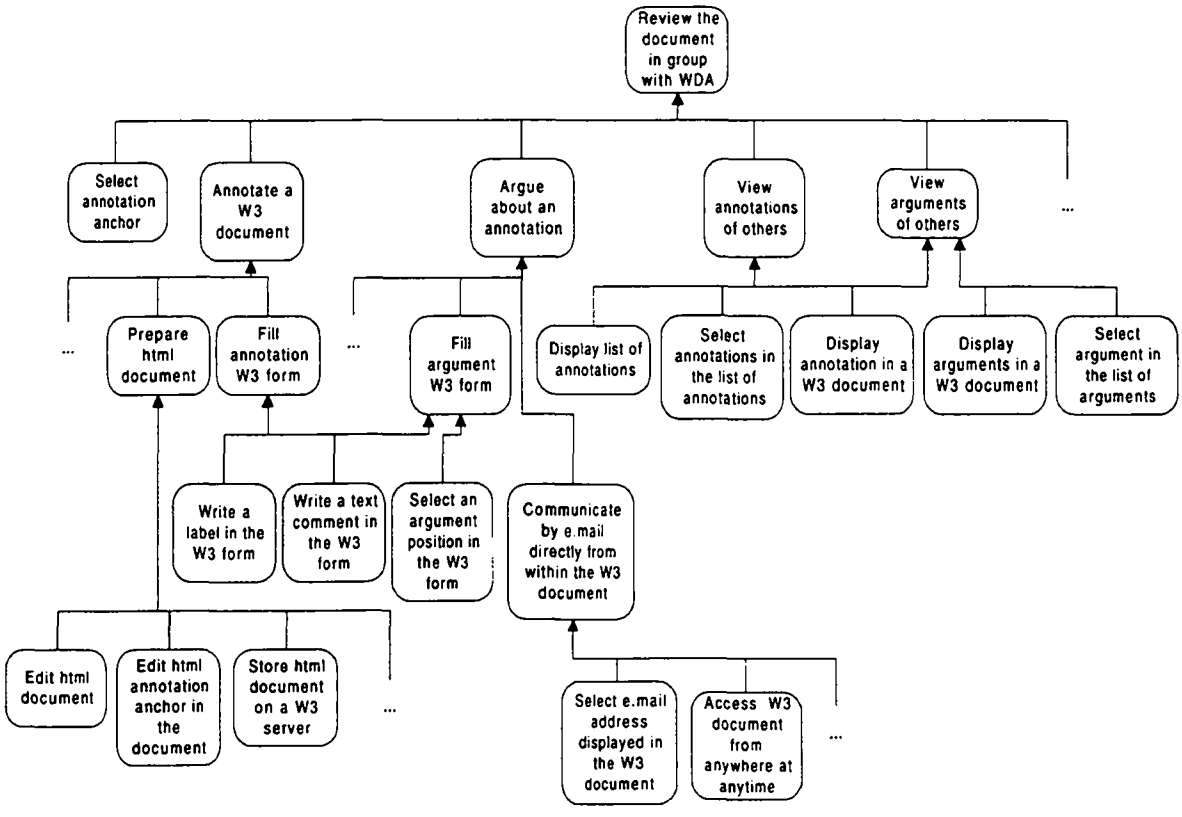

Figure 12. The function lattice of the WDA technology

WWW-BASED FORMAL REVIEW PROCESS SOLUTION

When the function lattices of both the process and the technology have been constructed, the next step is to find if certain functions in the process lattice can be replaced or implemented by the functions in the technology lattice by systematically comparing the two lattices. First the matching process is described, followed by the description of the matching results.

Matching process

First, for each function of the review lattice, the WDA lattice is searched[29] for a similar function, or for a function which could implement it. A similar function is an action that is described by the same verb or by a synonym, a hyponym or a hyponym of the action's verb used in the review lattice. For example, the software review process function *Reviewers argue* is similar to the WDA function *Argue about an annotation* (cf. Figure 9).

