Title : Enabling Global Software Development via Cloud-Based Software Process Enactment

Authors: Sami Alajrami, Barbara Gallina, Alexander Romanovsky

TECHNICAL REPORT SERIES

No. CS-TR-1494 March 2016

TECHNICAL REPORT SERIES

No. CS-TR-1494 March 2016

Enabling Global Software Development via Cloud-Based Software Process Enactment

Sami Alajrami, Barbara Gallina, Alexander Romanovsky

Abstract—Global software development (GSD) is a software development model where the development effort spans across distributed locations. Although GSD has gained vast popularity due to its economical benefits, it faces various challenges as a result of cultural, temporal and spatial distances. Cloud computing is becoming the norm for consuming computing resources due to its economies of scale. While the potential for using the cloud for GSD has been investigated in the literature, in this paper, we go one step forward and propose a cloudbased software process enactment architecture. This architecture facilitates bridging the spatial and temporal distances and aims at addressing communicational, managerial and technical GSD challenges. We use EXE-SPEM -an extension of SPEM2.0 which supports cloud-based executability of software process models to model software processes. These models are then enacted in the cloud where the type and amount of resources to be used can be configured. We demonstrate our approach using a simple verification process example that we enact in a proof-of-concept implementation of the architecture.

Bibliographical details

Title : Enabling Global Software Development via Cloud-Based Software Process Enactment

Authors: Sami Alajrami, Barbara Gallina, Alexander Romanovsky

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1494

Abstract

Global software development (GSD) is a software development model where the development effort spans across distributed locations. Although GSD has gained vast popularity due to its economical benefits, it faces various challenges as a result of cultural, temporal and spatial distances. Cloud computing is becoming the norm for consuming computing resources due to its economies of scale. While the potential for using the cloud for GSD has been investigated in the literature, in this paper, we go one step forward and propose a cloudbased software process enactment architecture. This architecture facilitates bridging the spatial and temporal distances and aims at addressing communicational, managerial and technical GSD challenges. We use EXE-SPEM -an extension of SPEM2.0 which supports cloud-based executability of software process models to model software processes. These models are then enacted in the cloud where the type and amount of resources to be used can be configured. We demonstrate our approach using a simple verification process example that we enact in a proof-of-concept implementation of the architecture.

keywords

Software Processes, SPEM2.0, Process Enactment, Cloud-Based Software Development, Cloud Computing

Enabling Global Software Development via Cloud-Based Software Process Enactment

Sami Alajrami
Newcastle University
Newcastle upon Tyne, UK
s.h.alajrami@ncl.ac.uk

Barbara Gallina
Mälardalen University
Västerås, Sweden
barbara.gallina@mdh.se

Alexander Romanovsky
Newcastle University
Newcastle upon Tyne, UK
alexander.romanovsky@newcastle.ac.uk

Abstract—**Global software development (GSD) is a software development model where the development effort spans across distributed locations. Although GSD has gained vast popularity due to its economical benefits, it faces various challenges as a result of cultural, temporal and spatial distances. Cloud computing is becoming the norm for consuming computing resources due to its economies of scale. While the potential for using the cloud for GSD has been investigated in the literature, in this paper, we go one step forward and propose a cloud-based software process enactment architecture. This architecture facilitates bridging the spatial and temporal distances and aims at addressing communicational, managerial and technical GSD challenges. We use EXE-SPEM -an extension of SPEM2.0 which supports cloud-based executability of software process models- to model software processes. These models are then enacted in the cloud where the type and amount of resources to be used can be configured. We demonstrate our approach using a simple verification process example that we enact in a proof-of-concept implementation of the architecture.**

I. INTRODUCTION

Engineering software systems is a complex set of tasks performed by multiple collaborating stakeholders. Over time, several factors have changed how software engineering is undertaken. Globalization (as an economical factor) has transformed the way software firms work; moving from monolithic development (one team at one location) to multiple geographically-distributed teams collaborating on a development project. This business model is attractive as it makes it possible to: a) utilize cheaper labour in different countries hence implying cost reduction. b) have multiple teams working in different time zones which leads to a shorter development life-cycle. c) be in closer proximity to customers and emerging markets.

Despite the benefits, teams collaborating in GSD projects face cultural, temporal and spatial distances which make managing such projects a challenging task. Synchronization between teams and ensuring no deadlocks happen, allocating the right resources/tasks to each team and global awareness of the project progress are examples of the issues that should be considered in a GSD project.

Some of these issues have been addressed by offering recommendations on how to undertake GSD projects [1], [2], [3]. Authors in [4], [5], [6], [7] argue that the use of cloud computing can facilitate GSD and overcome some of its challenges. Cloud computing is becoming the norm of accessing computing resources thanks to its elasticity and economies of scale. Its use for software development has not been widely discussed in the literature though. Industrial cloud-based solutions for software development have been made available in the last few years. Examples include: Codenvy [1], IBM Jazz [2], Cloud9 [3]. However, these solutions support particular phases of the software process which forces development teams to use different tools and platforms from different vendors. This will inevitably result in interoperability and synchronization problems.

