J Braz Comput Soc (2010) 16: 97–115
DOI 10.1007/s13173-010-0010-6

ORIGINAL PAPER

A multi-criteria distribution model for global software development projects

Ansgar Lamersdorf · Jürgen Münch

Received: 18 December 2009 / Accepted: 23 April 2010 / Published online: 30 May 2010

Abstract The allocation of development tasks to sites is one of the most important activities in the management of global software development projects. Its various influences on the risks and benefits of distributed projects require careful consideration of multiple allocation criteria in a systematic way. In practice, however, work is often allocated based on only one single criterion such as cost, and defined processes or algorithms for task allocation are typically not used. Existing research approaches mainly focus on selected aspects such as the minimization of cross-site communication and are difficult to adapt to specific environments. This article presents a customizable multi-criteria model for task allocation in global software development projects. Based on an analysis of the state of the practice, a set of requirements was derived and used for evaluating existing task allocation models from different domains. The Bokhari algorithm was identified as a suitable starting point and modified with respect to the fulfillment of the requirements. The modification includes the development of mechanisms for customization, the incorporation of cause-effect relationships, and the use of probabilistic modeling of uncertainty with Bayesian networks. The application of the model is demonstrated in different scenarios that represent typical hypothetical and real distribution decision problems in industrial contexts. Experience from applying the model to such problems has shown, for instance, that depending on the weight of different criteria, very different task distributions will result. This demonstrates, in consequence, the need for systematic multi-criteria task allocation support in global software development.

Keywords Global software development · Project management · Task allocation · Work distribution · Work assignment · Bayesian networks

A. Lamersdorf (✉)
Software Engineering Research Group: Processes and Measurement, University of Kaiserslautern, PO Box 3049, 67653 Kaiserslautern, Germany
e-mail: a_lamers@informatik.uni-kl.de

J. Münch
Fraunhofer Institute for Experimental Software Engineering (IESE), Fraunhofer Platz 1, 67663 Kaiserslautern, Germany
e-mail: Juergen.Muench@iese.fraunhofer.de

1 Introduction

Global Software Development (GSD) can be seen ambivalently: On the one hand, there are enormous potential benefits in distributing development work globally, such as decreased labor cost rates, access to a worldwide talent pool, and the possibility of developing in a follow-the-sun model [1, 2]. On the other hand, many problems are reported that are due to the immanent characteristics of distributed development and that, despite the recent improvements in collaboration technology, have not been overcome yet: Problems in communication, coordination, and control (for example, caused by cultural and language differences) [3–5], difficulties in knowledge transfer [6], and issues regarding the protection of intellectual property [7] are just a few examples. These problems can result in decreased productivity [8], loss of motivation [9], and a generally lower project success rate [10] and may also contribute to the trend toward "nearshoring" [11] or raise voices that argue for not starting GSD at all [12].

The high number of both potential benefits and problems of GSD show that successful implementation of distributed development requires an emphasis on project management [13], which in GSD especially includes project planning and

organizational design [3, 14]. One central aspect here is the allocation of work to distributed teams [15, 16]. In contrast to collocated development, the assignment of work not only determines the quality and expertise of the workforce, but also establishes the communication structure between sites, and thus has a direct impact on the severity of GSD-specific risks such as communication and coordination problems. Therefore, the distribution of work across sites (i.e., "task allocation") is an activity that needs to be thoroughly planned and systematically done in GSD.

In practice, however, work organization and task assignment in GSD are usually not done systematically and the result of different planning strategies and their impact on the performance of distributed development projects is often unknown [13, 14, 17]. Instead, work is often distributed based only on simple criteria such as labor cost [18, 19].

In this article, we present TAMRI (Task Allocation based on Multiple cRIteria), a model for supporting a systematic task allocation decision in distributed development projects that is based on multiple criteria and influencing factors. Reusing algorithms from task allocation in distributed systems as well as Bayesian networks for describing causal relationships under the aspect of uncertainty that is inherent to human behavior, the model is able to deliver a weighted list of task assignment suggestions with respect to specific project goals and characteristics. The influencing factors and causal relationships of the decision model were collected in an empirical study consisting of both a literature review and an interview study with experienced practitioners and managers of GSD projects. We will demonstrate the applicability of the approach in two scenarios. One is based on a detailed scenario for planning a distributed project [20]. The second scenario is based on a detailed study of global software development at Lucent [8], for which task allocation models have already been proposed [21].