[29] The WDA lattice is searched systematically by following a tree traversal method, i.e. by following each branch of the lattice from the top nodes, down to the detailed subfunctions.

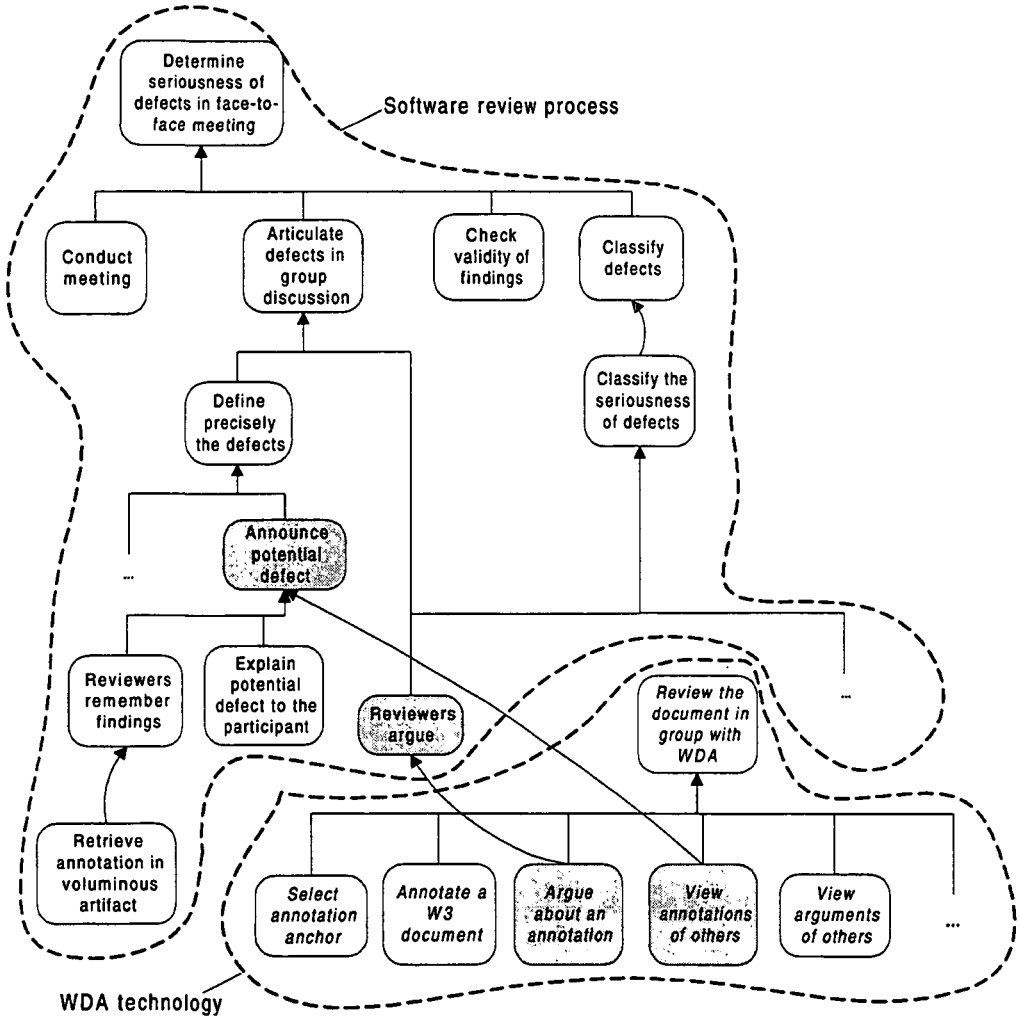

Figure 13. Matching software review functions with WDA functions

Another example is the function *Announce potential defects* which is currently implemented by the functions *Reviewers remember findings* and *Explain potential defect to the participant.* It could be implemented by the WDA function *View annotations of others*, if the reviewers would annotate the potential defects in the html document, in order that it can be retrieved and displayed to the other reviewers.

