Reference:
Ansgar Lamersdorf, Jürgen Münch. Model-Based Task Allocation in Distributed Software Development. In Proceedings of the 4th International Conference on Software Engineering Approaches for Offshore and Outsourced Development (SEAFOOD), volume 54 of Lecture Notes in Business Information Processing, pages 37-53. Springer Verlag, 2010.
DOI: 10.1007/978-3-642-13784-6_5
URL: http://link.springer.com/chapter/10.1007/978-3-642-13784-6_5

Model-Based Task Allocation in Distributed Software Development

Ansgar Lamersdorf[1] and Jürgen Münch[1,2]

[1] University of Kaiserslautern
[2] Fraunhofer IESE
a_lamers@informatik.uni-kl.de,
Juergen.Muench@iese.fraunhofer.de

Abstract. Task allocation is one central aspect in planning and managing global software development projects. To date several models that support task allocation have been proposed, including cost models and risk-based approaches. However, systematic integration of such models and a guiding process for task allocation activities is widely missing. In this article, we integrate existing models that reflect different viewpoints and abstraction levels of task allocation decisions. Based on the integrated approach, we sketch a process for systematic evaluation and selection of task assignments that defines the model interfaces and the sequential order of their use. In detail, the approach presented here integrates a risk model that is able to identify the possible risks for each assignment individually, an optimization model that uses Bayesian networks to suggest assignment alternatives with respect to multiple criteria, and an effort overhead model that is able to estimate the project effort for each assignment alternative. All three models are significantly grounded in empirical studies. Besides the introduction of all three models and the description of the process, the article provides an example application, sketches related work, and presents an overview of future work.

1 Introduction

Every software development project – be it in offshoring or outsourcing – that involves work from different, distributed sites requires a decision about task allocation: How can the individual tasks that together constitute the development project be distributed across the available sites?

There are many different ways to allocate tasks to different sites, especially with respect to the division of work into different tasks and with respect to criteria for deciding where tasks will be enacted: The work can be divided by process steps, along architectural components, by functionality, or using combinations of these models [1] [2]. In an extensive empirical study on the practice of distributed development, we identified three main criteria for task allocation: labor cost rates, availability of people, and expertise [3]. The cost rate, in particular, is often reported as a major criterion for task allocation: In global development, work is mainly assigned to those sites that have the lowest labor cost rates [4] [5].

M. Nordio et al. (Eds.): SEAFOOD 2010, LNBIP 54, pp. 37–53, 2010.

However, despite the differences, we observed that, in general, the assignment of work is done rather unsystematically in practice, without any specific process or defined decision procedure [3]. The criteria for work allocation are not defined systematically and are often just applied by the responsible project manager based on personal experiences. This highly increases the risks of distributed development projects, as there are many problems in global software development that are influenced by the task allocation decision: Language, cultural, and time zone differences are just some examples of barriers between sites that may cause problems such as decreased productivity [6] and increased lack of trust [7]. Assigning work to sites with low differences reduces these barriers and the resulting problems (often referred to as "nearshoring" [8]). Other problems such as high turnover rates [9] and little knowledge of the customer or application domain are also influenced by assigning work to different sites. Together with the already mentioned criteria for work allocation (labor cost rates, availability, expertise) this shows that the task allocation decision is highly complex and should take into account multiple criteria and influencing factors in order to reduce or avoid the problems immanent in distributed and global software development (GSD).

There are different perspectives under which a task allocation decision could be regarded: From a risk management perspective, there are various risks specific to GSD that are influenced by the work assignment (e.g., assigning work to sites with large cultural differences increases the risk of mistrust between sites). Thus, any given task allocation alternative can be evaluated under the perspective of the potential risks it creates. From the perspective of a project planner, one specific task allocation has to be selected from the (potentially large) pool of alternatives, taking into account multiple, sometimes conflicting, goals and influencing factors. Finally, the cost perspective requires special attention in GSD, as low labor costs are one driving factor for global development. However, since the distribution of work has an impact on both labor cost rates and productivity, any task allocation alternative should also be evaluated under the perspective of effort and cost estimation.

While all three perspectives are clearly different, they all regard the same decision and phenomena. Thus, they underlying influencing factors and causal relationships are common to all perspectives. In this article, we present an approach for systematic task allocation that includes models for all three of these perspectives and integrates them into one process for evaluating and selecting task allocation alternatives. The individual models have already been published separately [10], [11], [12], [13] but have not yet been integrated into one decision process as presented here.

The remainder of this article is structured as follows: Section 2 gives an overview of related work in task allocation decision support from different perspectives. Section 3 explains the task allocation process by briefly introducing the three sub-models and then presenting the process that integrates all models into one task allocation decision. Section 4 gives an example of how the process can be applied to an industrial context. Finally, Section 5 concludes the article and gives an overview of future work.

2 Related Work

Task allocation in global software development has been the focus of several research approaches and has already been analyzed in detail in another publication

[14]. We will thus just briefly highlight some related work in this area. There exist some research papers on the processes and criteria currently applied in practice that are based on empirical studies and observations [1] [15] [3]. However, they do not or only rudimentarily provide decision support for future GSD projects.

