ATLANTIS PRESS International Journal of Networked and Distributed Computing, Vol. 5, No. 4 (October 2017) 221–232

Enabling GSD Task Allocation via Cloud-based Software Processes

Sami Alajrami [1] , Barbara Gallina [2] , Alexander Romanovsky [1]

[1] School of Computing, Newcastle University,
Address,
Newcastle upon Tyne, NE1 7RU, United Kingdom
E-mail: {s.h.alajrami, alexander.romanovsky}@ncl.ac.uk

[2] Mälardalen University
Address,
Västerås, Sweden
E-mail: barbara.gallina@mdh.se

Abstract

Allocating tasks to distributed sites in Global Software Development (GSD) projects is often done unsystematically and based on the personal experience of project managers. Wrong allocation decisions increase the project's risks as tasks have dependencies that are inherited by the distributed sites. Decision support can help make the task allocation a more informed and systematic process. The challenges in allocating tasks to distributed sites exist because of three distance dimensions between sites (geographical, temporal and cultural). An informed task allocation decision needs to consider these distances. Therefore, in this paper, we propose to integrate and semi-automate the calculation of an existing Global Distance Metric (GDM) into an architecture that supports executing cloud-based software processes. We analyze the potential of integrating the GDM into this architecture and identify the needed extensions to the architecture.

Keywords: Global software development , Distributed tasks allocation decision support, Cloud-based software processes, Global distance

1. Introduction

Global Software Development (GSD) [11] has moved software firms from monolithic development (one team at one location) to multiple geographically-distributed teams collaborating on a development project. GSD benefits are established in literature [6,11,8] and include: a) utilizing cheaper labour in different countries hence implying cost reduction, b) having multiple teams working in different time zones which leads to shorter development cycles, and c) being in closer proximity to customers and emerging markets.

Despite the benefits, teams collaborating in GSD projects face geographical, temporal and cultural distances which make managing such projects a challenging task. Naturally, dependencies exist between the distributed tasks. These task dependencies (during process enactment) make it essential to ensure no deadlocks happen between distributed sites.

The distances between distributed sites introduce management challenges that can increase the risks for GSD projects. Such management challenges are inherent in GSD projects and are linked to issues of communication, control and supervision, coordination, creating social bonds, and building trust [7].

Among the main GSD challenges, allocating the right resources/tasks to each site is of critical importance.

The complexity of the dependencies in GSD projects is reflected on the task allocation decisions [12]. Task allocation can either decrease or increase the risks associated with GSD projects (such as: decreased productivity and lack of trust between sites) [14]. Despite the importance of task allocation decisions, in practice, the decision making process is not very systematic and often is based on the personal experience of the managers [13]. For example, allocating activities to sites with low differences (nearshoring [7]) seems to reduce GSD risks, while having large cultural differences between sites affects the trust between them. Therefore, a systematic decision support is needed to support allocating GSD activities.

The larger the distance between distributed sites, the larger the difference. Nearshoring [7] (allocating tasks to sites with low differences) reduces the risks associated with GSD projects management [14]. Carmel and Abbott argue that the rise of nearshoring proves that distance still matters [7]. Therefore, in this paper, we explore how we can make informed decisions about task allocation in GSD projects based on the distances between the distributed sites. In order to base the decision making on the distance factor, this factor needs to be quantified. For that purpose, we use the *Global Distance Metric* [17] which assesses and quantifies the distance between collaborating sites.

In a previous work, we proposed a reference architecture for supporting Software Development as a Service (SDaaS) in the cloud [5]. The potential for using the cloud to facilitate GSD projects has been discussed in [10]. The SDaaS architecture goes one step further and uses a model-based approach to execute software processes (which can be distributed processes). The SDaaS architecture facilitates by default: 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 [2]. Therefore, in this paper, we propose to extend the SDaaS architecture to support semi-automatic calculation of the *Global Distance Metric* in order to provide task allocation decision support for project managers.

The rest of the paper is structured as follows: Section 2 provides brief background on the SDaaS architecture and Global Distance Metric (GDM). Section 3 describes our proposed extension of the SDaaS architecture to provide GSD task allocation decision support. Section 4 explains the paper proposal using an example process. Section 5 reviews some existing works that target task allocation support in GSD projects. Finally, Section 6 concludes the paper and discusses the current limitations.

2. Background & Motivation

In this section, we briefly cover essential background information on our architecture for executing cloud-based software processes and on GSD distance metric and task allocation.

2.1. The SDaaS architecture

We proposed a reference architecture for supporting executing software process models in the cloud [5]. As shown in Fig. 1, the architecture consists of two main services: the design time service and the run-time service. The design time service deals with modelling and manipulation of software processes while the run-time service deals with scheduling, executing and monitoring software processes execution in the cloud. The execution takes place in a set of distributed workflow engines (with different computational and privacy specification). The workflow registry component tracks and manages the active workflow engines. During the execution, process models consume and produce software artefacts (code, docs, models, tests etc.). These artefacts are maintained along with meta-data describing them by the artefact manager component. The tools needed to support each process activity can be integrated within the environment or can be interfaced as a service. Activities can be: a) automated