Alternatively, for each function describing the WDA system, the software review lattice is searched for similar functions or for functions that could be potentially implemented by the WDA function. For example the WDA function *Annotate the W3 document* could potentially implement the software review function *Annotate artifact*, as shown in Figure 10.

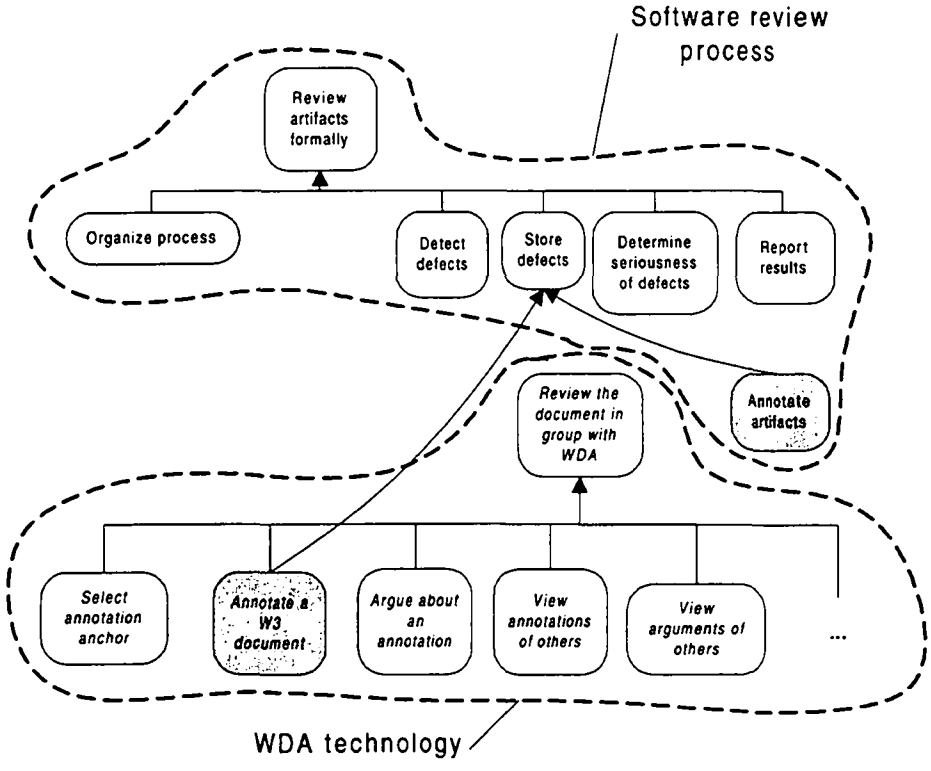

Figure 14. Matching WDA functions with software review functions

In many cases, one need to fully understand the context of the current function, before marking it a potential alternative. For instance, before deciding that *View annotations* in the WDA lattice can be used as an alternative implementation of the software review function *Announce potential defect*, one needs to know that reviewers propose their finding based on their previous annotations and that announcing a finding is made in the context of reviewing a work product or "document". This context information can be obtained by examining the parent nodes of the process. For example, one can examine the parent nodes of *Review artifacts formally*, shown in Figure 3, to make sure that the objects of the review are those that can be automated. However a large part of the comparison process still has to rely on the analyst's knowledge of both domain. Successful use of the FPA method usually requires that the analyst be the same one who build the function lattices to ensure that the analyst is familiar with the domains as well as the ways in which they have been represented.

In order for the revised lattice to work, however, it needs to make sure to incorporate not only the nodes implementing the present function but also all the other nodes required by any of the nodes being imported from the technology lattice. For example, the function *Annotate the W3 document* is a potential alternative to the software review function *Annotate artifact* for implementing the parent function *Store defects*. However, simply replacing this function (along with all of its subfunctions) for the *Annotate artifact* would not work because it also requires the function, *Edit html annotation anchor in the document*, which appears in another branch of the WDA lattice, i.e. under the function *Prepare html document* (cf. Figure 11 and Figure 8).

