Hochschule Reutlingen
Reutlingen University

Parallel and Distributed Computing Group
Department of Computer Science
Reutlingen University

Structured collaborative workflow design

Markus Held and Wolfgang Blochinger

(Accepted Peer-Reviewed Manuscript Version)

The formal publication is available at:
http://dx.doi.org/10.1016/j.future.2008.12.005

BibTeX

```
@article{Held2009,
       author = "Markus Held and Wolfgang Blochinger",
       title = "Structured collaborative workflow design",
       journal = "Future Generation Computer Systems",
       volume = "25",
       number = "6",
       pages = "638--653",
       year = "2009",
       issn = "0167-739X",
       doi = "https://doi.org/10.1016/j.future.2008.12.005",
       url = "http://www.sciencedirect.com/science/article/pii/S0167739X08002100"
}
```

Structured Collaborative Workflow Design

Markus Held Wolfgang Blochinger *

Eberhard-Karls Universität Tübingen
Symbolic Computation Group
Sand 14, 72076 Tübingen
Germany

Abstract

Workflow design is often an effort of distributed and inhomogeneous teams, thus making tool support for collaboration a necessity. We present a novel concept of collaborative workflow design which combines cooperation and workflow model analysis. Workflow analysis is simplified using workflow metrics, which help identifying problematic aspects of the workflow model. Our findings are implemented in a collaborative workflow design system, which is easily accessible on the Web, but provides a desktop like user experience.

Key words: Collaborative Workflow Design, CSCW, BPEL, Web 2.0, Rich Internet Applications

1 Introduction

1.1 Workflows as a Means and Product of Collaboration

In recent years Scientific Workflows have enormously gained importance [1], while Workflow Management has been a major aspect of enterprise systems since the 1990s. Based on Web Services, the Business Process Execution Language (BPEL) [2] has gained the status of a standard language for enterprise workflows. Scientific workflow management, on the other hand, still lacks common standards [3], although BPEL has successfully been applied [4] [5].

* *Email address:* blochinger@informatik.uni-tuebingen.de (Wolfgang Blochinger).
URL: http://www-sr.informatik.uni-tuebingen.de (Wolfgang Blochinger).

Contemporary software development in general is a task for teams, whose members are often scattered among different sites. A trend toward global software development makes it desirable to access resources anywhere and to coordinate development at distant sites [6]. Software development teams can be inherently distributed, as scientific research is often conducted in multi-institutional projects. Hence, collaborative software development tools become a necessity [7]. As discussed subsequently, this is especially true for workflow development.

The scientific workflow design process can involve people from very different backgrounds. Though some scientists will be eager to work with graphical workflow editors, it cannot be expected that they will fully comprehend the details of a workflow language. As [8] points out for engineering applications, we have to distinguish between domain and grid experts involved in workflow design. Domain experts bring in their knowledge about the high-level scientific processes to be modeled, while grid experts utilize their knowledge about grid infrastructure and implementation details.

Workflows often integrate capabilities and resources of distributed sites. Different research groups pool their services, thus forming *Virtual Organizations* [9]. *Gil* et al. point out that workflows are used by scientists "to collaborate with each other across organizations and physical locations" [10]. The activities of a workflow can thus include services provided by different vendors, business units, or research groups. Hence, the correct integration of these services may require special knowledge only available at a remote site. Moreover, workflow design is the most dynamic part in building and executing advanced grid applications, further driving the demand for sophisticated tool support with collaborative features. *Gil* et al. conclude [10]: "Scientific applications are driving workflow systems to examine issues such as [...] collaborative support for workflow design [...]."

In 1987 *Osterweill* has pointed out that "Software Processes are software too", which greatly inspired Software Process Improvement during the 1990s [11]. We argue, that workflows are at least as much a product of collaboration as they are a means of organizing collaboration. Summing up, we see a need of tool support for collaborative workflow design in a way that enables easy integration into existing Problem Solving Environments.

The same arguments for collaborative workflow design, we have presented here, also apply to enterprise workflows. In business, distributed software development teams are an effect of globalization and outsourcing. Moreover, enterprise workflows frequently involve several departments, which may be scattered across countries or even continents. Finally, business or engineering processes are designed by domain experts as well.

As a use case, we assume a life science project, that spans several research groups in different countries. In the project, a biologist from Germany and a physician from France are designing an in-silico experiment composed of several Web Services. New, non-trivial Web Services needed in the experiment are provided by research institutions in several European countries. To achieve a correct integration of these services, partners from the provider institutions have to be contacted. The results of the workflow are to be stored in a database located in Belgium. Figure 1 illustrates the situation of the project partners.

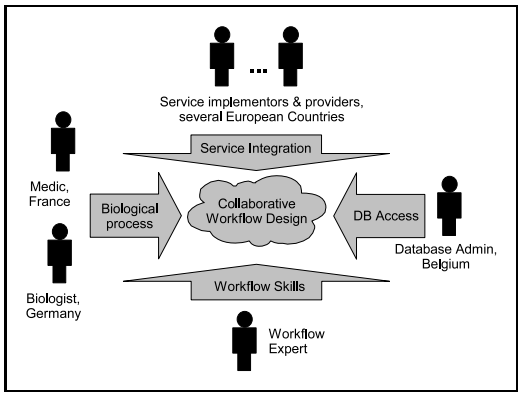

Fig. 1. A workflow design collaboration.

The workflow project is a collaboration of specialists that are too far away from each other to meet face-to-face. We can further observe, that the domains of expertise of the participants differ from each other, e.g. the biologist and the physician share knowledge about the biological process to be modeled, while the service providers know best how to configure their services. As none of them is very experienced in workflow design, they consult a workflow expert.

Bridging the gaps between distinct domains and cutting the geographic distances between team members requires support by a collaborative software system. Collaborative manipulation of a workflow model implies that no single person has a complete view of the model. However, a team leader needs to make decisions about how to proceed in the collaboration, e.g. by assigning tasks to team members. A collaborative workflow design system should thus help the team leader in assessing workflow models. Such workflow analysis support has to consider that the workflow model is ”under construction” and that it is the result of the cooperation of distinct partners. Verification techniques can prove the syntactic correctness of a workflow model, but will not help in identifying syntactically correct but troublesome parts of the workflow, i.e. design flaws. Although collaborative development improves the integration of domain expertise and technological expertise in a workflow, it also yields the risk that design flaws are left unnoticed, since no single member of the design team has a complete view of the model all of the time. Finding design flaws in source code can be simplified by using software met-

rics. As the team leader needs to identify design flaws in the workflow model, we propose to use workflow metrics in the collaborative workflow design system, i.e. workflow-specific software metrics. The team leader can use these metrics to gather information, which can help identifying design flaws in the workflow model.

Collaborative development of Scientific Workflows is a natural fit for Problem Solving Environments, which are often implemented as web portals. Moreover, department policies can prevent the installation of yet another software package on client machines. Thus, the collaborative workflow design system should be provided as a web application.

However, classical web applications cannot provide real-time interactivity and lack the look and feel of desktop GUIs. Many users, on the other hand, will only accept the system if they feel comfortable with its interface. Hence, we will use Web 2.0 technology, which enables desktop-like user experience and synchronous interaction of team members.

1.3 Scope of the Article

In this article we make the following contributions:

(1) We describe fundamental issues in realizing collaborative workflow design systems. We point out, how collaborative tools are to be used, which functionalities they are to provide and how to keep the workflow models on each client consistent.
(2) We augment collaborative design with workflow analysis using a set of workflow metrics, which can be used to assess the maturity and quality of a workflow model and to identify sub-workflows which should be refactored.
(3) We show how these principles have been implemented in a Rich Internet Application.

This article is an extended version of our paper "Collaborative BPEL Design in a Rich Internet Application", presented at *CCGrid'08* [12]. We extend our previous work by giving a more detailed description of collaborative workflow design and introducing new mechanisms for coordinating the work of team members. We also present workflow metrics as a means of workflow model analysis, which help team leaders in steering collaborative design.

The rest of the article is organized as follows. In Section 2 we present the background of our work, by introducing BPEL, Workflow Metrics and Web 2.0 technology. Afterward we will describe the general process of collaborative workflow design in Section 3 and consider requirements on collaborative workflow design in Section 4. Subsequently, in Section 5 we will present methods for achieving model consistency in web-based collaborative workflow design. Section 6 specifies a set

of pertinent workflow metrics for workflow analysis. In Section 7 we present our collaborative workflow design tool HOBBES. In Section 8, we will give an example of collaborative workflow design and analysis. We will review related work in Section 9 and conclude in Section 10.

2 Background

2.1 Workflows and BPEL

The XML-based *Business Process Execution Language* (BPEL) has become the *de-facto* standard for business workflows and is a key element of the *Service Oriented Architecture* (SOA) [2]. *Ezenwoye* et al. have shown the general applicability of BPEL for grid services [5].

BPEL control flow is a mixture of block-oriented and graph-oriented elements. Atomic tasks like service invocations or waiting are called *basic activities*. Control structures are expressed as *structured activities* (e.g. Sequence, If, While), which can contain child activities. Concurrency can be modeled using the structured activities Flow and ForEach. Flow allows the definition of Directed Acyclic Graphs of activities, while ForEach loops may be marked as "parallel". BPEL employs XPath as a standard expression language [13]. Expressions are used for conditionals, for triggering links between activities and for assignments.

2.2 Workflow Metrics

A Software Metric is an algorithm which computes a numeric value from source code to measure properties of a software system. Software Metrics are a means of steering development processes by assessing the quality of a software product and finding imbalances in the code base. They typically cover aspects like modularity, cohesion and coupling, source code complexity and size. Though no single metric can serve as a sufficient indicator of software quality, sets of different metrics can give hints about the structure of a software product.

