A Workflow-based Environment to manage Software-Testing Process Executions

Duncan Dubugras A. Ruiz, Karin Becker, Bernardo Copstein,
Flavio Moreira de Oliveira, Angelina Torres de Oliveira,
Gustavo Rossarolla Forgiarini, Cristiano Rech Meneguzzi
and Rafaela Lisboa Carvalho

Computer Science Post-Graduate Program
Faculty of Informatics
Pontifical Catholic University of RS
Av. Ipiranga, 6681, Predio 16, Sala 106-FACIN
Porto Alegre - RS, Brazil

Abstract. This work describes a workflow-based environment that manages the execution of software-testing processes. Testing processes require that human and computer resources be handled as dedicated resources, previously scheduled for testing activities, with no overlapping. Two striking features of this environment are: a) the efficient handling of resources by taking into account the capabilities offered by resources required by testing activities, and b) it provides a broader view of all execution steps in a software-testing plan. Hence, it enables a better planning of software-testing process executions, as well as of human and computer resources involved.

1 Introduction

Quality assurance in software products has increased the interest on software testing processes. New software development life cycles have stressed the importance of starting testing as early as possible in the software development life cycle [18]. It also demands increasingly efficient tools and techniques for the description and management of testing processes, as well as qualified staff to execute them, with various profiles.

Testing software means running it in an effort to find errors [6,11]. Testing a software product implies the definition of the Software Test Plan (STP), which defines a sufficiently encompassing number of test cases. According to [11], an STP should describe: (1) the computing environment in which the tests will run, (2) the capabilities required from the testing team, and (3) the sequence of test cases execution, as well as the procedures to handle errors. To properly execute STP, software testing centers are composed of appropriate computing and human (testers and test engineers) resources. Test engineers define and manage the execution of STP, and create the corresponding test reports. Testers develop the individual test cases, according to their competences. For a better management of the testing process, test centers usually split each STP in blocks of test cases, referred to as test activities, which are distributedd among testers.

Dubugras A. Ruiz D., Becker K., Copstein B., Moreira de Oliveira F., Torres de Oliveira A., Rossarolla Forgiarini G., Rech Meneguzzi C. and Lisboa Carvalho R. (2005).
A Workflow-based Environment to manage Software-Testing Process Executions.
In *Proceedings of the 2nd International Workshop on Computer Supported Activity Coordination*, pages 66-74
DOI: 10.5220/0002575300660074

Independent test activities may be allocated to different testers, enabling a parallel execution of these activities, and, consequently, saving STP elapsed time.

To properly manage a test center, there must be a deep understanding of its main characteristics: (1) resources are limited; (2) many of these resources need previous configuration or set-up before use; (3) tests described in the STP must be run in a predefined sequence; (4) distinct test sequences from the same STP can be executed in parallel; (5) unexpected results in a specific test may abort the test, a sequence or the entire test plan; and (6) delays caused by software development teams can significantly impact in all schedule planning of tests of the test center. As a result, the test manager should be aware of (a) which test plans are being executed and at which test activity each test plan is, (b) which resources are currently allocated and which are free to be utilized; (c) what the future resource schedule is; (d) which test plans are waiting to be run; (e) the average time taken to configure a computing system for the execution of a test; and (f) which test plans are waiting for developer feedback before resuming execution. All this information is important to achieve the optimized use of (limited) resources, agility in test execution, as well as to identify bottlenecks in the process. To achieve such level of control, it becomes necessary to provide adequate tools that allow test engineers or process managers to manage its execution and use of resources.

An important aspect that software testing processes share with production processes [24] is the need of full and exclusive dedication of human and computing resources to their respective test activities, durig the whole process. The Enterprise Reference Architecture CIMOSA[1] (Computer-assisted Industry Management - Open System Architecture) states that “Enterprise Activities of a particular enterprise define elementary tasks to be performed in the enterprise which consume inputs to produce outputs and need allocation of time and resources for the full duration of their execution”. In other words, neither the tester nor the computing system allocated to run a test can perform other activities concomitantly. Workflow management systems (WfMS) are targeted at handling the execution business process activities [8,12].However, workflow technology fail to provide support for handling human and computing resources as resources in production-processes [4].