Consequently one must also place the latter function in the revised lattice. In selecting an adequate place under which to place this function, one must first look for a function which could be implemented *by Prepare the html document.* In the present example, the function *Organize process* is a good candidate because it contains the activities of collecting and distributing the material to be reviewed and can be implemented by *Prepare the html document.*

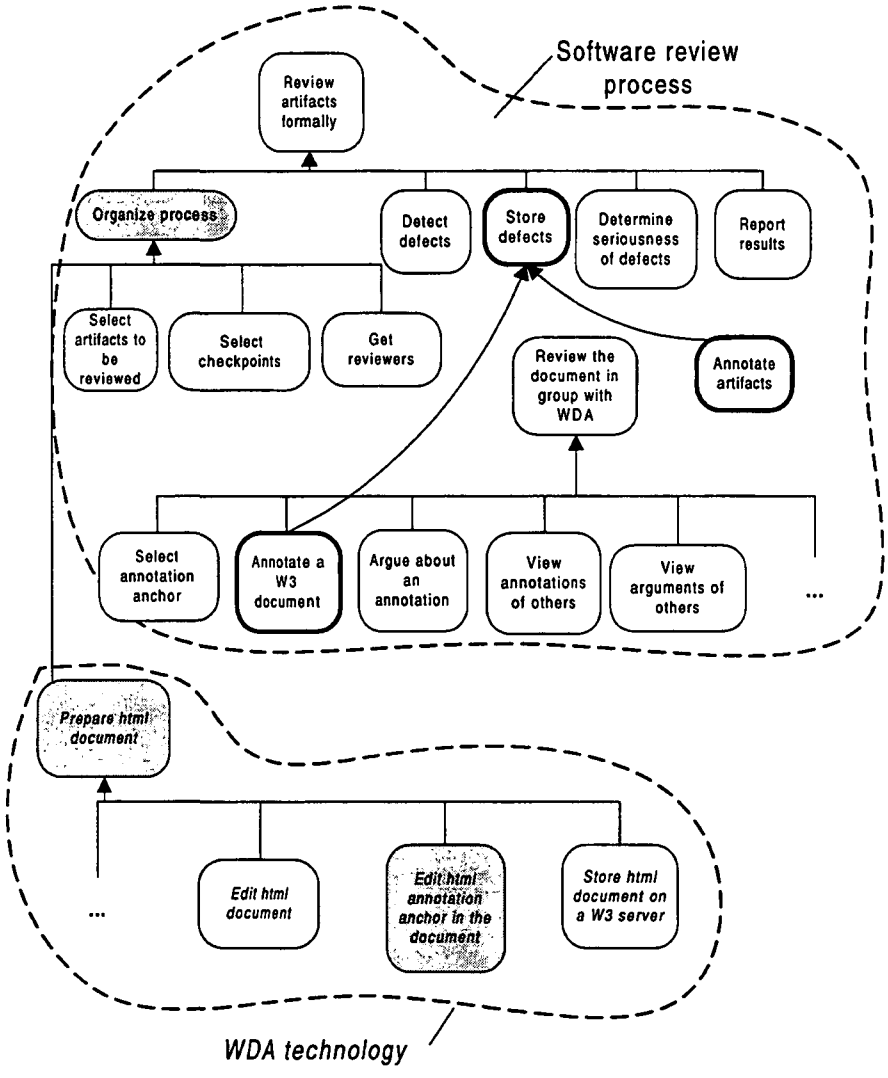

Figure 15. Placement of prerequisite functions

In some cases, one has to create intermediary functions between an existing parent function and the function to be placed, in order to better represent the rationale of the new function or to better represent alternatives composed of a group of functions. In Figure 12, *Determine seriousness of defects asynchronously* is created between *Determine in group the seriousness of defects* and *Review document with WDA.*