Specific aspects of the model development and application have already been published [20, 22–24]. In addition to these publications, this article contributes a demonstration on how these aspects are built on each other, an integration with our current status of work, and an extension that presents new application scenarios and provides an extensive discussion of the model.

The remainder of this article is structured as follows. First, the state of the practice in GSD task allocation is analyzed, resulting in a list of requirements for a task distribution model (Sect. 2). In Sect. 3, these requirements are used to analyze existing task distribution approaches both in GSD and in other domains. Section 4 introduces both the theoretical concepts and the implementation of the model. In Sect. 5, the empirical study conducted for gathering the causal relationships and influences of work distribution on project performance is presented. Section 6 demonstrates the application of the model in different scenarios, followed by a discussion of the model in Sect. 7.

2 State of the practice

The following section describes the results of a study on the practice of task allocation in GSD [23] and derives a set of requirements for a task distribution approach.

2.1 Task allocation practices

We conducted a qualitative interview study with practitioners from various companies in order to identify the practice of task allocation and the applied criteria [23]. The main research questions of the study were:

Question 1: What is the organization's general background in global software development?
Question 2: Which criteria are applied for task allocation in GSD projects?

The complete study can be found in the respective publication [23]; here, we will focus only on a brief summary of the findings related to task allocation.

We interviewed 12 subjects that came from 11 companies in many different application domains, such as satellite development, educational software, and software services, and were based in the US, India, and Europe. They thus cover a large proportion of existing global software development types.

Interviews were conducted in a semi-structured form, containing a mixture of both open and closed questions, which is very common for qualitative studies in Software Engineering [25]. The interviews were held primarily in person. Some of them, however, had to be conducted over the telephone since some interviewees were unavailable in person. With the exception of two interviews for which detailed notes were taken, all of the interviews were recorded and transcribed literally. The transcribed interviews were analyzed using selective and axial coding [25].

One of the first results of the interview analysis was that the interviewees' answers regarding task allocation within a specific project differed to a very large extent: They did not only describe different criteria for task allocation but also talked about very different ways of distributed development and task allocations. Therefore, in order to make the results comparable, the answers had to be grouped into several types (see Fig. 1 for the types).

A first type of distributed development identified was software development outsourcing. However, from the client's perspective, task allocation here means rather partner selection. Usually, after the activities have been assigned to a contractor, the client has little or no influence on the allocation of tasks to sites by the contractor. We thus excluded the case of software development outsourcing from our further studies.

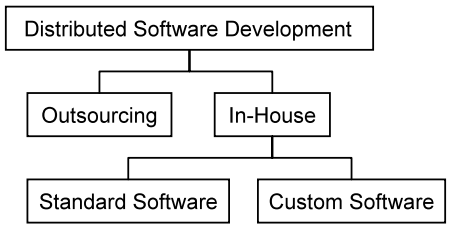


Fig. 1 Types of distributed software development

Table 1 Criteria for task allocation in standard and custom software development

Standard software development	Custom software development
Expertise	Availability
Proximity to market	Expertise
Labor costs	Proximity to client
Turnover rate	Labor cost
Availability	Strategic planning
Strategic planning	Personal reasons
Maturity of site	Political decisions
Development quality	
Personal trust	
Product architecture	
Time differences	
Cultural differences	
Willingness at site	

Within in-house development, we found a significant difference between standard and custom software development. A standard software development organization sells a standardized software product to many different customers. Often, those organizations create a relatively large amount of proprietary knowledge and technology that is contained in specialized teams. Consequently, the expertise of the teams was the most important criterion for task allocation.

In custom software development, an organization develops individual software for clients. Compared to standard software development, the specialization and the creation of proprietary technology are usually much lower. Here, the expertise of the workforce was also an important criterion for task allocation. However, in contrast to standard software development, work was also frequently assigned based only on availability—tasks were assigned to sites just because there were people available. Table 1 gives an overview of the identified criteria for task assignment with descending order of importance.