This paper presents a workflow-based environment for the management of the software testing processes, which regards human and computing resources as production-process resources, in the context of the CWf-Flex project. The main contributions of this environment are: (1) the efficient management of resources for the execution of test activities in view of the competences required; (2) efficiency, reliability and an encompassing view of the entire course of the STP by a workflow automation standpoint; and (3) the use of open-source software tools and solutions.

2 A Motivating Scenario

In 1999, our University and a major IT company launched a partnership which established a software test center (STC). This center has been able to identify the specific needs of this kind of process, like the need to work with limited resources with varied

[1] http://cimosa.cnt.pl/Docs/Primer

capabilities. As indicated by the characteristics already mentioned, workflow technology presents an advantageous solution to all these needs.

Between 1999 and 2000, an experiment was made in this test center, using a WfMS to support test management. The WfMS chosen was Changengine [9], and the following evaluation of the advantages and disadvantages of using a workflow approach showed that WfMS did not provide two important characteristics in a test process: the support for human and computing resources as production- process resources and the STP as single execution instance for each process model. Before the effecting of an STP, the computing system to be used must be configured with the proper operational system and a clean software environment. This is necessary in order to ensure the detection of errors happening strictly in the software being tested, and not errors in anything unrelated with the test specification. When identifying bugs and non-conformities, it must be possible to isolate and replicate the error, not only by the testing team but also by the development team. This is essential to ensure the quality of the testing. Since it is impossible to cover all possible hardware and software configurations at the same time in a test center, a prior setup time is frequently needed to reconfigure the machines before running a different set of tests.

A study was also conducted to identify if the standards for the description of workflows and the modeling of production-processes met the requirements of a software test center. The result was the proposition of the conceptual reference CWf [4,20], which merges the WfMC interface 1 [22] with the CIMOSA standard [24] in the description of production-processes. For the design of CWf models, a UML-extension was proposed to support CWf additional concepts: Workflow Activity Diagrams WAD [5]. The *CWf-Flex* project is a direct evolution of the union of these research efforts.

3 Environment Architecture and Description

The main goal of the *CWf-Flex* project is to specify and implement an open, flexible environment for the description of software-testing process, and management of executions, providing support for definition and management of human and computing resources. This environment is composed of three parts: (1) design module, (2) formal description model and (3) execution environment. The design module allows to model testing processes using WAD (Workflow Activity Diagram) [5] WAD is an extension of UML-Activity diagrams and aims at supporting CWf designs. The prototype of the design module [21] is able to generate a corresponding XML specification from a WAD test process modeling.

The formal description model is the standard by which the two parts (description and execution) communicate. A test process specification is made by using XPDL [23], with extensions proposed by CWf [4], in the form of a XML-Schema named XCWf. The main extensions present in XCWf are: (a) capability and capability-set, which may be associated to activities (required capabilities) and resources (available capabilities), (b) the definition of machines as a resource type, and (c) definition of synchronous transitions for the synchronized start of parallel activities or sub-processes. Inherited from CIMOSA, XCWf introduces the synchronized start (S-AND-split) as an additional routing construct besides AND-split, AND-join, OR-split and OR-join. An XCWf+XPDL