Few approaches have been developed in research that explicitly provide decision support in task allocation. Mockus and Weiss [2] developed an algorithm that identifies pieces of work (in this case, the responsibility for modules of the software) as candidates that should be reassigned from one site to another. The criterion for the assignment is minimization of cross-site communication as this highly influences distributed development overhead. Other criteria and influencing factors (e.g., cost rates, expertise, barriers between sites) were not regarded, however.

Setamanit et al. [16] [17] suggested a simulation model that is able to compare different task allocation strategies with respect to productivity and development time. It uses different influencing factors as input, such as familiarity, work coupling, and site-specific productivity. However, the model is rather designed for evaluating general strategies than for giving specific decision support for individual projects.

There are some other research approaches that aim at supporting task allocation in GSD projects, for example by developing an index for comparing coordination between sites [18], describing the work of a central coordination team [19], or defining a reference process for project planning in GSD [20]. However, none of them provides a comprehensive approach for task allocation in distributed projects.

From a risk management perspective, there exist different research approaches that try to assess risks specifically for distributed development projects. Ralyte et al. [21] developed a framework for potential risks in distributed and global software development. The risks are categorized along the two dimensions distance and activity. The framework only names possible risks and mitigation strategies; it does not say under which circumstances the risks might occur, nor does it describe the impact of task allocation on project risks.

Similarly, Ebert et al. [22] name risks and problems that are likely to occur in GSD projects together with mitigation strategies. This approach, too, does not name the impact of work allocation on risks.

Smite [23] developed a risk barometer that can help to identify project-specific risks. Thus, this approach is less generic and can be customized to individual projects. However, the described risks are again not impacted by task allocation.

In general, these risk management approaches show that the impact of task allocation on project risks has not yet been the focus of research. However, many risks described in the literature can be traced back to the barriers between sites and thus are clearly dependent on the decision about which sites work should be assigned to.

From an effort and cost estimation perspective, there also exist approaches that try to describe the cost overhead of distributed development. They are usually based on COCOMO2 [24], as this cost model is widely known and accepted.

One approach followed by different researchers [5] [25] is the extension of COCOMO2 using new cost drivers that specifically address distributed development. This includes drivers that are dependent on task allocation, such as cultural differences or the outsourcing maturity of the vendor. However, this does not take into account that many of the already existing cost drivers of COCOMO (e.g., programmer capability) might be different for every involved site and cannot be regarded globally (as it is done in COCOMO).

This problem is addressed by Madachy [26]. In this model, the original model of COCOMO is split up into several sub-models, which describe the characteristics of every involved site individually and which are then aggregated. Thus, the approach can support task allocation decisions by giving cost estimations that depend on the characteristics of the involved sites. However, it does not regard the overhead resulting from the barriers between sites and its impact on effort and costs.

In general, the existing cost estimation approaches do not regard all relevant aspects of distributed development, namely the different characteristics at the involved sites and the overhead due to distributed collaboration. Thus, they cannot be used to evaluate different task assignment alternatives with respect to the expected effort and costs. In addition, none of the analyzed approaches provides a sound empirical basis for the selection and quantification of the used cost drivers.

Looking at the related work from the three perspectives task allocation suggestion, risk identification in task allocation, and effort estimation, it can be seen that even though there exist several approaches in all of these areas that can support selected aspects of task allocation in GSD projects, there exists no comprehensive approach that can be used for systematic task allocation. This underlines the need for an approach that integrates all the different perspectives in order to provide decision support for task assignment.

3 Process Overview

In this section, the approach for task allocation decision support will be introduced. It consists of three different models, each regarding a specific perspective. Figure 1 gives an overview of the models together with the input needed for model development and the interfaces between them. In the following, we will present all three models individually, followed by a definition of the overall process for using the three models in task allocation decisions.

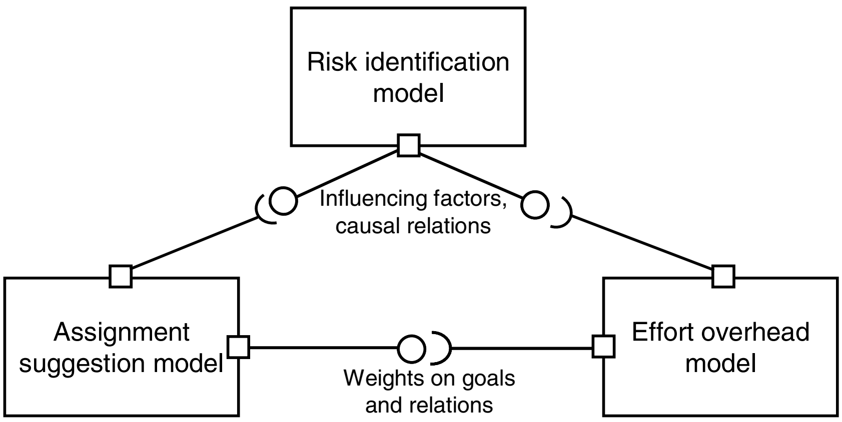

Fig. 1. Overview of models and relationships in model development

3.1 Risk Identification Model

The risk identification model is able to identify specific risks and problems of distributed development based on a characterization of the individual project and its involved sites. Using experiences from past projects as input, the model consists of logical rules describing potential threats and the influences that cause these risks.

