A Rule-based Model for Customized Risk Identification and Evaluation of Task Assignment Alternatives in Distributed Software Development Projects

Ansgar Lamersdorf
University of Kaiserslautern
Kaiserslautern, Germany
a_lamers@informatik.uni-kl.de

Jürgen Münch
University of Helsinki
Helsinki, Finland
Juergen.Muench@cs.helsinki.fi

Alicia Fernández-del Viso Torre
Indra Software Labs
Madrid, Spain
afernandezde@indra.es

Carlos Rebate Sánchez
Indra Software Labs
Madrid, Spain
crebate@indra.es

Markus Heinz
University of Kaiserslautern
Kaiserslautern, Germany
m_heinz@informatik.uni-kl.de

Dieter Rombach
University of Kaiserslautern and Fraunhofer IESE
Kaiserslautern, Germany
Dieter.Rombach@iese.fraunhofer.de

Distributed software development imposes new project risks that are very different from the ones in collocated development and are overlooked easily. At the same time, they depend to a large extent on project-specific characteristics. Therefore, new methods for identifying these risks in distributed projects have to be developed. This article presents a model for identifying these risks at the beginning of a project. The model systematically captures experiences from past projects in a set of logical rules describing how project characteristics influence typical risks in distributed development. Thus, it is able to assess risks individually for each project. In addition, the model can be used for evaluating different task assignment alternatives, which makes it possible to allocate tasks systematically. An instance of the model was developed by applying qualitative content analysis to 19 interviews with practitioners. An evaluation using expert interviews showed that the risks identified by the model matched the actual experiences in 81% of the cases; of these, 40% had not been regarded at project start.

This is the peer reviewed version of the following article: Ansgar Lamersdorf, Jürgen Münch, Alicia Fernández- del Viso Torre, Carlos Rebate Sánchez, Markus Heinz and Dieter Rombach (2012), JOURNAL OF SOFTWARE: EVOLUTION AND PROCESS 2012; 24:661–675, doi: 10.1002/smr.576, which has been published in final form at http://onlinelibrary.wiley.com/doi/10.1002/smr.576/abstract. This article may be used for non-commercial purposes in accordance With Wiley Terms and Conditions for self-archiving http://olabout.wiley.com/WileyCDA/Section/id-817011.html'.

1. Introduction

The literature on distributed or global software development (GSD) is full of failure stories [1] [2] [3] caused by the inherent characteristics of GSD. In contrast to collocated software development, GSD consists of several teams that have to cooperate across various barriers (e.g., language and cultural differences, time zone barriers, different backgrounds with respect to expertise and knowledge). As a result, decreases in productivity [4] [5] or increases in the number of defects [6] have been reported. Their causes include communication problems between sites [7] [8] [9], insufficient knowledge at one of the sites [6] [10], mistrust between sites [11], or decreased workforce motivation due to the fear of job loss [12]. Therefore, they represent a set of GSD-specific project risks that might be relevant in addition to the typical project risks.

This indicates that risk management in global development should specifically address the risks caused by the distributed nature of GSD projects. In practice, however, GSD-specific risks are often not considered at project start [13]. Instead, when distributed projects are initiated, they often focus only on possible benefits such as low labor cost rates, while neglecting the problems of distributed development [14] [15].

Knowing risks and potential problems together with their typical causes already at project start would help to initiate customized countermeasures (e.g., plan in extra resources, increase trainings, or initiate travelling) and therefore reduce risks. In addition, this knowledge could also be used to make systematic decisions regarding the distribution of work to different development sites: If it is known which characteristics might cause certain problems (e.g., low expertise level, high turnover rate) and if these characteristics are known for the involved sites (e.g., the expertise level and turnover rate at each site), the decision on how to allocate work can take into account the possibility of specific problems at each site and weigh this against the potential benefits (e.g., a low labor cost rate).

In this article, we present a model for identifying and predicting GSD-specific project risks as well as its instantiation. The instantiation is based on a detailed qualitative content analysis of 19 interviews with practitioners regarding their experiences in distributed and global software development. This was done by applying systematic coding to the transcriptions of the interviews. From the interview analysis, we derived a set of rules that describe under which circumstances certain problems can occur. The rules use a set of influencing factors as independent variables that represent characteristics of the software development project environment. This allows for assessing the risks individually for any project: By setting the influencing factors according to the project-specific characteristics and the distribution of work, every rule can be evaluated and the corresponding risk can be assessed.

The remainder of the article is structured as follows. First, related work in risk identification for GSD is discussed. Section 3 presents an overview of the model concepts. In Section 4, the interview study and content analysis method used for model development are described in detail. In addition, the model evaluation within a Spanish software development company is presented. Section 5 sketches the integration of the model into an approach for systematic task allocation, followed by a discussion of the results and an outlook on future work.

2. Related Work

According to the Project Management Body of Knowledge (PMBOK) [16], risk identification addresses the question “Which risks might affect the project?” In the following, we will only focus on the risk identification aspect of risk management.

There exists a large body of research on risk management and risk identification for software development projects [17] [18] [19]. However, these approaches usually do not consider distributed and global development. Consequently, we will concentrate on specific approaches for risk identification in GSD.