Table 1. Table 1 Partial Use Case Description

Use Case	Actor	Description
Insert Project	Test Engineer	Loading of the XPDL+XCWf (text file) specification. Process and resource consistency is verified.
Schedule Activities	Test Engineer	Allocates and schedules testers and computing resources for every activity of a new project. These resources should fulfill the capabilities required by the activities. This scheduling takes into account the work calendar and availability of the human participants. The activity scheduling must abide by the sequence of their execution, as defined by the XML specification.
Execute Project	Test Engineer	Enables the execution of the process. Activities which can be executed are inserted in the allocated participants worklists.
Review Project	Test Engineer	This interface is provided to enable the managing of the test process itself. Its progress can be checked at any time during execution, and the engineer may view which activities are scheduled, which are ready to execute, currently running or complete.
Review Worklist	Tester	Displays the worklist, informing which test activities are ready to be executed and which are currently being executed.
Review Schedule	Tester	Displays the schedule for human and computing resources according to their planned activities.
Execute Activities	Tester	Either sets an activity for execution or notifies that it is concluded. The environment evaluates the specification and enables the execution of the subsequent activities.

specification complies with XPDL using the XPDL extended-attributes option. This allows an XCWf+XPDL specification to be executable by an WfMS according to the XPDL, Interface1 of the WfMC reference model [8]. S-AND-split can be roughly simulated by the use of AND-split plus a set of deadline constraints. In fact, this solution does not guarantee the simultaneous start of activities.

The execution environment, described in the remaining of this section, is a workflow engine that enables the management of software testing processes. Its striking feature is the ability of effectively allocating human and computing resources, besides implementing the typical routines of a STC.

3.1 Overview of the Workflow Execution Engine

Table 1 presents use cases representing the main functionalities of the execution workflow engine. The maintenance of the STC basic data-sets and of human and computing resources is not shown. Only information pertaining directly to the process is kept by the environment, such as work schedule, capabilities (for human resources) and basic configuration (for computing systems).

Activity scheduling is a crucial functionality, since it supports the resource allocation planning activity. In the current prototype it is performed manually by the test engineer, but an automated allocation tool is under consideration, based on the M-DRAP approach [3]. M-DRAP is a multi-agent resource allocation approach where every single resource is managed by an intelligent agent, which, in turn, negotiates its commitments to test activities with the other agents. The data model for the execution environment was designed so as to easily perform this extension, such that only very exceptional cases would require the assistance of a test engineer during scheduling.

3.2 Execution Engine Data Model

The execution environment data model extends the model presented in [14] in two ways: a) it provides support the extensions proposed by the CWf reference model, and b) to persist data related to the work calendar of human resources, their capabilities and the resource activity schedule. Figure 1 shows the class diagram with the main abstractions. Human and computing resources, with their respective capabilities, are represented by the Human, Machine, Capability and Configuration classes. They represent the work force and computational infra-structure the STC has at its disposal.

When a specification is loaded, objects referencing the control flow are created in the Process, Activity and Transition classes. Each activity is associated to one or more Capability, to reference the required set of capabilities from a Human, and to one or more Configuration to reference the required system configurations. Each configuration corresponds to a different machine that will be employed in the activity. There is a special type of activity, named route activity, with no association to Capability and Configuration. It serves only to describe the control flow when its description is not possible in common activities. The activity schedule is established through the Schedule class, where the period in which the activity will be executed is defined, also associating: a Human with all required capabilities, and one or more Machines corresponding to each configuration.

During a process execution, the ProcessHistoric and ActivityHistoric classes store all state transitions of Process and Activity, including the starting states when a specification is loaded. The TransitionHistoric class stores the actual passage of the process through a transition in the model. The state diagrams adopted for Process and Activity are described in [8,12]. Activities ready to be executed are made available to their respective participants through their worklists, taken from the activity states and scheduling, which are stored in Schedule. Every time an activity is completed, the execution environment checks what are the next activities to be executed, according to the corresponding specification. Each process has its set of relevant data, visible to the execution environment, to select which paths are enabled on an Or-splitting activity.