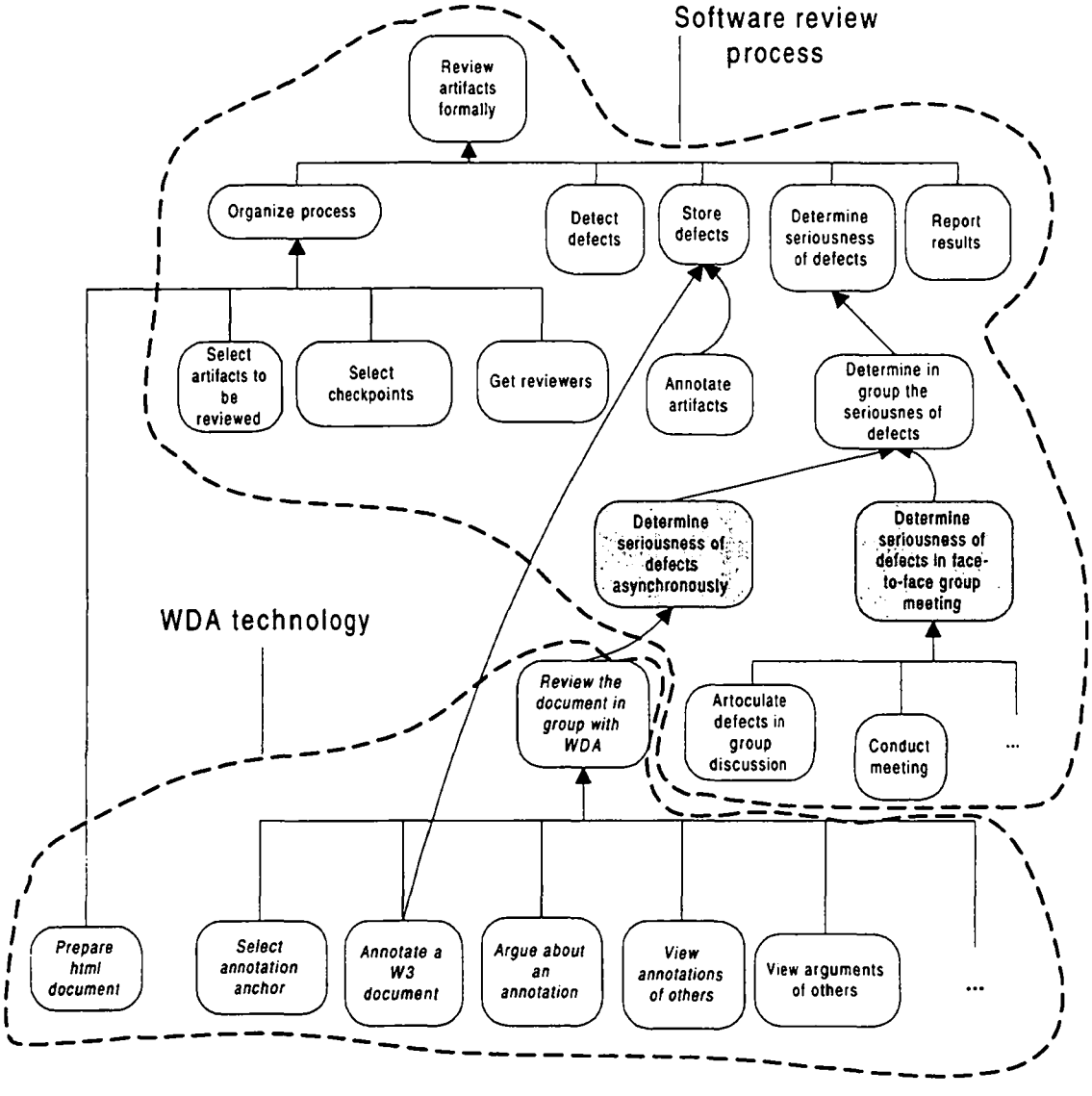

Figure 16. Adding functions for better understanding

Alternative WWW-based solution

A number of possible redesign alternatives have been already mentioned briefly while describing the matching process. Two of them are described below in detail: *Select artifact* and *Detect defects* processes.