Prikladnicki et al. suggest a process for risk management that is integrated into processes for distributed software development [20][21]. This approach mainly delivers a generic process without giving guidelines on how to identify the specific risks based on project and site characteristics. It can thus be seen as a generic process framework that needs to be filled with specific risk models for GSD.
Ralyte et al. present such a specific model for GSD, which includes a fixed set of risks that may occur due to the distributed nature of GSD projects [22]. Their risk framework is divided into the two dimensions distance (geographical, temporal, socio-cultural, organizational, technological, knowledge) and activity (communication, coordination, control, development, maintenance). For each combination of these two dimensions, they list specific problems that may occur in a project. For applying the framework to specific projects, the project-specific risks and solutions have to be selected from the proposed list.
A similar approach is given by Ebert et al. [23]. Here, several problems and risks that may occur in GSD projects are categorized into four drivers of global distribution: efficiency, presence, talent, and flexibility. The approach names a large number of possible problems and mitigation strategies and it is left to the user to identify which problems might occur in a specific project.
Smite [24] presents a risk identification approach that is more suited for identifying specific risks for an individual project situation. It is possible to identify project-specific risks if the individual threats for each project are known. This approach relies on very detailed historical data and does not give an explanation on why a specific threat might lead to certain problems or consequences.
In general, current research on risk identification in GSD focuses very much on providing lists of possible problems and risks while giving no explanations or rules as to which problems might occur under which circumstances or in which environments. However, this is very important in assessing project-specific risks, as significant project risks can be the result of certain characteristics and constellations: Research shows, for example, that, depending on maturity, geographical distance between sites is seen very differently by project managers, from “no problem at all” to “a major barrier” [6], and that the consequences of staff turnover depend on the type of development project [25]. Therefore, we need risk identification approaches that consider the causal relationships between project characteristics and problems and that can assess risks individually for a specific project situation.

3. Model Overview

3.1. Model Goal

Based on our previous work [25], we state the following two assumptions:
1) The specific problems and risks of distributed and global software development are often not known or underestimated at the beginning of GSD projects. 2) Most risks are not vital in all GSD projects but only under specific circumstances. 3) The assignment of work across sites in a distributed development project can have a significant impact on the risks of GSD projects. Therefore, the following goal was formulated for the risk identification model:
Goal: Develop a model that can be systematically used for the identification of risks in specific global software development projects as well as for the risk analysis of different task assignment alternatives. The model should be based on previous experiences of practitioners in distributed and global development.
As the goal is to identify project-specific risks, the model has to use the characteristics of a project environment as input for its predictions. Therefore, we decided to build the model as a set of rules stemming from interviews with practitioners in GSD and the experiences reported there. It can thus be seen as a formalized collection of lessons learned from previous projects.

3.2. General Concept

A general overview of the model concept is given in Figure 1. The main idea of the model is that unsystematic lessons learned are documented in a semi-formal way that allows for automatic evaluation. These lessons learned describe problems in GSD and the circumstances under which these problems can occur or be prevented.

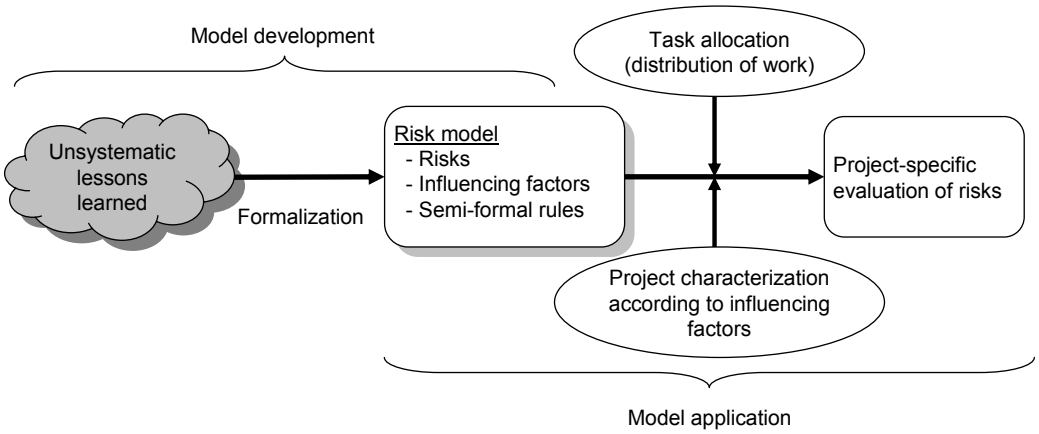

Figure 1. General model concept

The model development process consists of a transformation of the lessons learned into the risk model, which consists of influencing factors, risks, and logical rules. This transformation is done by formalizing each lesson learned as a semi-formal rule and identifying the corresponding influencing factors and risks described in the rule. Given the risk model, the model application process is able to predict the risks for a specific project by evaluating all rules of the model. In order to do so, the project is characterized according to the identified influencing factors and the analyzed task allocation.

3.3. Risks, Influencing Factors, Rules, and their Evaluation

As described above, the main elements of the model are (a) risks, (b) influencing factors, and (c) rules.

In our model, risks describe possible problems that might occur in a GSD project. For every risk, there exists a short textual description of its possible negative impact on the project (e.g., communication problems can decrease productivity).

Influencing factors describe characteristics of the project environment that have an impact on the existence of a certain problem. Influencing factors can be of different types:

- Characteristics of remote sites (e.g., the process maturity or the staff experience at the site),
- Relationships between sites (e.g., the cultural difference or the existence of previous working experience between two sites),
- Task characteristics (e.g., the complexity of a task),
- Relationships between tasks (e.g. the coupling of tasks assigned to different sites), or
- Characteristics of the overall project (e.g., the time pressure or the type of project).