3.3 Implementation Architecture

A client-server architecture has been adopted, where the client is a Java-enabled Internet browser, and the server side is composed of three tiers: presentation, business rules and data persistence. The J2EE technology has been used in the development of environment as a whole. The presentation and business rules layers are enclosed in the

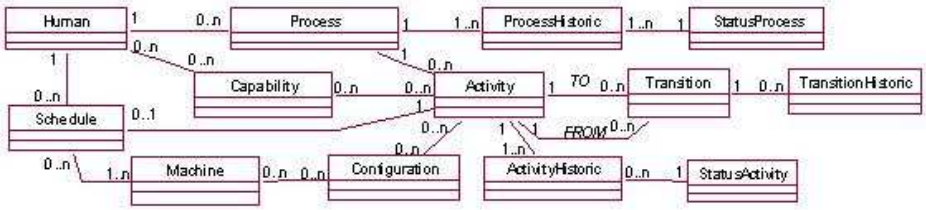

Fig. 1. Partial Class Diagram

Web server, along with the Java Server Pages (JSP). The presentation layer is based on the Model-View-Controller standard. The WebWork[2] framework has been used in this layer, mainly because it offers better functionalities for the validation and conversion of types when compared to other solutions. For the business rules, the Hibernate[3] framework was adopted because it provides abstraction mechanisms for database access, which allows an easier migration among data persistence tools. The use of these frameworks has made possible a better standardization in the source code, and also allowed greater efficiency and quality in the development process. For the data persistence layer, the PostresSQL[4] DBMS was chosen for both its transaction support and level of conformity to the SQL standard.

Figure 2 shows a typical Web page, the commitments of a tester, with the visual standard adopted. The upper menu in the screen belongs to the browser being used. The left side menu presents a hierarchic structure consistent with the modeled functionalities, and allows the easy access to the different system functions. These menus options may vary from user to user, depending on the logged users profile. Through this menu, all data relevant to the STP may be retrieved, including test plans being executed and completed, and resource commitments to the STP (by reviewing participants schedules and worklists, etc).

3.4 Innovative aspects of the Execution Environment

The implementation solution presented here fulfills the needs identified in section 2 above. It is a system that offers an efficient management of the test processes and the commitment of resources to these executions. In order to do that, the environment enables the resource scheduling for the entire process even before it starts its execution. Such scheduling takes into consideration testers capabilities and work calendars and previously scheduled activities of other STP. This permits test engineers to predict the involvement of the work force with test activities, to size up their test teams and computational infra-structure and to properly plan the growing of the test center. Besides, it guarantees the simultaneous start of activities, when specified in the model.

[2] http://www.opensymphony.com/webwork/wikidocs/Documentation.html
[3] http://www.hibernate.org/5.html
[4] http://www.postgresql.org/docs/