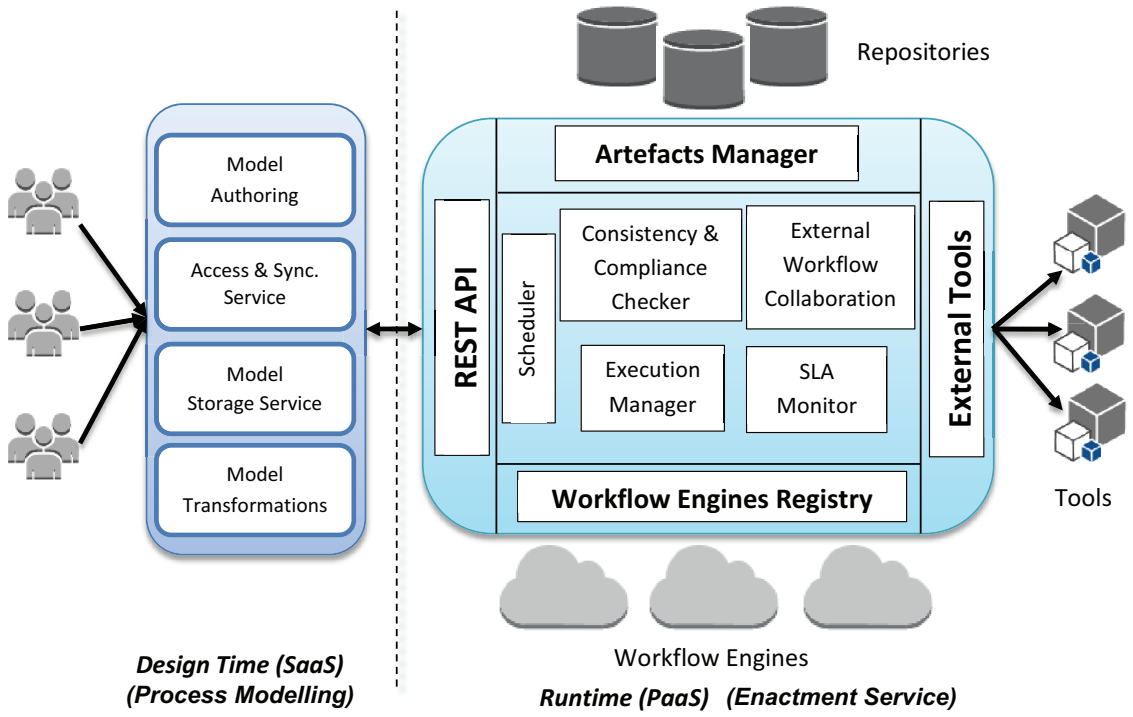

Fig. 1. The SDaaS reference architecture [5].

(triggering tools to perform certain tasks e.g., testing), b) interactive (receiving input from users e.g., for editing artefacts), or c) decision points (deciding -automatically or interactively- on which branch of the process to follow).

SDaaS facilitates distributed development. It uses a unified SaaS user interface which enables teams across distributed sites to access a shared development environment. This means that teams will be collaborating within the same virtual environment which is highly accessible and available via the cloud. The cloud model is based on provisioning of services and the SDaaS architecture provisions development environments and tool-chains as services. Hashmi et al. [10] argue that GSD challenges can be overcome via the use of services (Service Oriented Architecture - SOA). Their argument is that SOA increases the interoperability and technology and business alignment between sites [10]. Since the SDaaS architecture adopts a SOA, we argue that it can overcome GSD challenges.

In addition, the SDaaS architecture adopts a model-driven approach and supports modelling of dynamic processes like the ones that would be found in GSD projects. The use of models allows for raising the levels of abstraction and improves communication and understanding between distributed sites. The artefact manager of the SDaaS architecture allows for tracing and maintaining shared artefacts. Finally, SDaaS leverages the scalability of cloud to allocate computing resources and tools as services on the fly to meet the needs of individual tasks in a GSD project. However, the SDaaS architecture does not provide decision support for task allocation.

2.2. *EXE-SPEM*

The SDaaS architecture uses EXE-SPEM [3] as the modelling language for modelling cloud-based executable software processes. EXE-SPEM is an extension of the OMG Software Process and System Engineering Meta-model (SPEM 2.0 [1]). EXE-SPEM enables modelling important information needed for cloud-based process enactment such as: control flow

(i.e., order, conditions and loops), the responsible team/team member for enacting each activity (task) in the process, and the cloud-specific enactment information such as: the choice of cloud deployment model (private vs. public) and the amount of computational resources required. EXE-SPEM is created by extending the meta-model of SPEM2.0 as shown in Appendix A (which is simplified for readability) where meta-classes with dark grey background are added to the original SPEM2.0 meta-model and the ones with light grey background have new attributes.

Using model-to-text transformational rules, EXE-SPEM process models are mapped into XML-based textual representations which are compliant with the schema shown in Fig. 2.