Table 1 gives an example of how an experience (for example collected in interviews with experienced project managers) can be transferred into a rule describing the impact of an influencing factor on a project risk.

Table 1. Development of risk model based on collected experiences

Collected experience	*"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"*
Influencing factors and risks	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 considered project risks stem from the collected experiences but can be further specified and refined based on individual project goals and priorities.

As a result, the model consists of a set of influencing factors in distributed software development and logical rules that describe how these factors affect project risks (and thus, indirectly, project goals). The influencing factors are categorized into (1) characteristics of the project, (2) characteristics of the involved sites, (3) relationships between the involved sites, and (4) characteristics of the tasks assigned to the sites.

The model can be used twofold: On the one hand, it can be used for evaluating GSD projects and task assignment alternatives. Based on the characteristics of the involved sites (and therefore, indirectly, on the assignment decision), the rules predict different threats and problems that are likely to occur in a project and can help to compare different alternatives. On the other hand, the identified influencing factors and causal relationships are used as a basis for the subsequent models.

A prototype of a risk identification model has already been developed based on the analysis of 19 qualitative interviews with experienced practitioners in GSD. The created model was then evaluated at a Spanish software development company by comparing the predictions with the experiences from past projects. The result indicated that over 80 percent of the predicted risks had actually occurred in past projects. In addition, a group of five experienced managers all judged the model as very helpful for project management in future GSD projects.

3.2 Assignment Suggestion Model

The assignment suggestion model is able to identify assignment alternatives from the pool of all possibilities. This is done based on an internal Bayesian network model and the project characterization and goal weights done by the responsible project manager. It is described in detail in several other publications [14] [10] [11] [12]. Bayesian networks were chosen for different reasons: On the one hand, they are able to describe causal relations under uncertainty which, in our view, reflects the inherent nature of human behavior. On the other hand, they possess a graphical representation that makes it very easy for other persons (such as project managers) to understand the modeled concepts and causal relationships. Therefore, they have been used for software development project management in several occasions [27].

The model builds on a Bayesian network that describes influencing factors (again categorized into characteristics of tasks, sites, and the projects and relationships between sites), project goals (e.g., cost, time, quality), and causal relationships that model how the project goals are affected by the influencing factors. These relationships are weighted according to their relative impact. As Bayesian networks describe statistical relationships between factors, the relationships are able to cover the uncertainty in modeling human behavior. Figure 2 gives an excerpt of a Bayesian network used in the model. The network consists of variables (influencing factors or project goals), depicted as ovals, and causal relationships between them.

Using this Bayesian network, the model is able to assess the outcome of an assignment alternative with respect to weighted project goals. We implemented an algorithm that is able to automatically evaluate all possible assignments of tasks to sites for a given project and, as a result, rank them according to their expected fulfillment of the weighted goals [10]. This result thus represents a ranked list of assignment suggestions with respect to individual project characteristics and goals.

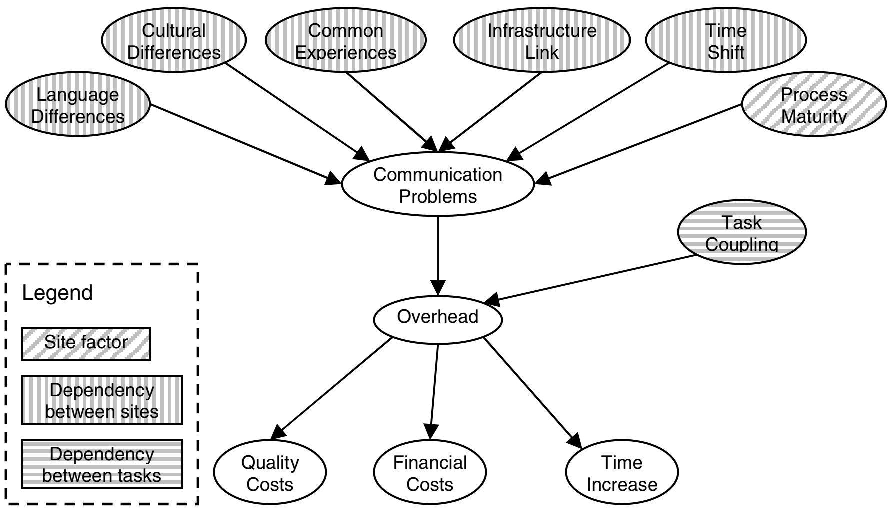

Fig. 2. Excerpt from Bayesian network model [10]

The underlying Bayesian network model can be easily modified and adapted to individual organizational contexts. For this, the results of the risk identification model can be used as input: In the risk identification model, influencing factors and causal relationships have already been defined. Thus, it remains for the Bayesian network to define how the identified risks affect project goals and to weigh the causal relationships against each other. This should again be done based on the estimations of experienced practitioners.

We developed a first prototype of the assignment suggestion. The underlying Bayesian network was developed based on an intensive literature study [12] and refined in interviews with nine highly experienced practitioners [10]. As a result, we implemented it as a Java tool, which has been presented in different publications [10] [11].

