Exploring Architecture Design Alternatives for Global Software Product Line Engineering

Bedir Tekinerdogan
Department of Computer Engineering
Bilkent University
Ankara, Turkey
e-mail: bedir@cs.bilkent.edu.tr

Semih Cetin, Ferhat Savcı
Cybersoft Information Technologies, Ata Plaza 3/3, 34758,
Atasehir, Istanbul, Turkey
e-mail: {semih.cetin, ferhat.savci}@cs.com.tr

***Abstract* — Current trends in software engineering show that large software projects have to operate with teams which are working in different locations. An analysis of current global software engineering literature shows that the focus has been basically on single system development. Yet, very often organizations do not aim to develop a single product but a *product line* for a particular market segment. Unfortunately, the notion of global software development has not been explicitly addressed in product line engineering. We introduce and define the notion of *global software product line engineering (GSPLE)* to integrate global software engineering paradigm with the software product line engineering paradigm. Based on an analysis of architectural approaches in both paradigms we define the space of the different software architecture design alternatives for GSPLE. We illustrate the architecture design alternatives using examples of an industrial context.**

Keywords-Product Line Engineering; Global Software Development; Business Strategies

I. INTRODUCTION

Current trends in software engineering show that large software projects have to operate with teams that are working in different locations. The reason behind this globalization of software development stems from clear business goals such as reducing cost of development, solving local IT skills shortage, and supporting outsourcing and offshoring [1]. There is ample reason that these factors will be even stronger in the future, and as such we will face a further globalization of software development [8]. To cope with these problems the concept of global software engineering (GSE) is introduced [9]. GSE is a relatively new concept in software development that can be considered as the coordinated activity of software development that is not localized and central but geographically distributed.

An analysis of current global software engineering literature shows that the focus has been basically on single system development. Yet, very often organizations do not aim to develop a single product but a *product line*. A product line is defined as a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [11]. Despite earlier software reuse approaches, software product line engineering (SPLE) aims to provide pro-active, pre-planned reuse at a large granularity to develop applications from a core asset base. The key motivation for adopting a product line engineering process is to develop products more efficiently, get them to the market faster to stay competitive and produce with higher quality [14]. In alignment with these goals different software product line engineering approaches have been proposed [5][11].

Unfortunately, the notion of global software development has not been explicitly addressed in product line engineering. On the other hand, an analysis of the current product line engineering approaches shows that global software development is not explicitly addressed. We can observe valuable knowledge on defining organization structures for product lines [4][11] but these do not explicitly consider the concern of globalization of the product line engineering process. To apply systematic, anticipated reuse for global software development we believe that global software development will substantially benefit from software product line engineering. In parallel, similar to single system development in which teams might be spread over different locations [3], it is also expected that product line engineering projects might operate with teams which are working in different locations. The reason for this globalization of product line engineering will also be based on the general motivations for global software development.

In this paper, we introduce and define the notion of *global software product line engineering (GSPLE)* to integrate global software engineering paradigm with the software product line engineering paradigm. The motivation for GSPLE stems from the industrial context of Cybersoft, a leading company in global software development in Turkey. The efforts to define the architecture for GSPLE have shown that the integration of SPLE and GSE can be done in multiple different ways. Based on an analysis of architectural approaches in both paradigms and our experiences we define the space of the different software architecture design alternatives for GSPLE. We illustrate the architecture design alternatives using examples of an industrial context.

The remainder of the paper is organized as follows. In Section II we briefly introduce a conceptual model for GSE, followed by an analysis to software product line engineering in section III. Section IV discusses the stakeholder analysis for GSPLE. Section V describes the strategies for integrating SPLE with GSE. Section VI discusses the design alternatives. Section VII provides the related work and section VIII concludes the paper.

II. Conceptual Model for GSE

GSE is a software development approach that can be considered as the coordinated activity of software development that is not localized and central but geographically distributed. Overall we can identify four important key concerns in GSE:

Development - the software development activities typically using a software development process. This includes activities such as requirements analysis, design, implementation and testing. Each product development site will address typically a subset of these activities.

Communication – communication mechanisms within and across sites. Typically the different sites need to adopt a common communication protocol.

Coordination – coordination of the activities within and across sites to develop the software according to the requirements. Coordination will be necessary to align the workflows and schedules of the different sites. An important goal could be to optimize the development using appropriate coordination mechanisms.

