Framework for Verification of Preserved and Redeployed Processes

Tomasz Miksa
Stefan Pröll
Rudolf Mayer
Stephan Strodl
Secure Business Austria
Vienna, Austria
{tmiksa, sproell, mayer, sstrodl}@sba-research.org

Ricardo Vieira
Jose Barateiro
INESC-ID Information Systems Group
Lisbon, Portugal
{rjcv, jose.barateiro}@ist.utl.pt

Andreas Rauber
Vienna University of Technology
& Secure Business Austria
Vienna, Austria
rauber@ifs.tuwien.ac.at

ABSTRACT

Preservation of processes requires not only identification of all process components, but also interception of all interactions of the process with the external components. In order to verify if the collected data is sufficient for the purpose of redeployment, as well as to verify that the redeployed process performs according to expectations, a framework for verification is needed. This paper presents a framework for verification of preserved and redeployed process. The applicability of the framework is demonstrated on an eScience use case. The preservation and the redeployment of the eScience process is tested by migration to another substantially different environment.

1. INTRODUCTION

Traditionally research in the area of digital preservation deals with preservation of static information like documents, scans, and other kinds of data. The long term preservation of entire systems and processes was not in the centre of attention. Addressing this new challenge requires new methods and processes which ensure that the process context is described adequately, as well as the sufficient information, which enables future redeployment, is collected. No matter how well-engineered the process for preservation of processes is, it cannot guarantee that all necessary information required to run the process were recorded. Given the complexity of preserving entire systems and processes, we thus need to derive means for solidly verifying, whether a process being re-deployed performs correctly according to preservation goals. We need to ensure that not only sufficient information is collected during planing and preserving of the process, but also confirms that the redeployed process performs according to the expectations of the redeployment scenario.

The verification of redeployed processes is a complex task which may vary in its form due to several factors: the way the processes are specified, the drivers for their preservation, the preservation strategies applied; the reasons for the redeployment, the redeployment environments, etc. However, regardless of these differences all of the processes must be verified at the end of the redeployment. Otherwise, there is no guarantee that the process running in the redeployed environment is the one which was meant to be redeployed. Such evidence is crucial in litigation cases when the correctness of the original process executed at some time in the past could be questioned and the only way to check this is to rerun the original process. In such cases, the method for verification of redeployed processes should provide irrefutable evidence that the redeployed process is behaving exactly the same way as the original. While in the domain of eScience [7] and Research Infrastructures [11], where scientists make scientific discoveries by creating and constantly improving the processes which transform the Big Data [10], the verification of redeployed processes is used to verify research results or apply earlier models on a new data.

In this paper we present the VFramework which is a framework for verification of preserved process. It is a refinement of a conceptual framework presented in [6]. It consists of 7 steps that describe key actions which have to be performed in order to verify any kind of redeployed process. The VFramework can be applied not only to fully redeployed processes but is also capable of evaluating partial redeployments. Moreover, the VFramework can also verify both identical and re-engineered processes. We present application of the VFramework for verification of a redeployment of an eScience process in the domain of sensor data analysis, which was extracted from its original environment and was redeployed in various environments different from the original one. The VFramework was applied successfully to assess these redeployments.

The paper is organized as follows. Section 2 presents the state of the art. In Section 3 the requirements set to the VFramework and the steps of the VFramework are described. Section 4 describes application of the VFramework to the use case. We provide conclusions and future work in Section 5.

2. STATE OF THE ART

In [6] a conceptual framework for evaluation of emulation results was presented. It was demonstrated in [5], that the framework can be successfully applied to evaluate the conformance and performance quality of applications and processes redeployed in an emulator. This was demonstrated on case studies in which the framework was used to evaluate the emulation of a video game and an accounting program. The VFramework presented in this paper is a refinement of the framework for complex, potentially distributed processes. It provides detailed specification of actions which have to be performed for verification of redeployed processes.

In ISO 12207 [2] the life cycle processes for systems and software were defined. "It contains processes, activities, and tasks that are to be applied during the acquisition of a software product or service and during the supply, development, operation, maintenance and disposal of software products" [2]. It does not consider the redeployment as a part of the life cycle and hence provides no guidance for the scenario considered in this paper. The standard defines also the Software Specific Processes and lists actions which are needed for the Software Verification Process and the Software Validation Process. However, these processes belong to the Software Support Process category which assists the software implementation process. As a consequence, these processes are highly coupled with the software development, what is not in scope of our investigations. Summing up, the ISO 12207 does not specify a process for verification of redeployed software processes as presented in this paper.

The IEEE 1012 standard [1] specifies a process for software verification and validation. This process addresses the following software life cycle processes: acquisition, supply, development, operation and maintenance. It is compatible with ISO 12207. It defines tasks, required inputs and outputs to conduct verification and validation (V&V) of the software at all aforementioned life cycle processes. The V&V process for the maintenance process considers migrations to other environments. This overlaps with some of the requirements we set to the framework for verification of redeployed processes (see Section 3), i.e. the system is migrated to the other platform when the original system is still available. However, it does not consider the situation when the system or the process is disposed, deposited and redeployed after some time. Furthermore, the standard specifies only a high level list of activities applicable in several maintenance scenarios which are rather focused on verification and validation of the activities performed to keep the the system running (e.g. system updates, bug fixing, enhancements to the functionality), rather than on digital preservation scenarios. The VFramework proposed in this paper provides more detailed guidance and can be applied to a broader range of digital preservation scenarios.