In this paper, we propose a cloud-based software process enactment architecture which utilizes the cloud elasticity, accessibility and availability to facilitate GSD and to overcome some of the technical and communicational challenges associated with it. We use EXE-SPEM [8] (a software process modelling language which supports modelling cloud-based-executable processes) to model software processes. The process models can be mapped into an executable XML format which is then enacted in a cloud-based software process enactment service. The architecture comes with enabling features for GSD which support monitoring of processes, activities and computing resources allocation, global change awareness and enforcing quality policies.

Software process models are workflows of software development activities. A process model may include activities from one or more phases of software development and can be created and customized to fit for the development team's needs. In this paper, we do not focus on any particular software development model. Instead, we provide a way to model and enact any process model.

The paper is structured as follows: Section II provides a brief background about GSD, software processes and EXE-SPEM. Section III describes the cloud-based software process enactment architecture and a prototype implementation. In Section IV, we demonstrate an example process of software verification task and how it is enacted in the prototype. Section V reviews the related work. In Section VI, we discuss our approach to evaluate this work. Then the paper concludes in Section VII.

[1]https://codenvy.com/
[2]https://jazz.net/
[3]https://c9.io/

II. BACKGROUND

This section provides background information on GSD's benefits and challenges and how modelling software processes can help addressing some of these challenges. In addition, we recall EXE-SPEM (the modelling language used in this paper) and the cloud as an execution environment.

A. *Global Software Development*

Facilitated by the internet and inspired from other business domains, Global Software Development (GSD) has become a widely used practice in the software industry. GSD -which is also known as: Global Software Engineering (GSE) and Distributed Software Development (DSD)- is defined as "software work undertaken at geographically separated locations across national boundaries in a coordinated fashion involving real time (synchronous) and asynchronous interaction " [9]. It can be conducted in two forms: intra-organization (different locations for the same organization) and inter-organization (two or more organizations outsourcing parts of their software development).

GSD is an attractive business model due to the benefits that it brings to different aspects of software development. These benefits are now established in the literature [10], [11], [12]. Ågerfalk et al. [13] have categorized the benefits based on their positive impact on organization, team and process as shown in Table I.

Many software firms have adopted a GSD model where they outsourced parts or all of their software production. A study at Microsoft [14] has shown that more than 50% of the surveyed employees were involved in some form of GSD. However, GSD has to face challenges related to geographical, temporal and spatial distances. Such challenges might cancel out the benefits. Most of the literature discussing the GSD benefits provide a thorough discussion on the risks and challenges associated with it. The challenges can be summarized as follows:

- **C1**: Inadequate communication: communication is an essential part of software development. The geographical and temporal distances may cause delays and deadlocks that can threaten the success of a software project [11]. This challenge mostly threatens the team and process related benefits.
- **C2**: Cultural issues: different cultural backgrounds amongst teams may cause difficulties and misunderstandings. For example, people's respect of time (deadlines) depends on the value of time in their culture [11]. This challenge mostly threatens the organization-related benefits.
- **C3**: Management issues: a GSD project is like a puzzle consisting of several pieces. The role of the management is to ensure these pieces fit with each other and ensure that the coordination and awareness between the teams is well-maintained [11]. This challenge mostly threatens the team and organization related benefits.
- **C4**: Recognizing mythical benefits: the GSD benefits listed in literature should not be taken for granted as they may be associated with risks [12]. This challenge threatens all GSD benefits.
- **C5**: Technical challenges: GSD needs technical support to enable it. The current development approaches face technical issues like: global awareness of change, incompatible data formats and using different versions of the same tool [11]. This challenge threatens the team and process related benefits.

Addressing these challenges will make it possible to benefit fully from GSD. In this paper, we focus on addressing challenges C1, C3 and C5.

B. *Software Processes*

Paulk et al. [15] describe a software process as "a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, and user manuals). As an organization matures, the software process becomes better defined and more consistently implemented throughout the organization". Over the years, several process models have emerged, such as: the waterfall model [16], the spiral model [17], extreme programming (XP) to name a few.Each of the models has its strengths and weaknesses and may or may not be suitable for particular projects. A GSD project can adopt any process model which better suits the project's needs. The adopted process model can then be customized to fit for the project.

Whichever process model is chosen, modelling the process can enhance understanding and communication between team members. Furthermore, from a management perspective, process models provide a mean of quality assurance by enforcing particular policies and quality measures in the process that will be followed by distributed teams. Process models also facilitate monitoring and synchronization of tasks (which relates to challenges C1 and C3 in Section II-A). The high emphasis on models leads to high emphasis on the software process design and on making the process well-documented [13].

However, if the process model is not executable, it loses a lot of its value for many stakeholders. A non-executable process model would lose the global awareness view and would not guarantee that the distributed teams are following the process in the same way. Furthermore, the lack of standardized execution environment/tools may lead to synchronization and integration problems across distributed teams. This raises the need for executable models and a supporting execution environment that can be used across development locations.