Control – systematic control mechanisms for analyzing, monitoring and guiding the development activities. This does not only include controlling whether the functional requirements are performed but also which and to what extent quality requirements are addressed.

Each of these concerns and the way they are allocated in the GSE environment will have a direct impact on the architecture. In principle, we assume that each of these concerns can be mapped to a separate implementation unit, or *layer*. Based on this assumption we have defined the conceptual layered model for GSE system as defined in Figure 1.

Here we have depicted GSE system as consisting of a structure with separate activity layers that depend on each other. The layering is defined based on conceptual relations. Activities in the *development layer* are coordinated by the *coordination layer*. The coordination of the activities will be controlled by functionality in the *control layer.* Finally, the development, coordination and control layers will require suitable communication mechanisms which are provided by the *communication layer*. In Figure 1, we have provided communication layer as a sidebar indicating that all layers will use this layer. Alternatively, a separate specific communication mechanism could be provided for each layer.

Based on this layered view of GSE system we need to decide how to allocate each layer to different nodes in the GSE environment. In the following sections we will define the different concrete deployment alternatives for GSE systems based on this model.

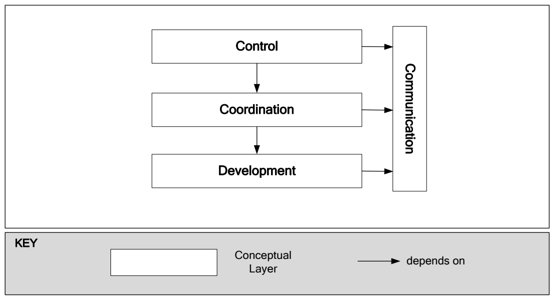

Figure 1. Layered View of GSE system with four key concerns

III. Software Product Line Engineering

Global software development can be focused on single software development or *product line engineering* [5]. Although different product line engineering processes have been proposed they share the same concepts of *domain engineering*, in which a reusable platform and product line architecture is developed, and *application engineering*, in which the results of the domain engineering process are used to develop the product members.

In general the adopted product line engineering approach has not been directly considered for global software engineering. Integration of both paradigms would in principle mean to define and align the common product line engineering process to a given GSE software architecture.

Since each unit can be considered as a separate, independent unit, the GSE system can be also set up as a *production line*. The concept of *production line* is defined in the industrial engineering and denotes a set of sequential operations established in a factory whereby materials are put through a refining process to produce an end-product; or components are assembled to make a finished article. Although the notion of software *product* line engineering is quite popular this does not seem to be the case for software *production* line engineering. Nevertheless, we think that this is important for GSE. In principle, the development units in GSE can also be considered as separate domain specific entities that aim to develop particular intermediate products, and likewise a production line can be set up.

IV. Design Space for GSPLE Architecture

It appears that we can combine the three different concepts of Global Software Engineering, Software Production Line and Software Product Line Engineering in different ways. We depict the different possibilities in Table 1. The names of the alternatives indicate whether the development is local (L) or global (G), whether production line (Pn) is applied or a conventional approach is used (C), and whether the focus is on product line (Pl) or single-system development (S). As such, the first four alternatives define the case of local software development in which the development units are co-located. The last four alternatives define the alternatives for global software development.

To denote the integration of global software engineering with product line engineering we define the notion of *global software product line engineering*. GSPLE spans the last two rows of Table 1 (GCPl and GPnPl). GSPLE can be considered as a special form of product line engineering process in which the development teams are not collocated but distributed as it is defined by the GSE paradigm. The integration of both paradigms might be based on practical necessity but in parallel will also combine the benefits of both product line engineering and global software development. From a reuse perspective we could state that GSPLE even further broadens reuse by also reusing development teams and not only artefacts. In the following we describe each alternative and provide the architectural template and an example.

TABLE 1. DESCRIPTION OF PRODUCT LINE INTEGRATION ALTERNATIVES WITH GLOBAL SOFTWARE DEVELOPMENT

Strategy	Description
LCS	Software development at a single site without product and production lines.
LPnS	Software development at a single site with production line but not focused on product variability management
LCPl	Software development at a single site focused on product variability management without production line
LPnPl	Software development at a single site with production line and focused on product variability management
GCS	Software development at multiple sites without product and production lines
GPnS	Software development at multiple sites with production line but not focused on product variability management
GCPl	Software development at multiple sites focused on product variability management without production line
GPnP1	Software development at multiple sites with production line and focused on product variability management

A. *Local Single System Development*

Local Single System Development is the traditional way of software development located at a single site. In the following sections we will also introduce product line and production line engineering for GSE, but for now we assume that a single system is developed at a single site.
The deployment view for GSE system for this case is shown in Figure 2. Note that the four layers/concerns are mapped to a single deployment node. From a theoretical perspective we could consider local system development as a special case, the simplest one, of global software development.