Task and site characteristics can be different for every involved site and might have to be elicited for each site individually. Relationships between sites have to be determined for every combination of two sites that collaborate in a project. Based on the experience of the authors, characteristics of the product to be developed or maintained might also be relevant. We model these characteristics indirectly as task characteristics.

In our model, we concentrate on software development within one organization; thus, we only look at the characteristics of different sites that act as subsidiaries. However, the model could

also be used for evaluating risks in an outsourcing scenario (i.e., global software development between independent companies). In such a case, additional issues arising (such as legal problems [26]) could be modeled either as an additional category of influencing factors or within the existing types (e.g., the existence of certain contracts could be described as a relationship between sites).

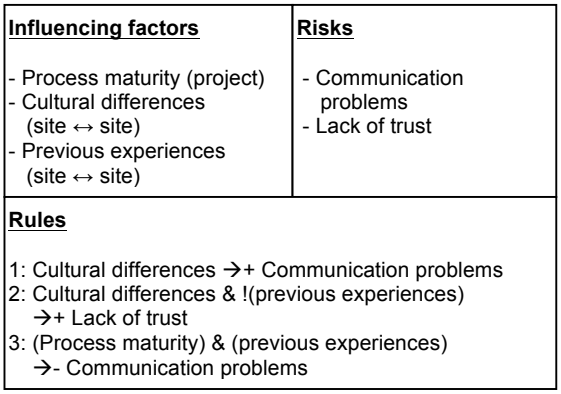

Figure 2. Exemplary model

Rules formalize how the influencing factors may impact the risks. The influencing factors in every rule can be combined using the logical operators ! ("not"), & ("and"), and | ("or"). Figure 2 shows a graphical illustration of exemplary rules. In every rule, the + operator indicates that the rule describes an increase in the risk, while the – operator describes a decrease.

The assessment of risks for a future global software development project is done in two steps: First, the project and the involved sites have to be characterized by the responsible project managers. Afterwards, the model is able to identify risks by evaluating the rules according to their project-specific relevance. Both steps are described in the following.

In order to reduce the complexity of the model, we decided to use only one ordinal five-level scale (very low – very high) for every variable, which is in accordance with other well-known estimation models such as COCOMO [27]. Thus, a project can be characterized by selecting one out of five values for every variable of the model. Project factors are assessed for the entire project; site and task factors are assessed for the involved sites; and relationships between sites are assessed individually for every two collaborating sites.

For evaluating the rules, we chose to derive a relevance value for every rule on the same five-level scale. Again, this decision was made to prevent the model from becoming too complex, which might reduce its applicability. The relevance value is calculated using two simple functions, `num` and `eval`.

The function `num` converts the ordinal value into a number and is defined as follows:

```
num(very low) = 0; num(low) = 1;...;num(very high) = 4
```

The function `eval` recursively applies Boolean logic to the rules:

```
eval(x → r) = num^-1(eval(x))
eval(a & b) = min(eval(a), eval(b))
eval(a | b) = max(eval(a), eval(b))
eval(!a) = 4 - eval(a)
eval(factor) = num(factor.value)
```

In other words, a rule is evaluated by recursively evaluating the combination of the values of its influencing factors. A logical "and" of two values is evaluated as the minimum (e.g., "factor1 & factor2 → risk X" with factor1="high" and factor2="low" has "low" relevance), a logical "or" of two values is evaluated as the maximum (e.g., "factor1 | factor2 → risk X"

with factor1="high" and factor2="low" has "high" relevance), and a logical "not" is evaluated as the complementary value (e.g., "!factor1 → risk X" with factor1="high" has "low" relevance).

Project characteristics:	**Site relations:**
Proc. mat.: medium	Cult. Diff.: high Prev. exp.: high
Rules relevance 1: Cultural differences →+ Communication problems **(Rule relevance: high)** 2: Cultural differences & !(previous experiences) →+ Lack of trust **(high & !high = high & low = low)** 3: (Process maturity) | (Previous experiences) →- Communication problems **(medium | high = high)**	

Figure 3. Assessment of influencing factors and evaluation of rule relevance for two sites

Figure 3 gives an example of the assessment of the influencing factors and the evaluation of the rules. It can be seen that rule 2 has the lowest relevance (due to high previous experiences), whereas the relevance for rules 1 and 3 is high.
Based on the rule evaluation, the rules can then be ordered according to their relevance and the project-specific risks and problems can be identified.

3.4. Evaluation of Different Task Assignment Alternatives

The example given in Figure 3 already shows how risk assessment is dependent on work distribution: If the work was assigned to a different remote site with (for example) low cultural differences but also low previous experience, the rules would be evaluated differently and other risks might have more relevance. In addition, a project might have more than two sites involved, resulting in different relevance values of each rule for every interface between two sites. For example, one site might collaborate with both a site with high and a site with low cultural differences. Thus, we use the model for analyzing different task assignment alternatives and assessing the risk for each task individually. This is done according to the following algorithm:

- *For every task T1 within the analyzed project:*
  - *For every task T2 that is assigned to a different site than T1:*
    - *For every rule R:*
      - *Evaluate R with respect to T1, T2 and the sites T1 and T2 are assigned to*

Figure 4 gives an example of how three tasks, assigned to three sites with different cultural differences, are evaluated with respect to one rule (cultural differences increase communication problems). The example shows that changing the assignment of task 2 from site 2 to site 3 would decrease the risk of having communication problems: In the current assignment (see Figure 4), the risk is of high relevance with respect to the interface between task 2 and task 3, and it is of medium relevance with respect to the interface between task 2 and task 1. If task 2 were to be assigned to site 3, the risk would have no relevance with respect to the interface between task 2 and task 3 (because both tasks would be assigned to site 3), and it would be of low relevance with respect to the interface between task 2 and task 1. (In a real-world risk model, however, other rules would have to be regarded, too).
As a result of this task-specific evaluation, a set of risks is identified for each task and each assignment individually. This can be used for analyzing different task assignment alternatives. After the task assignment decision, specific risks for every involved site can be identified by

summing up the risks for all tasks assigned to the site. These risks can then be communicated to the responsible site manager.