C. *EXE-SPEM*

Software & Systems Process Engineering Meta-model (SPEM2.0) [18] is an Object Management Group (OMG) standard for modelling software processes. SPEM2.0 lacks explicit support for process enactment, control flow semantics and cloud-based process modelling. Therefore, we introduced EXE-SPEM [8] which is an extension of a subset of the SPEM2.0 meta-model. We extended the *Process Structure* package of the SPEM2.0 meta-model. The extension added

TABLE I
BENEFITS OF GSD ADOPTED FROM [13].

Organizational-related Benefits	Team-related Benefits	Process-related Benefits
• cost savings • access to large skilled workforce • reduced time to market • proximity to market and customers • innovation and shared best practice • resource allocation	• improved task modularization • reduced coordination cost • increased team autonomy	• formal record of communication • improved documentation • clearly defined processes

subtypes of the activity stereotype to introduce modelling elements for interactive and control point activities. The cloud-based model execution information was embedded in the attributes of the activity stereotype. Software process models modelled in EXE-SPEM can be mapped to an executable XML notation. Further details on the mapping rules can be found in [8].

D. Cloud Computing

Cloud computing offers computing resources as a utility to consumers on a pay-as-you-go, scale-as-you-need model. Cloud services are offered in four delivery models (public, private, hybrid and community) and three service models (Software as a Service [SaaS], Platform as a Service [PaaS] and Infrastructure as a Service [IaaS]) [19]. Cloud can reduce effort (time and cost) spent on acquiring and maintaining computing infrastructure by delegating this task to the cloud service provider. This means that software firms can focus their resources on the core business problem they are trying to solve rather than being distracted by setting up and configuring the development environment. Cloud services are usually: accessible, available, multi-tenant and elastic. *Accessibility* refers to accessing the services from different types of devices and from anywhere that is connected to the internet. *Availability* refers to the reliability of cloud services which guarantees that the service will be accessible any time and anywhere for a minimum percentage of the time. In fact, many cloud providers promise a 99.9% availability. *Multi-tenancy* refers to the fact that multiple tenants will be sharing the same computing resources which allows the cloud providers to utilize their resources better. Finally, *Elasticity* refers to the fact that consumers can instantly scale the resources they are using up or down as they need. Section III-A illustrates how these features can help addressing some of the GSD challenges.

III. GSD IN THE CLOUD

In this section we introduce our a cloud-based software process enactment architecture which addresses challenges C1, C2 and C5 mentioned in Section II-A. We highlight the motivation for this work (Section III-A), then we establish a set of requirements to be met (Section III-B). Based on the requirements, we introduce a cloud-based architecture for software process enactment (Section III-D) and explain how it meets the requirements (Section III-E). Finally, we describe a prototype implementation of our architecture (Section III-F).

A. Motivation

As explained in Section II-B, modelling software processes can be utilized to support managing GSD projects and relates to addressing some of its related challenges such as: global project awareness, enhancing communication and understanding amongst distributed teams and supporting global monitoring and synchronization of tasks. In addition, executable process models (when supported with the appropriate execution environment) can help addressing technical GSD challenges such as: incompatible data formats and tools.

General cloud benefits such as: costs and time savings, minimizing upfront investment and reducing carbon footprint can be utilized for GSD. In addition, accessibility entitles different teams to access the same development environment from their remote locations (in a Software as a Service - SaaS- manner). Cloud providers take different measures to ensure availability of their services (e.g. availability zones, data replications, etc). This minimizes the risks of being not able to access the development environment or losing data. Furthermore, since software development involves multiple stakeholders, a multi-tenant development environment is essential and facilitates group-oriented tools which has been identified as a need for software engineering by Boehm [20]. SaaS solutions offer such multi-tenancy. Cloud also provides elasticity on both resources and tenants levels, which means more computationally intensive tasks can be satisfied and growing teams' needs can be met. Finally, using the cloud opens the door for transforming tools into services as proposed in [21].

Using the cloud as an execution environment for executable software process models can aggregate the benefits of both cloud and modelling in order to address some of the GSD challenges, in particular, challenges C1, C3, C5 in Section II-A.

B. Requirements for Cloud-based GSD Process Enactment

In order to facilitate GSD in the cloud, we define a set of requirements that needs to be met. The requirements come either from GSD needs or from cloud considerations.

1) GSD needs:

- **R1: Awareness and synchronization support** In order to avoid misunderstandings and misalignment between teams, it is required to keep the distributed teams and their management aligned and aware of the overall progress. A unified and accessible development platform where everyone can be aware of the process being followed and tools being used and the overall progress of the project would be helpful to both management and individual team members to enhance awareness and synchronization.