3. VFRAMEWORK

The VFramework was created to verify that the redeployed process performs according to expectations. There were two main requirements set to the framework.

Firstly, the framework has to be independent of the situation in which different digital preservation actions were applied to the full process or to different parts of the process. In such situations some of the process' parts may be substituted, re-engineered, emulated, migrated, etc. As a result, the redeployed process which is to be verified is not necessarily an exact copy of the original process. The framework has to be capable of verifying the execution of similar processes or their parts. By similarity of processes we mean a situation, in which the functionality or characteristics of the process have been altered, but the deviation is either desired (e.g. faster computation) or acceptable (e.g. some functionality is limited but for the purpose of redeployment it is not required). Such situations may be an inevitable side effect of the digital preservation actions or a consequence of deliberate actions (e.g. improved implementation of the process). The framework has to support such situations regardless of its origin and be capable of evaluating full and partial redeployments of processes.

Secondly, due to the high variety of the nature and implementation of the processes and a wide range of potential users requirements that has to be considered, the framework has to be elastic to cover all these requirements and settings. Therefore it has to remain at a relatively high level of abstraction and be customizable for the concrete processes which are going to be preserved. The guidance on customization has to be provided by the framework in order to achieve the comprehensiveness of the process verification.

The VFramework fulfils these two main requirements. The VFramework is depicted in Figure 1.It consists of two sequences of actions. The first one (depicted in blue) is performed in the original environment. The results of the execution of each of the steps of this sequence are stored into the VPlan. The VPlan is a machine readable document in which all of the information about the original environment are kept. The second sequence (depicted in green) is performed in the redeployment environment. The necessary information for completion of each of the steps is obtained from the VPlan.

Original environment denotes the system in which the process, which is going to be preserved, is deployed and operates. The redeployment environment is the system in which the process will be installed once the decision to redeploy the preserved process is taken. It is very likely, that the redeployment will take place at some distant time in the future, when the original platform does not exist any more and the process may need to be re-engineered to fit it into the new system.

Apart from descriptive metadata, the VFramework uses two kinds of data: verification data and redeployment performance data. The verification data is collected during the execution of the process in the original environment. It provides information on details of the execution of process instances, focusing on measuring significant properties. External interactions data is part of verification data representing a record of all interactions of the process with external components during the execution of a specific process instance in a scenario to be used for verification. This data is reapplied in the redeployment environment to ensure determinism, by recreating the same external interactions. The redeployment performance data collected during the execution of the process in the redeployment environment. It provides information on details of the execution of the process instances, focusing on measuring significant properties. It is used for comparison with verification data to assess the redeployment.

The steps of the framework are described below.

1. Describe the original environment The aim of this step is to describe the process and document its context by identifying environment dependencies in which the process is deployed. Also motivation for preservation of the process, considered redeployment scenarios and a set of example instances to be used for verification are determined. This corresponds largely to steps 1-3 of the "Define Requirements" phase in preservation planning [4], with the first step being subdivided into two more fine-grained steps.

1.1 Describe the process The information should describe the process itself but also the context in which the original process operated. A detailed description of not only software and hardware requirements, but also legal aspects is needed. Such information can be provided in multiple forms. One of them could be the context model [8], which is an ontology based model for description of processes and their dependencies.

1.2 Define set of potential redeployment scenarios The purpose of the redeployment has to be defined. This information has significant impact on the process of verification, because it impacts the type of measurements and the results they are supposed to fulfil. For example, different requirements are set to the process which is supposed to be an exact copy of the original process redeployed for a purpose of litigation case when the correctness of the original process has to be proven and therefore the redeployed process is verified for being identical. Different requirements are set to the eScience process which is redeployed with some of its parts substituted with components of the same functionality but improved quality (e.g. faster computation, more accurate results, etc.). In such cases some of the measurements may be ignored or interpreted differently, e.g. accuracy of results should not be worse than the original, but does not need to be exact. Verification focuses in this case on ensuring the functionality is achieved, but the significant properties related to part where the changes were introduced should be treated differently.

1.3 Select process instances to be used for verification Process may have several execution paths and therefore instances of the same process may vary considerably. In this step the instances of the process which will be used for verification are selected. They have to be chosen according to the considered redeployment scenarios. The instances selected at this step will be used to collect both verification data from the original environment, as well as the performance redeployment data. The description of selected instances should specify in a comprehensive way all actions which were performed when running the process. These could be depicted by sequence diagrams, activity diagrams, use case diagrams, textual description, etc. The way it is specified depends on the level of automation of the process, e.g. if it is a manual process or formally specified in BPMN or executed within a work-flow engine. Furthermore, the values of all parameters and input values must be documented.

1.4 Identify significant properties to be preserved The significant properties which have to be preserved and then evaluated have to be specified. They can either be collected at this step or obtained from preceding activities, e.g. preservation planning. However, regardless of the source, it is important that the significant properties reflect both functional and non-functional requirements of the process. It is important to determine which significant states of the object are to be measured as the significant property. These significant states could be: target state, continuous stream or series of states.

2. Prepare system for preservation The aim of this step is to identify the interactions of the process, i.e. all inputs and outputs of the process, but also configurations of process parameters, as well as influences of other components sharing the process environment or used indirectly by process components. This information is needed in order to ensure deterministic execution of the process and thus ensure reliable assessment. The steps should be conducted in view of redeployment scenarios and significant properties defined for the process.