In both types of distributed software development, however, many commonalities were also identified in the way work is allocated: No company had a specific or systematic process of allocating work; it was in all cases done based solely on the expertise and estimations of the involved decision makers. Even though a large list of criteria was named over all interviews, the individual decisions at the specific companies were usually based only on a few criteria with no systematic process of eliciting them.

Finally, the interviewees were also asked about problems in distributed development and factors causing these. Interestingly, they named a large number of factors here that were not considered in task allocation. For example, most of them reported problems caused by cultural differences between sites. However, only one interviewee said that in his project, cultural differences were regarded in the decision on where to assign work. This demonstrates a need for systematic decision-making in task allocation for global development that includes all relevant criteria and influencing factors that can cause the specific problems of global development.

2.2 Requirements for systematic work distribution

Based on the result of the interview study, the main problems for task distribution in industrial global software development we identified are: (1) There is no awareness of the impact of task assignment on project results and (2) task distribution in GSD is neither goal- nor risk-oriented. We thus formulated the following requirements for an approach to supporting systematic task allocation in GSD:

REQ 1. Evaluation of assignments based on criteria and influencing factors: It must be possible to evaluate and compare different assignments in order to select the most suitable. This must be done with respect to several possible criteria (e.g., development cost, expected quality) and influencing factors (e.g., cultural differences, workforce ability).

REQ 1.1. Use of multiple criteria and factors: As there are various possible criteria and influencing factors, it must be possible to regard several of them at the same time (and, for example, not just focus on labor cost rates alone). The criteria may sometimes even represent conflicting goals (e.g., cost minimization vs. development time reduction).

REQ 1.2. Empirical basis for criteria and factors: There are many possibilities regarding criteria and influencing factors that could be considered in task assignment. The specific impact of a certain influencing factor on project goals is also unknown. This data can only be gathered in empirical studies that analyze the practice of distributed development. The approach therefore has to be based on empirical studies.

REQ 1.3. Consideration of site-specific characteristics: The interviews show that task allocation has to take into account specific characteristics of sites, such as the maturity of a site and the ability to work on a specific task. Thus, it must be possible to consider these characteristics.

REQ 1.4. Consideration of distributed overhead: Similarly, the interviews show that dependencies between tasks and

between sites have to be considered, as they impact the problems of distributed development (e.g., cultural differences between sites and coupling between tasks at different sites impacting communication overhead). They thus have to be regarded, too.

REQ 2. Causal relationships between influencing factors and criteria: Some influencing factors might have complex interactions with each other and with possible criteria: For example, a time shift between sites can, one the one hand, reduce total development time in a follow-the-sun scenario (however, only under certain circumstances such as high process maturity) and, at the same time, it can increase development costs due to the increased overhead in communicating across time zones. Therefore, it must be possible to systematically regard these types of causal relationships between influencing factors and criteria.

REQ 3. Ability to make assignment suggestions: An approach for decision support should be able to deliver suggestions for task assignment. This should be done on an algorithmic basis, since there may exist an exponentially large number of possible work assignments. The approach should thus also have a certain degree of formality.

REQ 4. Adaptability: In different projects, there might be different project goals with different weights (e.g., cost, time, quality). It must be possible to adapt the goals to project-specific environments by adding or removing goals. In addition, it can depend on the organizational and project-specific context which factors are relevant for a certain task allocation decision. Thus, it must also be possible to add or remove influencing factors.

3 Related work

The following section analyzes related work in decision support for task allocation. There exists a small body of research on task allocation for GSD, but similar problems are also considered in other domains such as distributed production (where the problem is to distribute production tasks across a network of sites) and distributed systems (where computation tasks are assigned to processor nodes). The approaches presented here will be evaluated against the previously stated requirements. Afterward, selected approaches are introduced in greater detail.