- **R2: Availability of tools in real time** Acquiring software development on the fly saves time and cost for setting up, configuring and maintaining your won environment. Similarly, to keep the focus on the business problem, tools should be available as services that can be accessed on demand. This also makes it easier to guarantee that all teams are aligned in the tools (and versions) they are using. In addition, expensive commercial tools can be made available on a pay-as-you-go model.
- **R3: Organizational policy enforcement** In certain cases, management may need to enforce a particular process to be followed (e.g. for certification or quality purposes). It should be possible to define a process model which distributed teams can adhere to.
- **R4: Capturing process & provenance data** Today, there is no need to emphasize on the importance of data. Both research and industry are pushing the limits to collect more data, reason about it and process it faster. Software development data is no exception. Capturing data about the software process execution such as: artefacts, versions, time, people involved etc. can be useful for different purposes, particularly, to support accountability and traceability. For example, in safety critical systems, safety cases are built to describe the process being followed in order to prove that it meets the certification standards. Another example is to use this data to help improving the process.
- **R5: Accessible artefacts** Artefacts is an important part of software processes. They should be stored and maintained in an accessible way by all authorized stakeholders regardless of their locations.

2) Cloud considerations:

- **R6: Privacy and legal compliance** Using the cloud raises the concern of privacy. As the software development will take place on the cloud provider's infrastructure, companies may not be willing to put confidential artefacts on public clouds. Similarly, regulations may impose restrictions as to where processes can take place. For example, the EU regulations require that EU data should be processed and stored within the EU. Therefore, there is a need for process execution to be configurable and to be deployable on hybrid cloud (private and public).
- **R7: Governance and Inter-organization collaboration** Facilitating outsourcing of parts of the software development processes to sub-contractors while ensuring privacy and confidentiality of data and processes is essential in modern software industry. Companies should be able to host the software development process of their sub-contractors on their private cloud infrastructure while giving them access to the artefacts they need. This eliminates the risks associated with sending confidential artefacts outside of the company's network.

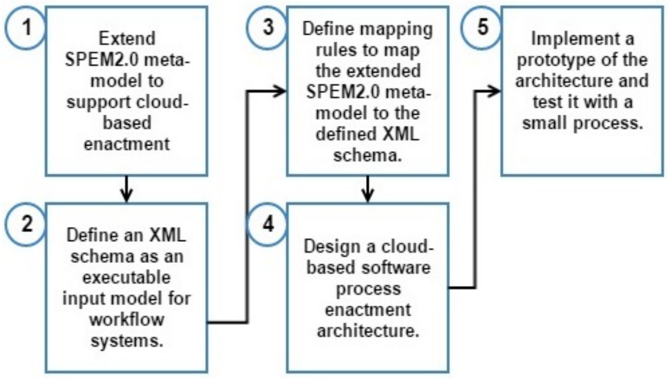
1
Extend SPEM2.0 meta-model to support cloud-based enactment
2
Define an XML schema as an executable input model for workflow systems.
3
Define mapping rules to map the extended SPEM2.0 meta-model to the defined XML schema.
4
Design a cloud-based software process enactment architecture.
5
Implement a prototype of the architecture and test it with a small process.

Fig. 1. Our approach to supporting cloud-based executable software processes.

C. Approach

Our approach is a model-driven approach in the sense that software processes are modelled in order to be enacted. It is based on our previous works [22], [8]. However, instead of creating our own modelling language as in [22], we developed EXE-SPEM [8] for modelling the software processes. EXE-SPEM enables modelling cloud-based executable software processes and can be mapped into an executable XML notation. We propose a cloud-based architecture to enact those XML models. This approach is illustrated in Figure 1. Steps 1-3 relate to the work in [8] while steps 4,5 relate to this paper.

D. Cloud-based Architecture for Software Process Enactment

In this section, we describe a cloud-based architecture for software process models enactment. The architecture is designed to meet the requirements established in Section III-B and is also based on the following set of assumptions:

- A software process is a workflow which consists of multiple activities.
- Activities are the smallest unit of execution. They are supported with tools and assigned to an actor.
- Tools (Activities) are provided by a community of tool producers or can be developed by the users.

Figure 2 illustrates the architecture and its components. Actors (stakeholders involved in a software development process) can interact with the platform using a *multi-tenant SaaS platform*. The SaaS layer interacts with the *Enactment Service (PaaS layer)* through a *REST API*. The enactment service consists of the following components:

- Workflow Engine Registry: keeps track of all workflow engines that are in service and their status.
- Scheduler: handles the planning of a process execution. This involves checking the needed resources (based on the cloud configurations expressed in the process model). Since the execution granularity is set to the level of

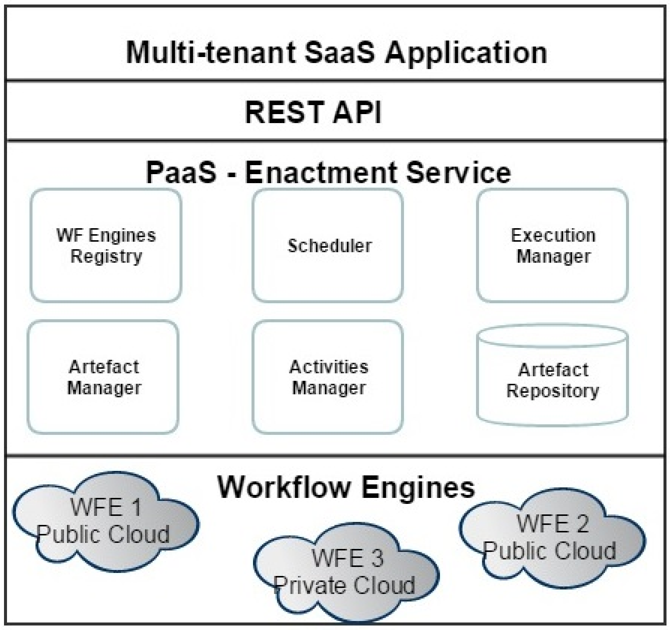

Fig. 2. Cloud-based software process enactment architecture.