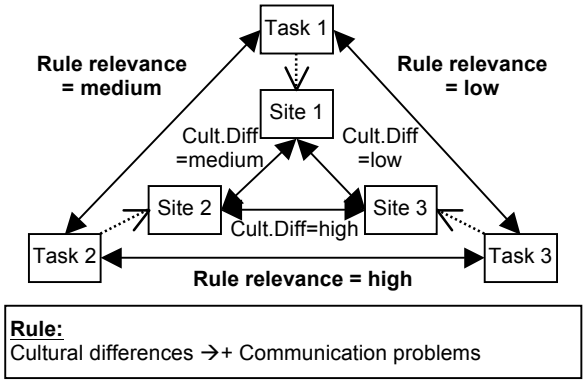

Figure 4. Individual assessment of a risk for every task in a specific assignment

3.5. Maintenance of the Model

Evaluation and feedback are of central importance in experienced-based software project management [28]. Thus, a possibility to maintain and update the model based on experiences made while applying the model should be included.

Maintenance of the model can, in principle, be performed by adding new rules, removing old rules, or changing existing rules. In addition, rules might become more credible once they were applied successfully. Therefore, we use the concept of storing experiences together with significance [29] and suggest adding a significance variable to every rule, describing its credibility. Initially, every rule has a significance of 2. After a project is finished, every rule is updated according to the following process:

- *If the predictions of the rule were correct (i.e., if rule relevance was "high" or "very high" and the risk actually occurred or if rule relevance was "low" or "very low" and the risk did not occur), increase its significance by 1.*
- *If the predictions of the rule were incorrect:*
  - *If an influencing factor not yet regarded caused the difference between actual and predicted risk, add the factor to the rule and reduce significance by 1.*
  - *Otherwise, divide the significance of the rule by 2.*
- *If the significance of the rule is below 0.5, remove the rule from the model.*

In addition, new rules describing new risks can be added after every project with a significance of 2. Figure 5 gives an example of different possibilities for updating a rule.

In this model, the significance variable has two uses: On the one hand, it can be seen as a counter that provides notification when to remove a rule from the model because correctness is too low. On the other hand, it can be used as a guideline for project managers indicating the probability of a risk actually occurring.

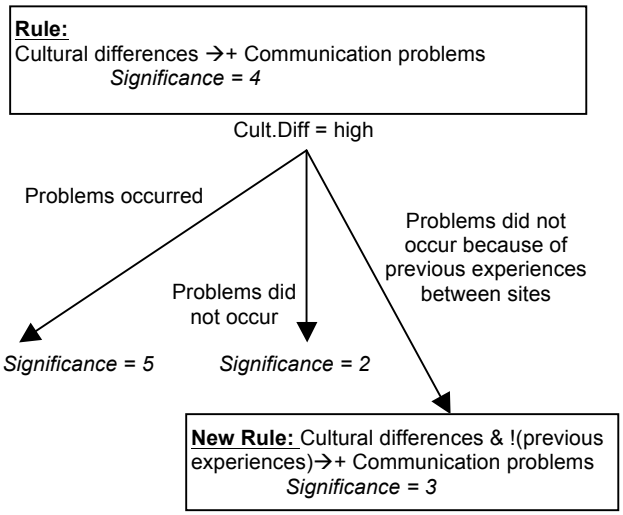

Figure 5. Model update after application of rule

The process for updating the model also implicitly handles the problem of contradictory rules: One conceivable case is that two contradictory rules are stored in the model (e.g., "time zone differences →- productivity" according to most experiences but "time zone differences →+ productivity" according to publications on follow-the-sun). In this scenario, two cases are possible: Either the contradictions are based on other influences such as the type of project, the sites or regions involved, or the technologies used (e.g., time zone differences only increase productivity if the organization has a certain maturity level), or no additional influences can be found. In the first case, the update process would lead to two new rules with additional influencing factors that are not contradictory anymore (e.g., "(time zone differences) & !(process maturity) →- productivity" and "(time zone differences) & (process maturity) →+ productivity"). In the second case, each application of the model in a new project would result in one rule increasing its significance level and the other rule decreasing it until finally one rule would be removed from the model.

4. Model Application and Evaluation

4.1. Instantiation Process

As a basis for the experiences captured in the model, we used a series of qualitative interviews with practitioners in global software development conducted between spring 2008 and fall 2009. Some of the interviews were conducted for a different study on task allocation practices in distributed development [25]. However, they also included questions on general experiences in distributed development and on factors causing problems in GSD.

In total, 19 interviews were conducted with experts from 14 different companies in the US, India, and Spain. The experts came from different domains such as aerospace, educational software, and custom software development for the financial industry. With the exception of two interviewees who reported from a researcher perspective, all of them came from management positions, with 9 being project managers and others holding positions such as quality manager, product manager, or CIO. All interviewees could report from several (up to 20) years of experience in distributed and global software development projects.

