2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations

Process Migration Framework - Virtualising and documenting business processes

Johannes Binder, Stephan Strodl, Andreas Rauber
SBA Research
Vienna, Austria
Email: {jbinder, sstrodl}@sba-research.org, rauber@ifs.tuwien.ac.at

***Abstract*—The process migration framework (PMF) aims to extract a process from a shared system that hosts multiple processes, and redeploy it in a dedicated virtual machine. Moreover, the PMF supports the documentation of the business process implementation. Main issues when migrating processes are the identification of the process environment (i.e. software packages, files, system settings) and the recreation of this environment. Existing migration tools mainly focus on entire systems or operate on a low level that is difficult to maintain. Main concepts of the PMF are the static and dynamic extraction of process information, monitoring of the process execution, and the usage of a process context model to describe the process environment. The application of the PMF will be demonstrated in the context of a civil engineering business process and an e-Science workflow. Using the PMF it is shown that maintainability, flexibility, shareability, and preservability of business processes is improved by deploying them in dedicated systems, and a comprehensive documentation about the process environment can be provided.**

I. Introduction

Keeping multiple business processes deployed on one shared system poses severe threats to maintenance and flexibility of individual processes as well as the hosting system itself. Shared dependencies between processes limit the possibility to update the system for one process, because of the potential side effects on the other processes. The processes are bound to a system, even to a specific state of the system, and it is difficult to transfer them in a minimal way, so that only resources and dependencies of the respective process are considered. If documentation exists it might be outdated or complex to transform to an environment that is able to execute the process. Besides maintenance issues preservation of the state of processes in a shared system also consumes more resources than necessary. This makes it difficult to archive and share processes.

The aim of this work is to provide a process migration framework (PMF) that frees processes from the chains imposed by the shared system where they are deployed. Processes in this context are an orchestration of activities that are hosted in execution environments and may access both local and remote resources. The first step of the PMF is to identify the process environment, so the resources that a process requires during execution, which is done by static and dynamic observation of the process and its environment. Moreover, custom extractors can be used to capture additional information about the process and its context from the system and its resources (e.g. a model of process steps from process execution engines). The resulting information is stored in the business process context model. It is an ontology-based model allowing to specify aspects of a process that are essential for management and redeployment. The model can be adapted and modified, e.g. to upgrade software. Next, a virtual system is created from the model where the process environment is deployed. Finally it is verified that the target system is able to execute the process correctly. The result of the application of the PMF is a documentation of the process and its environment as well as a virtual system that contains only resources relevant to the process.

The advantages of this approach are that processes can be redeployed independently of any physical machine, processes are able to evolve independently of other processes, and process environments can be shared and archived in a sustainable way, including the local dependencies that are required to execute the process. Remote dependencies are detected and documented but not migrated. The evaluation of the PMF on different scenarios has shown its ability to create models of the process environments in a level that is sufficient to recreate the process environment.

II. System design

The goals of the PMF are to create a virtual system (target system) where a specified process that is embedded in an existing system (source system) is able to execute in, as well as to create a documentation of the process context. In the design it is also considered that the process environment might be adapted, i.e. to have newer software versions of specific packages available on the target system. The process migration consists of four main steps.

- **Capturing the process context**
  The source system is analysed and the process environment is captured and represented as model.
- **Refinement of the model**
  The model created in the previous step is adapted by e.g. replacing or adding software components. This step is optional, but adds the flexibility to handle changing requirements, e.g. in tool versions.
- **Building the target system**
  A virtual machine that corresponds to the refined model and where the process can be executed on is created.
- **Verification of the model and the target system**
  It is verified that the process on the target system shows the same behavior as the process on the source system. Also the model is verified for correctness and completeness.

978-1-4799-5467-4/14 $31.00 © 2014 IEEE
DOI 10.1109/EDOCW.2014.66

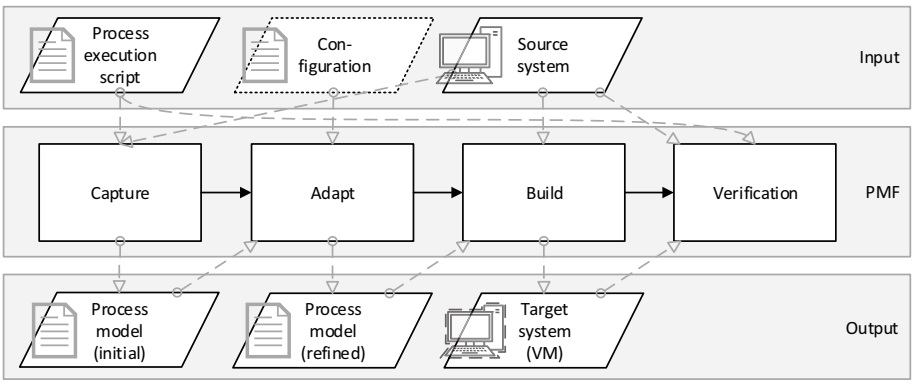

Fig. 1. High-level representation of the PMF process

The steps are executed consecutively and each step except verification returns a result which is used by the next step. The components are described in the following subsections. Figure 1 shows the components and their input and output.