Since workflow languages differ from general purpose programming languages, *Workflow Metrics* are a special class of Software Metrics. For example, Vanderfeesten et al. consider modularity as an aspect which does not apply to workflows, because workflow languages do not have any module concept and usually treat their activities as black boxes [14].

Workflow metrics are useful for collaborative development, since they enable reasoning about workflows in quantitative terms, thus improving communication about

manipulated workflow models. Moreover, some metrics help identifying unusually structured sub-workflows, thus guiding workflow development.

2.3 Web 2.0 Technology

The term *Web 2.0* has been popularized by Tim O'Reilly as a synonym for a set of advanced web application paradigms [15]. Web 2.0 applications share several common features [16][17]:

- Web 2.0 applications enable *collaboration*, e.g. via sharing and annotation (tagging) of artifacts.
- *Virtual communities* and *social networks* encourage user participation.
- APIs are exposed as *REST-ful* web services. The acronym *REST* ("Representational State Transfer") in this context denotes the usage of plain Text/HTTP or POX/HTTP-based ("Plain Old XML") protocols to invoke services [18].
- Sophisticated GUIs are provided as *Rich Internet Applications* (RIAs).

Rich Internet Applications provide both web access and desktop like usability. They often are identified with "AJAX" technology [19], though alternatives exist. AJAX stands for "Asynchronous Javascript And XML". It denotes the combination of Javascript and HTML Document Object Models with an *XMLHttpRequest*-API, that enables asynchronous server queries.

The Adobe Flex framework, which is based on the Flash plugin, appears to be the most stable and most complete RIA solution [20]. It supports declarative GUI design in an XML-based language (MXML) with direct mapping to the object-oriented scripting language Actionscript. Actionscript strongly resembles Javascript, but includes (optional) build-time type checking and a class system. Flex supports HTTP requests and its communication facilities can be enhanced with optional server side components called "Lifecycle Data Services" (LCDS), which among other things enable messaging according to the Publisher-Subscriber pattern, i.e. server-to-client notification.

3 The Collaborative Workflow Design Process

Weiseth et al. have identified three sub-processes of collaboration, which they call *coordination*, *production* and *decision-making* [21]. Coordination denotes the planning and assignment of tasks to collaborators, while production means collaborative development. Decision-making is the base of coordination and requires analysis of the manipulated artifacts. Our Collaborative Workflow Design approach accounts for each of these sub-processes in distinct phases (see Figure 2).

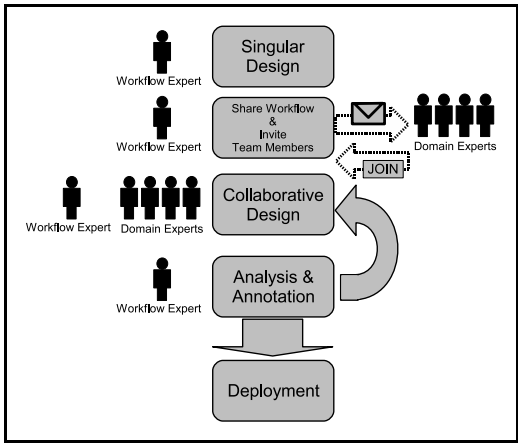

Fig. 2. Overview of the collaborative workflow design process.

First, the team leader instantiates a new session in the collaborative workflow design system. She can make initial decisions by laying out and configuring elements. Finally, the team leader will decide to share the workflow and invite team members.

Team members can join or leave at any time after sharing the workflow design session. The team synchronously works on the design until the team leader decides, it is time to analyze the quality of the workflow model. During the analysis and annotation phase the workflow is globally locked, so that only the team leader has full access. The team leader uses analysis tools, e.g. workflow metrics, to asses the workflow and find sub-workflows that should be refactored. If the team leader decides that the workflow needs further refinement, she will annotate the workflow model to delegate responsibility for specific tasks in sub-workflows to individual team members. Afterward she will switch the session to a new collaborative phase.

Figure 3 provides a more detailed description of the collaborative workflow design process, modeled as an Event-Driven Process Chain [22].

It shows, that workflow analysis (decision-making) and annotation (coordination) are closely related, as coordinative actions depend upon decisions. Both aspects involve only the team leader and require a workflow model that is left unchanged by other users actions. Thus, in a collaborative workflow design system both can be handled within the same phase.

In this paper we describe methods and present a software system to support this Process Chain, with the exception of the hatched area. The Process Chain shows that the same tools used for enabling collaborative workflow design can be used for collaboration between a workflow expert and an administrator in case a workflow deployment has led to problems.

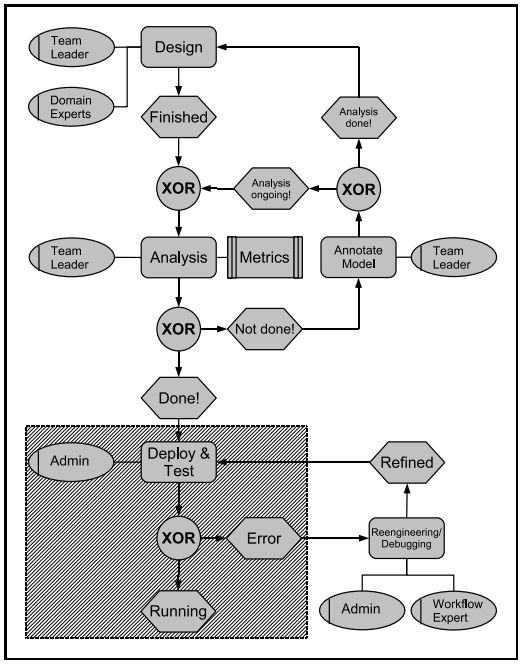

Fig. 3. The complete collaborative workflow design process as an Event-Driven Process Chain.

4 Requirements of Collaborative Workflow Design Systems

We will now discuss requirements for collaborative workflow design systems by proceeding from general to more specific issues.

4.1 General Requirements

To enable the seamless integration of workflow technology in Problem Solving Environments, workflow design tools will have to support collaborative work, must be accessible on the web, and should provide a nearly desktop-like user experience. Thus, we need a client that can be accessed from the web browser, but provides a user experience similar to desktop applications. Users need to be able to create team sessions, whose members work on the same workflow project and can synchronously add, remove, and connect activities.

4.2 Relaxed WYSIWIS and Late Joining

A fundamental design decision for collaborative software is, which information should be forwarded to other participants. A naive approach would be to give all users exactly the same view on the shared document (*strict* "What You See Is What I See" (WYSIWIS)) [23]. In the case of complex documents (like workflows) this

would diminish productivity because it is unlikely that all parts of the document can be shown in one view at the same time. A special concern arises if users attempt to make conflicting changes to the view, e.g. scroll into different directions or open pop-up windows. Such conflicts are sometimes dubbed ”scroll-war” or ”window-war”. Hence, a user’s actions should only affect another user’s view if it is semantically necessary.

Begole et al. discuss several usability requirements for the collaborative adaptation of legacy applications [24]. They especially point out that a user should be able to begin sharing a document at any time after its creation, that users should be able to join a shared session at any time (”late-joining”) and that users need to be allowed to have independent views of the shared document (”location-relaxed WYSIWIS”).

4.3 Workflow Locking Mechanisms

Synchronous collaboration encourages users to work on the same objects at the same time. This stands in contrast to each individual user’s demand for privacy while working on a specific hard problem in a sub-workflow. Sub-workflows, however, may contain complex domain-specific logic. One way to enhance group awareness — knowledge about other participants’ actions — is showing representations of foreign mouse cursors, called telepointers. Telepointers make much less sense in a web-based infrastructure due to the relatively high latency. Thus as an alternative means of raising group awareness, we have to allow synchronous work but also provide mechanisms to temporarily protect a user from being disturbed. To prevent the possibility of one user blocking the progress of the workflow design process, the team leader may grant or release such locks.

4.4 Collaborative Document Navigation

Additionally, there should be a way to lead another user’s view to a specific part of the workflow. This resembles face-to-face collaboration, where participants hint at a part of an object when communicating about it. Its implementation can be simplified, if the graphical representation reflects the tree structure of BPEL. Collaborative navigation is not only a better alternative to telepointers when collaborating in large, structured documents, but also improves awareness of dependencies between team members, since it enforces direct communication.

The configuration of an individual activity may depend on knowledge from a domain expert and from an infrastructure expert. Consequently, it makes sense to provide means that encourage a direct negotiation of these parameters. We thus have to provide means to enable two partners to collaborate on configuring a workflow activity.

4.5 Design Project Phases

As the design process will often be split up into distinct phases (see Section 3), which can be subject to a number of iterations, the team leader needs tool support for terminating a collaborative phase. Phase switching has to be a very light-weight operation to anticipate a quick alternation of phases. During the analysis phase team members should not be forced to leave the session. The analysis and annotation phase is non-collaborative, since any changes of the workflow by users other than the team leader would make analysis impossible. Effectively, switching to the analysis and annotation phase is equivalent to locking the workflow for other users than the team leader. Hence it can be seen as a special case of the issues presented in 4.3.

4.6 Workflow Metrics

As has been pointed out in Section 3, the team leader will need support for workflow model analysis, which has two main purposes. First, the team leader needs to assess whether the workflow model is finished, i.e. it is runnable and it mirrors the requirements. While the first can easily be verified from the source code, the second has to be checked by a human being.

The second purpose is the detection of design flaws in sub-workflows. Design flaws are not identical with errors in the workflow model. In contrast to errors, design flaws do not affect syntactical correctness or prevent the execution of a workflow, and thus can only be detected by a human user.

We are particularly interested in comparing manipulated artifacts according to their size and complexity, to enable searching for ”code smells” or ”bad smells” which indicate a need of refactoring [25]. In this context, complexity informally can be understood as the intricacy of understanding a specific artifact. For instance, *Cardoso* defines workflow complexity as ”the degree to which a process is difficult to analyze, understand or explain” [26]. It is not to be confused with algorithmic complexity of mathematical problems.