While in most cases, only one interview was conducted per company, four interviewees came from Indra Software Labs (ISL). ISL was later also used for evaluating the risk model (however, in a different interview session).

Each interview lasted for 30-75 minutes and was done (mostly) in person or over the telephone. With the exception of four interviews, all interviews were recorded and transcribed literally. For the other four, detailed notes were taken during and after the interview. This made it possible to analyze the interviews under various viewpoints.
According to the basic model, the interviews were analyzed with respect to statements on risks in distributed development, on factors influencing these risks, and on rules that describe experiences regarding how the factors impact the risks positively or negatively. This was done using qualitative analysis [30] and coding [31]: The code categories "Risk" and "Influencing factor" were created and all interviews were searched for codes that fit into these categories. Afterwards, the interviews were analyzed again and where passages containing influencing factors and risks were identified, the experiences were extracted as rules combining the influencing factors and risks. Table 1 gives an example of how an interview passage was analyzed.

Table 1. Example of text analysis

Original Passage	*"If you have a distributed team then it needs to be informed every time. If you have one single team, the management needs to inform only one team. […] the more sites, the more the number of teams that have to be coordinated"*
Identified Codes	Influencing factors: • Number of sites Risks: • Coordination problems
Textual Rule	The more sites there are, the more people have to interact with each other in order to make any kind of decision and let the others know about it
Logical Rule	Number of sites →+ Coordination problem

The first analysis of the interviews revealed a very large number of findings (42 influencing factors, 140 identified rules). Therefore, they and some of the factors and rules were removed based on the experience and evaluation of the authors and thus represent a threat to validity. However, as this was done following a defined process and documented throughout the process, the decisions were made transparent and can be traced back to the original findings in the interview transcriptions. As a result, we identified 31 influencing factors and 9 problems, which are used in 46 rules formally describing the collected experiences on problem enablers and barriers in GSD.

4.2. The Instantiated Model

In the following, a short overview of the identified model will be given. In order to make the model applicable in a specific environment, it was further customized and simplified. Table 2 shows all identified factors. They are categorized into relationships between the sites, characteristics of the site, characteristics of the task, relationships between tasks, and project characteristics.

Table 2. Identified factors

Type	Factor	Explanation
nships betwe en	Time zone difference	Differences between time zones at the sites
	Language difference	Differences in language or dialects in language (e.g., UK – India)

	Cultural difference	Differences in national or regional culture
	Personal relationships	Relationships between persons at different sites (have they met or talked personally?)
	Common working experiences	Experience of the two sites having worked together in the past
	Communication infrastructure	Quality of communication tools and network speed between sites
Characteristics of the site	Application knowledge	Expertise and knowledge in the application domain
	Technical knowledge	Expertise and knowledge regarding the technology (e.g., programming framework)
	Process knowledge	Knowledge about the development and communication processes used
	Transparency	Insight into remote site (plans, involved persons, status…)
	Staff motivation	Motivation of the staff for working on the project and in a distributed fashion
	Project experience	Experience of the personnel in similar projects
Characteristics of the tasks	Criticality	Criticality of the work (do failures threaten project success?)
	Complexity	Complexity of the work (e.g., required documentation)
	Formality of the description	Degree of formality and fine granularity of task description
	Novelty of the product	Degree of novelty of the product for the involved persons
	Process phase	E.g., requirements, coding, testing
Relationships between tasks	Coupling to other tasks	Dependency and required communication between tasks
Project characteristics	Process maturity	E.g., CMMI level (can also be seen as site characteristic – in this environment, the sites in a project were of equal maturity)
	Product size	Size of the product to be developed
	Requirements stability	Degree of change in the requirements during the project
	Number of involved sites	Number of sites that need to collaborate
	Time pressure	Pressure on people working on the project

An example of the identified rules was already given in Table 1. The complete set of identified rules is presented in the appendix together with a textual description of each rule. Based on the identified factors and rules, the model was implemented in Microsoft Excel.

4.3. Evaluation

In the following section, we describe the evaluation of the prototype model with respect to the context and evaluation process, the results, and the threats to validity.

4.3.1 .Context and Evaluation Process

The model was evaluated at Indra Software Labs (ISL), where four of the interviews for building the model were conducted (see Section 4.1). For the evaluation, we did not use the individual application of the model to different tasks and assignment alternatives as described in Section 3.4. Instead, we applied the model only to projects with one local and one remote site in order to simplify the evaluation process.

ISL is the network of software labs of Indra that develops customized software solutions for Indra's markets. It has 20 development sites, half of which are located in Spain and the others

in Latin America, Slovakia, and the Philippines. Most of the software development projects at ISL are distributed either within Spain or globally. Therefore, there exists a lot of experience at ISL regarding working in GSD projects and related risks and problems.
We evaluated the model in interview sessions with five practitioners at ISL. Four of the interviews were conducted in person at an ISL site in Madrid, while the last one was done during a videoconference with the interviewee being located at Ciudad Real, Spain. The persons interviewed for the evaluation were different from the ones interviewed for the model development (see Section 4.1) and reported from different projects. There was thus no threat to validity, which might have arisen from using the same experiences for model development and evaluation.
Of the five interviewees, three were project managers, one was a director at ISL (responsible for one business area), and one was working in the quality department. From their perspective, they all had insights into various distributed projects in different constellations and had several years of experience. They were thus highly experienced in distributed software development.
The evaluation process was done as follows:

1. A questionnaire was sent to the interviewees in advance, asking them to recall one specific historical distributed development project. For this project, they were asked to characterize it and identify values for the 23 influencing factors.
2. In the interview, the model was used and all factors were set to the values named by the interviewees in the questionnaire. This resulted in an evaluation of the 36 rules with respect to the project characteristics.
3. Every rule that was identified as relevant (where relevance was either “high” or “very high”) by the model was presented to the practitioner (both as a logical rule and with the textual description) asking (a) whether the rule complied with the project experience (i.e., if its described impact on risks and problems could be observed) and – in the event the rule did comply – (b) whether this rule was known at project start (i.e., if the project manager was aware of the phenomenon described by the rule).
4. Finally, the interviewees were asked if they found the use of such a model helpful and whether they would like to use it for future projects.