3.3 Effort Overhead Model

The effort overhead model is able to predict project effort based on a specific assignment decision and can thus be used to compare and evaluate assignment alternatives with respect to the expected project costs. It relies on the assumption that most software development organizations are able to estimate the effort for collocated projects effectively (e.g., using standard cost models like COCOMO or expert estimations), but experience unexpected overhead in distributed development projects. Consequently, the model aims at assessing the effort overhead that occurs in GSD projects.

We assume that the effort overhead in GSD is caused by two main phenomena that were already considered in existing effort models for GSD: First, there are different abilities (e.g., with respect to programmers' skills) at the involved sites [26]. This can be described using characteristics of the sites and the tasks assigned to these sites. Second, there are additional overhead drivers that are caused by the distances between sites [5] and that mainly depend on the relationships between sites. The effort overhead is thus influenced by the same factors and causal relationships as the overhead already identified in the other models.

For evaluating the effort overhead, the influencing factors and effort drivers have to be quantified. For this, we selected the CoBRA process [28] [29], which describes how a cost model can be developed based on a causal model and systematic expert quantifications. In a different publication [13], we show how the effort model can be used to assess and compare different assignment alternatives. By multiplying the predicted effort per site with the site-specific labor cost rates, the model is also used for evaluating task assignments with respect to overall project costs.

The influencing factors and causal relationships can be reused from the risk identification model (however, only those factors and relationships that influence productivity should be considered here). In addition, the weights identified in the assignment suggestion model can be used as a starting point for quantifying the impact. Thus, the effort overhead model should be developed last.

We also developed a prototype for the effort overhead model using expert estimations at a Spanish software development company. With an initial set of influencing factors taken from risk model development, we had the practitioners select the most important factors, develop a causal model, and quantify their impact on effort overhead. Figure 3 shows the causal model developed in this study. The resulting model will be used in future evaluations at the company.

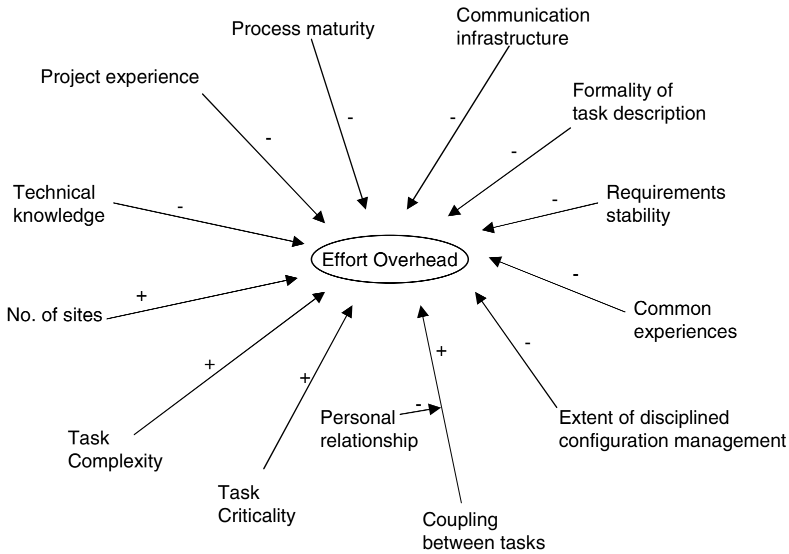

Fig. 3. Causal model for effort overhead

3.4 Integration of the Models

The three models described in the previous section have several commonalities: They all operate on a project management level and use the concept of influencing factors that impact the outcomes of a global software development project. However, at the same time, they all regard different perspectives and thus are important for making a systematic task assignment decision: The risk model can analyze existing and potential assignment alternatives with respect to project risks and problems in distributed development. Therefore it can not only support task assignment but also risk management for GSD projects. The assignment suggestion and the effort overhead model help identifying task assignment alternatives while considering multiple influencing factors and criteria.

As the task assignment and the effort overhead model both aim at the same goal, it could be argued to use only one of the models for task allocation. However, we see both of them as important: One the one hand, a systematic task allocation process should be able to regard multiple goals and measures (e.g., cost, time, or quality – but also other goals such as workforce motivation) which can only be ensured by a generic task assignment model. On the other hand, effort and productivity are a central aspect in software project management and effort models are well-known in software development practice and research [26] [5]. These models should be adapted to GSD and integrated into task allocation, as done in the effort overhead model.

As all models have influencing factors and causal relationships in common and should be used for a systematic task allocation process, we integrated them into one approach for systematic task allocation. This was mainly done following two principles:

1. The models are built on each other: Both the assignment suggestion and effort overhead model use the results of the risk model development as input for their underlying causal models. The risk model can be derived easily from lessons learned as it operates on a textual basis. During risk model development, influencing factors, goals, and causal relationships are identified. These are enhanced into a causal model that is the basis for the other two models. In addition, the weights identified during the assignment suggestion model development are the foundation for the quantification of the effort overhead model.