Scientific and business workflows are supposed to be reusable, so they should be maintainable, adaptable, and comprehensible. Scientific workflows often are larger than business workflows, making it harder to fulfill these demands. For scientific workflows, performance is an additional factor, so parallelism should be exploited and data-manipulation be avoided where possible.

Hence, a collaborative workflow design system should provide a view on several workflow metrics, which help the team leader in decision-making. In particular, recursively defined metrics are needed, that enable analyzing sub-trees of the BPEL

model.

4.7 Delegating Responsibility

After analyzing the workflow model, the team leader will prepare the next collaborative phase. This preparation especially includes decisions about who is supposed to work on which workflow parts. As sub-workflows in BPEL are induced by structured activities, this can be done by annotating activities. Hence, the team leader should have a means of annotating an activity to express the delegation of a specific task to a set of team members. In addition to the delegation itself, in some cases the affected sub-workflow should be locked for other team members. A special case arises, if a sub-workflow is to be locked for all team members to prevent further modification. The team leader also needs the right to remove these annotations at any time in the future. The different delegations should be individually visualized according to their effects for each team member.

4.8 Operation Granularity

One important difference between collaborative text editing and collaborative workflow design is the granularity of the manipulations. A workflow model consists of entities like activities, links and control structures. These are relatively complex objects, represented by parameter maps and object relationships. Inserting, deleting and connecting activities is handled by mouse input. Changing the configuration of an activity is a very coarse grained operation, when compared to inserting or deleting a letter. Hence operation messages are more coarse-grained when compared to collaborative text editing systems.

4.9 Scalability

The whole software system defined by a workflow and its services will probably be the product of a large group of developers. However, the design team for the workflow itself will be significantly smaller. We expect workflow design teams to consist of less than ten persons, according to *Brooks* the upper limit of a ”*small* sharp team” [27].

Brooks uses the analogy of a surgical team, which consists of team members that assume different roles. He stresses the role of a chief surgeon, similar to the team leader in our approach. We believe that this estimation is realistic as e.g. the case study on a collaborative workflow design process presented in [8] has three participants who bring in their different kinds of expertise.

5 Collaboration and Consistency

5.1 Manifestations of Inconsistency

A major concern when implementing a synchronous collaborative workflow design tool is to keep the workflow models on all participating clients consistent.

Inconsistency can manifest itself as *divergence*, *causality violation*, or *intention violation* [28]. Divergence means, that the resulting model of a collaborative design process differs on the participants' editors. Then it is not possible to decide, which model is the "correct" one or even if a correct model exists at all. Causality violation can happen if two operations, one of which is depending on the other, arrive at a client in the wrong order. For example, the deletion of an element can only succeed if the element has already been created. Intention violation means that the *intended* effect of an operation is compromised by the order in which operations are processed, e.g. if two clients demand to move an element of a sequence up one step, the result could be that the element moves two steps up. Obviously the real intention behind an operation cannot be decided upon in every case. If two users simultaneously add an activity of the same type to a workflow it is unclear, if they intended to model the same activity of the underlying process.

5.2 Algorithms for Enforcing Consistency

A classification of consistency algorithms can be found in the fourth chapter of Jürgen Vogel's dissertation [29]. Vogel distinguishes between *soft state* and *hard state* approaches to sharing a model. In a soft state approach, one or more participants periodically announce the current state of the model. This method is simple and does not need a reliable transport protocol. Its disadvantages are possibly high notification times, frequent temporary inconsistencies, and very high data rates, which makes it unsuitable for use on the web.

In a hard state approach, all operations are explicitly and immediately sent to all clients. It needs a reliable transport protocol, extra support for late joining clients and consistency has to be enforced by a special algorithm. *Optimistic* consistency algorithms execute operations locally at once and try to recognize and fix inconsistencies if they appear. Since they rely on every editor instance to enforce consistency, they are well suited for peer-to-peer systems. Optimistic algorithms enable a high responsiveness, but are hard to implement and verify. An example of an optimistic approach is *Operational Transformation* [30][31]. Local operations are immediately executed and then broadcast to the other editors. All operation messages are tagged with vector clocks, thus implying a partial ordering of the operations. Incoming operations from the peers are transformed to fit into the local model, thus

preventing inconsistencies.

Pessimistic consistency algorithms first check the validity of an operation and only then propagate it. This means, that either those parts of a document which can be affected by an operation have to be locked temporarily, or that the operations have to be processed in a sequential order by one designated host.

Since we assume small teams (see Section 4.9), a pessimistic, centralized approach can be applied, thus enabling an implementation within a client/server architecture. Incoming operations are processed sequentially by a ”controller” routine to prevent both divergence and causality violation.

Before actually executing an operation, the server gathers all needed elements. If a resource is unavailable, i.e. if it is not a part of the workflow or if it has been locked, the operation is aborted. Checking the preconditions of an operation is simplified by the tree structure of BPEL, the small set of possible operations and their precise semantics.

Operations are impossible, if they depend on a workflow element which does not exist. For example, *A* wants to add an activity α to a structured activity σ. *B* has deleted σ, which has not yet been propagated to *A*. The controller fails to look up σ and then decides to deny the operation. *A*'s client gets informed about the deletion of σ. It automatically navigates to the top element of the workflow and notifies *A* about the deletion.

A special case is adding an activity to a structured activity, that can only have one child (e.g. ForEach). If *A* and *B* concurrently try to add a child to such an activity, only one operation will succeed. If user *A*'s add-operation fails, she will see the addition of *B*'s activity and receive additional feedback by a pop-up notification.

Another special case is changing the sequence number of an activity that is a child of a Sequence or another structured activity that depends on the order of its children. Sequence numbers can only be changed by stepwise increments or decrements, i.e. by switching the positions of two activities α_1 and α_2. Thus, the controller has to check the existence of σ, α_1 and α_2 and that $pos(\alpha_1) - pos(\alpha_2) = 1$ to prevent intention violation.

Operations can either affect the BPEL structure defined by the model or its visual structure. Changing the position of an element within a Flow affects the visualization of the workflow, but not the BPEL model itself. Thus, concurrent drag operations within a Flow can lead to overlapping activities, which could be considered as an intention violation. However, attempts by the system to prevent such a situation would probably lead to irritation of the users who can easily resolve it by themselves by dragging the activity on top. If two concurrent drag operations of the same element occur, it is impossible to decide, which one should succeed from both user's point of view, because drag operations are solely characterized by their

outcome.

5.3 Late Joining

As has been noted in Section 4.2, we cannot assume all clients to be present at the beginning of a design session. A client can join a session after any number of operations have been performed on the model. All announcements of changes by the server are tagged with a counter. If a client joins a session, it first subscribes to operation notifications and begins buffering all incoming operation messages. The client then sends a join-request to the server, whose reply includes the current BPEL model and the value of the operation counter at the time the server received the join-request. Joining and leaving are handled atomically on the server. When the client receives the reply, it constructs the BPEL model and then processes any messages that have been lost between the join-request and reply. This ensures that after the join process is complete, the client has a consistent document model.

6 Workflow Analysis for Collaborative Workflow Design

6.1 Evening out Intricacies of Collaboration

Though collaborative workflow design fosters the integration of domain-specific and IT-specific knowledge within a workflow, it may increase the probability of design flaws. While every collaborator brings in additional expertise, he also is a source of unpredictable behavior. On the other hand, no single person has a view of all changes taking place in the model. Thus, the structural quality of a workflow model can deteriorate in the course of collaborative workflow design. To even out these effects, the team leader will repeatedly analyze the workflow model and actuate refactorings (see Sections 3 and 4.6).

6.2 Workflow Metrics for BPEL

Workflow metrics are needed to support the decision-making sub-process of collaborative workflow design. The following aspects of BPEL affect what can be measured:

(1) Web Services are treated as ”black boxes”, giving no insight into their functionality or internal structure.
(2) BPEL lacks any concept of modules or objects.
(3) It does not include any notion of functions or procedures.

(4) It is a structured programming language, since control structures are block-oriented and links between activities are limited to form directed acyclic graphs.
(5) In contrast to other workflow languages, control-flow and data-flow are defined separately.
(6) BPEL enables parallel task execution.
(7) BPEL workflow models are inherently structured as a tree.

Item 7 indicates, that we can develop recursively defined metrics that indicate the status of individual sub-workflows. Our choice of metrics is also affected by the necessity to use only such information which is available at design time.

We will now present a set of metrics for measuring *size*, *control-flow*, *data-flow* and *parallelism* of (sub-)workflows, some of which have already been presented by other authors. Afterward we will evaluate the set and present its application to identify code smells.

6.3 Size Metrics

Size is often measured using the Lines-Of-Code metric (LOC), which counts the number of statements within a program source code. Since workflows are usually designed with graphical editors, using the LOC metric does not make sense.

Cardoso et al. have proposed to count the number of basic activities (*NOA*) in a workflow as an alternative to the Lines-Of-Code metric [32]. In contrast to the LOC metric this leaves out any information about control structures. Thus, they also propose to count the number of activities and control structures (*NOAC*) as well.

In BPEL, the Assign activity may contain any number of copy statements. As an additional size metric, we propose counting the number of activities, control structures, and copy statements (*NOACC*).

6.4 Measuring Control Flow

6.4.1 McCabe's Cyclomatic Complexity

McCabe has proposed to measure the cyclomatic complexity of programs, i.e. the number of linearly independent paths in the control flow graph [33]. McCabe's Cyclomatic Complexity (MCC) is often used to evaluate the complexity of individual modules or functions. According to [34], it "measures the amount of decision logic in a single software module". The cyclomatic number of a graph g with n vertices, e edges and p connected components is defined as:

$$V(g) = n - e + 2 \cdot p$$

In structured programming languages, MCC can be computed by counting the number of binary decision points d, and setting $MCC = d + 1$. We calculate the MCC metric by counting the appearances of

- Receive with the attribute `createInstance = "true"`
- conditional branches, loops and events
- logical conjunctions and disjunctions in XPath-Expressions.