2.1 Determine process boundaries The process boundary specifies which elements belong to the process and which elements belong to the external environment in which the process operates. It is possible to define different process boundaries depending on the scenarios for redeployment. For example, if the scenario assumes redeployment of only a part of the process which will be fitted into another process, then only the redeployed parts of the original process are within the boundary. However, there may be a second scenario in which the full process is redeployed, then a second boundary has to be defined which covers the entire process. Boundaries may also be influenced by the degree of control one can exert on specific components (e.g. external web services) and their importance for redeployment as well as their stability. In all cases, the description should ensure that the process boundaries are specified clearly, i.e. a distinction between elements which are part of the process to be preserved and which are external services with which the process exchanges data has to be made. This is particularly important in case of distributed processes which are using the Service Oriented Architecture for their implementation, or those deployed in the cloud.

2.2 Determine external interactions For each of the specified boundaries the external interactions have to be identified. External interaction denotes situations in which elements within the process boundary interact with elements from outside of the boundary. External interactions may be critical for the correct execution of the entire process because any changes in the external components may cause changes in process execution. For example, the web service which provides data for one of the process steps may change (change of interface, implementation of algorithms, etc.) or become unavailable [9]. As a result, the process can perform differently (providing different outputs) or cannot be run any more. Another example could be encryption and necessity to access an authentication centre. When the certificate is not available any more, then the communication

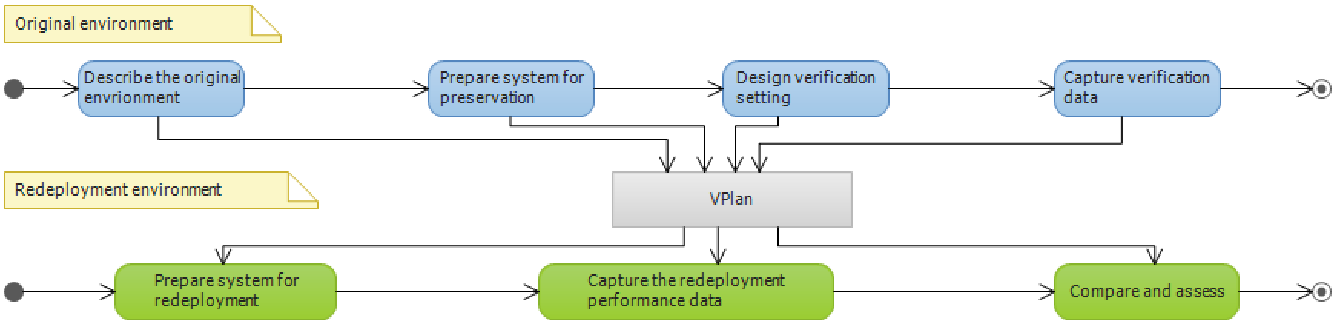
Original environment
Describe the original envrionment
Prepare system for preservation
Design verification setting
Capture verification data
VPlan
Redeployment environment
Prepare system for redeployment
Capture the redeployment performance data
Compare and assess

Figure 1: VFramework - framework for verification and validation of preserved business processes

cannot take place unless the authentication is removed (if the redeployment scenario allows for this).

Special attention has to be paid to indirect external interactions and consequences for the process which might not always be visible at the first sight. For example the operating system if not included within the process boundary, its version and all system updates may alter the execution of the process. For all the requirements which focus on the visual presentation, the installed fonts, appearance settings, colour schemes of the system may be such influencers. Other digital objects which coexist in the system may also have impact. For example, processes running in the background (e.g. virus scan software, remote desktop software) can significantly affect the performance of a system. Moreover, other processes may share common data with the examined process and may modify the data what may result in the non-deterministic execution of the analysed process. Furthermore, all user or system I/O (e.g. keyboard, network, specific hardware components such as system clock, etc.) that are outside the process boundaries need to be identified.

2.3 Determine internal interactions The process may consist of several components which have their own settings. All these settings must be determined at this step. Furthermore, some of the process components depend on further software tools or libraries which may vary in version or settings. Some examples of these could be: virtual machine, database software, libraries, software device drivers, fonts, codecs, etc. The detected versions of components have to verified if the detected versions have not been modified or customized. If some of them were modified (e.g. modified config files) and this has an impact on the process, then they have to be preserved as well. Besides the software dependencies, the underlying hardware has to be considered when searching for potential internal interactions. The process may depend on some proprietary and unique hardware equipment or the underlying hardware may have some specific implementations of algorithms affecting the results obtained in the process. For example, some of the hardware bugs may affect the results delivered by the process. These results will only be achievable on a particular hardware platform (e.g. well-known Pentium FDIV bug had an impact on the results of floating point calculations, and therefore could alter the results of the whole process upon correct redeployment).

2.4 ensure deterministic behaviour To allow verification of redeployment we need to ensure a process performs deterministic. Thus, all interaction identified in 2.2 and 2.3 need to be verified for completeness to ensure deterministic re-execution. If this is not possible within the generic process, adaptations have to be made specifically for verification. If the determinism cannot be ensured, the verification of processes is not very likely to be possible. The investigation of determinism of the process should be conducted in view of considered redeployment purposes. In some settings, some of the non-deterministic influencers are affecting measures which are not important for the purpose of the redeployment. For example, when the exact execution speed does not have to be achieved, then all the non-deterministic influencers of this criterion does not have to be considered.