activities, the scheduler allocates each activity to the least-busy workflow engine at the time of allocation.
- Execution Manager: once the scheduler has allocated activities to workflow engines where they will be executed, the distributed execution needs to be managed. The execution manager keeps track of the progress of the execution, and logs execution outcomes.
- Artefact Manager: software processes involve producing large number of artefacts such as: code, models, test cases, requirement documents, documentation, etc. These artefacts capture invaluable information about both the software process and product evolution. The artefact manager stores the artefacts themselves and meta-data about them into the artefacts repository. The meta-data includes: actors involved, version, tools used and the timestamp at which the artefact was created/modified on. Although mining the artefact repository is out of the scope of this paper, several approaches to mining software artefacts exist in literature. Some of which have been surveyed by Kagdi et al. [23].
- Activities Manager: similar to the artefact manager, it manages the meta-data about activities and the activities' executables which are all stored in the repository.

Workflow Engines are independent applications running on different cloud providers. Activities get executed in a workflow engine that is deployed on a machine (on private or public cloud) that meets the execution requirements expressed in the process model. The execution of activities is a black-box execution. This means that the workflow engine would not know any information about the process being executed which reduces the risks of privacy and confidentiality violations. In order to decouple the enactment service from the workflow engines, asynchronous communication between them is achieved through message oriented middleware. The enactment service pushes jobs to workflow engines by placing them into their designated jobs queue. The workflow engines place progress updates into the enactment service responses queue.

E. Meeting the Requirements

We explain below how each of the requirements in Section III-B have been met.

- **R1**: The enactment service is the unified development platform which is accessible by distributed teams from a web browser. This provides a global view of the project and raises global awareness.
- **R2**: Tools are represented by activities in our architecture. They are stored in the repository and made available on demand. Tools need a standardized way of taking input and generating output artefacts. In Section IV we show an example of wrapping an existing tool to operate in our architecture.
- **R3**: Organizations can enforce certain policies/standards by: providing a process model that actors are going to follow and by specifying the enactment details such as tools to use and their versions, input/output formats
- **R4**: The architecture captures all information related to process executions and artefacts evolution.
- **R5**: The enactment service REST API allows for accessing artefacts and their meta-data using a web browser regardless of location.
- **R6**: The cloud specific configurations in the process model allows for defining cloud-related constraints such as the type of cloud to be used. The enactment service considers those configurations when allocating an activity to be executed.
- **R7**: Inter-organizational collaboration can be supported by making calls between the enactment services of two or more companies. Company A may outsource part of its process to company B in which both of the companies would pass the required artefacts through the their enactment services APIs without reviling any other details of their internal processes.

F. Prototype Implementation

We implemented a prototype for the architecture described above. The prototype supports execution of only activities written in Java and have Java executable files (JAR). It also does not support interactive activities at the moment. The prototype consists of two applications (Enactment service and Workflow engine) both developed as a web service and accessible through an API. The enactment service can connect to any number of workflow engines and the workflow engines can be deployed on any machine type. The prototype has been deployed in Amazon Web Services EC2 machines where we run the example process described in the next section.

IV. EXAMPLE PROCESS

We apply our approach to a simple process as a demonstrating example. The process consists of a single activity performing distributed model checking in the cloud. Model checking (and software verification tasks in general) are computing intensive tasks. It involves traversing a massive tree of potential program paths and verifying that certain properties do/do not hold for the program. This requires a lot of memory and processing power. It is often possible to have a model checking task running for hours before it crashes due to reaching the computing limit of the machine running it. In a distributed model checking task, the load is distributed over several networked machines. This aims to harness the multiplicity of resources and making it possible to model-check large program models.

In this demonstrating example, we have wrapped the distributed model checker DiVinE [24] as an elastic activity and made it available for reuse in our prototype. Using the cloud for model checking is a double-edged sword. While you benefit from elastic resources on demand, you might end up spending a lot of money on scaling resources without actually achieving your goal. The model checking activity comes with several configurations parameters which specify how a model checking task will be performed in the cloud, when and how to scale resources and what type of cloud to use (public or private). Since it is almost impossible to tell how much resources a program model would require to be model checked, we have parameters to set the initial amount of resources to start the model checking task. Furthermore, the activity has a time-out parameter which stops the model checking task from running infinitely. Once the time-out is reached without finishing the model checking task, the activity automatically scale the amount of resources being used either in a linear or exponential way. After a pre-defined number of attempts, if the model checking task has not successfully terminated, the task will terminate with a failure. The values of these parameters are based on experience and estimation.