6.4.2 Control-Flow-Complexity

Cardoso has proposed a control flow complexity workflow metric (CFC), which he has also adopted to BPEL [35][26][36]. The CFC metric is defined recursively for every possible activity of a BPEL process.

The CFC definition does not mention the Scope activity. We propose to set the CFC value of a Scope to the CFC value of its child activity, since scopes do not affect the control flow.

The CFC value of the Flow activity is defined as $CFC(F) = (n-l)! \cdot \sum_{a \in F} CFC(a)$, where $n = |F|$, $l = |links(F)|$. Thus, it is implicitly assumed, that $n > l$, which is not necessarily the case. Hence, we propose to use the term $CFC(F) = \sum_{a \in F} CFC(a)$, iff. $l >= n$.

6.5 Measuring Parallelism

We have not found any hints on measuring workflow parallelism in the literature. In BPEL, Tasks can be created via the structured activities Flow and ForEach. Both differ significantly from each other in their level of abstraction.

The Flow activity defines static parallel execution of tasks in a directed acyclic graph. The ForEach activity can be used to create a number of tasks based on a numeric XPath expression. Hence, it is possible that the actual number of tasks generated by a ForEach activity cannot be known at compile time. The maximum number of concurrently running basic activities thus may depend on knowledge, which is not available at build time. In the terminology of [37], both constructs yield explicit parallelism. However, tasks are *implicitly* decomposed with ForEach and *explicitly* decomposed using Flow.

Thus, ForEach parallelism cannot be measured, but is defined on a higher level of abstraction than Flow parallelism. Hence, we propose a parallelism metric *DOP*

Activity	DOP
process	$DOP(P) = DOP(a), a \in P$
basic activities	$DOP(a) = 0$
flow	$DOP(F) = \omega_F + max \sum_{i \in \{i_1,\ldots,i_{\omega_F}\}} DOP(a_i)$
sequence, if, pick	$DOP(A) = max_{a \in A} DOP(a)$
loops, scope	$DOP(L) = DOP(a)$

Table 1
Degree of parallelism.

(degree of parallelism), which gives an upper limit to the maximum number of concurrently executed activities which have been created by explicit decomposition.

For any Flow activity F, parallel child tasks form a clique in the non-directed graph induced by the relationship τ_1 is parallel to τ_2 (See Figure 4).

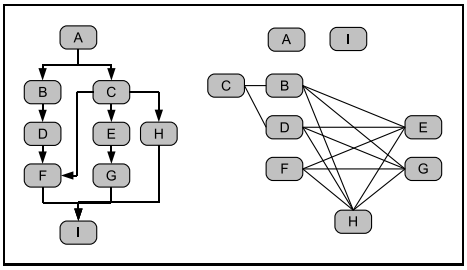

Fig. 4. Flow graph and parallelism graph.

Hence, we construct a parallelism graph G_p and then set

$$\omega_F := max_{c \in C(G(F))} \omega(c)$$

$$DOP(F) := \omega_F + max_{(A \subseteq F) \wedge (|A| = \omega_F)} \sum_{a \in A} DOP(a)$$

where $C(G)$ yields the connected subgraphs of G, $\omega(c)$ is the size of the biggest clique, and $G(F)$ is the parallelism graph induced by F. The Flow graph shown in Figure 4 has a DOP value of 3, if all activities are basic activities.

As the child activities of a Sequence are processed in a sequential order, the DOP of a Sequence is the maximum DOP of its children. The same applies the treatment of conditionals (If) and external events (Pick).

The algorithm for the DOP metric is presented in Table 1. In addition to DOP, we propose counting the number of parallel ForEach activities.

There has been little work on measuring data flow in BPEL, although data flow is a major aspect of workflows. We use a simple data flow intensity metric (DFI), inspired by *Liggesmeyer* [38]:

$$DFI = \frac{\#definitions + \#references}{\#decisions + 1}$$

where $\#references$ is the number of variable references both in predicates and computations, while $\#definitions$ is the number of assignments. In contrast to Liggesmeyer, we set the divisor to $\#decisions + 1$ instead of $\#decisions$. Thus, the metric is always defined, even if a workflow does not contain any decisions.

In BPEL we can identify $\#definitions$ as the number of copy statements and message receiving activities and $\#references$ as the number of variable references in different elements, while $\#decisions$ is the number of appearances of the activities If, Pick, For, RepeatUntil and While. $\#decisions$ can be interpreted as an additional control flow metric. Thus, instead of $\#decisions$, we can also use MCC or CFC to define variants of DFI.

To gain insight into the structure of the data flow, we need to augment DFI with other measures. For instance, the ratio of assignments per invocation and the ratio of *copy* operations per Assign, can indicate the complexity of integrating the given Web Services. The ratio of data-manipulating activities (receive, invoke, assign) to NOA can be used as a rough measure to assess the impact of data flow compared to control flow.

6.7 Additional Structure Metrics

We can augment the set of metrics by adding several derived metrics and other statistical values. These values are not necessarily meaningful by themselves, but can help interpreting the given metrics. The metrics proposed in this section are inspired by [38]. They should only be used in conjunction with complexity and size metrics:

- In addition to the NOA and NOAC metrics, it makes sense to count the appearance of any distinct element type within a workflow, so its impact on workflow complexity can be assessed.
- Similar to the decision density metric $\frac{MCC}{LOC}$, we propose using the ratio of decisions per activity/basic activity/ per invocation ($\frac{MCC}{NOAC}$, $\frac{MCC}{NOA}$, $\frac{MCC}{\#invoke}$).
- The number of simple predicates per decision and the number of simple and compound predicates per decision can indicate how complex the decisions in a

workflow are.
- The ratio of arithmetic operators to the number of copy statements can yield the average complexity of calculations within the workflow ($\frac{\#arithmetic\ operators}{\#copy}$).

6.8 Evaluation

6.8.1 Weyuker's Properties

For our purposes, the set of metrics that we propose both has to cover meaningful aspects of BPEL workflows and provide sufficient granularity. *Weyuker* has presented a set of nine desirable properties of software metrics [39]. The Weyuker properties assess the granularity of and the effects of program composition on software metrics. While we have not found any meaningful metric in the literature that satisfies all of the properties, if a set of metrics fulfills many or all of the properties, this can be interpreted as an indication of its usefulness. For a metric μ, with P, Q, R denoting programs, the Weyuker properties are defined as:

(1) $\exists P \exists Q : \mu(P) \neq \mu(Q)$
(2) $\forall c > 0 : |\{P, \mu(P) = c\}| \neq \infty$
(3) $\exists P \exists Q : P \neq Q \wedge \mu(P) = \mu(Q)$
(4) $\exists P \exists Q : P \equiv Q \wedge \mu(P) \neq \mu(Q)$, where $\equiv$ means that P and Q produce the same output data for the same input.
(5) $\forall P \forall Q : \mu(P) \leq \mu(P : Q) \wedge \mu(Q) \leq \mu(P : Q)$, where : means concatenation.
(6) $\exists P \exists Q \exists R : \mu(P) = \mu(Q) \wedge \mu(P : R) \neq \mu(Q : R)$
$\exists P \exists Q \exists R : \mu(P) = \mu(Q) \wedge \mu(R : P) \neq \mu(R : Q)$
(7) There exist programs P and Q such that Q is formed by a permutation of the statements of P, and $\mu(p) \neq \mu(Q)$.
(8) If P is a renaming of Q, then $\mu(P) = \mu(Q)$.
(9) $\exists P \exists Q : \mu(P) + \mu(Q) < \mu(P : Q)$

6.8.2 Weyuker Properties of the DOP Metric

Weyuker property 1 obviously holds for DOP. Property 2 does not hold, since an arbitrary number of workflows with the same degree of parallelism but differing sequential parts can be constructed. Property 3 holds for the same reason. Property 4 holds, since sequential and parallel invocations of web services with the same arguments yield the same output. BPEL workflows can be concatenated using either sequences or flows. For sequential concatenation $\mu(P : Q) = max\{\mu(P), \mu(Q)\}$, while for parallel concatenation $\mu(P : Q) = \mu(P) + \mu(Q) + 2$. Thus, property 5 holds, while property 6 does not hold.

Property 7 holds, since exchanging activities between sequential sub-workflows can lead to different DOP values.

Property 8 holds, since element names do not affect workflow structure or execution. Property 9 does not hold for sequential concatenation. However, it holds if two workflows are concatenated in parallel, as this increases DOP by two.

6.8.3 Weyuker Properties of the DFI Metric

Property 1 holds for this metric, since sequential processes contain only one decision but an arbitrary number of definitions and references. Property 2 does not hold, since the number of decisions and the number of variable references can be increased at the same time by using XPath custom functions. Property 3 holds, for the XPath expressions $\$x * \$y$ and $\$x + \$y$ contain the same number of references but a different operator. Property 4 holds since any number of unused assignments can be made.

Properties 5 and 9 do not hold, since the concatenation of two BPEL programs P and Q yields the term

$$DFI(P:Q) = \frac{def(P) + def(Q) + ref(P) + ref(Q)}{dec(P) + dec(Q) + 1}.$$

Property 6 holds. Assume $def(P) + ref(P) = 16$, $dec(P) = 3$, $DFI(P) = 4$ $def(Q) + ref(Q) = 8$, $dec(Q) = 1$, $DFI(Q) = 4$ $def(R) + ref(R) = 2$, $dec(R) = 2$. Then $DFI(P) = DFI(Q)$, but $DFI(P:R) = \frac{18}{6} = 3$ and $DFI(Q:R) = \frac{10}{4} = 2\frac{1}{2}$. Since DFI only depends on counts, property 7 does not hold, while property 8 holds.