2. The assignment selection is done from different perspectives: The task allocation is done based on the suggestions made by the assignment suggestion and effort overhead model. This assignment is then analyzed using the risk model, ensuring that the task assignment decision is done systematically and regarding multiple perspectives.

In the following, the assignment selection process is described in detail.

3.5 Assignment Selection Process

Figure 4 gives an overview of the assignment selection process. In short, it consists of the development and application of the three sub-models in sequential order. In the following, we will explain all steps of the process.

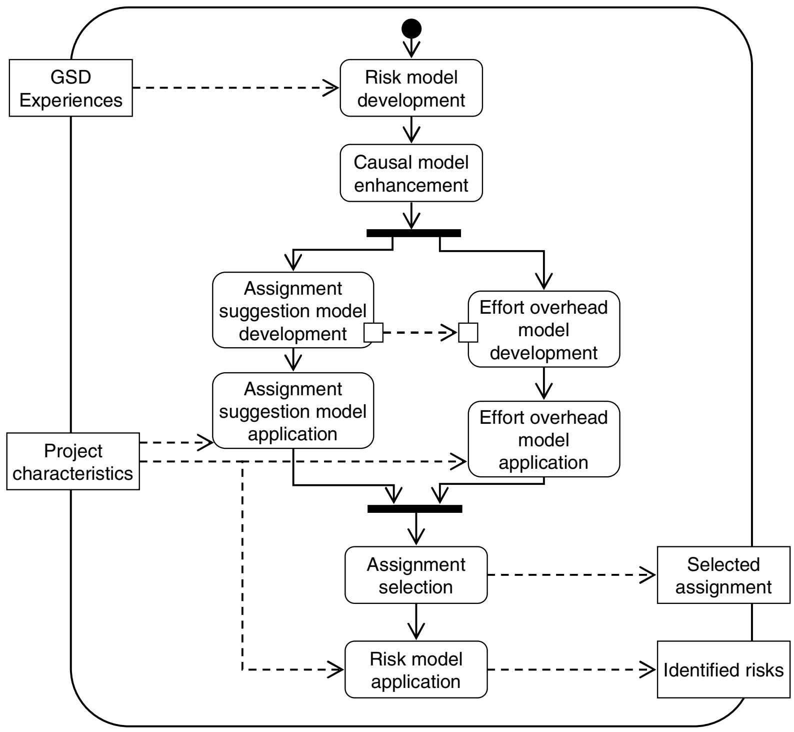

Fig. 4. Assignment selection process

1. Risk model development: The risk model is developed based on the experiences from past projects. By analyzing problems in previous GSD projects, their underlying causes, and their impact on project goals, rules are formulated as lessons learned that describe how project characteristics (i.e., influencing factors) impact the project. The influencing factors are categorized into different categories (e.g., characteristics of sites, relationships between sites).

Input: Experiences from past projects (e.g., lessons learned, expert interviews).
Output: Risk model; criteria for task allocation, risks, influencing factors, and causal relationships.

2. Causal model enhancement: The criteria, risks, influencing factors, and causal relationships identified in the risk model are refined into a causal model. This is done according to the following rules: First, all criteria and risks are linked to project goals by inserting new goals (e.g., cost minimization, workforce motivation, and product quality) and causal relationships. This ensures that for every influencing factor, the impact on project goals is described in the model. Second, new influencing factors are added that might not have been relevant in the risk model. This is done by checking for every goal in the causal model if all relevant influencing factors that might impact the goal are already included.

Input: Criteria for task allocation, risks, influencing factors, and causal relationships; expert estimations.
Output: Causal model.

3. Assignment suggestion model development: The causal model is refined into a Bayesian network model. In order to do so, the causal relationships have to be weighted and further specified. This has to be done using the estimation of experienced practitioners. The Bayesian network models can be automatically transferred into the assignment suggestion model using the TAMRI approach [10] [11].

Input: Causal model; expert estimations.
Output: Assignment suggestion model, weights on causal relationships.

4. Effort overhead model development: The effort overhead model is developed using the CoBRA methodology. For this, the causal model is used as input (however, only those factors and causal relationships that affect productivity). The weights of the Bayesian networks can be indicators for quantifying the impact.

Input: Causal model; weights on causal relationships; expert estimations.
Output: Effort overhead model.

5. Assignment suggestion model application: The assignment suggestion model is executed using the project (i.e., task, site, and general project) characteristics and project-specific weights on the assignment criteria as input. As a result, the model returns an ordered (with respect to the weighted criteria) list of assignment suggestions.

Input: Assignment suggestion model; project characteristics.
Output: List of assignment suggestions (1).

6. Effort overhead model application: Possible task assignments are analyzed using the effort overhead model. It returns effort estimations for each alternative, thus making it possible to order the alternatives based on the expected effort.

Input: Effort overhead model; project characteristics.
Output: List of assignment suggestions (2).

7. Assignment selection: The two lists of the effort overhead model and the assignment suggestion model might suggest different assignment alternatives, as they regard different viewpoints. Thus, the responsible project manager has to decide between both suggestions while keeping in mind that the effort overhead model solely regards productivity, while the assignment suggestion model considers multiple criteria but might be less accurate.