We used the model checking activity to construct an EXE-SPEM process model as shown in Figure 3. In the process, the model checking activity is performed by a verification engineer and consumes input model and produces the model checking results. The process also contains a control activity (an EXE-SPEM type of activity) which allows the engineer to edit the model (in the *Edit Model* task) and rerun the model. It is worth noticing that in EXE-SPEM, tasks refer to procedures that are not supported by tools (e.g. we did not have a tool to edit the model) while activities are procedures that have tool support. We then mapped this process model into an XML model -using the rules defined in [8]- which we run in our prototype.

We deployed our enactment service and a single workflow engine on two Amazon *t2.small* instances. The model checking activity was configured to start execution with two *m3.xlarge* Amazon instances which were created using a machine image that contained DiVinE and all of its dependencies. The enactment service has a REST API which allows to add a process model and then enact it. After running this example process, a new artefact (a text file) was created. The file contained the outcome of the DiVinE model checker.

V. RELATED WORK

With the increasing popularity of cloud, several domains have adopted the new computing delivery model. The potential of using clouds to support particular software development phases such as testing has been investigated in the literature [25], [26], [27]. Cloud-based commercial tools have been introduced to the market in the last few years supporting particular software tasks such as: code editing, compiling, testing and continuous integration. Examples include: Github [4], Jenkins [5], Jira [6]. However, to the best of our knowledge, supporting the entire software development process in the cloud has not been investigated before. Our approach supports modelling and enacting the entire development process rather than partial phases of it.

The potential of using cloud for GSD has been investigated by several authors. Al-qadhi et al. [5] identified seven areas that benefits from using the cloud for GSD. These areas are: increasing productivity, testing, GSD process, reducing IT operations costs, eliminating global distances, better content (artefacts) management and enforcing of standards. Arimura et al. [6] described how Fujitsu has converted their development centre which served 9 hubs in Japan to a private cloud and how it helped them to: reduce the costs by $9 million, increase the utilization of their infrastructure, expand their range of services and reduce their environmental load. Hashmi et al. [4] identified several benefits that the cloud could bring to GSD, particularly, to facilitate some GSD challenges such as: collaboration, geographical distance, project knowledge transfer and execution monitoring.

Mwansa et al. [28] argue that migrating agile software development processes to the cloud could result in: faster production, improved quality and more flexible change-embracing processes. They also propose a framework to help companies migrate their agile processes to the cloud. Paul [7] explains how cloud reinforces agile methodologies by providing enhanced business agility, faster time-to-market, increased productivity, high quality code and more efficient cost contract. Yara et al. [29] have argued that cloud would be beneficial in three areas in software development: code development (implementation), QA and testing and IT operation. Oza et al. [30] have empirically studied the risks and benefits of using cloud for distributed software development where they studied a GSD project involving three geographically-distributed teams.

The risks of using the cloud have also been considered in the literature. Some of these risks are inherent in GSD (i.e. they exist with or without cloud) such as: dependencies , unavailability and technical debt [5], [30]. On the other hand, vendor-lock-in, SLA control, privacy, reliability, data migration, auditing and regulation compliance are more cloud-related risks [29].

[4]https://github.com/
[5]https://jenkins-ci.org/
[6]https://www.atlassian.com/software/jira

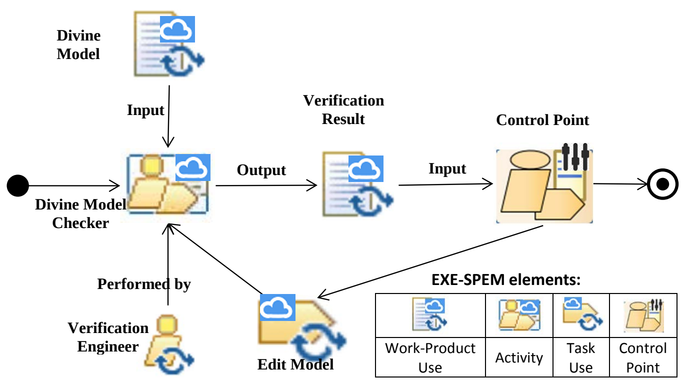

Fig. 3. EXE-SPEM model of the model checking process.

Although several authors have analysed the potentials and risks of using cloud to enable GSD, to the best of our knowledge, neither academic publications nor commercial tools have addressed the support of the complete software development cycle in the cloud. Our approach also mitigates some of the cloud-associated risks mentioned above. Since our architecture is open and can be deployed into any IaaS provider, vendor-lock-in is not considered a problem any more. Furthermore, having fine execution granularity which allows defining privacy requirements for each activity execution, mitigates the privacy risk. Moreover, with provenance data and the artefacts repository, auditing and accountability can be supported. Finally, as companies can use our architecture on their own private infrastructure and host their sub-contractors processes, the data migration problem is avoided.