6.8.4 Summary

Table 2 gives an overview of the adherence of the presented metrics to the Weyuker properties. It shows that all Weyuker properties are satisfied by at least one of the metrics. The same arguments Weyuker has used for investigating the LOC metric, hold for the size metrics NOA, NOAC and NOACC. MCC and DFI are defined non-recursively. Both can be computed for any sub-workflow, though, since they only depend on counts.

The other metrics presented in this paper are either derivations from the metrics listed in Table 2 or are only meant to be used in conjunction with them. Thus we do not evaluate their adherence to the Weyuker properties.

Workflow Metric	Category	recursive	1	2	3	4	5	6	7	8	9	remarks
NOA	size	yes	+	+	+	+	+	-	-	+	-	see text
NOAC	size	yes	+	+	+	+	+	-	-	+	-	and [39]
NOACC	size	yes	+	+	+	+	+	-	-	+	-	
MCC	control flow	no	+	-	+	+	+	-	-	+	-	[39]
CFC	control flow	yes	+	-	+	+	+	-	+	+	+	[35]
DFI	data flow	no	+	-	+	+	-	+	-	+	-	See Section 6.8.3.
DOP	parallelism	yes	+	-	+	+	+	-	+	+	+/-	See Section 6.8.2.

Table 2
Workflow metrics.

6.9 Applying the Metrics: BPEL Code Smells

The set of metrics proposed here enables investigating BPEL documents for code smells that indicate a need of refactoring, which has been required in Section 4.6. If the sub-workflows X and Y are expected to perform a similar functionality, the code smell

$$\begin{aligned} & MCC(X) >> MCC(Y) \\ \vee\ & NOACC(X) >> NOACC(Y) \\ \vee\ & CFC(X) >> CFC(Y) \\ \vee\ & DFI(X) >> DFI(Y) \\ \vee\ & \#invoke(X) >> \#invoke(Y) \end{aligned} \tag{1}$$

indicates that either one implementation is incorrect or X is overly complex. This can happen, if collaborators either differ in their idea about the functionality of the workflow or in their level of experience.

The following smell indicates, that sub-workflow X is linear, but contains at least one Flow element with *join* and *target* conditions and probably should be refactored to a Sequence:

$$\begin{aligned} & CFC(X) = NOA(X) \\ \wedge\ & MCC(X) > 1 \end{aligned} \tag{2}$$

When collaborators independently add assignments to a workflow, this can lead to redundancy. The following smell indicates that sub-workflow X may contain Assign activities that can be merged:

$$\frac{\#assign(X)}{\#invoke(X)} > 1 \wedge \frac{\#copy(X)}{\#assign(X)} \approx 1 \tag{3}$$

Here, sub-workflow X probably contains unnecessary nesting:

$$\begin{aligned} & NOAC(X) > NOA(X) + 1 \\ \wedge\, & CFC(X) \approx NOA(X) \end{aligned} \tag{4}$$

Unnecessary nesting can appear if designers work at different levels of a model tree and activities on the upper layer are removed. Another source of unnecessary nesting are inexperienced designers, who attempt to use the structured activity "of their choice".

Scientific workflows usually exploit parallelism for efficiency. If X contains compute-intensive tasks, the following smell should lead to questioning a domain-expert about potentially parallel execution of affected invocations:

$$\begin{aligned} & DOP(X) = 0 \\ \wedge\, & \#parallel\ forEach(X) = 0 \\ \wedge\, & \#invoke(X) > 1 \end{aligned} \tag{5}$$

If two sub-workflows X and Y are known to perform a similar functionality,

$$DOP(X) >> DOP(Y) \tag{6}$$

is another bad smell for unused parallelism.

Because BPEL is not intended as a general programming language, a high density of arithmetic operators or a very high data flow indicates a need of new custom functions or web services:

$$\frac{\#arithmetic\ operators}{\#copy} > c, \;\; c > 1\ const. \tag{7}$$

7 The HOBBES System for Collaborative Workflow Design

In this Section we show, how the principles we have presented in the previous sections have been implemented in our HOBBES workflow design system. A demo version of HOBBES is available at
`http://www-sr.informatik.uni-tuebingen.de/~held/ria.htm`.

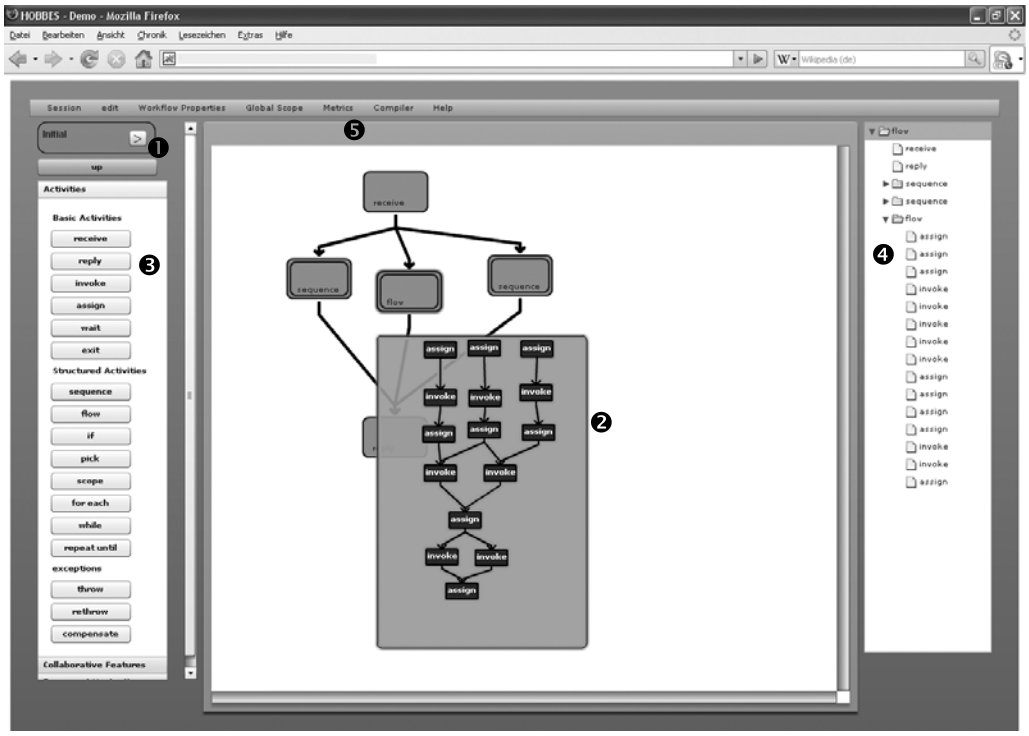

Fig. 5. A workflow model in HOBBES

7.1 BPEL-Editing

We will briefly describe the general features of HOBBES first and then elaborate on the collaborative features. The activities of the workflow are edited and arranged in a graphical view (see Figure 5). User dialogs can be opened to edit the references of the BPEL document to WSDL files, its Partner Links to Web Services, and other properties, like global variables, fault and event handlers.

7.1.1 Activities

Figure 5 shows the main screen of HOBBES. It consists of an accordion panel on the left (see Figure 5, ❸), which provides diverse editing options, a canvas ❷, which shows the children of a structured activity, and a navigation tree ❹ on the right. The editing canvas may differ according to the kind of structured activity currently shown.

Conditional and arithmetic XPath-expressions are edited graphically. The expressions are represented as trees, where nodes correspond to operators, functions, and constant values.

7.1.2 Document Navigation

HOBBES directly reflects the tree structure of the BPEL document by showing only the immediate child elements of a structured activity and their connections. Clicking on a flow or sequence activity reveals a preview of its child elements. Navigation through the document is simplified by a navigation tree widget, which is shown right of the editor view. The structured activity currently being edited is highlighted. HOBBES's document navigation adheres to relaxed WYSIWIS (see Section 4.2).

7.2 Collaborative Features

7.2.1 Workflow Sessions and Phase Management

At the beginning of a team session, the team leader has sole access to the workflow until she decides to share the session, after which other users may join. Clients are notified of users joining and leaving the session and display a list of present users. A chat feature enables communication without depending on external programs.

The current collaboration mode is shown as a screen icon (see Figure 5, ❶). The team leader may at any time switch to a different mode, which is propagated to all clients. When the system changes from the collaborative design mode to the analysis and annotation mode, the team leader gains private access to workflow metrics as well as the ability to annotate workflow elements, as described later. During the analysis mode, other team members can view the changes the team leader makes on the workflow. They are prevented from modifying the workflow themselves, though. The team leader may persist the current state of the workflow model at any time.

7.2.2 Collaborative Editing

We need to support synchronous collaboration (see Section 4.1) and also provide means for temporarily locking workflow parts (see Section 4.3). Team members can concurrently add, remove, drag, and connect activities. Such changes, which affect visual elements of the workflow model, are instantly propagated to all clients.

If a user actuates a modification of the workflow a request is sent to the server. The server then decides upon the validity of the operation and includes the necessary instructions in its response message and informs the other clients via a broadcast. The validity of an operation can be compromised if users concurrently actuate operations (see Section 5).

For example, if user *A* actuates the creation of an Invoke activity as a child of a Sequence activity, the server will only allow the modification if the Sequence still

exists within the model. If this is not the case because user *B* has deleted it, *A*'s client will receive notification about the delete operation. It will then automatically navigate to the root of the workflow and perform the delete operation.

However, users may lock parts of the workflow, thus preventing others from modifying it. If user *A* wants to lock a structured activity, she will click its "lock" button. The lock operation is rejected if another user still holds a lock on the activity or any of its descendants. Otherwise, the team leader is asked for allowing *A* to lock the activity and its children. If the team leader currently is logged out, the lock operation succeeds anyway. If a user has sole access to a specific part of the workflow, he cannot be disturbed by other user's actions. This can be of great importance for implementing details. Because all changes to the sub-workflow are still propagated to other clients, all users stay aware of the progress of the design process. Locks can be released by the lock owner herself or by the team leader.