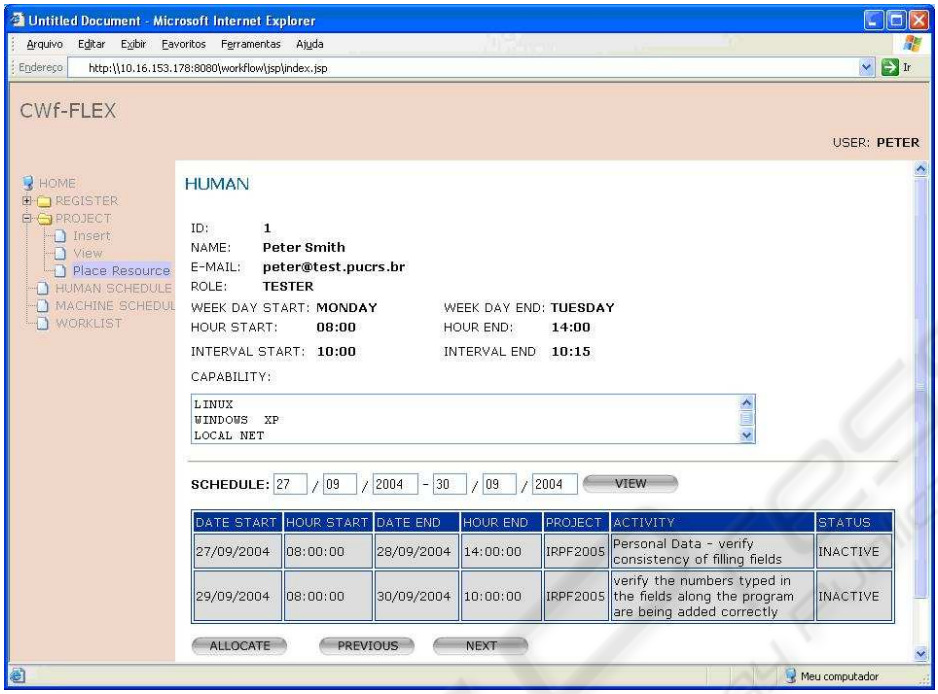

Fig. 2. Screen showing commitments of a Tester.

4 Related Work

[13] presents many WfMS development projects using open-source software, which supports XPDL at various degrees of conformance. Our approach, based on [4], looks at resources as production-process resources, which is not supported by the [23] and [16] specifications. Likewise, it is not supported by either other open-source projects such as YAWL [1] or commercial tools such as AQdevTeam [2].

TestDirector [15] is a leading workflow-based tool that addresses the management of software testing processes. It support resource allocation, permitting to view resource skills, assignments and load rates. Our approach works with exclusive use of resources by activities and not with load rates. Testlog [17] is a tool that permits the definition of resources similar to our approach: testers, hardware platforms, test configurations, etc. In addition, it permits the assignment of resources to test cases. However, it supports neither the testers capability description nor resource scheduling, as opposed to what our approach proposes.

[19] introduces a multi-agent approach for the modeling and scheduling of resources in activity coordination. Two types of resources are discussed, schedulable and not schedulable. An abstract resource model and four basic operations for its manipulation are presented: identification, reservation, acquisition and release. Even though the schedulable resources are adequately typified, the authors do not explore them fully.

They state that even in a process with a previously defined order, the agents will manage their appointments and execute activities at their own discretion. Our solution keeps the global scheduling of activities e supervises their execution, according to the test process specification, which is essential to the execution of testing activities.

[7] presents a WfMS for grid computing, named GridFlow, and address the workflow scheduling problem using a fuzzy timing technique. Similar to [3] approach, GridFlow is an agent-based resource manager, but oriented to grid resources. Grid computing means the execution of multiple parallel tasks with maximum resource utilization. [7] states that WPDL [22] is sophisticated and too generalized for grid computing. Resources have different capabilities and should be allocated properly in our approach, as opposed to GridFlow. However, the fuzzy timing technique can be useful to improve our resource management.

5 Conclusions

This work has detailed the research made in the context of the *CWf-Flex* project, for the specification and implementation of the execution environment. The characteristics and problems of test process management were described, and, in particular, the necessity to adequately support human and computing resources, especially during test activity scheduling. Besides that, it also references the adoption of the XPDL standard for workflow description, along with the extensions proposed by [4], such as the procedures for test process specifications exchange. The environment was developed using only open-source software tools, which allows its portability to different operational systems. The addition of new modules is to be considered for future versions. The main contributions are (1) the previous scheduling of resources to the activities, even before the execution of the test process, (2) the consideration of the capabilities of testers when selecting activities for them, (3) the control of collisions in the schedule, and (4) the beforehand knowledge of resource commitment in the test center.

The project's current stage is the installation of the environment in an STC, with the intention of assessing requirements compliance, accessible use and the quality of generated productivity information. Apart from that, the implementation of the [3] dynamic resource allocation approach in the project core, as well as a method for the definition of workflow processes, with resource allocation, is being considered in a near future. This method will be based on the description and execution environment specification and the formalization of the description model.

Acknowledgment

This work has been supported by CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico, a Brazilian Federal Research Agency.

References

1. Aalst, W.M.P.v.d.; Aldred, L.; Dumas, M.; Hofstede, A. (2003) Design and Implementation of the YAWL System. Queensland University of Technology (FIT-TR-2003-07).