VI. EXPERIMENTS & EVALUATION

The benefits of using cloud in general is undoubtable. Our vision was to bring those benefits to software development in general and GSD in particular. We started our investigation by using e-Science central(eSC)[31] (a cloud-based science workflow system). eSC allows scientists to collaborate on designing/running experiments and sharing their results. eSC works with data-flow processes in a fire-and-forget style (where you run the workflow and expect to get the results at the end of the execution). However, software development processes are more complex control-flow processes. Therefore, eSC and similar workflow systems do not fit for software processes.

During our investigation of the use of eSC for software processes, we have successfully developed and experimented several activities (integrated within eSC). The first one reused the *Spin* model-checker. Another one reused the distributed model checker *DiVinE* [24], we added here an extra controls to allow automatic scaling and elastic use of computing resources. The aim was to test how software processes can utilize the scalability of the cloud to obtain more computing resources as it needs at runtime.

In order to fulfil our vision, tools have to be offered as services. Today, more and more tools are moving to the web (as a service). We have already moved some verification tools to the cloud, including the *Why3* provers[7] and the *ProB* model-checker[8]. In particular, in our work with *Why3* we support the elastic and scalable execution of several provers to improve the quality of discharging the verification conditions [32].

Evaluating our approach and architecture is indeed challenging as the resources (time and workforce) are not available to move the tools -required to run a complex, real world process using our approach- into services. Therefore, in our work we rely on a phased evaluation in which we use the lessons learnt from our experiences with eSC, Divine and Why3 in addition to studying literature(e.g. [4], [5]) and learning industrial experience as in [6] to validate the potential of our approach. Furthermore, we are currently applying our approach to the safety-critical domain where we workflow and enact an automated safety case fragment generation process. This process generates both product-based and process-based argument fragments that can be used for certification purposes. However, there is a need to implement larger-scale case studies and conduct more empirical experiments.

VII. CONCLUSIONS

In this paper, we proposed our vision to enable GSD processes via cloud-based process enactment. We used EXE-SPEM to model software processes and we proposed a cloud-based architecture for software process enactment. The architecture executes XML models (mapped from EXE-SPEM) on a configurable hybrid cloud. Cloud accessibility, availability, multi-tenancy and elasticity can help addressing communicational, managerial and technical challenges which GSD faces. We defined a set of requirements that needs to be met by a software process execution environment to address those challenges. Then we have illustrated how those requirements are met by our architecture.

As a demonstrating example, we created a distributed model checking activity using the distributed model checker DiVinE and used it to construct and enact a simple process model.

[7]http://why3.lri.fr/
[8]https://www3.hhu.de/stups/prob

Empirical studies and more complex software processes are still needed to evaluate the impact of our approach on software development practices in terms of: performance, quality, productivity and cost. The process enactment platform represents a core that can be extended in the future. These are some of the other areas which we will be addressing in the longer term:

- Mining the software artefact repository to automatically produce process-based safety arguments for certification purposes in the context of safety critical software development.
- Adopting a smart scheduling mechanism for performance and cost optimization through cloud resources allocation and workflow engines selection.
- Investigate the potential of the pay-as-you-go pricing model for tools involved in software processes that are executed on the cloud.

ACKNOWLEDGEMENTS

We are grateful to the school of computing science at Newcastle University, UK for funding this work and to the following people: Andreas Roth, Paul Watson, Henry Muccini and Jacek Cala for the useful feedback on the earlier versions of this work. Barbara Gallina has been supported Swedish Foundation for Strategic Research (SSF) project SYNOPSIS and by the EU Artemis project 333053 CONCERTO.

REFERENCES

[1] E. Carmel and R. Agarwal, "Tactical approaches for alleviating distance in global software development," *Software, IEEE*, vol. 18, no. 2, pp. 22–29, Mar 2001.

[2] C. Ebert and P. De Neve, "Surviving global software development," *Software, IEEE*, vol. 18, no. 2, pp. 62–69, Mar 2001.

[3] R. Battin, R. Crocker, J. Kreidler, and K. Subramanian, "Leveraging resources in global software development," *Software, IEEE*, vol. 18, no. 2, pp. 70–77, Mar 2001.

[4] S. Hashmi, V. Clerc, M. Razavian, C. Manteli, D. Tamburri, P. Lago, E. Di Nitto, and I. Richardson, "Using the cloud to facilitate global software development challenges," in *Global Software Engineering Workshop (ICGSEW), 2011 Sixth IEEE International Conference on*, 2011, pp. 70–77.

[5] M. Al-qadhi and J. Keung, "Cloud-based support for global software engineering: Potentials, risks, and gaps," in *Proceedings of the International Workshop on Innovative Software Development Methodologies and Practices*, ser. InnoSWDev 2014. ACM, 2014, pp. 57–64.

[6] Y. Arimura and M. Ito, "Cloud computing for software developmnt environment," *Fujitsu Sci. Tech. J*, vol. 47, no. 3, pp. 325–334, 2011.

[7] S. K. Paul. (2015) Can cloud computing enhance agile software development? [Online]. Available: www.cloudsigma.com/can-cloud-computing-enhance-agile-software-development/

[8] S. Alajrami, B. Gallina, and A. Romanovsky, "Exe-spem: Towards cloud-based executable software process models," in *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development*, 2016.