III. Process Migration Framework

The Process Migration Framework implements the four components, `capture`, `adapt`, `build`, and `verification`, of the design as shown in Figure 1. The software architecture is implementing the four components as independently callable modules. The PMF uses and customizes existing software and services to implement the required functionality. The monitoring functionality of the capture component has high dependencies to the operating system. The here presented prototype implementation is focused on `Debian` GNU/Linux, but is extensible to support additional operating systems. The other components and tools are not bound to a specific platform.

A. Capture

For the capturing of the process context different approaches can be used. Depending on the system and the process implementation various tools can support the automatic capturing of the process and its context. The result of the capturing is a model specifying all resources required for the deployment and execution of the process. Moreover this model provides a holistic documentation of the process, specifically its business, service and infrastructure aspects.

For automatic capturing following approaches are used:

1) Dynamic monitoring: This approach monitors the resources usage and access of the process execution. It helps identifying the components that implement the infrastructure of the process, including files, software and services. A challenge for the dynamic monitoring is the identification of all required resources of the process. The execution of all possible process execution paths would be required to have a full coverage. As this is often not possible the resulting model of the dynamic monitoring can be completed by static approaches and expert input.

2) Static approach: The static approaches extract information about the process without execution. Different methods can be used to identify process information and required resources. Examples include the mapping of accessed files that have been identified by the dynamic monitoring to software packages, the extraction of process steps from workflow files, or using process mining tools to derive a process model from logs. This additional information leads to a better maintainable and more accurate model of the process and its context.

For the process virtualisation the PMF deals with infrastructure information of the process. The completeness of automated capturing of the process context depends on the setting and implications of the process. Manual expert input can help to add missing resources and business layer information of the process.

Process context model: The process context model represents the core information entity of the framework. The model is used to exchange information between the components. The model was developed within the TIMBUS research project[1]. The context model and its application is described in detail in [1]. It defines the process context and dependencies including business, services and infrastructure aspects of the process. The context model was designed to provide a holistic view of a business process allowing to specify domain-specific aspects using extensions, and supporting different viewpoints. A metamodel was needed that is able to describe core concepts and allows refinement of its concepts by extension. The context model defines the generic concepts in a domain-independent ontology (DIO). These concepts can be further refined by integrating and mapping of domain-specific ontologies (DSOs). Specifically, we adopted the Archimate language for the DIO, which provides a template to describe a business by around 30 different concepts on the business, application and technology layer. The model is used both internally as data structure that is passed between and used by individual components, but also serves as resulting documentation of the process. For the virtualisation the CUDF DSO (based on the CUDF, the Common Upgradeability Description Format [2]) and the Software DSO were used to describe the software and data context of the process. The context model is available online[2].

The capture process: The dynamic monitoring requires the execution of the process. The monitoring is done by intercepting system calls on the operating system layer. This approach is independent of the language and runtime of tools. The software prototype uses `strace` [3] which allows monitoring

[1]http://www.timbusproject.net, accessed 2014-04-25

[2]http://timbus.teco.edu/ontologies/, accessed 2013-08-22

system calls issued by a process. System calls are used to load resources, access files, spawn processes, inter-process communication and the like. Therefore strace is useful for identifying libraries and data that have been accessed, and to show socket connections, like connections to a database or requests to web services. External dependencies of the process are documented, but the migration of the process can only address local components of the process. For distributed processes therefore each node needs to be considered separately.

In order to get a complete list of required resources of the process, all potential process execution paths need to be executed and observed. This is a useful approach for processes that can be executed multiple times in an automated way to cover different process paths (e.g. workflow or script based processes). For other processes it may not be possible to cover all execution paths, and the resulting context model does not contain all required resources (i.e. only those accessed while capturing the process). In this case the model needs to be extended and completed by static approaches and expert input.

Static approaches investigate the process implementation, different approaches can be used to extract additional information about the process from the system. In [4] the process steps for the model are extracted from a workflow file as well as additional information about used resources such as software and web services. A list of static extractors for the process context is provided at the repository of the TIMBUS Project[3] for example for hardware components, Debian software, network or relational databases. These extractors work on the system level and expert input is required to select resources related to the process. The extractors reduce the workload for experts to model the process and its dependencies.