4.3.2. Evaluation Results

Table 3 shows the results of the evaluation. It shows that on average, one third of the 36 rules were relevant for each historical project. This indicates again that only a subset of the phenomena and problems of GSD described in the literature can be applied to a specific distributed development project.
A wide majority of the rules (81.4%) that were predicted as relevant could actually be observed in the projects. However, some rules were identified as irrelevant, as they could not be observed in most of the projects: Rule 11 stated that a certain product size decreases the risk of losing intellectual property. This could not be confirmed by the practitioners because in their opinion, loss of intellectual property was never an issue at ISL, independent of project size. Rule 32 stated that if the coding phase is transferred to another site, project risks will decrease, since coding tasks usually come with very detailed specifications. This could not be confirmed, as the practitioners could report about various problems that also occurred when coding was transferred to another site. With the exception of these two rules, nearly all relevant rules could be confirmed in each of the five projects.
Of the rules that complied with the real project experience, 59.5% were considered at project start by the project management. This means that the project managers were aware of a majority of the experiences stored in the model. One reason for this might be the fact that the interviews were conducted with highly experienced project managers who were aware of most of the risks and problems in distributed development and were able to incorporate these

experiences into their project planning. In less experienced environments, this number would therefore presumably have been lower.

Table 3. Results of evaluation

Project No	# relevant rules	# rules confirmed (from the relevant ones)	# rules considered at project start (from the ones observed)
1	14	12	8
2	16	12	5 out of 6 [1]
3	10	9	7
4	9	6	2
5	10	9	3
∑	**59**	48	25 out of 42[1]
		81.4%	**59.5 %**

[1] The quality manager did not know for all rules if they were considered at project start or not.

However, a rate of 40.5 % still demonstrates that a significant proportion of the experiences stored in the model were not systematically regarded at project start. In these cases, an application of the model at project start would probably have helped, as it would have drawn attention to the described risks and made it possible to consider them in project management and initiate countermeasures.

This hypothesis was also supported by the practitioners' answers to the applicability of the model: All of the five interviewed persons stated that they found the model useful and would like to use it in future projects. Even the managers who had already considered most of the experiences stored in the model (e.g., in projects 1 and 3) found the model very helpful: They reported that it was sometimes difficult for them to formulate their experiences and their predictions about possible risks and problems in meetings and discussions with other managers. In their opinion, such a model would help them demonstrate and communicate their experiences to others. Other managers stated that the model could also be used to identify and demonstrate project risks during project planning sessions with a customer.

Another advantage that was pointed out by one interviewed manager was the fact that this model can be used for evaluating different allocation scenarios: By inserting the characteristics of different remote sites into the model and assessing the predicted experiences and risks, the decision on how to select one site from a number of different sites for a project could be supported.

4.3.3. Threats to Validity

A threat to the internal validity might be the fact that the interviewees did not understand the rules correctly while applying them for their projects. However, this threat was reduced by explaining every rule to the practitioners.Conclusion validity is relatively low due to the small number of analyzed projects. To obtain higher significance, a larger study should follow. However, the results seem to indicate a general trend, as the degree of compliance (81%) is relatively high. Construct validity might be threatened by the fact that the evaluation was conducted by the same person who developed the model and the interviewees might have been biased towards giving pleasant answers. Particularly, the question of whether the interviewees would like to use the model in later projects might have produced biased results. However, the significant rate of rules not considered at project start (40.5%) supports the usefulness of the model. External validity might be threatened by the fact that all evaluation was done within one company. Therefore, it should be repeated at different organizations.

However, as most of the interviews for model development (15 out of 19) were done in companies other than ISL, the evaluation results can probably be generalized.

5. Integration of the Model into an Approach for Systematic Task Allocation

In Section 3.4, we already described how the model can be used for evaluating different task assignment alternatives as part of a systematic task allocation decision. It is the overall goal of our current research to support task allocation decisions using experiences from previous projects that are systematically stored in models. During this research, we developed two other models for decision support: an assignment suggestion model [32] [33], and an effort overhead model [34]. Describing the other two models in detail is beyond the scope of this article. However, in a different publication [35], we describe how these models as well as the risk model can be integrated into one coherent approach for systematic task allocation.
In order to demonstrate the use of this approach, Figure 6 shows a scenario of a task allocation decision in a GSD project: In this project, five tasks (i.e., the development of five sub-components) are to be assigned to sites in Cologne, Frankfurt, London, and Bangalore. While component 1 is already assigned to Frankfurt and component 5 is assigned to London, it has to be decided where components 2 to 4 shall be assigned to.