One of the software review functions that can benefit from the WWW technology is *Select artifacts to be reviewed* (cf. Figure 13). The subfunctions of *Select artifact to be reviewed* show that the coordinator must find the information about the risks and the potential defects associated with the artifacts. Currently this is usually done by contacting the people such as project managers who would have such information. This need for human contact often cause delays the whole review process. In examining the WDA lattice for an alternative, one finds the function *Access documents from anywhere at any time*, suggesting the possibility of eliminating this source of delay. The relevant information can be listed in a WEB page and updated by the project managers. Other WEB pages could contain the status of the monitored risks and the task assignments. The review coordinator can then access these documents and get the needed information asynchronously.

Another feature of the WWW-based technology appears in the WWW lattice as the subfunction *Email address embedded in WWW document* serving the function *Communicate by email directly from inside a WWW page* (cf. Figure 13). Many of the activities of the review coordinator appear in the review lattice as communication between him and the process participants (reviewers and author). This function of communication could be potentially served efficiently by the WWW communication feature, where a shortcut between the documents and information exchange with the participants is easily implementable.

Another software review function that can be improved by the WDA technology is *Detect defects.* In the software review lattice, the function *Detect defects* is implemented in two ways (cf. Figure 13). On the one hand, possible defects are found during a private examination by the reviewers before the group meeting. On the other hand, potentially remaining defects are discovered during the group meeting by discussing the previously found defects. These two subfunctions are very different from a logistic perspective, but both contribute to the same goal *Detect defects*. Detecting defects during the group discussion, in turn, is implemented by the functions *Announce potential defects* and *Store defects. Announce potential defects* is implemented by the reviewers explaining during the meeting the findings that they have previously annotated in the artifacts. *Store defects*, in turn, is implemented in two different ways. Firstly the reviewers annotate the artifacts during their private examination before the group meeting. Secondly the classified defects are recorded during the meeting.

In the WDA-based implementation of these functions, the entire process becomes asynchronous and simpler due to the ubiquity that the WWW technology provides. Moreover the functions appearing in the WDA lattice suggest that the storing of the findings as well as their announcements and discussions (argumentation and counter-argumentation in favor of or against the official classification of a defect) could be alternatively implemented by the WDA functions *Annotate the W3 document* and *Argue about a selected annotation* as shown in Figure 13. Each reviewer would go through the complete set of annotations created by all the reviewers and react by entering arguments and counter-arguments.