Example:
John Doe Software Co. develops an accounting system accustomed for Non-Exist Tech. Ltd.. The accounting system is developed at a single site using a traditional, non-product line engineering, development approach.

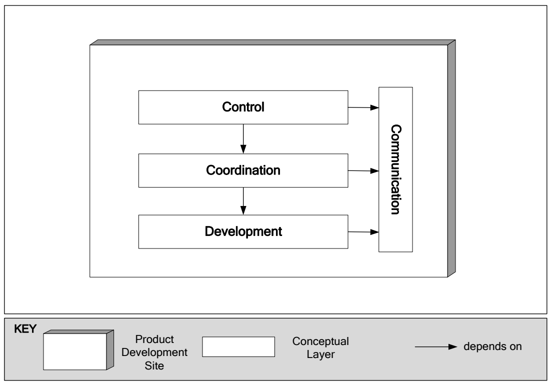

Figure 2. Local Single System Development

B. *Local Single System Development with Production Line*

We could define a software product line engineering as an application of the Pipes and Filters pattern [2]. Hereby the filters define processing units, whereas the pipes define the mechanism for distribution and communication. A conceptual model of software product line engineering is given in Figure 3. In principle a number of filters, i.e. production units can be defined which can be linked in different ways to each other. However, the key design principle for having independent filters as defined in the Pipes and Filters pattern also seem to apply for the software production line engineering process. This is to say that each production unit can be (largely) seen as a separate, black box unit that can accept input, process this and provide it to the output. In principle, the production units are not aware of each other.

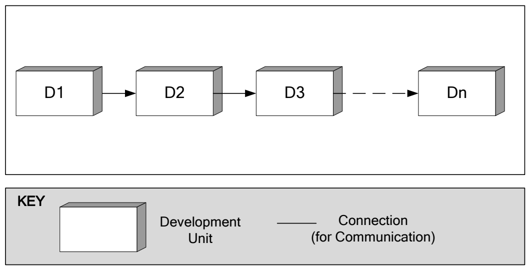

Figure 3. Software Production Line Engineering Process defined using the Pipes and Filter Pattern

Figure 4 shows the deployment view when we apply production line engineering to single-site single system development. Here the Pipes and Filters pattern has been applied to the development process units within a single site. These could be typically the applied workflows of the software development process. In Figure 4, we assume that we apply a centralized control and coordination mechanism.

However, these could also be equally distributed leading to a distributed coordination and control system of the development process.

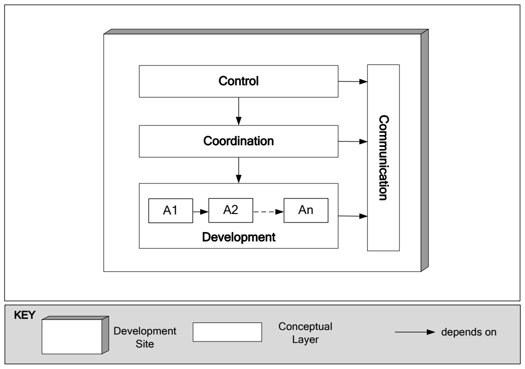

Figure 4. Local Single System Development with Production Line

Example:

John Doe Software Co. has a custom software production line based on SpringSource [12], which is used to develop an accounting system accustomed for Non-Exist Tech. The company intentionally employed the production line to reuse infrastructural modules such as logging, content management, object to relational mapping, etc.

C. Local Software Product Line Development

A product line is defined as a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [5]. The key motivation for adopting a product line engineering process is to develop products more efficiently, get them to the market faster to stay competitive and produce with higher quality. In alignment with these goals different software product line engineering approaches have been proposed. These approaches seem to share the same concepts of domain engineering, in which a reusable platform and product line architecture is developed, and application engineering, in which the results of the domain engineering process are used to develop the product members [5][9].