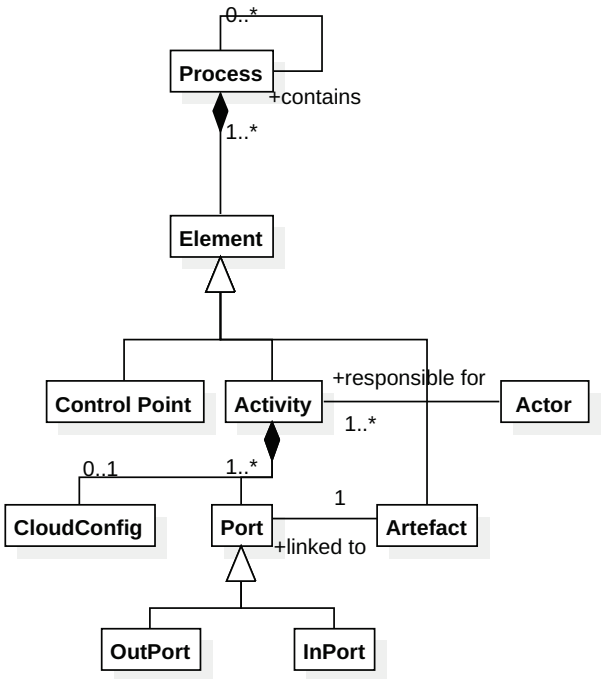

Fig. 2. The XML schema for representing EXE-SPEM process models

2.3. GSD task allocation

Allocating GSD tasks to distributed sites has a direct impact on the risks associated with distributed development projects. Allocation is often done based on multiple criteria (labor cost rates, availability of workforce and expertise) [14].

Lamersdorf et al. have reviewed several tactics followed in practice to avoid the risks associated with distance between distributed sites [13]. The first tactic is to minimize the collaboration needed (separation of concerns between sites) which reduces the GSD communication problems. Another tactic is to minimize the differences (e.g., cultural, temporal) between sites. Grinter et al. [9] proposed the use of strategies from organizational theory to task allocation in GSD projects.

The optimal task allocation decision needs to be based on understanding of the capabilities, differences and distances among the distributed tasks. Distance between sites is the main source of risk in GSD projects and it takes different dimensions (geographical, temporal and cultural). Thus, quantifying these dimensions of distance helps to make an effective and informed task allocation decision by project managers.

2.4. Global Distance Metric

Noll and Beecham [17] have developed the global distance metric (GDM) to measure global distance between distributed sites collaborating on GSD projects. The metric combines and quantifies the three dimensions of GSD distance: geographic, temporal, and cultural between two sites. The metric is then calculated as follows:

$$D_{global} = \sqrt{D^2_{geographic} + D^2_{temporal} + D^2_{cultural}} \quad (1)$$

where D_c is the value of the distance dimension and $c \in \{geographic, temporal, cultural\}$. Each of the dimensions in Eq.1 is calculated as the sum of the impact values for different distance factors. A list of these factors and there impact values is provided in Table 1. Each team (site) computes the global distance metric from other collaborating sites. This provides a quantified representation of the perceived distances between the distributed sites towards each other.

Table 1 is taken from [17] and shows the factors contributing to each distance dimension along with their impact values. These impact values have been identified by surveying practitioners. As we can see

in the table, factors affecting both the geographical and temporal distances are straightforward to assess (based on the locations and timezones of distributed sites). However, the cultural distance depends more on the perception and trust between teams. For example, as noted by Noll and Beecham [17], having a team member from the same nationality (of a certain site) in another site may lead to increase the perceived trust and reduce the perceived language barriers.

3. SDaaS-based task allocation

In this section, we build on existing GSD support in the SDaaS architecture by facilitating decision making about allocating tasks across distributed sites. Since knowing the distance (in all its dimensions) between distributed sites is crucial for making the right allocation decision, we propose to integrate the measurement of the *Global Distance Metric (GDM)* [17] (see Section 2.4) within the SDaaS architecture.

The SDaaS architecture can automate the measurement of the geographical and temporal distances of the GDM based on knowing the collaborating sites and their locations. In addition, it can calculate the cultural distance perceived by each site towards each other site by relying on input from team members. These calculated values can then be used to calculate the overall GDM between each two sites using equation 1.

3.1. The SDaaS architecture extension

In order to support the GDM calculation, the SDaaS architecture needs to be extended. Task allocation is needed during the process design-time phase. The following extensions are needed in the SDaaS architecture:

1. **Extending the process models.**
   The teams which might be involved in executing the process and their respective sites need to be integrated in the process models (which are created using EXE-SPEM [3]). We extend the EXE-SPEM meta-model which defines EXE-SPEM process models elements. As shown in Appendix B, the extended meta-model of EXE-SPEM integrates the *Site* and *Team* meta-classes (in dark grey). The *Activity* meta-class has a new attribute stating the site that the activity has been allocated to. Finally, the *CulturalDistanceKind* enumeration is added to represent different cultural distance factors as shown in Table 1.

   In addition to extending the meta-model of EXE-SPEM, we extend the schema for defining the XML representation of EXE-SPEM process models as shown in Fig. 3 where the *Site* and *Team* have been added.