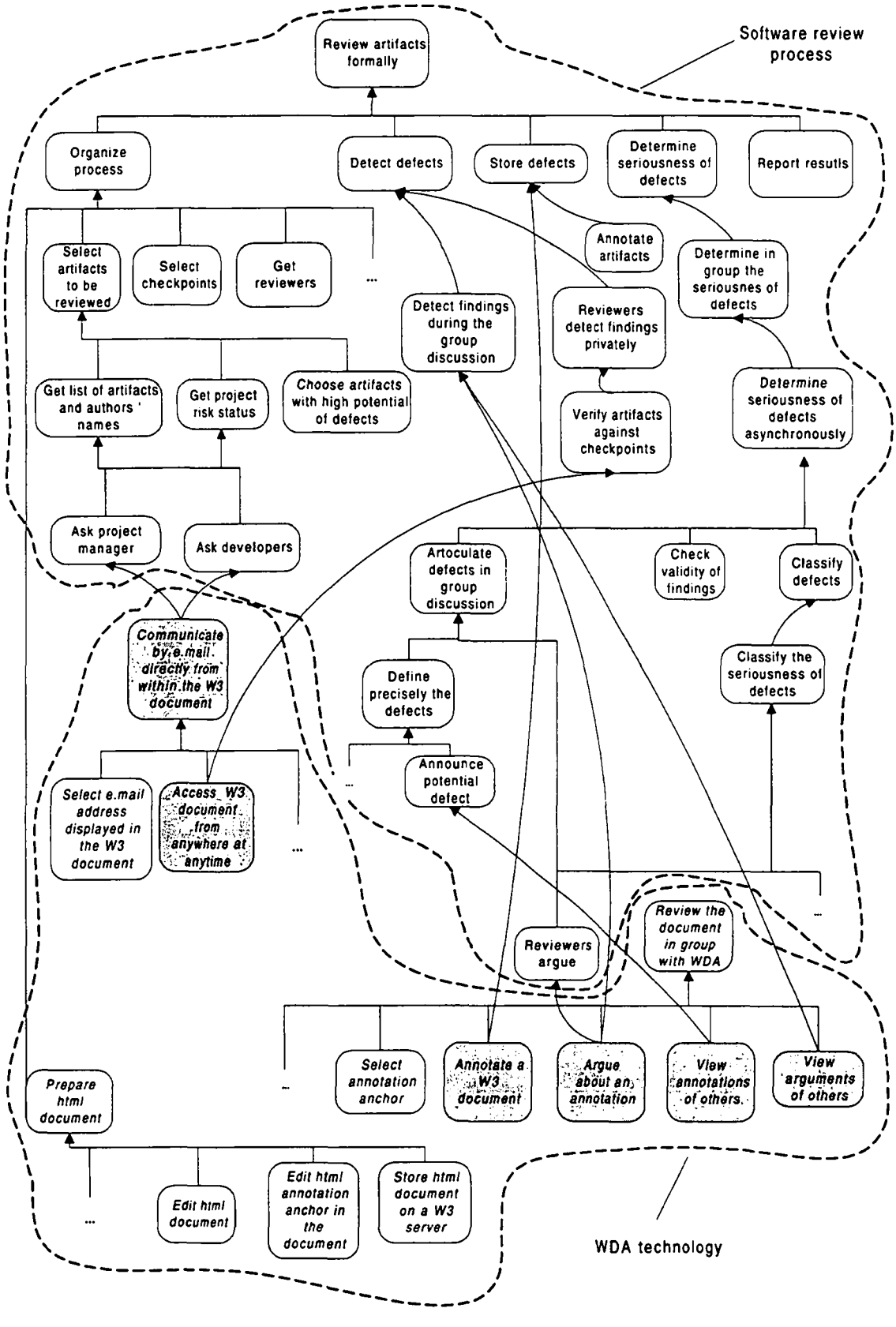

Figure 17. Partial view of the redesigned software review process with WDA technology

CONCLUSION

This paper describes a method for systematically exploring the ways in which a given technology can improve an existing process. It shows how a process can be modelled as a function lattice and how the construction of the lattice helps identification of potential problem areas. It also shows how a technology such as the WWW Document Editor can be represented as a function lattice.

With both a process and a technology represented in the common representation, i.e. function lattice, it shows how a systematic matching between them can help us identify the features of the technology that can implement some parts of the process. Whether the actual implementation of the technology will in fact improve the process is an empirical question that depends not only on the technology alone but also on many other factors such as how it is deployed and integrated in the overall workplace setting. However, this preliminary work has demonstrated the ways in which the method can help generate potentially promising alternatives based on the technology.

A few caveats about the use of the method are in order. As mentioned earlier, the method still requires an analyst with good understanding of the process and the technology because constructing and matching of function lattices requires detailed knowledge of the ways in which the functions have been represented as well as the context information that may not be explicit in the lattices.

Secondly, one has to be careful with the dependencies that hold among the process steps. A function lattice representation is not expressive enough to capture all the dependencies that may hold among the subfunctions or subprocesses. As mentioned briefly in Section 6, when considering whether a set of functions can implement a given function, currently one needs to check manually to make sure whether there are other functions on which this set depends. If so, one needs to add the additional functions and consider the implications. In Malone et al. (1993), we explore a representation that makes explicit these dependency relations so that a system can help explore the implications.

The third caveat is that so far in the discussion of FPA, it is assumed that a technology such as the Web comes in individual features. But sometimes a given technology has to be accepted as a whole or not at all. It implies that even though some part of a technology may help improve the current process, adopting it as a whole may introduce more problems, apart from the usual training problems, because of the other features that bring with it (e.g. security, incompatibilities).