Example:

John Doe Software Co. develops accounting products for different customers like Non-Exist Tech. by reusing the assets and managing the variability of these assets specific to accounting domain. The company developed the product by using conventional techniques, but not based on a production line infrastructure.

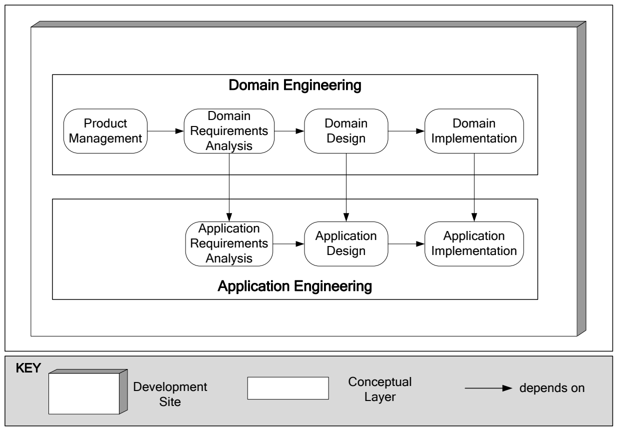

Figure 5. Local Single Product Line Development

D. Local Product Line Development with Production Line

A product line development can be realized on a *production* line platform. Hereby multiple variant products are developed based on set of sequential production units whereby components are assembled to make a finished article. Similar to the case for single system development with production line we could apply here the Pipes and Filters pattern.

Figure 6 shows an example of a local product line development with production line. Hereby, we have chosen for centralized control and coordination of the product line engineering activities (domain engineering and application engineering).

It appears that we could also have different interpretations and applications of local product line development with production line. For example, we could also apply production line engineering only for domain engineering, or only for application engineering.

Example:

John Doe Software Co. develops accounting products for different customers like Non-Exist Tech. by reusing the assets and managing the variability of these assets specific to accounting domain. The company developed the product by using its custom production line based on SpringSource. In this case, both the business domain specific assets and infrastructural modules are reused.

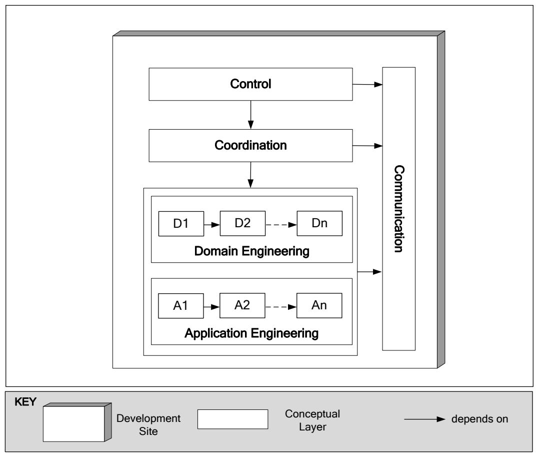

Figure 6. Local Product Line Development using a Production Line for both Domain Engineering and Application Engineering with Centralized Control and Coordination

E. Global Software Development with Single System Development

This section and the following three sections focus on defining the architecture design alternatives for GSE system in particular. We first consider GSE for single system development. We have defined the GSE with single system development alternative in Figure 7. Here the development of a single product is distributed over multiple sites (denoted by multiplicity 1..*).

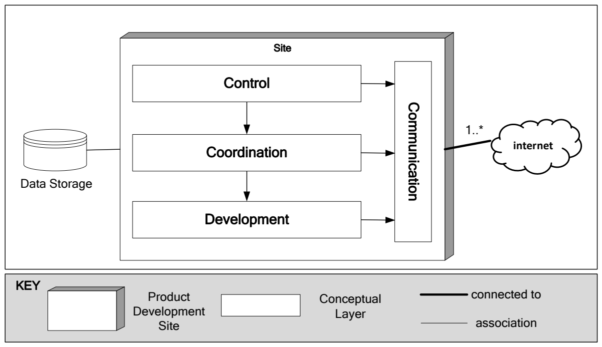

Figure 7. Global Software Development Single System Development

However, again we can observe here several sub-alternatives. These are defined basically due to the different application of the coordination and control mechanisms. In particular we can distinguish among the following alternatives as defined in Table 2.