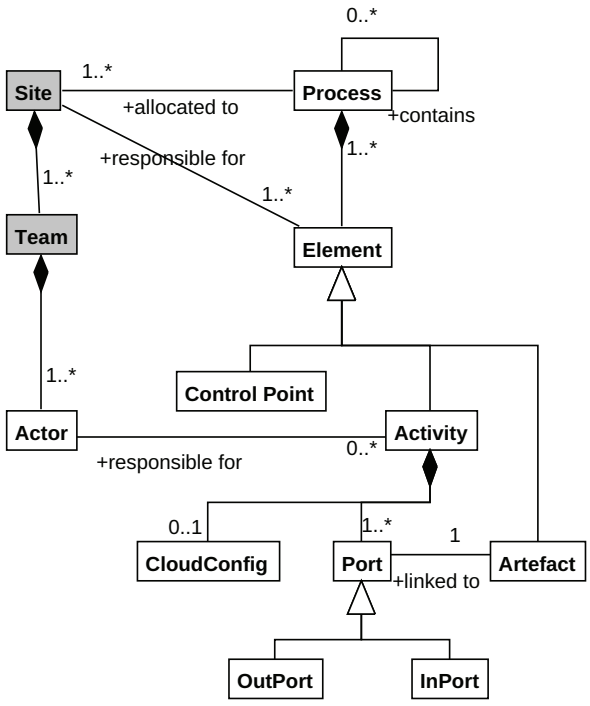

Fig. 3. The software process model XML schema

2. **Adding a GDM calculation module**
   The design-time part of the SDaaS architecture (see Fig. 1) needs to be extended by adding a module for calculating the GDM (following Eq 1). The geographical and temporal distance factors can be automatically calculated by this module using the team and

Table 1. Factors contributing to distances [17]

No.	Factors affecting geographic distance	Impact Value
1	Different building on same campus	1
2	Different towns in same region (two hour drive)	2
3	Less than three hour flight (Frankfurt to Helsinki)	3
4	Transcontinental flight (New York to San Francisco)	4
5	Intercontinental flight (London to Shanghai)	4
No.	Factors affecting temporal distance	Impact Value
1	Transcontinental (five hour overlap)	0
2	Intercontinental (three or four hour overlap)	3
3	Global (one or two hour overlap)	4
4	No overlap	4
No.	Factors affecting cultural distance	Impact Value
1	Uneven language skills	3
2	East/West divide in culture	3
3	Different national culture	2
4	Different organizational culture	3

site information from the process model. The cultural distance, however, is a subjective factor. Therefore, this module should interact with the team members to calculate their perceived cultural distance factors towards other teams at different sites. This can be done using the factors from Table 1.

3. **Visualizing the GDM between distributed sites**
Once the GDM between each pair of distributed sites is calculated, the project manager needs to view the overall perceived distances between distributed sites in order to make the best allocation decisions. The distances can be visualized following the example in Fig. 4 which is taken from Noll and Beecham [17] and shows the distances between three distributed teams (Germany, Spain and UK). The numbers represent the perceived distance from one site towards another. The larger the number, the larger the distance and consequently, the larger the differences and risks.

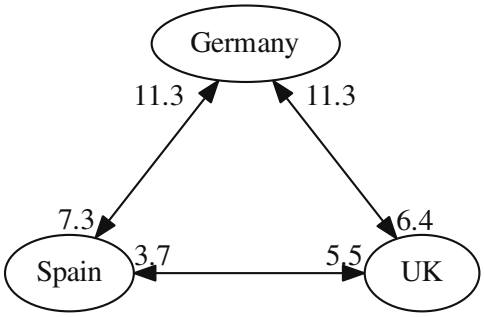

Fig. 4. The global distance between three distributed teams. Taken from [17]

The decision making process is depicted in Fig. 5. It starts with the project manager or process author creating the process model and specifying the teams that might be involved in this process. Then, the GDM between these sites is calculated and visualized. Finally, the project manager makes a decision to allocate specific tasks to specific teams based on a trade-off between multiple factors (e.g., labour cost, availability, expertise and GDM). Based on the trade-offs, the project manager may decide to make modifications to the process in order to

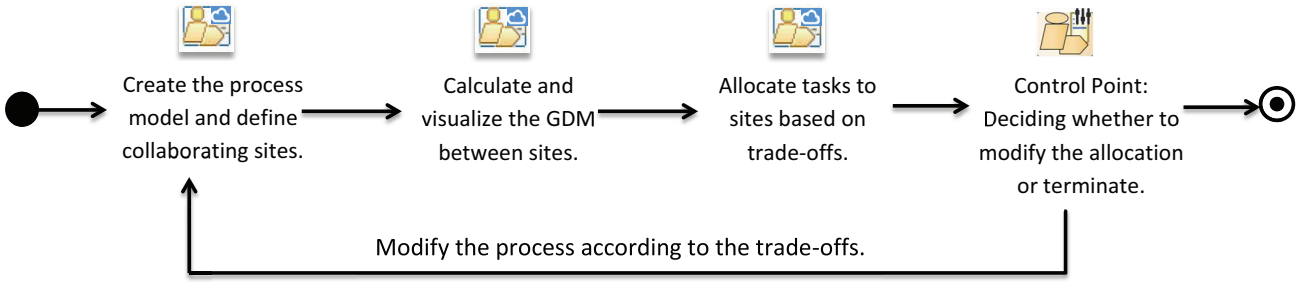

Fig. 5. The decision making process

reduce the risks associated with involvement of distributed teams. For example, to reduce dependencies between two teams with high GDM value.

4. Demonstrating Example

To demonstrate the proposed approach in this paper, we use a process model we developed in a previous work [4]. The process is a safety process for generating product and process safety arguments to be used in building safety cases for safety critical systems. Fig. 6 shows the original process (before introducing the extension for task allocation support) modelled in EXE-SPEM. The process consists of *activities* which consume and produce *work products* (artefacts) and are performed by *role use* (actors).