Input: List of assignment suggestions (1); list of assignment suggestions (2).
Output: Selected assignment.

8. Risk model application: The risk model can identify GSD-related project risks individually for a given project and task allocation decision. As the impact of influencing factors on project risks was refined in the two other models, it was used indirectly as basis for the task allocation decision. Thus, there is no need to apply the risk model in addition to the other models for making the task allocation decision. However, it should be applied as a final step in order to analyze the project risks for the selected task allocation. As a result, it returns a list of individual project risks that can be used for enacting countermeasures.

Input: Selected assignment; risk model; project characteristics.
Output: Identified project-specific risks.

4 Application Scenario

In the following, we will demonstrate the use of the approach by using an example scenario. In order to point out the relevant characteristics, we focus the scenario on relevant aspects and omit some details that are not relevant.

In the scenario, three components (A, B, and C) are to be developed and three sites are available (located in Germany, UK, and India). Due to project constraints, A has to be developed at the German site. B and C can now either be developed at the UK or at the Indian site. The Indian site has more expertise in working on component C and a generally lower labor cost rate. Figure 5 gives an overview of the scenario.

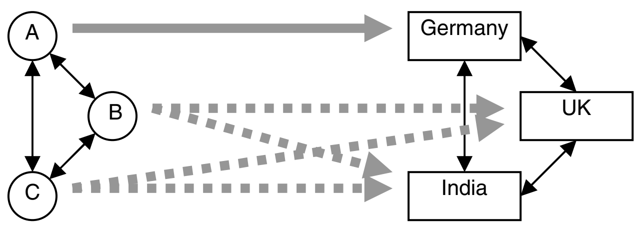

Fig. 5. Application scenario

The process steps of the approach will be applied as follows:

Step 1: A risk model is developed based on previous experiences. It results in three rules (in reality the set of rules might be significantly larger), which are demonstrated in Table 2.

Table 2. Risk model

Influencing factors	Rule	Textual description
Cultural differences, process maturity	Cultural differences and no process maturity → -productivity	Cultural differences between sites can reduce productivity if process maturity at the sites is not very high
Cultural differences	Cultural differences → -motivation	Cultural differences reduce workforce motivation
Expertise	Expertise → +productivity	Low expertise results in rework and thus reduces productivity

Step 2: The risk model is refined into a causal model. The experts name as project goals development costs (influenced by productivity), and workforce motivation. Further analysis reveals that workforce motivation is not only dependent on cultural differences but also on a site-specific motivation level. Thus, this is added as an influencing factor. Likewise, expertise is added as a factor influencing productivity.

Step 3: The Bayesian networks are built out of the causal model. Figure 6 gives an excerpt of the Bayesian networks together with the relative weights.

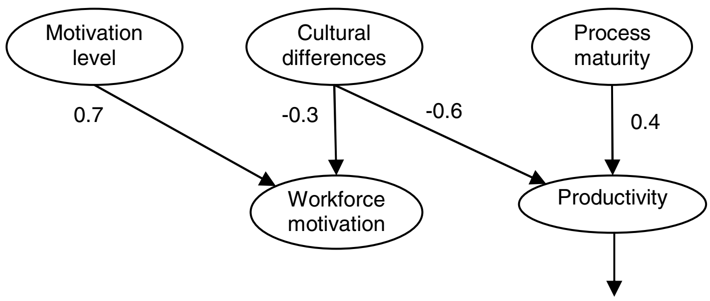

Fig. 6. Bayesian network excerpt

Step 4: For the effort overhead model, only the influences on productivity are regarded. This means that the impact of cultural differences on workforce motivation is left out. Figure 7 shows the resulting causal model.

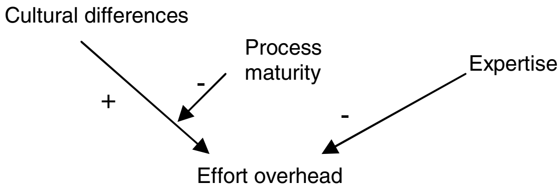

Fig. 7. Causal model for effort overhead

Step 5: The project characteristics are inserted into the assignment suggestion model. For the concrete example, this means, for instance, that the cultural differences between the UK and Germany are relatively low, while the differences between Germany and India are high. In addition, the weights on the assignment goals are set. In this scenario, keeping the workforce motivated is less important than achieving high productivity. Consequently, the model suggests assigning component C to India (due to the higher expertise) but assigning B to the UK site due to the lower cultural differences to Germany.

Step 6: Using the effort overhead model, the development costs for each assignment alternative are estimated. In this scenario, the cultural differences between India and Europe lead to only a minor productivity drop due to relatively high process maturity. Thus, the model suggests assigning B and C to India in order to profit from the lower labor cost rates.

Step 7: The project manager now has to decide between the different suggestions of the models. As both models suggest assigning C to India, it remains to be decided whether B should be assigned to the UK or to India. Finally, the project manager decides to assign B to the UK.

Step 8: The resulting assignment is: A → Germany, B → UK, and C → India. This is then analyzed using the rules of the risk model. The risk model predicts that the cultural differences between India and UK/Germany might cause low motivation among the workforce. Now, the project manager will be able to enact countermeasures such as initiating personal contacts between workers in India and Germany/UK.