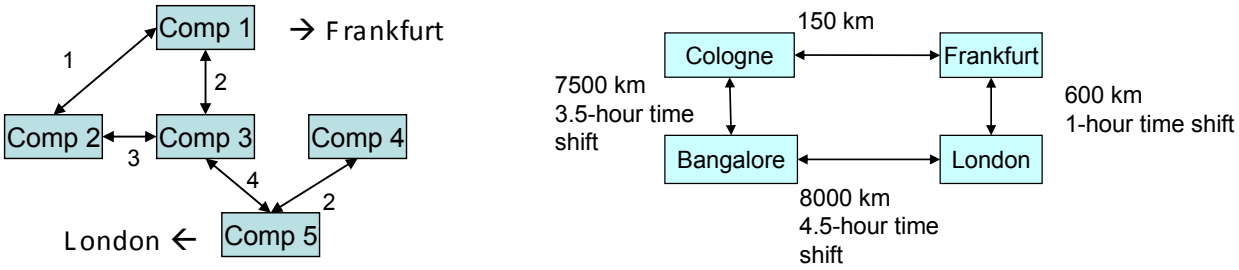

Figure 6. Tasks and sites of the example scenario.

Based on the project characteristics, the assignment suggestion model (which applies the weighted rules of the risk model) suggests the assignments shown in Table 4 (see [33] for details on the suggestion algorithm), and the cost estimation model predicts effort and cost for each assignment alternative. Due to the predictions of effort and cost, the decision maker decides to follow the second assignment suggestion and therefore assigns components 2 and 3 to Frankfurt while assigning component 4 to Bangalore.
The remaining risks can now be predicted by the risk model. Table 5 gives an overview of the risks predicted for component 1 in the selected assignment. It shows that the risk with the highest relevance is an increase in communication problems with respect to the interface between component 1 and component 4. This is caused by the language differences between Frankfurt (where component 1 is assigned to) and Bangalore (where component 4 is assigned to). The same rule is relevant with respect to the interface between component 1 and component 5, but here the relevance is lower because component 5 is assigned to London and the language differences between London and Frankfurt are not as high as the language differences between Bangalore and Frankfurt.

Table 4. Suggested assignments with cost and effort predictions

No	Suggested assignment					Effort	Cost
	1	2	3	4	5		
1	F	B	B	B	L	366	1,742,464

2	**F**	**F**	**F**	**B**	**L**	**230**	**1,375,059**
3	F	F	B	B	L	275	1,526,840

Table 5. Risks for component 1

Relevance	Rule	Other involved Task
Very high	Language differences →+ communication problems	Component 4
High	Language differences →+ communication problems	Component 5
High	Time zone differences →- motivation	Component 4
High	Time zone differences →- productivity	Component 4

6. Conclusions and Future Work

In this article, we presented a model for assessing project-specific risks and problems of distributed software development. The model was described in its basic concept, its instantiation based on a systematic qualitative analysis of 19 interviews with practitioners, and its evaluation at Indra Software Labs.

The evaluation showed that the model was able to make predictions that complied with the experiences in historic projects and that its applicability in practice was strongly supported by highly experienced managers.

However, the model as presented in this article still has some deficiencies that can be the basis for future work:

1) While the model already provides a relatively clear definition of the concept of "influencing factors" and categorizes them into four groups, the risks and problems are not yet specified on a detailed basis.

2) The current set of rules can be improved.

3) It is not clear if and how the current set of rules can be reused by other organizations.

4) Time considerations are difficult to address with the model. It is not clear how the risk model can be used for analyzing and predicting the impact of work distribution and task allocation on the duration of the project or on the probability that deadlines will be met.

Acknowledgements

The authors would like to thank all participants in the interview and evaluation studies. Some of the work was done during a stay at the Fraunhofer Center for Experimental Software Engineering, Maryland and was financially supported by the Otto A. Wipprecht Foundation. The authors also thank Sonnhild Namingha for proofreading the paper.

References

[1] Heeks R, Krishna S, Nicholson B, Sahay S. Synching or sinking: Global software outsourcing relationships. *IEEE Software* 2001; 18(2): 54-60

[2] Seshagiri G. GSD: Not a business necessity, but a march of folly. *IEEE Software* 2006; 23(5): 63-64

[3] Fabriek M, Brand M, Brinkkemper S, Harmsen F, Helms RW. Reasons for success and failure in offshore software development projects. *European Conference on Information Systems* 2008; 446-457

[4] Herbsleb JD, Mockus A. An empirical study of speed and communication in globally-distributed software development. *IEEE Transactions on Software Engineering* 2003; 29(6): 481-494