TABLE 2. POSSIBLE ALTERNATIVE CONFIGURATIONS FOR CONTROL AND COORDINATION CONCERNS IN GSE

Alternative	Control	Coordination
1	Central	Central
2	Central	Distributed
3	Distributed	Central
4	Distributed	Distributed

A selection of one of the four alternatives will result in a refinement of the architecture in Figure 7. For example, Figure 8 shows the alternative with a central control and coordination, whereby development is distributed. Figure 9 defines an alternative with distributed control and central coordination. Of course not all the possible deployment alternatives might make sense. These should be validated from the requirements in practice.

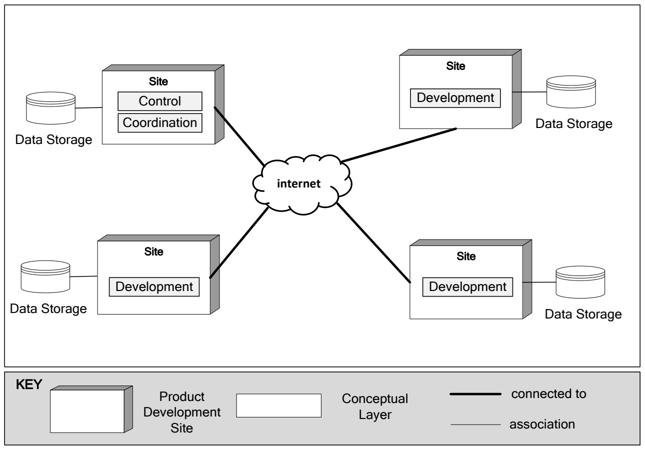

Figure 8. Deployment View of GSE with Single System Development using Central Control and Central Coordination

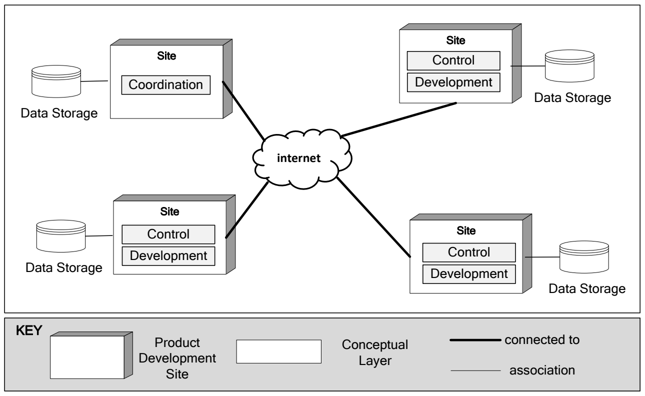

Figure 9. Deployment View of GSE with Single System Development using Distributed Control and Central Coordination

Example:
John Doe Software Co. distributes the development of an accounting system accustomed for Non-Exist Tech. Ltd. to different units all over the world. The company employed classical processes and approaches without having reuse insight for assets and infrastructural modules.

F. Global Single Software Development with Production Line

Figure 10 shows the case for global single software development with production line. Since GSE is used, the architecture will consist of multiple sites. The focus is on the development of a single system and as such the domain engineering process is missing. However, the development is based on the production line paradigm.

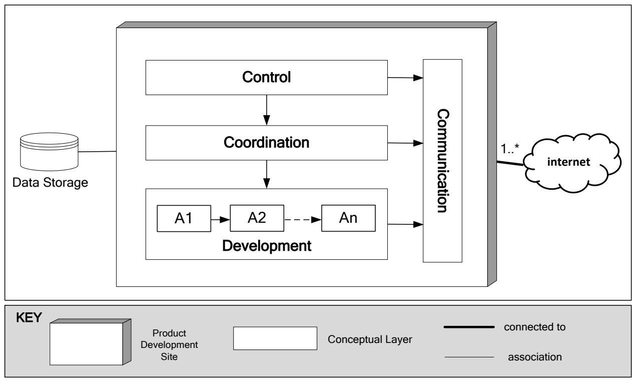

Figure 10. Global Software Development Single System Development with Production Line

Example
John Doe Software Co. has a custom distributed software production line based on SpringSource, which is installed at its business units all over the world to develop an accounting system accustomed for Non-Exist Tech. The company intentionally employed the production line to reuse infrastructural modules such as logging, content management, object to relational mapping, etc. within all its units.

G. Global Software Development for Product Line Engineering

Figure 11 represents the most difficult case for designing GSE system. It focuses on distributed development for a product line, in which the concept of production line is adopted.