When one of the process steps exchanges data with some third party component (external interaction), the communication can be recorded and replayed in the redeployment environment. If the process depends on the component which affects determinism of the process, it may be possible to substitute the component with a mock-up which does not have this deficiency. An example of such a solution for web services can be found in [9], if one of the steps of the non-deterministic process depends on a random number generator, then it may be substituted with a mock-up which always provides the same sequence of values as the one recorded and thus the process becomes deterministic. Of course, such changes to the process must be documented and possibly reverted after the verification process is finished in the actual redeployment, but for the purpose of verification they should be present.

3. Design verification setting The aim of this step is to identify the measurement points of the process, specify metrics used to assess quality of preservation actions and couple them with thresholds which are used as criteria for the assessment. The measurement points can be defined as points of the process where data enabling reasoning about correctness of the process execution is collected. The investigation should be conducted in view of redeployment scenarios and significant properties defined for the process.

3.1 Specify measurement points Measurements points for both internal and external interactions must be described unambiguously and precisely, because the given value can be measured in different ways and in different parts of the process and therefore not always the same values may be obtained. For example, the output of a process that transforms some images into PNG files is selected as a measurement point. This seems to be a clear requirement but without explicit definition of what is exactly measured the results may vary, because the bit streams which write the PNG file to the disk can be compared on the fly or the files already written to the disk can be opened and analysed by image recognition algorithms. In the first case, different libraries may have been used to transform the image (e.g. library was replaced in the redeployment) and as a result the outputs may be different at the bit level, while in the case of image recognition algorithms the images may turn out to be identical. Both approaches are valid and can be used. As the example shows, the choice of the measurement point depends on the requirements and intentions of the future redeployment. We thus need to identify, for each significant property of the process, on which level these must be captured. According to [6], the core levels are (1) bit level file storage, (2) the rendering of an internal state in a the system memory, (3) memory of an output device (e.g. video card memory (vitalized o real)), (4) port communication (e.g. VGA port, network interface, audio port) or (5) the actual output device (screen, speakers, actuator). If the verification aims to check if the rendering algorithms are exactly the same, then the bit comparison seems to be a better measurement point. But if it is allowed to modify the process and only the final visible product needs to be verified, then the second approach should be selected. It may be advisable to take measurements at multiple measurement points and collect the data for all of them. The choice of the measurement point which is most accurate for the redeployment environment will be left to the person redeploying the process who is aware of the reasons and requirements set to the redeployed process. While measurement points will usually relate to external interactions (e.g. result storage, communication with user or external system), internal interactions within process may be useful to capture for partial redeployments, to allow application and verification of a wider range of preservation actions (such as component replacement) and to allow more flexible redefinition of the boundaries identified in step 2.1.

3.2 Specify metrics for preservation quality comparison The significant properties which were selected in the first step have to be decomposed from high level significant properties into tangible and measurable metrics which can be measured and identified directly in the process. A wide range of techniques can be used for decomposition. Especially techniques stemming from requirements engineering may be particularly useful in this step, e.g. goal modelling [12], GQM method [3], etc. It is also advisable to specify metrics which can identify what the process should not do. In many cases it is easier and quicker to identify the forbidden behaviour or an incorrect state of the process. Then the redeployment can be rejected without a necessity of checking other metrics.

Having defined the metrics, the target values are assigned. These values will be used as the criteria for the assessment. They have to be specified in view of considered purposes of the redeployment. This information has significant impact on the process of verification, because it impacts the importance of available metrics and results they are supposed to achieve. Target values itself can be specified in different ways, e.g. metric A equals Y, metric B is maximum 120% of the original value, etc.

3.3 Aim for automated measurements capture When the VFramework is applied during planning of the preservation activities when different preservation scenarios and activities are considered, the possibility to automate measurements decreases the time needed for evaluation of alternative preservation strategies. This has lower importance when the VFramework is used during the preservation phase and redeployment phase, when the preservation strategies are already defined. Regardless of the phase, automation of measurements eases the process of verification.

4. Capture verification data This step has two main tasks. Firstly, to configure the capturing environment for collection of verification data. Secondly, to collect the verification data while the process is monitored by tools which trace process interactions.

4.1 Prepare system for capturing In this step the capture environment is configured. Either a clean environment is created in which the process is deployed, or the operational system is used directly.

4.2 Prepare data capture tools Tools for capturing external interactions, as well as verification data are introduced to the capture environment in the next two steps.

4.2.1 Set up tools for capturing external interactions Tools which will intercept external interactions of the process are installed in the capture environment. The captured information will be used to ensure deterministic execution of sample process instances (step 1) in the redeployment environment.

4.2.2 Set up tools for capturing verification data Tools which collect data in previously specified measurement points are installed in the capture environment. The captured information will be used to evaluate performance of the redeployed process.

4.3 Run the process and capture data When the capture environment has been configured and the tools for capturing data are in place, the instances of the process, which were identified in the first step, are executed. The data is being collected during and after the execution of the process.

4.4 Verify validity of captured data Once the execution of process instances has finished, the recorded data is verified for its correctness. This could be either manual or automatic action, which checks if all the measurements were stored correctly, e.g. if the log files are not empty. If all the data is correct then it is stored into the VPlan.

5. Prepare system for redeployment This is the first step performed in the redeployment environment. This step has three main objectives. Firstly, to configure the redeployment environment for collection of redeployment performance data. Secondly, to redeploy the process in the new environment. Thirdly, to execute process instances.