5 Conclusion and Future Work

In this article, we presented an approach for systematic task allocation evaluation in global software development that consists of three sub-models, each regarding task allocation from a different perspective, interfaces between these models, and a decision process that uses all three models in sequential order.

While the individual models have already been published and have also been evaluated in an industrial context, we have now integrated them into one coherent approach. The advantage is that, on the one hand, weaknesses of the individual models can be eliminated. For example, the effort overhead model is not able to take into account criteria other than overall effort and total development costs (however, the assignment suggestion model and risk model are) and the risk model is not able to evaluate a large set of assignment alternatives (which is possible using the other two models). On the other hand, the underlying concepts such as influencing factors and causal relationships can be reused between the different models. This reduces the effort for developing the other two models once one model has been developed.

However, there are still some limitations to this approach: Even though we tried to eliminate this threat by using multiple models, the impact of a task assignment decision is often influenced by many different influencing factors and criteria that are often very hard to collect and even harder to weight or quantify. Therefore, the final decision cannot be made by decision models but rather has to be made by experienced project managers. This was also a reason for us to include a manual assignment selection step in the process (step 7): The models only deliver weighted lists of assignment suggestions to help the decision maker; the final decision, however, is made by the responsible manager (this manual selection might not scale up with big projects and very large numbers of tasks but in the industrial projects we analyzed, only a limited number of tasks was freely assignable to the available sites). In general, we think that the use of our approach can help make the decision more systematic and can support the manager in making a decision that is based on aggregated organizational experience (thus making the quality of the decision less dependant on the individual expertise of the manager).

Another limitation is the fact that the approach assumes the existence of very specific knowledge at a certain point in the project (e.g., at project start or after high-level design): The project manager must be able to divide the work into distinct tasks and name the characteristics of the tasks, the available sites, and the overall project. This might not be possible in every project and also implies some restrictions for software development processes. For example, in a completely agile process the characteristics of the tasks might not be known, thus hindering the use of this approach. However, we think that for a systematic task allocation decision, this information has to be available independent of the use of this approach. We thus see this rather as a limitation of systematic task allocation in general than of this specific approach.

A further limitation might be the effort needed to develop and maintain these models. Especially creating a Bayesian network can be very costly. However, in the TAMRI approach, we used some simple mathematical formulas (e.g., calculating the weighted average value of two inputs) for creating the probabilistic tables of the networks which largely reduced the effort. In addition, we believe that a systematic task allocation decision can reduce the overhead in distributed development projects by a much larger amount than the additional effort for the model development and application. Furthermore, the models and Bayesian networks can be reused in between projects. In future work, we will concentrate on methods for systematically reusing the models and including feedback of project managers.

In further future work, we will evaluate the approach in a real industrial context (as already done with the risk model). This will result in much more complex model instances as shown in the example: The prototype risk model developed in an industrial context consisted of 36 rules instead of 3. The assignment suggestion model contained two Bayesian networks with 14 and 16 nodes, and the effort overhead model contained 14 effort drivers. In further future work, we will investigate other interdependencies between the models. For example, the results of the cost overhead model development (i.e., the quantification of the effort overhead drivers) could be used as input for developing the assignment suggestion model.

In addition, we are planning to implement the models together with the integration process as one task allocation support tool. Individual tools have already been implemented: A prototype of the risk model has been implemented using Microsoft Excel. The assignment suggestion model has been implemented as an extensible Java tool with its own graphical user interface [11]. For the effort overhead model, there exists a general implementation of the CoBRA approach as well as an Excel model for evaluating task allocation alternatives [13]. However, a coherent implementation that integrates all models is missing yet.

Another area of future work is the integration of the models in risk management approaches. The risk model already is able to name some GSD-specific risks. However, other aspects of risk management are not supported yet: The probability of meeting certain predefined project goals (e.g. cost, time, quality) cannot be assed using the current model specification. However, the standard CoBRA approach is able to do so by using simulation. In future work, we want to integrate this into our assignment suggestion and effort overhead model.

Acknowledgment

The authors would like to thank Sonnhild Namingha for proofreading this paper.

References

1. Grinter, R.E., Herbsleb, J.D., Perry, D.E.: The geography of coordination: Dealing with Distance in R&D Work. In: Proc. ACM Conference on Supporting Group Work (GROUP 1999), pp. 306–315 (1999)
2. Mockus, A., Weiss, D.M.: Globalization by Chunking: A Quantitative Approach. IEEE Software 18(2), 30–37 (2001)
3. Lamersdorf, A., Münch, J., Rombach, D.: A survey on the state of the practice in distributed software development: Criteria for task allocation. In: Proceedings Fourth International Conference on Global Software Engineering, pp. 41–50 (2009)
4. Bass, M., Paulish, D.: Global Software Development Process Research at Siemens. In: Third International Workshop on Global Software Development (2004)
5. Keil, P., Paulish, D.J., Sangwan, R.: Cost Estimation for Global Software Development. In: International Workshop on Economics Driven Software Engineering, Shanghai, China, pp. 7–10 (2006)
6. Herbsleb, J.D., Mockus, A.: An Empirical Study of Speed and Communication in Globally-Distributed Software Development. IEEE Transactions on Software Engineering 29(6), 481–494 (2003)
7. Smite, D., Moe, N.B.: Understanding a Lack of Trust in Global Software Teams: A Multiple-Case Study. In: Münch, J., Abrahamsson, P. (eds.) PROFES 2007. LNCS, vol. 4589, pp. 20–34. Springer, Heidelberg (2007)
8. Carmel, E., Abbott, P.: Why 'nearshore' means that distance matters. Communications of the ACM 50(10), 40–46 (2007)