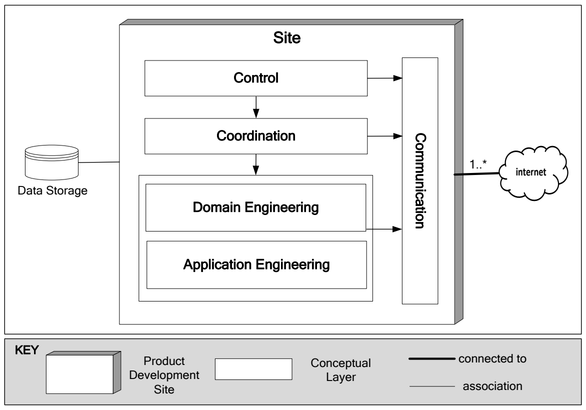

Figure 11. Global Software Development for Product Line Engineering

Example
John Doe Software Co. develops accounting products for different customers like Non-Exist Tech. by reusing the assets and managing the variability of these assets specific to accounting domain. The company distributed the development efforts of the product to different business units all over the world by using classical techniques, but not based on a production line infrastructure.

H. Global Software Development for Product Line Engineering with Production Line Engineering

Figure 12 represents the most difficult case for designing GSE system. It focuses on distributed development for a product line, in which the concept of production line is adopted.

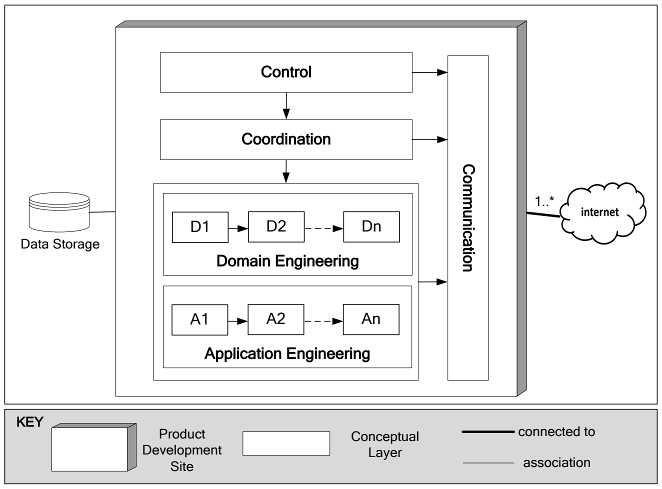

Figure 12. Global Software Development for Product Line Engineering with Production Line Engineering

Example:
John Doe Software Co. develops accounting products for different customers like Non-Exist Tech. by both reusing the assets and managing the variability of these assets specific to accounting domain. The company developed the product by using its custom distributed production line based on SpringSource, which can centrally control and monitor the whole development items and deliverables precisely. The production line based product variability management allows the reuse of business domain specific assets and infrastructural modules in a distributed way.

V. RELATED WORK

Notably, architecting in GSE has not been widely addressed. The key research focus in the GSE community seems to have been in particular related to tackling the problems related to communication, coordination and control concerns. Clerk et al. [4] report on the use of so-called *architectural rules* to tackle the GSE concerns. *Architectural rules* are defined as “principles and statements about the software architecture that must be complied with throughout the organization”. They have defined four challenges in GSE: time difference and geographical distance, culture, team communication and collaboration, and work distribution. For each of these challenges they list possible solutions and describe to what extent these solutions can be expressed as architectural rules. The work of Clerk et al. aims to shed light on *what* kind of architectural rules are necessary to guide the GSE. We consider our work complementary to this work. In our work the design actions that relate to the expected answers of questions are defined as design actions.

In a position paper of Siemens, Paulish [10] provides some guidelines about how to develop a product line using a centralized product line management team and distributed component development teams. For this, the author proposes to decompose the large-scale requirements into a well-structured set of software components that can be developed in parallel among globally distributed development teams. Likewise it is aimed to develop the product line using global software engineering practices. Further it is recommended to keep small teams that use agile processes and which are controlled by a central organization. Further, the author describes some best practices for formal requirements engineering and architecture design to develop the software components that will make up the product line. Using the approach it is aimed to reduce the time-to-market and increase productivity. The architecture as proposed by Paulish is one of the alternatives that we have defined in Table 1. In fact, Paulish focuses more on the overall process for supporting product line engineering using global software engineering. In our approach we have focused on the architectural design of global software product line engineering. We believe that both approaches are complementary to each other.