7.2.3 Collaborative Activity Configuration

As has been discussed in Section 4.4, sometimes a user will need another user's advice, when configuring an activity. For this case, HOBBES includes a special configuration "co-edit" window (see Figure 6). User *A* may select an activity and then choose the "co-edit" feature. A window dialog will pop up for inviting another user *B* to collaboratively edit the activity. *B* is informed of the invitation and may accept or reject it. In case *B* accepts, on both user's screens a window will appear, which shows three activity configuration forms, two of which are disabled. In one form the user can input a configuration and send it to his partner by pushing the "propose" button. The two disabled forms contain the opposite user's last proposal and the current configuration of the activity. Both users can exchange and discuss each other's proposals until *A* either commits one of the two proposals or cancels the operation.

Fig. 6. Collaboratively editing the properties of an activity.

7.2.4 Collaborative Document Navigation

With relaxed WYSIWIS, one user's actions do not influence other users' view on the model. In a face-to-face collaboration, however, the first step in communicating about a specific object is to show it to a partner. We have thus included a feature, that enables such a communication, as discussed in Section 4.4. If user *A* clicks on the "co-navigate" button, she can invite another user *B* to navigate to her view of the model. *B* is notified by a pop-up window and may accept or reject. If *B* accepts, his client will navigate to the same structured activity.

7.2.5 Workflow Metrics

In the analysis phase of the workflow design process, the team leader may view the current metrics of the workflow model (see Figure 5, ❺). The metrics window also includes a tree view for those metrics which are defined recursively. The team leader can also access a graph view that presents the development of the global metrics of a workflow over time (see Figure 7).

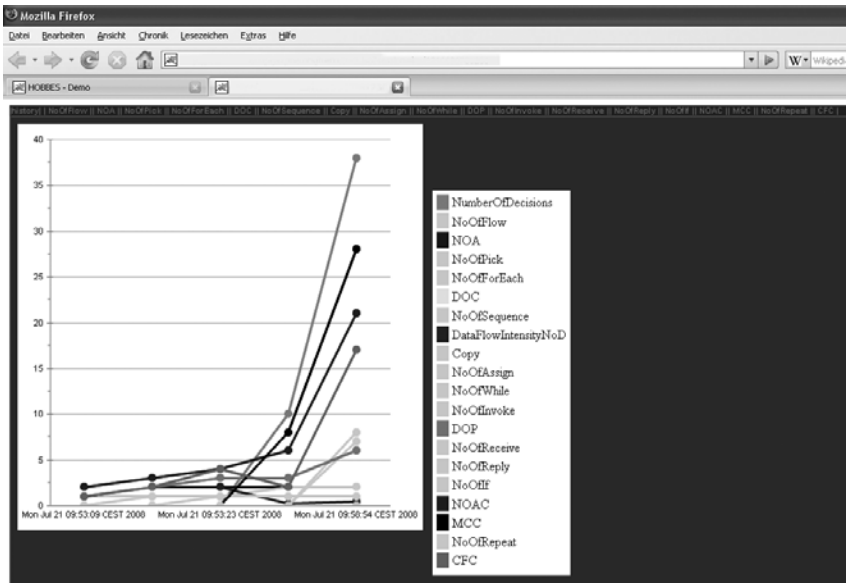

Fig. 7. Metrics history graph.

7.2.6 Activity Annotations

During the analysis and annotation phase, the team leader may annotate activities in the workflow to delegate tasks to team members (see Section 4.7). Annotations automatically are also applied to the sub-tree induced by structured activity.

The team leader may annotate an activity α in the following ways:

(1) α is delegated to a set of team members, and locked for others.

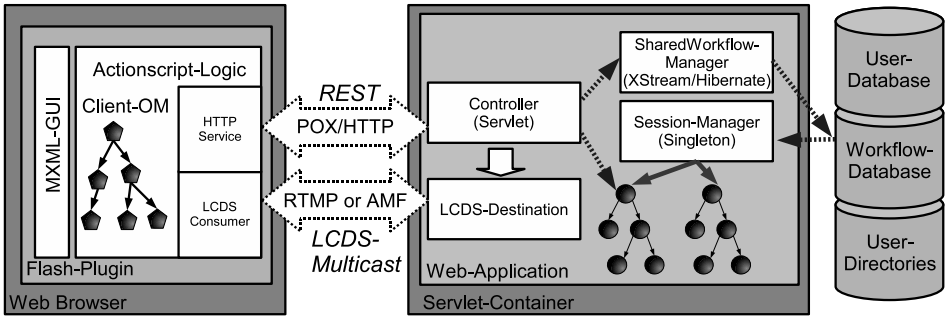

Fig. 8. The architecture of HOBBES.

(2) α is delegated to a set of team members, but not locked for others.
(3) α is locked for everyone.

Annotation 1 can be used to both grant privileged access to a workflow part to an expert and to instruct her about her responsibilities in the next collaborative phase. Annotation 2 delegates responsibility to a team member but does not prevent collaboration, while 3 can especially be useful to prevent those parts of a workflow model from further modification, which are considered complete. According to their meaning to a specific user, the annotated activities are marked with colors. Additionally the annotations may include a text which describes the expected task.

7.3 Implementation

7.3.1 The HOBBES *Architecture*

Figure 8 shows an overview of the architecture of HOBBES. Basically, it implements the Model-View-Controller Pattern [40] using the LCDS notification facilities of Flex (see Section 2.3). Concurrent input is processed on the server, which decides upon the validity of operations and subsequently notifies the clients of changes of the BPEL model. Clients post HTTP requests to the server, whose responses are handled via callback methods. The full functionality of a workflow design session is exposed via a RESTful interface using POX/HTTP messages. Hence alternative non-collaborative clients could be developed as well, e.g. for mashups.

7.3.2 The BPEL Object Model

The structure of the resulting BPEL document is reflected by a BPEL Object Model (BOM). It mainly consist of an object tree, where every node represents a BPEL activity and additional information concerning its visualization.

The full BOM is held on the server. The client side object model contains less information and thus is more light-weight than the server side object model. This

reduces the amount of data that has to be sent when joining a session and also simplifies the implementation of the client. Every relevant object (activities, links, etc.) has an identifier which is identical in the client model as well as in the server model.

Activity objects can be marked as unlocked or as locked by a specific user. The server uses this information to decide about the validity of an operation, while the client needs it for visualization purposes.

An activity object holds information relevant for compilation to BPEL (its configuration, its incoming and outgoing links, etc.), for internal management (e.g. its parent activity) and for visualization purposes (e.g. its x- and y-position within a flow). Structured activities also contain references to their children.

Client side activity representations contain some, but not all of the information of the server side objects, e.g. the configuration information is only held on the server. Server side activity representations contain a map of the properties of the affected activity. For manipulating its configuration, the client fetches the current configuration from the server and presents it in a configuration form.

7.3.3 The HOBBES *Server*

The controller decides upon the validity of a user request, manipulates the concerned BOM and sends an answer message via HTTP reply, if necessary. If a manipulation can affect the views of other clients (e.g. creating an activity), it is propagated via an LCDS destination.

Operations which affect the BOM are processed in short critical regions within the controller servlet. Every critical region first checks whether the resources necessary to fulfill a request exist and whether the affected activities are available to the user. This may not be the case if an activity has been locked. If this check and further consistency checks as described in Section 5 succeed, the operations are performed and then announced to the requester and all other clients. If a check does not succeed, the reaction of the system depends on the nature of the request. If e.g. two users tried to delete the same activity at the same time, only one deletion request will succeed but no special information about the failed deletion request needs to be published. If on the other hand, a client requests to import a WSDL file which is not available on the server any longer, the reply will contain an error message.

8 An Example of Workflow Design

8.1 Purpose

Our example is a pipe process, which converts gene sequence identifiers of the plant Arabidopsis thaliana to gene sequences. Arabidopsis thaliana is widely used as a model organism for plant science. We have chosen this particular example, since it is a small and simple workflow, yet it represents a common case in bioinformatics. We have developed this example with the aid of the BioMoby Web Service database [41]. The example can be downloaded from `http://www-sr.informatik.uni-tuebingen.de/~held/example.htm`.

The example workflow consists of the sequential execution of three tasks and takes a list of AGI (Arabidopsis Genome Initiative) Locus Identifiers as input. The first task invokes a service, which converts the AGI Locus identifiers to NASC (Nottingham Arabidopsis Stock Centre) code identifiers. The second task invokes another service, that converts the NASC identifiers to EMBL (European Molecular Biology Laboratory) accession numbers, while the third task uses yet another service to yield actual DNA, RNA or amino acid sequences from the EMBL accession numbers.

8.2 Original Design

The designers have attempted to implement the process in a semi-modular fashion, where every service invocation is encapsulated within a Sequence activity, thus representing every one of the tasks as a sub-workflow. Each of these sequences consists of two phases. First assignments are made to configure the invocation, second the actual invocation is actuated. The sequences are embedded in a Flow activity and connected by links, because originally it had not been obvious, whether the workflow would contain concurrent invocations. In the first phase, three designers have worked in parallel to implement the workflow model, with each designer being responsible for one sub-workflow. However, one of the designers is inexperienced in workflow design and has committed several design flaws when implementing sub-workflow 1. The workflow is visualized in Figure 9.

8.3 Analysis and Redesign

While the rationale behind the workflow at hand is a relatively clear cut design, a look at some of the metrics reveals that it can be improved (see Table 3).