Figure 7 shows the same process modelled with the extended EXE-SPEM. As the figure shows, the model now describe the collaborating sites (one in the UK and another in India). By analyzing calculating the GDM between these two sites from the process model, the distances can be reported and visualized to project managers who can then make an informed decision to allocate certain activities to certain sites. For example, as shown in Fig. 7, the decision could be to allocate the *Product-based Argument Generation* activity to the UK site and the *Process-based Argument Generation* to the Indian site. After allocating the activities to sites, the process model can be executed in the SDaaS architecture.

5. Related Work

Several approaches for task allocation in GSD projects have been studied in literature. Some studies have reviewed these approaches (e.g. [12,13]). Imtiaz and Ikram [12] have identified several factors that impact task allocation in GSD projects such as: labour cost, expertise, task-site dependency,temporal and cultural differences, etc. Task allocation approaches often target one or few of these factors and a trade-off between them need to be performed based on the situation and the project [12].

Task allocation for GSD projects can be categorized into two groups [16]: a) *optimization approaches* (aiming to decide on the best task allocation with respect to a specific goal) and b) *predictive approaches* (aiming to evaluate different task allocations individually).

Mockus and Weiss [15] propose an optimization algorithm which aims to minimize the communication needed between sites and thus reducing the communication overhead. However, this approach only addresses a single criterion (i.e., communication overhead). Another approach developed by Setamanit et al. [18] uses a simulation model to compare different task allocation strategies with respect to productivity and development time. This approach, however, does not provide task allocation decision support for individual projects and instead compare the strategies generally. Lamersdorf and Münch [14] study the risk identification and effort estimation perspectives in GSD task allocation and conclude that although some approaches can be used to support certain aspects of task allocation, there is no comprehensive

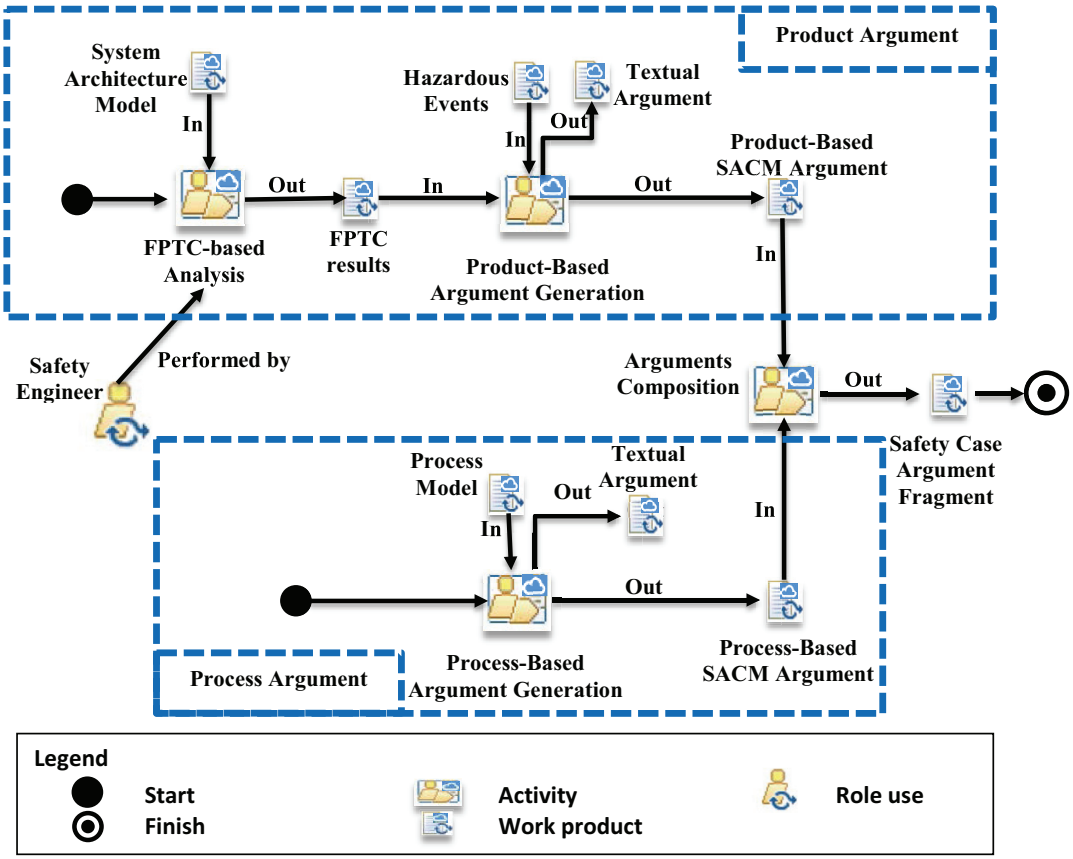

Figure 6: Safety process modelled using EXE-SPEM. Adapted from [4]

approach for systematic task allocation covering all the needed aspects.

6. Conclusion & Future Work

In this paper, we extend the SDaaS architecture [5] to provide task allocation decision support for GSD projects. SDaaS facilitates conducting GSD projects in the cloud and automate the computational ad tool resources allocation on demand. The extension uses the Global Distance Metric (GDM) [17] to quantify the three dimensions of GSD distance (geographical, temporal and cultural). This extension allows projects managers to make task allocation decisions baring in mind the distances (differences) between the collaborating distributed tasks and the risks associated with it.

In practice, the decision on task allocation is made based on multiple factors (e.g. labour cost, expertise, availability, etc.) Although this paper focuses only on one factor which impacts task allocation in GSD projects (the distance factor), other factors could similarly be integrated within the SDaaS architecture in future works. The motivation for extending the SDaaS architecture is that it already support other aspects of GSD projects (as discussed in Section 2.1).