The FPA work is continuing in several directions. First additional ways to support the process of comparing function lattices, are explored. A generic taxonomy of functions is articulated, in order to facilitate the search, creation, navigation, and matching of function lattices. In this taxonomy, a new function is created as a specialisation of an existing function and inherits the attributes of the parent function (such as the agent, objects, and instruments, associated with the function). These attributes, in turn, can be used to acquire relevant information from the user in characterizing a new function or searching in existing function.

Although the use of FPA has been described in exploring how a given technology can improve an existing process, its use in evaluating which of the technologies should be most appropriate for a given process is also being explored. For this purpose, a library of function lattices representing different technologies such as the WWW, the Lotus Notes, and other groupware tools, is created. One may then use the same matching procedure as described in this paper not against one technology but against all the technologies in the library and see which of them produce the best result.

The FPA can also be used as a framework for comparing and contrasting a given group of processes or technologies that reveals what is generic and different about them. This use of FPA, however, requires, in addition to the decomposition relation, the specialisation relation which was not described in this paper and is a topic of another paper.

ACKNOWLEDGEMENT:

The authors would like to thank the anonymous reviewers for their useful comments and suggestions. The authors would like to also thank Cecile Bruellhart, Thomas Mueller, Juerg Braun and Franz Brunner for their important participation in the gathering of the data. The work and the people at the MIT Center for Coordination Science have been also influential in the shaping of the ideas underlying this paper.

This work would not have been possible without the funding of the Swiss National Fund for Scientific Research (Grant no 1214-039627.93), Bank of Vaud, Centre Informatique de l'Etat de Vaud, Credit Suisse, Unicible, and by the support of the Department of Decision Sciences at the University of Hawaii (USA) and Elca Informatique in Lausanne (Switzerland).

REFERENCES

Baum, D. (1997) "Intranet Politics and Technologies" in **Byte**, North-America issue. May, 1997, Vol. 22, Number 5. Page 88F.

Conklin, E., J. & Yakemovic K. C. B. (1991) "A Process-Oriented Approach to Design Rationale" in **Human-Computer Interaction**, the Special Issue on Design Rationale. 6 (3-4), pp. 357-392.

Fagan, M. E. (1976). "Design & Code Inspection to Reduce Errors in Program Development". **IBM System Journal**, Vol. 15, No 3

Freedman, D. P. & Weinberg D. (1990). **Handbook of Walkthroughs, Inspections & Technical Reviews - Evaluating Programs, Projects & Products**. Dorset House Publishing. New York

Gibson, M. L., & Hughes, C. T. (1994) **Systems Analysis & Design**. Boyd & Fraser Publishing Company. One Corporate Place. Danvers, Mass.

International Function Point Users Group (1994). **Guidelines to Software Measurement**, Release 1.0, International Function Point Users Group (IFPUG) Standards, 1994. Westerville, OH

Kalin, S. (1998) "Kaiser Permanente Online" in **CIO WebBusiness Magazine.** CIO Communications, Inc. <http://www.cio.com/archive/webbusiness/050198_kaiser.html> 1.5.1998

Lee, J. (1990) "What's in Design Rationale?" in **Human-Computer Interaction**, the Special Issue on Design Rationale. 6 (3-4), pp. 251-280

Lee, J. (1993) "Goal-based Process Analysis". **Proc. ACM Conference on Organisational Computing** 11-93 Milpitas, CA

Malone, T. W., Crowston, K., Lee, J. & Pentland, B. (1993) "Tools for inventing organisations: Toward a handbook of organisational processes". Proceedings of the 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative, **IEEE Computer Society Press**

Martin, J. (1990) **Information Engineering - Planning & Analysis**, Prentice Hall, Englewood Cliffs, New Jersey

Schenk, M.-A. (1995) "W3 Document Annotator". Inforge Institute, Business School HEC, University of Lausanne. Technical Report 5-1995. Lausanne