5.1 Prepare redeployment environment The environment in which the process will be redeployed has to be selected. Tools which ensure determinism during execution of the process, as well as the tools used for data collection have to be installed.

5.1.1 Set up redeployment system Either it will be a clean system or a system in which some other processes already exist. This depends on the purpose of the redeployment. If the process is run in an environment shared by other process an analysis of possible external interactions has to be conducted in order to ensure that the determinism of the redeployed process is not affected by the new environment.

5.1.2 Set up external interactions replay to ensure determinism The external interactions data is used in this step to recreate the interactions of the system. Tools which allow replaying of this data have to be installed in the redeployment environment.

5.1.3 Set up data capture tools Similarly to the step 4.2, the tools which extract redeployment performance data are installed in the redeployment environment. These tools will collect data needed for verification of the redeployed process at the predefined measurement points.

5.2 Redeploy preserved process The preserved process is redeployed in this step. Required adjustments required to run the process in the new environment are done and the same instances of the process which were used in the original environment are executed.

5.2.1 Identify required preservation actions to enable redeployment The aim of this step is to ensure that the process becomes operational in the new environment and that all of the process instances defined in the first step can be executed.

It is very likely that the preserved process will have to be re-engineered in order to be fitted into the new environment. For example, in the given environment a certain library responsible for encrypted communication with a web service cannot be used. However, a substitute library which allows to communicate with a web service without the encryption might be available. Then such substitution has to be made in order to make the process operational (only if the redeployment scenario does not exclude such action). In this step all kinds of preservation actions such as replacing a library with another one, cross-compiling code, migrating a file, putting an additional wrapper around the component, etc. may be applied.

5.2.2 Re-run the set of process instances Process instances, which were defined in the first step and executed in the original environment to collect verification data, are executed in this step in order to create redeployment performance data. The execution is controlled by the tools which ensure determinism of the process.

6. Capture redeployment performance data The aim of this step is to collect the redeployment performance data from the new system and verify if the data collection conditions were fulfilled.

6.1 Collect redeployment performance data The redeployment performance data is recorded by the tools which are monitoring the execution of process instances. All this data is collected and will be used for comparison with the verification data.

6.2 Verify validity of captured data Before the data can be used for comparison, its validity and fulfilment of assumed level of determinism of the environment needs to be checked.

6.2.1 Verify if required level of determinism was reached Results has to be analysed considering the required level of determinism in the environment. If it was possible to ensure it and the tools which were introduced for this purpose in the step 5 performed its task correctly then the requirements are fulfilled. Otherwise, the procedure has to be repeated starting from the step 5 and new ways of ensuring deterministic execution of the process have to be introduced.

6.2.2 Verify correctness of capture data Similarly to the step 4.4, the collected redeployment performance data needs to be verified before it can be used for further analysis. This could be either manual or automatic action which checks if all the measurement were stored correctly, e.g. if the log files are not empty.

7. Compare and assess The comparison of significant properties measured in both environments is conducted in this step. The comparison is described in a report and a decision about fulfilment of redeployment purposes is made.

7.1 Compare redeployment performance data and verification data In this step the comparison between verification data and redeployment performance data is conducted. The comparison has to be done by comparing the data collected at each of the measurement points of the original process with the data collected at each of the measurement points of the redeployed process. Due to the changes which might have been introduced to the process, some of the measurement points may not be available, if so, the comparison is either omitted or another corresponding point is used.

7.2 Conduct preservation quality comparison The metrics which were specified in Step 3.2 are calculated for the redeployed process. These metrics allow to assess the quality of preservation action. These metrics are always interpreted depending on the redeployment scenario, because they may have different target values depending on the scenario. In some scenarios the required functional or non-functional metric must be fulfilled, while in the other scenario it is not a requirement.

7.3 Provide summary report A report summarising the comparison is created. The report is supposed to deliver credible information about the state of the redeployed process, measurements made, metrics and their expected values and any alterations detected which are not compliant with the purpose of the redeployment.

7.4 Make the final decision The final decision is taken by the auditor who knows the reason for the redeployment and using the report can make a credible decision.

7.5 If positive, remove tools used for verification If the process is positively evaluated, then the tools for ensuring determinism are removed from the environment, unless they are needed for the redeployment. The original implementations or substitute services providing the full functionality are used instead. Similarly the tools for data collection can be removed from the environment.

4. VFRAMEWORK EVALUATION

In this section we test the applicability of the framework on an eScience use case. The origins of the use cases and reasons for its preservation, redeployment and verification are provided in Section 4.1. In Section 4.2. an explanation is given, how each of the steps of the VFramework was executed to verify the process migration from Windows to Linux environment.

4.1 Use case description

The institution which provides the use case comes from the domain of civil engineering. It owns and maintains a system for supporting the process of acquiring and managing information captured from sensors installed in dams, with the objective of studying the structure behaviour and thus prevents any accidents that might happen. The experts working for this institution execute many processes which are used for managing the visual inspections, physical models, mathematical models, and technical documents. They also use data analysis tools such as tabular and chart reports and graphical representation of geo-referenced information. Only the long term data analysis allows to make correct predictions about the state of the dams. In case of any disasters the sensor data may need to be reprocessed to look for mistakes in the original process. Therefore being able to rerun the process using either the original sensor data or the new data is crucial requirement for this organisation. Digital preservation of processes was selected as a strategy to address this requirement.