This paper comes as a first step towards a comprehensive approach for task allocation decision support within the SDaaS architecture. In the future, other factors affecting task allocation decisions need to integrated. It is also possible to adapt the model-based approach developed by Lamersdorf and Münch [14] which integrates three models: a risk

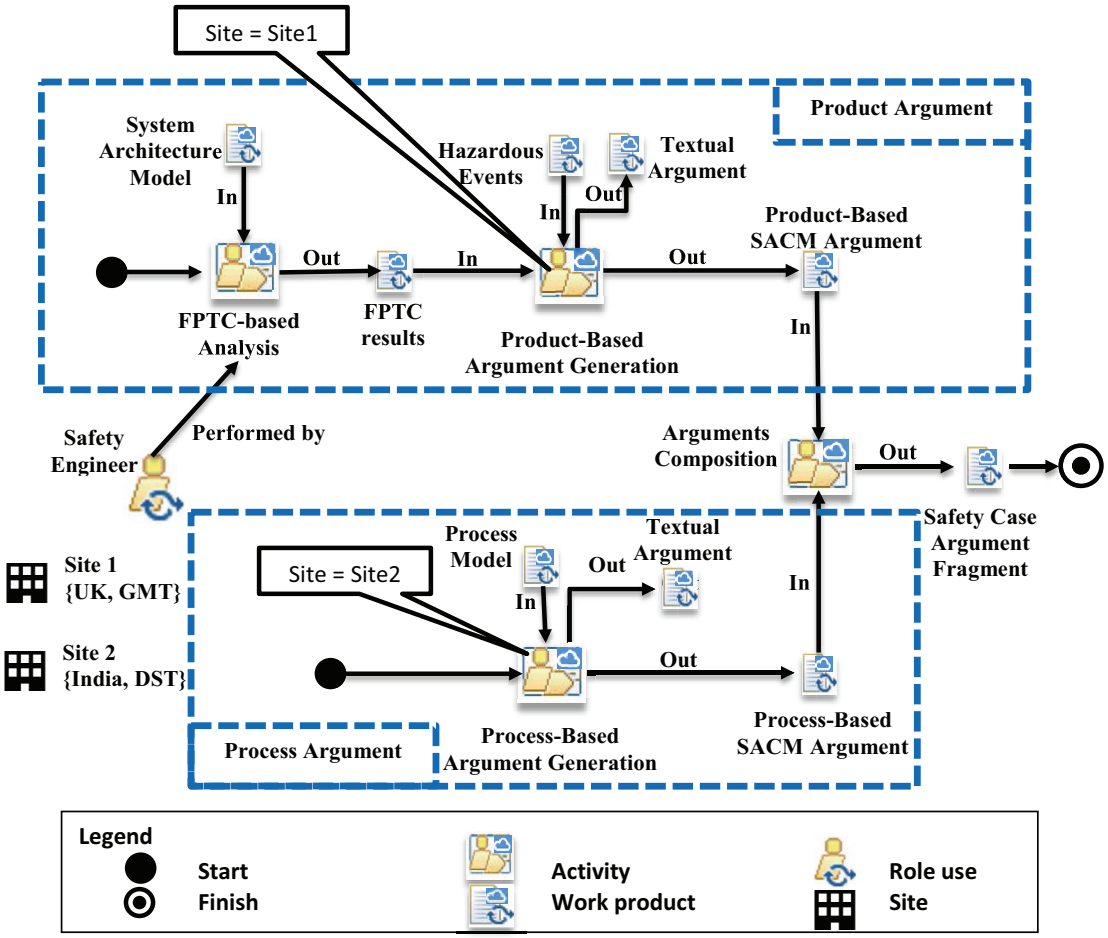

Figure 7: Safety process modelled using the extended EXE-SPEM

model which identifies risks for each allocation alternative, an optimization model which suggests alternative allocation based on multiple criteria, and an effort overhead model which estimates the effort needed for each allocation alternative.

Acknowledgements

B. Gallina is partially financially supported by EU and VINNOVA via the ECSEL Joint Undertaking under grant agreement No 692474, project name AMASS.