A common practice is to model and document different architectural views for describing the architecture according to the stakeholders' concerns [6][9]. An architectural view is a representation of a set of system elements and relations associated with them to support a particular concern. Having multiple views helps to separate the concerns and as such support the modeling, understanding, communication and analysis of the software architecture for different stakeholders. Architectural views conform to viewpoints that represent the conventions for constructing and using a view. An architectural framework organizes and structures the proposed architectural viewpoints. Different architectural frameworks have been proposed in the literature. Examples of architectural frameworks include the Kruchten's 4+1 view model [9], the Siemens Four View Model and the Views and Beyond approach (V&B)[6]. In our work we have defined the architecture that represents the deployment view of the system. This view appeared to be one of the most useful views since it is able to depict the multi-site character of GSE. However, we could easily consider other views such as decomposition view or uses view. We consider this as part of our future work.

VI. CONCLUSIONS

We have defined the notion of *global software product line engineering* that considers the application of product line engineering in a global development environment. Our study shows that we can in essence identify 8 possible integration alternatives of product line engineering with global software engineering. We have made a distinction between two global software product line engineering approaches: (1) GSPLE without production line and (2) GSPLE with production line. Obviously the latter GSPLE approach is the most difficult alternative but on the other hand will also lead to enhanced reuse.

The goal of this work was primarily to shed light on the challenges related to the architecture design of GSE system. The alternatives that we have shown can be used as templates for GSE architect to derive the architect for a particular project. Further, we consider this work as an initial step towards integrating product line with global software engineering. Our future work will focus on enhancing the concepts that we have discussed in this paper and applying this within an industrial context of Cybersoft.

REFERENCES

[1] R.D. Battin, R. Crocker, J. Kreidler, K. Subramanian. Leveraging Resources in Global Software Development. IEEE Software, 18(2), p. 70-77, Mar/Apr, 2001.

[2] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, Pattern-Oriented Software Architecture Volume 1 - A System of Patterns, Wiley, 1996.

[3] E. Carmel and R. Agarwal. Tactical Approaches for Alleviating Distance in Global Software Development. IEEE Software, March/April,p. 22-29, 2001.

[4] V. Clerc, P. Lago, H. van Vliet. *Global Software Development: Are Architectural Rules the Answer?* In: Proc. of the 2nd International Conference on Global Software Engineering, pp. 225–234. IEEE Computer Society Press, Los Alamitos, 2007.

[5] P. Clements, L. Northrop. Software Product Lines: Practices and Patterns. Boston, MA:Addison-Wesley, 2002.

[6] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford. Documenting Software Architectures: Views and Beyond. Second Edition. Addison-Wesley, 2010.

[7] J.D. Herbsleb. Global Software Engineering: The Future of Socio-technical Coordination. International Conference on Software Engineering. p. 188-198, 2007.

[8] J.D. Herbsleb and D. Moitra. Global Software Development. IEEE Software, March/April, p. 16- 20, 2001.

[9] P. Kruchten. The 4+1 View Model of Architecture. IEEE Software, 12(6):42–50, 1995.

[10] D. Paulish, Product Line Engineering for Global Development, Siemens AG, pp. 1-6, 2005.

[11] K. Pohl, G. Böckle, F. van der Linden. Software Product Line Engineering – Foundations, Principles, and Techniques, Springer, 2005.

[12] SpringSource Tool Suite. http://www.springsource.com/products/sts

[13] T. Stahl, M. Voelter. Model-Driven Software Development, Addison-Wesley, 2006.

[14] K. Schmid, M. Verlage. The Economic Impact of Product Line Adoption and Evolution. IEEE Software, Vol. 19, No. 4, July/August 2002, 50-57.

[15] B. Sengupta, S. Chandra, V. Sinha. A research agenda for distributed software development, In Proceedings of the 28th international conference on Software engineering, pp. 731-740, 2006.

[16] J. Whitehead, *Collaboration in Software Engineering: A Roadmap*, In FOSE '07: 2007 Future of Software Engineering, pp. 214-225, 2007.

[17] J.A. Zachman. A Framework for Information Systems Architecture. IBM Systems Journal, Vol. 26. No 3, pp. 276-292, 1987