The process which is used for testing of applicability of the VFramework is depicted in Figure 2. This process is run on by a scientists, who use their desktop workstations with Windows XP as the operating system. The process consists of 5 steps which fetch the sensor data from an external web service (*Get Data Files*) download the R (*Get R script*)and TEX (*Get Tex script*) scripts for processing the data, generate PNG and TEX files (*Generate Plots*) which are finally compiled into PDF report (*Generate Report*).

4.2 VFramework application

1. Describe the original environment The first step consists of sub steps which describe the process in details, choose its significant properties and process instance used for data collection, as well as specify potential redeployment scenarios.

1.1 Describe the process We have described the purpose of the process, identified the users of the process and documented the components which build the process. We have used tools for detecting software dependencies and documentation provided by the owner of the process.

1.2 Define set of potential redeployment scenarios In this step we have defined two potential redeployment scenarios. Scenario 1 assumes that the process will be redeployed in order to be fully operational. It assumes that the external communications (e.g. web service) will be available. Scenario 2 assumes that the process will be redeployed in order to confirm that the values and plots presented in the scientific paper were obtained from a cited data set. The data set which was used for processing will be provided and there will be no need for communication with the web service. Further steps of this framework are always performed in view of requirements of these scenarios.

1.3 Select process instances to be used for verification For each scenario we have selected 10 instances which were differing in the configuration of parameters. For scenario 1, parameters for fetching data from a web service were randomly altered. For scenario 2, 10 data sets from 10 different locations were used.

1.4 Identify significant properties to be preserved We have conducted interviews with the owner of the process in order to collect the list of significant properties important for each of the considered scenarios. We have collected both functional requirements, e.g. the system must be able to generate sensor data for quantitative interpretation, and non-functional, e.g. the system provides correct results. The significant properties were grouped by the scenario for which they are important (sometimes both).

2. Prepare system for preservation In this set of steps we have identified the process boundaries and described its interactions. Our analysis also included determination of non-deterministic influencers and strategies for their mitigation.

2.1 Determine process boundaries There are two scenarios of redeployment considered. In the first one, which assumes that the process is redeployed in order to be fully operational, the presence of the web service is assumed. Therefore we put the web service outside of the process boundary. In case of the second scenario, when the redeployment is done for the purpose of validation of experiment's results, we exclude first three steps of the process (*Get Data Files*, *Get R script*, *Get Tex script*) from the process boundary. Data used in the original experiment will be applied to the process directly. In both cases the scripts which are implementation of process steps: *Generate Plots*, *Generate Report* are within the process boundary. The operating system and the software required to run the scripts is not included as part of the process.

2.2 Determine external interactions In case of the scenario 1 the process transforms the data obtained from a web service. This is identified as an external interaction. In both redeployment scenarios, steps of the process are executed manually by executing commands using the keyboard. Therefore the input from the keyboard is another type of external interaction. Finally, the files produced as the output of the process are displayed on the LCD screen in the form of PDF document. The visual presentation of the results on the screen is also identified as external interaction.

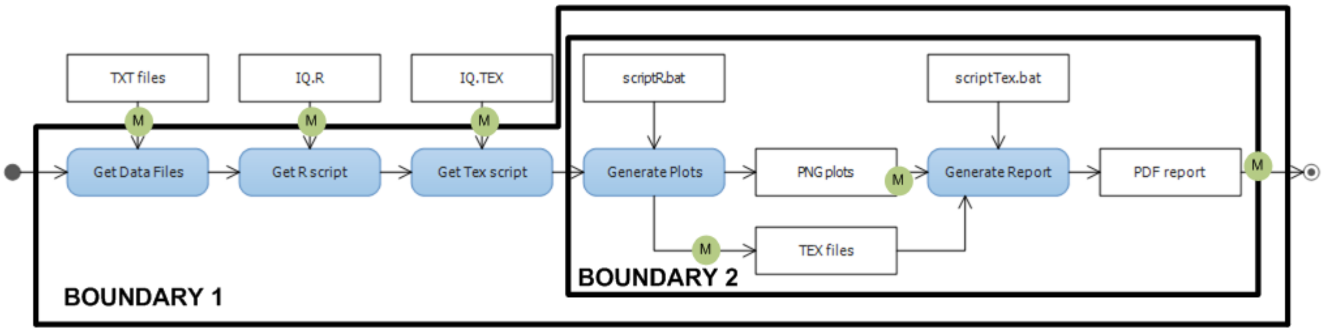

Figure 2: Sequence diagram for eScience process with two process boundaries and measurement points (green circles with 'M') marked.

2.3 Determine internal interactions For both scenarios, the process uses default settings of R and Latex. The scripts are invoking the software from its original locations. However, after careful analysis of software files, it turned out that the default style file of latex *article.cls* was modified. Therefore, this file has to be preserved along with the process and used in the redeployment environment to ensure same effects. Otherwise the final PDF reports will vary in their layout (i.e. number of pages, alignment of content, etc.).

2.4 Ensure deterministic behaviour In case of scenario 1 the execution of the process is deterministic and there are no actions needed. For the purpose of scenario 2, the interaction between the web service and the process has to be captured and data files provided by the web service have to be provided directly in the redeployment environment.

3. Design verification setting In this set of steps, we have identified suitable measurement points enabling us to measure significant properties defined for the redeployment scenarios.