As the dynamic monitoring identifies only accessed files and services we developed a component to clean and aggregate the captured information. Accessed files that are part of a software are aggregated to software packages representing a higher level abstraction. This improves the documentation and maintainability of the process model, i.e. updates or other maintenance activities. The prototype implementation uses `dpkg` [3] to map files to software packages. The software identifies files of software packages that were modified on the source system, such as config files, which are excluded from the aggregation, such that information loss is avoided. For removing unnecessary information that is not needed for the process to execute and irrelevant for describing the process environment a blacklist is used. The blacklist consists of files like `/etc/hosts`, which contains network settings specific to a system, or `/proc/`, which contains information about running processes [3].

The (semi-)automated ways to capture the process context mainly focus on the infrastructure layer of the business process. The service and business layer is very challenging to extract and depends strongly on the execution environment (e.g. process execution engines). Manual expert input can help to complete the model and add business layer information to the model to document a holistic view of the process. Ontology tools, such as Protégé[4], can be used to edit the context model.

[3]http://opensourceprojects.eu/p/timbus, accessed 2014-04-15

[4]http://protege.stanford.edu, accessed 2014-04-18

B. Adapt

The adapt component supports experts in making changes to the model that has been created by the capture component. The aim of the module is to allow modification in the implementation such as allowing alternative tools to be used within the process. This is necessary to be able to use the PMF not only to migrate the process to a virtual system, but also to bring the process environment up to date or to respond to changed requirements. The modifications are done on the process context model by using editors such as Protégé.

C. Build

The build component is used to build the target system from the model that has been generated and refined in the previous components. Additionally to the model, the source system needs to be available to the build component, to be able to transfer the required files to the target system. The output is a virtual system that corresponds to the model of the process environment, and therefore is able to execute the process. By embedding the process in a virtual system, it is not bound to a dedicated physical host. The main tasks of the build component are to generate a machine interpretable setup instruction from the model, and to use this instruction to instantiate a new virtual machine that is able to execute the process. The build module uses `Puppet`[5] to provision the virtual machine, and consequently the Puppet language as setup instruction. The build module converts relevant aspects of the model to a Puppet manifest. `Vagrant`[6], a tool for the management of virtual machines, is used as wrapper for `VirtualBox`[7]. By utilizing the `Vagrant-Binding API`[8] for Java, a new virtual machine is prepared, and the manifest applied to this machine. Data files are transferred directly from source to target system, so also the build module is executed on the source system. The base system from which the target system is derived can be created manually, or chosen to be downloaded from existing Vagrant base systems. There are several base boxes public available e.g. from Vagrantbox.es[9]. The result of the component is a virtual image deploying the infrastructure components defined in the process context model.

D. Verification

Verification is necessary to confirm that the process on the source system behaves the same as the process on the target system. It is currently a manual task and requires access to the source system, the target system, and the model. Using those it decides whether the model and the target system correspond to the source system in respect to the process environment. The aim is to be able to determine if the target system provides an environment for the process that is suitable to generate valid results. An important and efficient way to verify the migration is to compare the results of the process when executed on the source system against the result of the process executed on the target system. For non-deterministic processes or for more detailed verification the VFramework can be applied [5].