2. AutomatedQA Corp. (2004) AQdevTeam. http://www.automatedqa.com /products/aqdevteam.asp
3. Bastos, R. M., Oliveira, F. M., Oliveira, J. P. M. (1998) Decentralised Resource Allocation Planning through Negotiation In: Inteligent Systems for Manufacturing: Multi-Agent System and Virtual Organization. Kluwer Academic Publishers, p. 67-76.
4. Bastos, R. M.; Ruiz, D. D. A. (2001) Towards an Approach to Model Business Processes using Workflow Modeling Techniques in Production Systems. In: HICSS-34, Proceedings. IEEE Computer Society.
5. Bastos, R. M.; Ruiz, D. D. A. (2002) Extending UML Activity Diagram for Workflow Modeling in Production Systems. In: HICSS-35. Proceedings. IEEE Computer Society.
6. Beizer, B. (1990) Software Testing Techniques. New York: Van Nostrand Einhold.
7. Cao, J.; Jarvis, S.A.; Saini, S.; Nudd, G.R. (2003) GridFlow: Workflow Management for Grid Computing. In: 3rd IEEE/ACM CCGrid 2003. Proceedings. IEEE Computer Society.
8. Hollingsworth, D. (1995) The Workflow Reference Model. Hampshire, UK: WfMC.
9. Hewlett-Packard Company. (2000) HP Process Manager: (former Changengine v. 4.2). http://www.ice.hp.com/ cyc/af/00/index.html
10. Jacobson, I.; Booch, G.; Rumbaugh, J.(1999) The Unified Software Development Process. Addison-Wesley.
11. Kaner, C.; Falk, J.; Nguyen, H.Q. (1999) Testing Computer Software. John-Wiley & Sons.
12. Leymann,F.; Roller,D. (2000) Production workflow: concepts and techniques. Prentice Hall.
13. Manageability.org (2004). Open-source workflow engines written in Java. http://www.manageability.org/blog/stuff/workflow_in_java
14. Meneguzzi, C.R. (2002) TC-Wf: Applying Workflow Technology on Software-Testing Planning. Porto Alegre-RS, Brazil, PPGCC-PUCRS. (MSc. Dissertation, in Portuguese)
15. Mercury Interactive Corp. (2003) Implementing an Effective Test-Management Process. White Paper. http://www.mercuryinteractive.com
16. Object Management Group.(2000) Workflow Management Facility Specification, V1.2. www.omg.org/docs/formal/00-05-02.pdf
17. PassMark Software (2003) Testlog User Guide. www.testlog.com/ftp/TestLogUserGuide.pdf
18. Pressman, R. S. (1997) Software Engineering A Practitioners Approach. McGraw-Hill.
19. Podorozhny, R.M.; Lerner, B.S.; Osterweil, L.J. (1999) Modeling Resources for Activity Coordination and Scheduling. In: COORDINATION99. Proceedings. LNCS 1594.
20. Ruiz, D. D.; Bastos, R. M. (2002) C-Wf: a Model to Represent Workflow Business Processes in Production Systems. Journal of Applied System Studies, Cambridged, England, v.3, n.1.
21. Velasco, L.H. (2004) Workflow Designer: A workflow-process design tool. Porto Alegre, FACIN-PUCRS (Undergraduate conclusion project, in Portuguese).
22. Workflow Management Coalition. (1998) Interface 1: Process Definition Interchange Process Model. Hampshire, UK: WfMC. (Official Release 7.04)
23. Workflow Management Coalition. (2002) Workflow Process Definition Interface XML Process Definition Language. Hampshire, UK: WfMC. (TC-1025, Final Draft 1.0)
24. Zelm, M.; Vernadat, F.B.; Kosanke, K. (1995) The CIMOSA business modelling process. Computers in Industry, v. 27, n. 2, p.123-142. October.