[5] Espinosa A, Slaughter SA, Kraut RE, Herbsleb JD. Familiarity, Complexity, and Team Performance in Geographically Distributed Software Development. *Organization Science* 2007; 18(4): 613–630
[6] DeLone W, Espinosa JA, Lee G, Carmel E. Bridging Global Boundaries for IS Project Success. *38th Hawaii International Conference on System Sciences* 2005; 48.2
[7] Lindqvist E, Lundell B, Lings B. Distributed development in an intra-national, intra-organizational context: An Experience Report. *International workshop on global software development for the practitioner*, 2006
[8] Alami A, Wong B, McBride T. Relationship Issues in Global Software Development Enterprises. *Journal of Global Information Technology Management* 2008; 11(1): 49-68
[9] Kommeren R, Parviainan P. Philips experiences in global distributed software development. *Empirical Software Engineering* 2007; 12(6): 647-660
[10] Herbsleb JD, Paulish DJ, Bass M. Global software development at Siemens: Experience from nine projects. *International Conference on Software Engineering (ICSE)* 2005; 524-533
[11] Smite D, Moe NB. Understanding a Lack of Trust in Global Software Teams: A Multiple-Case Study. *International Conference on Product Focused Software Development and Process Improvement PROFES* 2007; 20-34
[12] Casey V, Richardson I. Uncovering the reality within virtual software teams. *International workshop on global software development for the practitioner*, 2006
[13] Betz S, Mäkiö J. Amplification of the COCOMO II regarding offshore software projects. *Workshop on offshoring of software development-methods and tools for risk management at the second International Conference on Global Software Engineering* 2007
[14] Bass M, Paulish D. Global software development process research at Siemens. *Third International Workshop on Global Software Development*, 2004
[15] Keil P, Paulish DJ, Sangwan R. Cost estimation for global software development. *International Workshop on Economics Driven Software Engineering* 2006; pp. 7–10
[16] Project Management Institute. *A guide to the project management body of knowledge: PMBOK guide*. 3rd edition, Project Management Institute, Newton Square, Pennsylvania, USA: 2004
[17] Boehm BW. Software risk management: principles and practices. *IEEE Software* 1991; 8(1): 32–41
[18] Fairley RE. Risk management for software projects. *IEEE Software* 1994; 11(3): 57-67
[19] McManus J. *Risk Management in Software Development Projects* 2004, Butterworth-Heinemann Publishing: Oxford, UK
[20] Prikladnicki R, Yamaguti MH. Risk management in global software development: A position paper. *Third International Workshop on Global Software Development* 2004
[21] Prikladnicki R, Yamaguti MH, Antunes DC. Risk management in distributed software development: A process integration proposal. *5th IFIP Working Conference on Virtual Enterprises at 18th IFIP World Computer Congress* 2004; 517-526
[22] Ralyte J Lamielle X Arni-Bloch N Leonard M. A framework for supporting management in distributed information systems development. *Second International Conference on Research Challenges in Information Science, RCIS* 2008: 381-392
[23] Ebert C, Murthy BK, Jha NN. Managing risks in global software engineering: principles and practices. *International Conference on Global Software Engineering* 2008; 131-140
[24] Smite D. Project outcome predictions: Risk barometer based on historical data. *International Conference on Global Software Engineering* 2007; 103-112
[25] Lamersdorf A, Münch J, Rombach HD. A survey on the state of the practice in distributed software development: Criteria for task allocation. *Fourth International Conference on Global Software Engineering* 2009; 41-50
[26] Kobitzsch W, Rombach HD, Feldmann RL. Outsourcing in India. *IEEE Software* 2001; 18(2): 78-86

[27] Boehm B, Abts C, Brown A, Chulani S, Clark B, Horowitz E, Madachy R, Reifer D, Steece B. *Software Cost Estimation with COCOMO II*, Prentice-Hall: New York, 2000
[28] Basili VR, Caldiera G, Rombach HD. Experience factory. In J. Marciniak, *Encyclopedia of Software Engineering*, volume 1, 1994; 469-476.
[29] Heidrich J, Münch J, Riddle W, Rombach HD. People-oriented Capture, Display, and Use of Process Information. *New Trends in Software Process Modeling, Series on Software Engineering and Knowledge Engineering*, volume 18, World Scientific Publishing Company, 2006, pp. 121-179
[30] Gibbs G. *Analyzing Qualitative Data (Qualitative Research Kit)*, Sage Publications: Los Angeles, 2008
[31] Seaman C. Qualitative methods. in F. Shull et al., *Guide to Advanced Empirical Software Engineering*, Springer: Heidelberg 2008: pp. 35-62
[32] Lamersdorf A, Münch J, Rombach HD. Towards a Multi-Criteria Development Distribution Model: An Analysis of Existing Task Distribution Approaches. *Third International Conference on Global Software Engineering* 2008; 109-118
[33] Lamersdorf A, Münch J, Rombach HD. A decision model for supporting task allocation processes in global software development. *International Conference on Product Focused Software Development and Process Improvement PROFES* 2009; 332-346
[34] Lamersdorf A, Münch J, Rombach HD. Estimating the Effort Overhead in Global Software Development, *Fifth International Conference on Global Software Engineering* 2010
[35] Lamersdorf A, Münch J. Model-based Task Allocation in Distributed Software Development. *Fourth International Conference on Software Engineering Approaches for Offshore and Outsourced Development* 2010: 37-53

Appendix: Identified rules

Table 6. Identified rules (Excerpt)

	Logical rule	**Textual description**
4	Time zone difference & (cultural difference | language difference) → +Coordination problems	When there is a time zone difference, there remains little time to communicate with each other. Having language and cultural differences in addition makes this even worse because the little time available can't be used efficiently when there is repeated mutual misunderstanding.
14	!(Process knowledge) & size → +Communication problems	A bigger project needs more communication and coordination. If there is a manager without experience in managing and coordinating a project correctly, there are a lot more problems in communication.
18	!(Requirements stability) & (!(communication infrastructure) | !(maturity)) → +Coordination problems	If there are no stable requirements and a requirement changes, this change has to be communicated. This is not easily possible if there is no maturity or no good communication infrastructure between sites.
23	!(Communication infrastructure) & (!(personal relations) | time zone difference) → +Communication problems	When there are bad tools and no personal relationships or a time zone difference, nobody wants to communicate because it costs too much time or is too difficult, and all that for the price of talking to a stranger who isn't trustworthy.
34	Time pressure & !(personal relations) → +Communication problem	If people are under pressure, they focus more on their work and are less willing to communicate. This is aggravated by a large distance and the lack of trust. So it is even more unlikely for them to communicate with the other site.