[5]http://puppetlabs.com/, accessed 2014-04-15

[6]http://www.vagrantup.com/, accessed 2014-04-15

[7]http://www.virtualbox.org/, accessed 2014-04-25

[8]http://github.com/guigarage/vagrant-binding/, accessed 2013-10-08

[9]http://www.vagrantbox.es/, accessed 2013-10-28

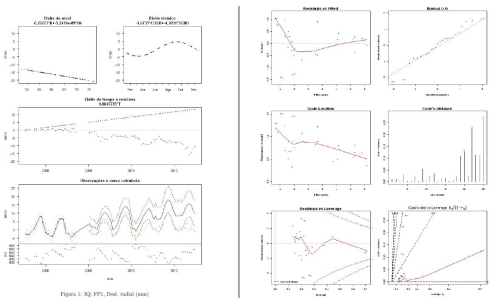

Fig. 2. Resulting report of the civil engineering process

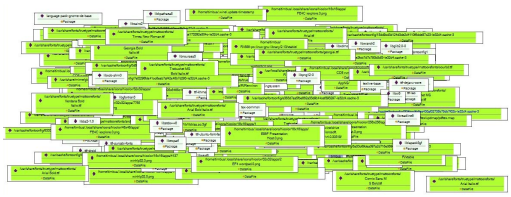

Fig. 3. Visualisation of the infrastructure layer

IV. Evaluation and demonstration

The here presented framework was evaluated on different scenarios. Two were chosen for the demonstration. The first business process is from the civil engineering domain. Sensor data installed in infrastructures for monitoring are captured from a central database and prepared as a report. The report is used by experts to evaluate and monitor the structural state of buildings. The sensor data is analysed and visualised using different physical and mathematical models implemented as `R` scripts[10]. The output is compiled to a single report in PDF format by using TEX[11] as shown in Figure 2. The second process is an e-Science workflow which has been selected to demonstrate the applicability of the framework for processes executed by workflow engines. Both scenarios use local tools as well as web services. The demonstration shows the capturing of the process by using dynamic and static approaches. The prototype demonstrates that processes can be observed at a very detailed level with standard GNU/Linux tools. The resulting model represents a good documentation of the process environment and can be further extended by detailed process information that are not automatically captured, e.g. licenses or involved stakeholders. By querying and visualising the ontology specific information about the process can be extracted from the model. Figure 3 shows the visualisation of involved software and data of the civil engineering process. The model can be modified and updated to address new requirements, e.g. software updates or the use of alternative services. The demonstration shows the creation of a virtual instance based on the model. The resulting instance is tested and verified comparing its behavior with the ordinal process. The demonstration will further show limitations of the capturing process, in particular of the business level, and discuss potential extension for different scenarios (e.g. process execution engines).

[10] http://www.r-project.org, accessed 2014-04-15

[11] http://www.tug.org, accessed 2014-04-15

V. Conclusion

The PMF not only can be applied to extract and relocate a process from a shared existing system, but also to share a specific system environment for teams that need to execute the same process, or to copy the environment e.g. to establish a test setting that allows the execution of the process. The PMF can be used for archiving processes by creating a self contained virtual system and the according documentation. Because the target system only contains resources that are accessed by the process it is more suitable for preservation than systems that result from a clone approach.

The main advantages of the PMF are:

- The migration scope is on individual processes, not the source system as a whole (compared to P2V converter tools like the `vCenter Converter`[12]).
- A documentation of the process is generated, on which the migration is based upon (compared to application packaging tools like `CDE` [6]).
- The framework operates on high level concept (i.e. packages vs. bitlevel), which improves the maintainability of both model and target system (compared to file based migration, e.g. using `rsync`[13]).
- The framework is independent of the architecture and technology that the process uses (compared to frameworks like CloudMIG [7]).

Acknowledgment

This work has been co-funded by COMET K1, FFG and by the TIMBUS project, co-funded by the EU under the FP7/2007-2013 under grant agreement no. 269940.

References

[1] G. Antunes, M. Bakhshandeh, R. Mayer, J. Borbinha, and A. Caetano, "Using ontologies for enterprise architecture analysis," in *Proc. of the 8th Trends in Enterprise Architecture Research Workshop (TEAR 2013)*, Vancouver, Canada, September 9-13 2013.

[2] R. Treinen and S. Zacchiroli, "Description of the CUDF Format," Tech. Rep., 2008, http://arxiv.org/abs/0811.3621.

[3] A. Robbins, *Unix in a Nutshell*. O'Reilly Media, 2008.

[4] S. Strodl, R. Mayer, G. Antunes, D. Draws, and A. Rauber, "Digital preservation of a process and its application to e-science experiments," in *Proc. of the 10th Int. Conf. on Preservation of Digital Objects (IPRES2013)*, Lisbon, Portugal, September 2013, pp. 117–125.

[5] T. Miksa, S. Proell, R. Mayer, S. Strodl, R. Vieira, J. Barateiro, and A. Rauber, "Framework for verification of preserved and redeployed processes," in *Proc. of the 10th Int. Conf. on Preservation of Digital Objects (IPRES2013)*, Lisbon, Portugal, September 2–6 2013.

[6] P. J. Guo and D. Engler, "CDE: Using system call interposition to automatically create portable software packages," in *Proc. of the 2011 USENIX conference on USENIX annual technical conference (USENIXATC'11)*. Berkeley, CA, USA: USENIX Association, 2011, pp. 21–21.

[7] S. Frey and W. Hasselbring, "Model-based migration of legacy software systems to scalable and resource-efficient cloud-based applications: The CloudMIG approach," in *Proc. of the First Int. Conf. on Cloud Computing, GRIDs, and Virtualization (Cloud Computing 2010)*, Lisbon, Portugal, November 2010, pp. 155–158.

[12] http://www.vmware.com/products/converter/, accessed 2014-04-15

[13] http://rsync.samba.org/, accessed 2014-04-15