3.1 Specify measurement points Analysis of significant properties resulted in decision to collect files produced after execution of each step. The TEX files will be compared with a use of text comparison tools. The PNG plots will be compared in their rendered form. The PDF report will also be compared in their rendered form as well as by examination of its metadata. In case of scenario 1, the communication to the web service will be measured by text comparison of files received from the web service.

3.2 Specify metrics for preservation quality comparison Using goal modelling techniques, we have decomposed high level significant properties into measurable metrics and coupled them with expected target values. For example: the number of pages in PDF report must be equal, the representation of the 'Residuals vs Leverage' in the PNG plot is the same, the duration of calculations is not longer than in the original system, etc.

3.3 Aim for automated measurements capture The analysed process is not formally specified (e.g. by being specified within a workflow engine). All of the process steps must have been performed manually and therefore most of the the measurements had to be taken manually. Only in case of the first scenario, the requests to the web service were captured by the tool described in [9].

4. Capture verification data In this step we collected verification data from the original environment by running process instances defined in the first step of the VFramework.

4.1 Prepare system for capturing We have decided to deploy the process in the new clean environment. The system was configured according to descriptions of the process and the required dependencies were added. Thus we validated that all necessary information about the process was collected and there are no interactions which we might have missed.

4.2 Prepare data capture tools In this two steps we introduced tools enabling us to extract verification data from the original system.

4.2.1 Set up tools for capturing external interactions In case of the first scenario, the communication to the web service will be captured by the tool described in [9]. We have also used key logging applications installed in the operating system to collect the inputs from the keyboard.

4.2.2 Set up tools for capturing verification data The data will be collected with a use of tools provided by the operating system by default.

4.3 Run the process and capture data We have run the instances and collected data from all measurement points in a separate folder. The folder structure and naming convention ensured that the verification data can easily be associated with the executed process instance.

4.4 Verify validity of captured data Due to manual collection of files, each of them was inspected by us before moving to archive.

5 Prepare system for redeployment We decided to use Ubuntu Linux[1] as a redeployment environment. Ubuntu Linux is an open source project, that is based on the GNU Linux kernel. Ubuntu is easy to use, easy to install, well established and widely used. All of the native operating

[1] www.ubuntu.com

system components are available with open source licenses. Additional packages might be proprietary (such as the Acrobat Reader), but are available free of charge at the moment.

5.1 Prepare system for redeployment We chose Ubuntu Linux 12.10 as the re-deployment environment. The operating system was installed with standard configurations within a virtual machine (VM). Our vitalisation environment was VirtualBox[2] 4.1.26. We installed all required updates and ensured the system was up to date. A desktop environment was needed in order to setup the run time environment (see Step 5.1.1).

5.1.1 Set up Redeployment System For redeploying the process, we had to analyse which packages are needed in order to substitute the Windows tools that implement the steps of the use case. For the local steps we were able to use the packages available from the Ubuntu repositories. This includes the mathematical statistics package R[3] and Latex[4] (via the texlive package).

5.1.2 Set up External Interactions Replay to Ensure Determinism Within our use case there was only one external interaction. The use case needs to retrieve data from a Web service hosted on a machine beyond our influence. In our first redeployment scenario, the Web service was still available and maintained. In the second scenario external dependencies could be removed, as a local data set was used.

5.1.3 Set up Data Capture Tools The tools for capturing the data produced by intermediate steps during the process execution are provided out of the box by the Linux Ubuntu operating system. All intermediate steps produce files that can easily be examined by tools such as diff[5].

5.2 Redeploy Preserved Process The redeployment step involves executing the original process within its new environment. To achieve successful redeployment, several adjustments have to be performed within the new execution environment. These are described in the following steps.

5.2.1 Identify Required Preservation Actions to Enable Redeployment

The original client software was implemented with Microsoft's .NET 4.0 platform and the the C#-Language, running on a Windows 7 operating system. The .NET platform is exclusively available for Microsoft operating systems. Yet there exist compatible implementations and run time environments for Linux as well. Several attempts have been made in order to redeploy the process within a Linux environment.

First, we tried porting the software client from the Windows .NET 4.0 environment to the Mono Project[6], which is an open source software development platform and run time environment. Mono enables the development and execution of .NET software products, which are binary compatible. Although the mono migration tool[7] indicated full compatibility, direct replacement was not possible due to invalid attempts to access reserved areas of the memory. Having the source code of the client available, we were able to identify incompatible code between mono and .NET[8] implementations within the Web service security stack.

The second approach we considered was porting the client software into a Wine[9] (Wine Is Not an Emulator) environment. Wine allows to run many Windows applications on a Windows platform, by substituting required libraries and acting as a compatibility layer. Hence wine allows to run legacy Windows applications, without the need to maintain the full operating system. Using the package manager winetricks[10] the installation of the required runtime libraries could be scripted. Hence we could use the original Microsoft .Net framework 4 component, that can be installed within the wine environment. This enabled the execution of the client software within the Linux redeployment environment and the data could be retrieved from the Web service within a Linux environment.

The next challenge was to orchestrate the packages, that we used for executing the intermediate steps of the process. Adjustments were required regarding naming conventions of applications and paths. Differences occurred in encoding standards and in the scope of included features in the packages. Missing libraries were indicated at the runtime and could be easily installed.

5.2.2 Re-run the Set of Process Instances After the environment has been set up correctly, the use case could be executed. This involved invoking the Web service, which provided the data for further processing. Next, the R script has to be invoked with the retrieved data. The final step produced the PDF document based on the retrieved data.