[9] S. Sahay, "Global software alliances: The challenge of 'standardization'," *Scand. J. Inf. Syst.*, vol. 15, no. 1, pp. 3–21, Jan. 2003.

[10] E. Carmel, *Global Software Teams: Collaborating Across Borders and Time Zones*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

[11] J. Herbsleb and D. Moitra, "Global software development," *Software, IEEE*, vol. 18, no. 2, pp. 16–20, Mar 2001.

[12] E. O. Conchúir, P. J. Ågerfalk, H. H. Olsson, and B. Fitzgerald, "Global software development: Where are the benefits?" *Commun. ACM*, vol. 52, no. 8, pp. 127–131, Aug. 2009.

[13] P. Ågerfalk, B. Fitzgerald, H. Holmstrm Olsson, and E. Conchir, "Benefits of global software development: The known and unknown," in *Proceedings of the Software Process, 2008 International Conference on Making Globally Distributed Software Development a Success Story*, ser. ICSP'08, 2008, pp. 1–9.

[14] A. Begel and N. Nagappan, "Global software development: Who does it?" in *ICGSE,IEEE International Conference on Global Software Engineering.*, Aug 2008, pp. 195–199.

[15] M. Paulk, W. Curtis, M. B. Chrissis, and C. Weber, "Capability maturity model for software (version 1.1)," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-93-TR-024, 1993.

[16] W. W. Royce, "Managing the development of large software systems: Concepts and techniques," in *Proceedings of the 9th International Conference on Software Engineering*, ser. ICSE '87, 1987, pp. 328–338.

[17] B. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, May 1988.

[18] OMG, "Software and Systems Process Engineering Meta-Model Specification, V2.0," Object Management Group (OMG), Tech. Rep., April 2008.

[19] P. M. Mell and T. Grance, "Sp 800-145. the nist definition of cloud computing," National Institute of Standards & Technology, Tech. Rep., 2011.

[20] B. Boehm, "Some future trends and implications for systems and software engineering processes," *Systems Engineering*, vol. 9, no. 1, pp. 1–19, 2006.

[21] M. A. Chauhan and M. A. Babar, "Cloud infrastructure for providing tools as a service: Quality attributes and potential solutions," in *Proceedings of the WICSA/ECSA 2012 Companion Volume*, ser. WICSA/ECSA '12, 2012, pp. 5–13.

[22] S. Alajrami, A. Romanovsky, P. Watson, and A. Roth, "Towards cloud-based software process modelling and enactment," *In CloudMDE 2014*, p. 6, 2014.

[23] H. Kagdi, M. L. Collard, and J. I. Maletic, "A survey and taxonomy of approaches for mining software repositories in the context of software evolution," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 19, no. 2, pp. 77–131, 2007.

[24] J. Barnat, L. Brim, M. Češka, and P. Ročkai, "DiVinE: Parallel Distributed Model Checker," in *Parallel and Distributed Methods in Verification and High Performance Computational Systems Biology*. IEEE, 2010, pp. 4–7.

[25] S. Bucur, V. Ureche, C. Zamfir, and G. Candea, "Parallel symbolic execution for automated real-world software testing," in *Proceedings of the Sixth Conference on Computer Systems*, ser. EuroSys '11. ACM, 2011, pp. 183–198.

[26] M. Oriol and F. Ullah, "Yeti on the cloud," in *Software Testing, Verification, and Validation Workshops (ICSTW), Third International Conference on*, 2010, pp. 434–437.

[27] A. Pakhira and P. Andras, "Leveraging the cloud for large-scale software testing - a case study: Google chrome on amazon," in *Software Testing in the Cloud: Perspectives on an Emerging Discipline*. USA: IGI Global, 2013, pp. 252–279.

[28] G. Mwansa and E. Mnkandla, "Migrating agile development into the cloud computing environment," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, June 2014, pp. 818–825.

[29] P. Yara, R. Ramachandran, G. Balasubramanian, K. Muthuswamy, and D. Chandrasekar, "Global software development with cloud platforms," in *Software Engineering Approaches for Offshore and Outsourced Development*, ser. Lecture Notes in Business Information Processing, O. Gotel, M. Joseph, and B. Meyer, Eds. Springer, 2009, vol. 35, pp. 81–95.

[30] N. Oza, J. Mnch, J. Garbajosa, A. Yague, and E. Gonzalez Ortega, "Identifying potential risks and benefits of using cloud in distributed software development," in *Product-Focused Software Process Improvement*, ser. LNCS, J. Heidrich, M. Oivo, A. Jedlitschka, and M. Baldassarre, Eds. Springer, 2013, vol. 7983, pp. 229–239.

[31] H. Hiden, S. Woodman, P. Watson, and J. Cala, "Developing cloud applications using the e-science central platform," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1983, 2013.

[32] A. Iliasov, D. Adjepon-Yamoah, P. Stankaitis, and A. Romanovsky, "Putting provers on the cloud," in *23rd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP 2015), March 04-06*, 2015.