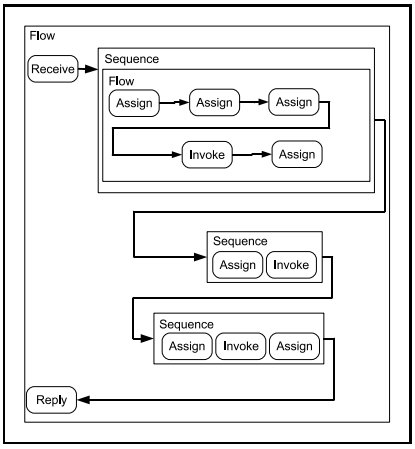

Fig. 9. A pipeline workflow.

Since the number of basic activities is equal to its control-flow-complexity and its degree of parallelism is zero, the workflow is a pipe.

Moreover, the ratio of copy-elements per assign is one, while there are more than one assign-activities per invocation. The metrics resemble code smell 4 from Section 6.9. Since the workflow does not include any conditionals, some of the assign activities can be merged.

While it is known that all three sub-workflows are responsible for a similar purpose, i.e. the invocation of a service, there is an imbalance between the metrics of the sub-workflows. As stated in Section 6.9 (code smell 2), this indicates that the first sub-workflow is overly complex. Code smell 5 applies to sub-workflow 1, so it contains unnecessary nesting. Closer inspection also reveals that it contains a redundant Assign activity. Refactoring leads to the new pipe shown in Figure 10.

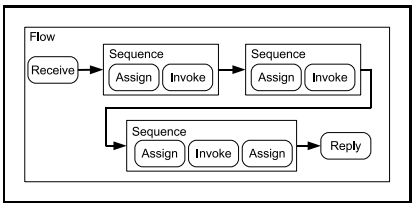

Fig. 10. The refined pipeline workflow.

8.4 Further Analysis and Redesign

Still, the MCC value is too high for a simple pipe, as it is the result of join and target conditions (code smell 3). Hence, the workflow can still be improved by refactoring all structured activities to a single sequence element. Size and complexity of the new workflow have diminished substantially (see Figure 11).

Compiling the initial workflow yields about 180 LOC, while the refactored Workflow contains about 100 LOC. Since the workflow is comparatively small (contain-

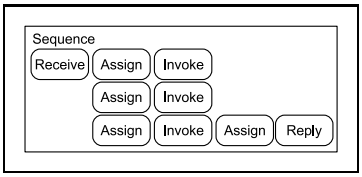

Fig. 11. The final pipeline workflow

Workflow Metric	old value	sub-workflow 1	sub-workflow 2	sub-workflow 3	refined design	final design
NOA	12	5	2	3	9	9
NOAC	16	6	2	3	13	10
NOACC	29				25	22
# of Structured Activities	4				4	1
# of decisions	0	0	0	0	0	0
$\frac{\#assign}{\#invoke}$	4				1.33	1.33
$\frac{\#copy}{\#assign}$	1				3	3
MCC	17	8	1	1	6	1
CFC	12	5	2	3	9	9
DOP	0	0	0	0	0	0
DFI	36	10	10	12	34	34

Table 3
Workflow metrics.

ing only three invocations), we can see that larger workflows absolutely require metrics support for effective steering of collaborative design.

9 Related Work

9.1 Collaborative Development Software

Since the 1990s several real-time collaborative text editors have been developed. A popular example is Subetha-Edit, which uses Apple's Bonjour protocol to automatically find peers in local area networks but also can be used on the Internet [42]. Groups of users can synchronously edit the same text document. Whenever a group member types or deletes a character, this event is immediately reproduced on his

co-workers' editors.

CoWord and CoPowerpoint are examples of collaborative extensions for office software [43], where operational transformation is being added by intercepting the user's local interactions (see Section 5.2). Collaborative editors like Subetha-Edit or CoWord prove the concept of synchronous collaborative editing. They cannot be easily embedded into web environments, though.

Google Docs is a web-based collaborative office suite including a word processor, a spreadsheet and a presentation software, which supports real-time collaboration by HTTP polling [44]. Buzzword is a web-based word processor which allows users to share a document [45]. Like our collaborative workflow design tool HOBBES it is implemented in the Flex framework. Compared to other web-based word processors it more strongly resembles desktop applications and clearly shows the value of browser extensions for implementing advanced user interfaces on the web.

9.2 Workflows in Science and Business

A plethora of industrial workflow design tools exists. These either employ a specific modeling language (e.g. BPMN, UML Activity Diagrams, EPC) or reflect the execution language (e.g. BPEL). One typical example of an industrial strength BPEL editor is the ActiveBPEL Designer [46], which is based upon Eclipse. It inherits many advantages of the Eclipse platform, like project management features and different perspectives. Its graphical representation of the workflow is directly derived from the structure of BPEL. All activities are presented on one plane. ActiveBPEL Designer shows structured activities as collapsible containers and allows to zoom in and out of the workflow graph. Unlike the Oracle JDeveloper IDE [47] it does not allow to "drill into" structured activities, which means showing only the workflow subgraph defined by a structured activity. Apart from version control system integration, ActiveBPEL does not support collaborative development.

The Eclipse-based BPEL-Editor Sedna has been used for grid computing in computational chemistry [1][4]. Sedna does not provide any collaborative features. All activities are presented on one plane and structured activities are represented by non-collapsible containers. Sedna lacks further means to represent more complex BPEL documents.

Friese et al. have presented an Eclipse-based collaborative BPEL Editor [8], which supports a synchronous editing process based on the Eclipse Communication Framework, where the collaboration initiator acts as a coordinator. They motivate collaborative workflow design by an example, where a casting engineer, a numerical simulation expert, and a grid expert cooperate on the design of an engineering workflow in the domain of metal casting. They argue, that especially small companies will gain from collaborative workflow development, since such companies will often

have to hire external experts for implementing aspects of a workflow. Users can join a session and create, delete, connect and move basic activities concurrently. They neither address the issue of collaboratively editing complex workflows consisting of several structured activities nor how participants should negotiate the configuration of a specific activity.

The P-GRADE bioinformatics portal provides a Java-based workflow editor with collaborative features [48]. Out of the necessity to connect Grid jobs of different users, P-GRADE provides the ability to lock, locally edit and submit parts of a workflow. P-GRADE neither supports synchronous collaboration nor a structured collaborative design process steered by workflow metrics. The P-GRADE Editor uses Java Web Start deployment, which can be seen as a distant relative of RIA technology but prevents embedding the application in a web site. It uses a proprietary workflow language, limited to the design of Directed Acyclic Graphs.

The ESCOGITARE bioinformatics portal contains a workflow design tool, which compiles BPEL processes and can be accessed online [49]. ESCOGITARE provides a simplified editor, which hides BPEL features from the user and only presents a palette of available web services. Although it is integrated in a web site it does not support collaborative design.

The BioWMS workflow management system contains a web-based workflow editor [50]. Workflows are presented as static HTML pages with embedded graphics. Changes to a workflow lead to reloading the page. BioWMS neither supports BPEL nor does it include any collaborative features.

The LEAD portal (*Linked Environments for Atmospheric Discovery*) gives its users access to the TeraGrid infrastructure for research on meteorological phenomena [51]. It provides a simple, data-flow oriented pipeline editor using Java Web Start. The pipeline workflows can be compiled to Java Python (Jython) and be executed via the portal, while support for BPEL and more complex graphs is planned.

The A-WARE Grid portal includes a workflow designer based on the Business Process Modeling Notation (BPMN) [52][53]. The resulting BPMN-models are compiled to BPEL. It is implemented as a Java-Applet but does not provide any collaborative features.

Yahoo Pipes [54] is an AJAX-based Mashup-editor which resembles a dataflow-oriented workflow editor. Users may connect web forms and simple REST-style Web Services which provide String input and output to DAGs with a set of input parameters and one output data sink. The Pipes notation supports iterating single activities over input collections. Different string operators can be embedded directly into a pipe.

Table 4 shows an overview of the presented workflow design tools compared to HOBBES. We have included Yahoo Pipes in the table, because it is a sophisticated

example of mashup technology, which shows strong similarities to workflow editors. The character "*" signifies the use of Java Web Start.

	BPEL support	web integration	synchronous collaboration	locking workflow-parts	collaborative document navigation	hierarchical presentation	graphical XPath editor	Workflow Metrics	collaboration phases	task delegation
ActiveBPEL Designer	2.0	no	no	no	no	no[11]	no	no	no	no
Sedna	1.1	no	no	no	no	no	no	no	no	no
Friese et al.	2.0	no	yes	no	no	no	no	no	no	no
P-GRADE	no	no*	no	yes	no	n/a	n/a	no	no	no
ESCOGITARE	1.1	yes	no	no	no	no	no	no	no	no
BioWMS	no	yes	no	no	no	no	n/a	no	no	no
Yahoo Pipes	no	yes	no	no	no	no	n/a	no	no	no
LEAD	planned	no*	no	no	no	no	no	no	no	no
A-WARE	2.0	yes	no	no	no	no	no	no	no	no
HOBBES	2.0	yes	yes	yes	yes	yes	yes	yes	yes	yes

Table 4
Workflow design tools.

As the table shows, in contrast to the other systems presented here, HOBBES enables the full integration of collaborative workflow design in Web 2.0-based Problem Solving Environments.

10 Conclusion

We have presented a novel method of workflow design, which supports synchronous collaboration, metric-based workflow analysis and annotation-based coordination of collaborators. The possibility to work synchronously on a workflow, while structuring the design process itself encourages cooperation between software engineers and domain experts. Workflow metrics help the team leader in decision making, identifying design flaws in sub-workflows and estimating the maturity of a workflow project. They also simplify communication between the team leader and other team members because they enable reasoning about workflow models in quantitative terms.

While our article has focused on collaborative BPEL development, our findings can be applied to other workflow systems as well. As the Web is already used as a platform for scientific Problem Solving Environments and for workflow enactment, the time has come to broadly support web-based collaborative workflow development.