6 Capture redeployment performance data Data produced during the process execution and captured in the measurement points is collected and verified for its correctness during the course of this step.

6.1 Collect Redeployment Performance Data The selected measurement points overlap with the outputs of three steps of the use case which produce intermediate data. This data is persistently stored in files on the hard disk of the redeployment environment.

6.2 Verify Validity of captured data The following sub steps aim to verify if the data is not affected by lack of deterministic environment and if the measurements are free of unexpected errors.

6.2.1 Verify if Required Level of Determinism was Reached In the first scenario, the Web service is still available. Hence it can be compared to the original data set. In the second scenario, the data is static, hence deterministic.

[2] www.virtualbox.org

[3] R, Version 2.13.1

[4] pdfTeX, Version 3.1415926-1.40.10

[5] http://linux.die.net/man/1/diff

[6] www.mono-project.com

[7] www.mono-project.com/MoMA

[8] http:// www.mono-project.com/WCF_Development

[9] www.winehq.org

[10] http://winetricks.org/winetricks

6.2.2 Verify Correctness of capture Data Once the execution of process instances has finished, the recorded data was verified for its correctness. All of the files could be opened and the screening of their contents was made to verify their correctness.

7. Compare and assess In this set of steps we conducted the comparison of verification data and redeployment performance data. The final assessment about fulfilment of the requirements of the redeployment scenarios was made.

7.1 Compare redeployment performance data and verification data For both scenarios all of the measurement points were available in both environments. We were able to match all of them at the corresponding levels of comparison (see Section 3.1). For example, the TEX files produced by the step Generate Plots were matched for comparison at the file level. Similar matchings were made for other measurement points.

7.2 Conduct preservation quality comparison In this step we calculated the metrics, which were defined in Section 3.2. The examples are: the original PDF report has 169 pages, the new PDF report has 169 pages, values are equal (fulfilled); the process executes in 16,93 s in the original system, the process executes in 12,96 s in the redeployed environment, execution time is not higher (fulfilled), etc.

7.3 Provide summary report Having calculated the metrics, we have created a summary report. It is a document in which all the metrics for each of the scenarios are collected. Clear indication whether the target values are fulfilled is given.

7.4 Make the final decision The report presented that all significant properties of the process were preserved correctly. The requirements of redeployment scenarios were fulfilled. The final decision was made, that the redeployment is meets requirements of redeployment scenarios.

7.5 If Positive, remove tools used for verification There was no need to remove the tools.

5. CONCLUSIONS AND FUTURE WORK

In this paper, the VFramework for verification of preserved and redeployed processes was presented. The applicability of the framework was demonstrated on an eScience use case from the domain of sensor data analysis. The preservation and the redeployment of the eScience process was tested by migration to another substantially different environment. For the purpose of redeployment, the process must have been re engineered and adjusted to work in the new environment. The VFramework was capable of verification of the redeployment in both of considered redeployment scenarios.

Future work will focus on automation of the verification process. The tools needed for extraction and comparison of measurements taken for significant properties in the measurement points will be created. Furthermore, the VFramework will be tested on further use cases and in different redeployment scenarios.

ACKNOWLEDGMENTS

This research was co-funded by COMET K1, FFG - Austrian Research Promotion Agency and by the European Commission under the IST Programme of the 7th FP for RTD - Project ICT 269940/TIMBUS.

6. REFERENCES

[1] IEEE Std 1012 - 2004 IEEE Standard for Software Verification and Validation, 2005.

[2] ISO/IEC 12207:2008: Systems and software engineering - Software life cycle processes, Feb. 2008.

[3] V. R. Basili, G. Caldiera, and H. D. Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994.

[4] C. Becker, H. Kulovits, A. Rauber, and H. Hofman. Plato: a service-oriented decision support system for preservation planning. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL'08)*. ACM, June 2008.

[5] M. Guttenbrunner and A. Rauber. Evaluating an emulation environment: Automation and significant key characteristics. In *Proceedings of the 9th International Conference on Digital Preservation (iPres 2012)*, pages 201–208, Toronto, Canada, October 1-5 2012.

[6] M. Guttenbrunner and A. Rauber. A measurement framework for evaluating emulators for digital preservation. *ACM Transactions on Information Systems (TOIS)*, 30(2), 3 2012.

[7] T. Hey, S. Tansley, and K. Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research, 2009.

[8] R. Mayer, A. Rauber, M. A. Neumann, J. Thomson, and G. Antunes. Preserving scientific processes from design to publication. In P. Zaphiris, G. Buchanan, E. Rasmussen, and F. Loizides, editors, *Proceedings of the 16th International Conference on Theory and Practice of Digital Libraries (TPDL 2012)*, volume 7489 of *Lecture Notes in Computer Science*, pages 113–124, Cyprus, September 23–29 2012. Springer.

[9] T. Miksa, R. Mayer, and A. Rauber. Ensuring sustainability of web services dependent processes. *International Journal of Computational Science and Engineering (IJCSE)*, 2013. Accepted for publication.

[10] C. Thanos, S. Manegold, and M. L. Kersten. Big data - introduction to the special theme. *ERCIM News*, 2012(89), 2012.

[11] M. Van der Graaf and L. Waaijers. A Surfboard for Riding the Wave. Towards a four country action programme on research data. A Knowledge Exchange Report, 2011.

[12] Young, R. R. (2004). *The Requirements Engineering Handbook.*