Appendix A The meta-model of EXE-SPEM

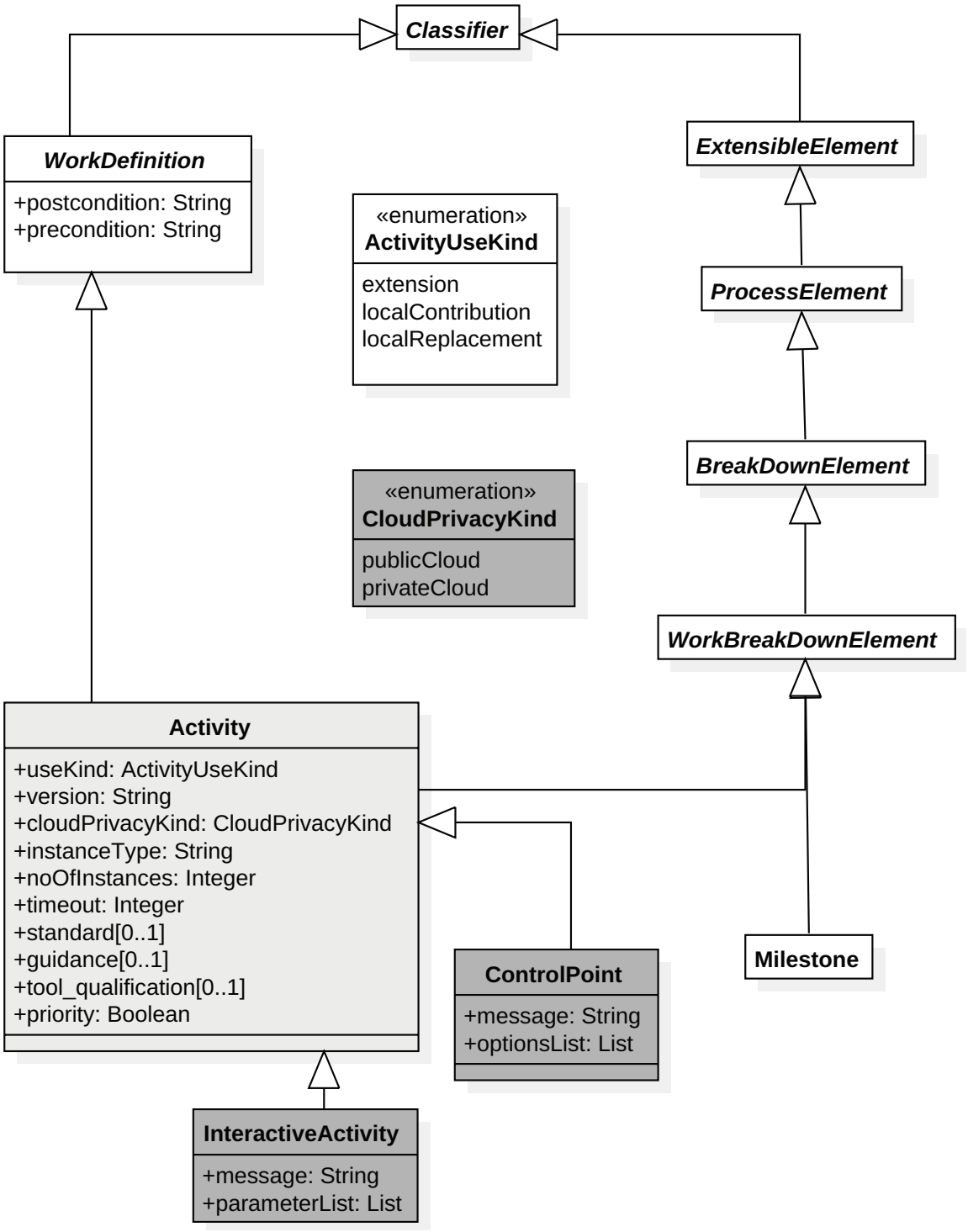

Figure A.1: The meta-model of EXE-SPEM

Appendix B The extended meta-model of EXE-SPEM

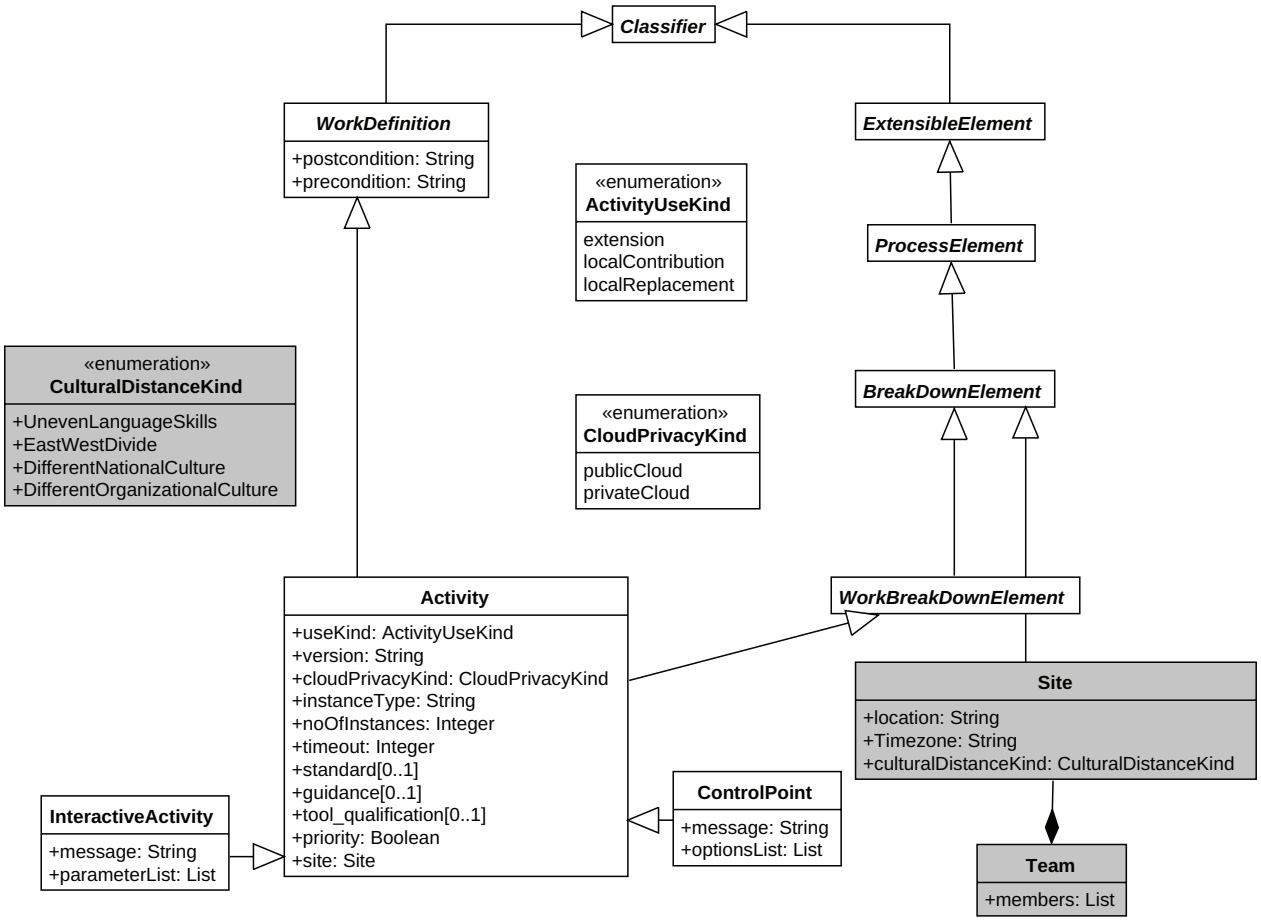

Figure B.1: The extended meta-model of EXE-SPEM

References

1. *Software and Systems Process Engineering Meta-Model Specification, V2.0*. Number formal/2008-04-01. Object Management Group (OMG), MA, USA, April 2008.
2. Sami Alajrami, Barbara Gallina, and Alexander Romanovsky. Enabling global software development via cloud-based software process enactment. Technical Report TR-1494, Newcastle University, School of Computing Science, 03 2016.
3. Sami Alajrami, Barbara Gallina, and Alexander Romanovsky. EXE-SPEM: Towards Cloud-based Executable Software Process Models. In *MODELSWARD'16 - Proceedings of the 4rd International Conference on Model-Driven Engineering and Software Development, Rome, Italy, 19-21 February.*, pages 517–526. Scitepress, 2016.
4. Sami Alajrami, Barbara Gallina, Irfan Sljivo, Alexander Romanovsky, and Petter Isberg. Towards Cloud-Based Enactment of Safety-Related Processes. In Amund Skavhaug, Jérémie Guiochet, and Friedemann Bitsch, editors, *Computer Safety, Reliability, and Security - 35th International Conference, SAFECOMP'16, Trondheim, Norway, September 21-23, Proceedings*, pages 309–321. Springer, 2016.
5. Sami Alajrami, Alexander Romanovsky, and Barbara Gallina. Software Development in the Post-PC Era: Towards Software Development as a Service. In Pekka Abrahamsson and Andreas Jedlitschka, editors, *The 17th International Conference on Product-Focused Software Process Improvement, PROFES'16, Trondheim, Norway, November 22-24, Proceedings*. Springer, 2016.
6. Erran Carmel. *Global Software Teams: Collaborating Across Borders and Time Zones*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.
7. Erran Carmel and Pamela Abbott. Why 'nearshore' means that distance matters. *Commun. ACM*, 50(10):40–46, October 2007.