9. Palmer, J., Speier, C., Buckley, M., Moore, J.E.: Recruiting and retaining IS personnel: factors influencing employee turnover. In: ACM SIGCPR Conference on Computer Personnel Research, pp. 286–288 (1998)
10. Lamersdorf, A., Münch, J., Rombach, H.D.: A Decision Model for Supporting Task Allocation Processes in Global Software Development. In: PROFES 2009, LNBIP, vol. 32, pp. 332–346. Springer, Heidelberg (2009)
11. Lamersdorf, A., Münch, J.: TAMRI: A Tool for Supporting Task Distribution in Global Software Development Projects. In: International Workshop on Tool Support Development and Management in Distributed Software Projects, collocated with the IEEE International Conference on Global Software Engineering, ICGSE 2009, pp. 322–327 (2009)
12. Lamersdorf, A., Münch, J.: Studying the Impact of Global Software Development Characteristics on Project Goals: A Causal Model. The Open Software Engineering Journal (accepted for publication)
13. Münch, J., Lamersdorf, A.: Systematic Task Allocation Evaluation in Distributed Software Development. In: Meersman, R., Herrero, P., Dillon, T. (eds.) OTM 2009 Workshops. LNCS, vol. 5872, pp. 228–237. Springer, Heidelberg (2009)
14. Lamersdorf, A., Münch, J., Rombach, H.D.: Towards a Multi-criteria Development Distribution Model: An Analysis of Existing Task Distribution Approaches. In: Third IEEE International Conference on Global Software Development, pp. 109–118 (2008)
15. Edwards, H.K., Kim, J.H., Park, S., Al-Ani, B.: Global Software Development: Project Decomposition and Task Allocation. In: International Conference on Business and Information (2008)
16. Setamanit, S., Wakeland, W.W., Raffo, D.: Planning and Improving Global Software Development Process Using Simulation. In: International Workshop on Global Software Development for the Practitioner (2006)
17. Setamanit, S., Wakeland, W.W., Raffo, D.: Using simulation to evaluate global software development task allocation strategies. Software Process: Improvement and Practice 12(5), 491–503 (2007)
18. Sooraj, P., Mohapatra, P.K.J.: Developing an Inter-site Coordination Index for Global Software Development. In: International IEEE Conference on Global Software Engineering, pp. 119–128 (2008)
19. Raghvinder, S., Bass, M., Mullick, N., Paulish, D.J., Kazmeier, J.: Global Software Development Handbook. Auerbach Publications, London (2006)
20. Prikladnicki, R., Audy, J.L.N., Evaristo, R.: A Reference Model for Global Software Development: Findings from a Case Study. In: International IEEE Conference on Global Software Engineering, pp. 18–28 (2006)
21. Ralyte, J., Lamielle, X., Arni-Bloch, N., Leonard, M.: A Framework for Supporting Management in Distributed Information Systems Development. In: Second International Conference on Research Challenges in Information Science, pp. 381–392 (2008)
22. Ebert, C., Murthy, B.K., Jha, N.N.: Managing Risks in Global Software Engineering: Principles and Practices. In: International Conference on Global Software Engineering, pp. 131–140 (2008)
23. Smite, D.: Project Outcome Predictions: Risk Barometer Based on Historical Data. In: International IEEE Conference on Global Software Engineering, pp. 103–112 (2007)
24. 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, Englewood Cliffs (2000)

25. Betz, S., Mäkiö, J.: Amplification of the COCOMO II regarding Offshore Software Projects. In: Workshop on Offshoring of Software Development-Methods and Tools for Risk Management at the Second International Conference on Global Software Engineering (2007)
26. Madachy, R.: Distributed Global Development Parametric Cost Modeling. In: Wang, Q., Pfahl, D., Raffo, D.M. (eds.) ICSP 2007. LNCS, vol. 4470, pp. 159–168. Springer, Heidelberg (2007)
27. Fenton, N., Marsh, W., Neil, M., Cates, P., Forey, S., Tailor, M.: Making Resource Decisions for Software Projects. In: International Conference on Software Engineering, pp. 397–406 (2004)
28. Briand, L.C., El Emam, K., Bomarius, F.: COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment. In: International Conference on Software Engineering, pp. 390–399 (1998)
29. Trendowicz, A., Heidrich, J., Münch, J., Ishigai, Y., Yokoyama, K., Kikuchi, N.: Development of a Hybrid Cost Estimation Model in an Iterative Manner. In: International Conference on Software Engineering, pp. 331–340 (2006)