11 Acknowledgments

Markus Held is supported by a grant from the Ministry of Science, Research and the Arts of Baden-Württemberg (Az: 23-7532.24-4-18/1).

References

[1] I. J. Taylor, E. Deelman, D. B. Gannon, M. Shields, Workflows for e-Science: Scientific Workflows for Grids, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] Web Services Business Process Execution Language Version 2.0, `http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf`, last accessed 24/10/2007. (April 2007).

[3] J. Yu, R. Buyya, A Taxonomy of Workflow Management Systems for Grid Computing, Journal of Grid Computing 3 (3-4) (2005) 171–200.

[4] W. Emmerich, B. Butchart, L. Chen, B. Wassermann, S. Price, Grid Service Orchestration Using the Business Process Execution Language (BPEL), Journal of Grid Computing 3 (3-4) (2005) 283–304.

[5] O. Ezenwoye, S. M. Sadjadi, A. Carey, M. Robinson, Grid Service Composition in BPEL for Scientific Applications, in: Proceedings of the International Conference on Grid computing, high-performance and Distributed Applications (GADA'07), Vilamoura, Algarve, Portugal, 2007.

[6] J. D. Herbsleb, Global Software Engineering: The Future of Socio-technical Coordination, in: FOSE '07: 2007 Future of Software Engineering, 2007, pp. 188–198.

[7] J. Whitehead, Collaboration in Software Engineering: A Roadmap, in: FOSE '07: 2007 Future of Software Engineering, 2007, pp. 214–225.

[8] T. Friese, M. Smith, B. Freisleben, J. Reichwald, T. Barth, M. Grauer, Collaborative Grid Process Creation Support in an Engineering Domain, in: Proceedings of High Performance Computing - HiPC 2006, 13th International Conference, 2006.

[9] I. Foster, C. Kesselman, S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International Journal of High Performance Computer Applications 15 (3) (2001) 200–222.

[10] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, J. Myers, Examining the Challenges of Scientific Workflows, IEEE Computer 40 (12) (2007) 24–32.

[11] L. Osterweil, Software Processes Are Software Too, in: ICSE '87: Proceedings of the 9th international conference on Software Engineering, IEEE Computer Society Press, Los Alamitos, CA, USA, 1987, pp. 2–13.

[12] M. Held, W. Blochinger, Collaborative BPEL Design in a Rich Internet Application, in: CCGRID '08: Proceedings of the 8th International Symposium on Cluster Computing and the Grid, IEEE Computer Society Press, Lyon, France, 19-22 May 2008., 2008, pp. 202–209.

[13] XML Path Language (XPath) Version 1.0, `http://www.w3.org/TR/xpath`, last accessed 18/07/2008. (November 1999).

[14] I. Vanderfeesten, J. Cardoso, J. Mendling, H. Reijers, W. van der Aalst, BPM and Workflow Handbook 2007, Future Strategies Inc., Lighthouse Point, Florida, USA, 2007, Ch. Quality Metrics for Business Process Models, pp. 179–190.

[15] T. O'Reilly, What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software, `http://www.oreillynet.com/lpt/a/6228`, last accessed 11/06/2008.

[16] M. Jazayeri, Some trends in web application development, in: FOSE '07: 2007 Future of Software Engineering, IEEE Computer Society, Washington, DC, USA, 2007, pp. 199–213.

[17] G. C. Fox, R. Guha, D. F. McMullen, A. F. Mustacoglu, M. E. Pierce, A. E. Topcu, D. J. Wild, Web 2.0 for Grids and e-Science, in: INGRID 2007 - Instrumenting the Grid 2nd International Workshop on Distributed Cooperative Laboratories, 2007.

[18] R. T. Fielding, R. N. Taylor, Principled design of the modern web architecture, ACM Transactions on Internet Technology 2 (2) (2002) 115–150.

[19] J. J. Garrett, Ajax: A New Approach to Web Applications, `http://www.adaptivepath.com/ideas/essays/archives/000385.php`, last accessed 11/06/2008.

[20] Adobe Flex, `http://www.adobe.com/flex/`, last accessed 21/07/2008.

[21] P. E. Weiseth, B. E. Munkvold, B. Tvedte, S. Larsen, The wheel of collaboration tools: a typology for analysis within a holistic framework, in: CSCW '06: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work, ACM, New York, NY, USA, 2006, pp. 239–248.

[22] A.-W. Scheer, ARIS Business Process Modeling, Springer Verlag, 1999.

[23] M. Stefik, D. G. Bobrow, S. Lanning, D. Tatar, G. Foster, WYSIWIS revised: Early Experiences with Multi-User Interfaces, in: CSCW '86: Proceedings of the 1986 ACM Conference on Computer-Supported Cooperative Work, 1986, pp. 276–290.

[24] J. Begole, M. B. Rosson, C. A. Shaffer, Flexible Collaboration Transparency: Supporting Worker Independence in Replicated Application-sharing Systems, ACM Transactions on Computer-Human Interaction 6 (2) (1999) 95–132.

[25] M. Fowler, Refactoring: Improving the Design of Existing Code, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[26] J. Cardoso, Evaluating the Process Control-Flow Complexity Measure, in: 2005 IEEE International Conference on Web Services (ICWS 2005), IEEE Computer Society, 2005, pp. 803–804.

[27] F. P. Brooks Jr., The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1978.

[28] C. Sun, X. Jia, Y. Zhang, Y. Yang, D. Chen, Achieving Convergence, Causality Preservation, and Intention Preservation in Real-time Cooperative Editing Systems, ACM Transactions on Computer-Human Interactions 5 (1) (1998) 63–108.

[29] J. Vogel, Consistency Algorithms and Protocols for Distributed Interactive Applications, 2004, dissertation. `http://madoc.bib.uni-mannheim.de/madoc/volltexte/2004/671/`, last accessed: 21/07/2008.

[30] C. A. Ellis, S. J. Gibbs, Concurrency Control in Groupware Systems, SIGMOD Record 18 (2) (1989) 399–407.

[31] C. Sun, C. Ellis, Operational Transformation in Real-time Group Editors: Issues, Algorithms, and Achievements, in: CSCW '98: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, 1998, pp. 59–68.

[32] J. Cardoso, J. Mendling, G. Neumann, H. Reijers, A Discourse on Complexity of Process Models, in: J. Eder, S. Dustdar (Eds.), BPI06 - Second International Workshop on Business Process Intelligence, In conjunction with BPM 2006, LNCS 4103, Springer-Verlag, Berlin, Heidelberg, 2006, p. pp.115126.

[33] T. McCabe, A Complexity Measure, IEEE Transactions on Software Engineering 2 (1) (1976) 308–320.

[34] A. H. Watson, T. J. McCabe, NIST Special Publication 500-235: Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric, Tech. rep., National Institute of Standards and Technology (1996).

[35] J. Cardoso, Control-flow Complexity Measurement of Processes and Weyuker's Properties, in: Ardil, Cemal (Eds.), 6th International Conference on Enformatika, Vol. 8, International Academy of Sciences, pp. 213–218.

[36] J. Cardoso, Complexity Analysis of BPEL Web Processes, Software Process: Improvement and Practice 12 (2) (2007) 35–49.

[37] D. B. Skillicorn, D. Talia, Models and languages for parallel computation, ACM Computing Surveys 30 (2) (1998) 123–169.

[38] P. Liggesmeyer, A set of complexity metrics for guiding the software test process, Software Quality Journal 4 (4) (1995) 257–273.

[39] E. J. Weyuker, Evaluating Software Complexity Measures, IEEE Trans. Softw. Eng. 14 (9) (1988) 1357–1365.

[40] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns, Addison-Wesley Professional, 1995.

[41] M. D. Wilkinson, M. Links, BioMOBY: An Open Source Biological Web Services Proposal, Briefings in Bioinformatics 3 (4) (2002) 331–341.

[42] SubEtha-Edit, http://www.codingmonkeys.de/, last accessed 24/10/2007.

[43] C. Sun, S. Xia, D. Sun, D. Chen, H. Shen, W. Cai, Transparent Adaptation of Single-User Applications for Multi-User Real-time Collaboration, ACM Transactions on Computer-Human Interactions 13 (4) (2006) 531–582.

[44] Google Docs, http://docs.google.com/, last accessed 24/10/2007.

[45] Buzzword, http://preview.getbuzzword.com/, last accessed 24/10/2007.

[46] ActiveBPEL Designer, http://www.active-endpoints.com/active-bpel-designer.htm, last accessed 24/10/2007.

[47] Oracle JDeveloper, http://www.oracle.com/technology/products/jdev, last accessed 24/10/2007.

[48] G. Sipos, P. Kacsuk, Multi-grid, Multi-user Workflows in the P-GRADE Portal, Journal of Grid Computing 3 (3-4) (2005) 221–238.

[49] D. Laforenza, R. Lombardo, M. Scarpellini, M. Serrano, F. Silvestri, P. Faccioli, Biological Experiments on the Grid: A Novel Workflow Management Platform, in: CBMS '07: Proceedings of the Twentieth IEEE International Symposium on Computer-Based Medical Systems, 2007, pp. 489–494.

[50] E. Bartocci, F. Corradini, E. Merelli, L. Scortichini, BioWMS: a web-based Workflow Management System for bioinformatics., BMC Bioinformatics 8 Suppl 1. URL http://dx.doi.org/10.1186/1471-2105-8-S1-S2

[51] M. Christie, S. Marru, The LEAD Portal: a TeraGrid gateway and application service architecture, Concurrency and Computation : Practice and Experience 19 (6) (2007) 767–781.

[52] A-WARE Project, http://www.a-ware-project.eu, last accessed 24/06/2008.

[53] Business Process Modeling Notation, http://www.bpmn.org/, last accessed 21/07/2008. (February 2006).

[54] Yahoo Pipes, http://pipes.yahoo.com, last accessed 03/04/2008.