8. Eoin Ó Conchúir, Pär Ågerfalk, Helena Olsson, and Brian Fitzgerald. Global Software Development: Where Are the Benefits? *Commun. ACM*, 52(8):127–131, August 2009.
9. Rebecca E. Grinter, James D. Herbsleb, and Dewayne E. Perry. The geography of coordination: Dealing with distance in r&d work. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, GROUP '99, pages 306–315, New York, NY, USA, 1999. ACM.
10. S. I. Hashmi, V. Clerc, M. Razavian, C. Manteli, D. A. Tamburri, P. Lago, E. D. Nitto, and I. Richardson. Using the cloud to facilitate global software development challenges. In *2011 IEEE Sixth International Conference on Global Software Engineering Workshop*, pages 70–77, Aug 2011.
11. J D Herbsleb and D Moitra. Global Software Development. *Software, IEEE*, 18(2):16–20, March 2001.
12. Salma Imtiaz and Naveed Ikram. Dynamics of task allocation in global software development. *Journal of Software: Evolution and Process*, 29(1), 2017.
13. A. Lamersdorf, J. Munch, and D. Rombach. A survey on the state of the practice in distributed software development: Criteria for task allocation. In *2009 Fourth IEEE International Conference on Global Software Engineering*, pages 41–50, July 2009.
14. Ansgar Lamersdorf and Jürgen Münch. *Model-Based Task Allocation in Distributed Software Development*, pages 37–53. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
15. A. Mockus and D. M. Weiss. Globalization by chunking: a quantitative approach. *IEEE Software*, 18(2):30–37, Mar 2001.
16. Jürgen Münch and Ansgar Lamersdorf. *Systematic Task Allocation Evaluation in Distributed Software Development*, pages 228–237. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
17. John Noll and Sarah Beecham. *Measuring Global Distance: A Survey of Distance Factors and Interventions*, pages 227–240. Springer International Publishing, 2016.
18. Siri-on Setamanit, Wayne Wakeland, and David Raffo. Planning and improving global software development process using simulation. In *Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner*, GSD '06, pages 8–14, New York, NY, USA, 2006. ACM.