3.1 Description of the approaches

Some empirical studies in global software development aimed at identifying the practice of task allocation and tried to identify criteria, similar to the study presented in Sect. 2: Grinter, Herbsleb, and Perry [26] identified different ways of how distributed work is organized, e.g., by functional expertise, product structure, or process steps. Edwards et al. [16] looked at specific criteria for task assignment to individuals and identified individual expertise and availability as the most popular criteria. Westner and Strahringer [27] also performed a similar study from the perspective of selecting project candidates for offshore development. However, these studies only tried to analyze the practice of task allocation and did not aim at providing decision support. As the focus of our work is on decision support in task distribution, we will in the following concentrate on related work regarding the analysis of task distributions or support for decision making.

Mockus and Weiss [21] developed a model for optimizing work assignments (which in this case are represented as sets of files) that tries to minimize cross-site communication for modification requests, i.e., changes to existing files. Based on this model, it was possible to formulate an algorithm for automated identification of optimal work distributions. In the Global Studio Project [28, 29], a less formal distribution approach was developed, in which the work was split up into coherent packages together with their required knowledge and temporal dependencies. Based on this data and the knowledge of the teams, the packages were assigned manually. Madachy [30] developed a cost model for estimating effort in distributed projects that is based on COCOMOII [31] and allows for comparing different assignments and selecting the most cost-effective one. Similarly, Sooraj and Moahapatra [32] developed an index that can be used for evaluating different task allocation alternatives with respect to coordination overhead. Setamanit et al. [33, 34] suggest a simulation model for studying different task allocation alternatives.

A much different approach presented by Fenton et al. [35] uses Bayesian networks for analyzing and evaluating software development projects. As Bayesian networks are very flexible and can be adapted to organization-specific environments, they can also be used for evaluating task distributions. The model proposed by Fenton already contains a subnet for distributed communications and management.

In distributed productions, models have been developed for optimally assigning production work to distributed plants. As the problem of finding an optimal plant (i.e., site) for a production (i.e., task) while systematically considering the overhead for exchanging artifacts between plants (i.e., communication overhead) is, on an abstract level, very similar to the assignment problem in GSD, these models were also taken into consideration. Chen and Wang [36] present a distribution model for steel production with the goal of minimizing transport and production costs. The optimal assignment is calculated using linear programming. Similarly, Cohen and Moon [37] developed an approach and algorithms for assigning work to plants, with the goal of minimizing

overall costs while satisfying all demands. A more complex model was developed to optimize the distribution of car production at BMW [38]. Here, the goal is also minimization of the overall costs, but investment planning is regarded as well.

A third domain was the area of distributed computer systems. Here, the problem lies in assigning a set of interrelated computing tasks to processors with respect to both computation time and communication, which is again very similar to the problem of GSD. Bokhari [39] developed an algorithm that assigns software modules to nodes with the objective of minimizing the weighted sum of communication and execution costs. For dynamic job assignment, Amir et al. [40] developed a distribution model that, based on the expected resource usage of incoming jobs, assigns jobs with the objective of minimizing the overall slowdown. Very similar is the File Allocation Problem for assigning files to network nodes, for which Chu [41] presented a mathematical model aimed at minimizing both storage and transmission costs.

3.2 Evaluation

Table 2 presents the results of the evaluation of the presented approaches against the previously stated requirements [22]. The approaches already existing within GSD focus only on selected aspects in decision support: Except for the empirical studies, none of them regards both site characteristics and the effects of distribution overhead together. In addition, only the approach by Mockus and Weiss is able to give decision support by algorithmically selecting assignments out of all possibilities. Most of them are not truly multi-objective, either; only isolated goals are considered.

Distribution models within the production domain are more suited for making assignment suggestions. However, they only focus on cost minimization and are therefore not multi-objective. Besides, they have a strong focus on modeling production facilities while neglecting the details of tasks.

Approaches from the distributed systems domain seem to fit better to the requirements. Particularly, the model by Bokhari considers both site specific characteristics (i.e., characteristics of the processors) and the overhead of distribution properties while at the same time being able to make assignment suggestions.

Of all analyzed approaches from related work, none fulfills all requirements. However, it can be seen that a combination of the Bayesian network approach with the processor allocation algorithm by Bokhari would fulfill all requirements despite not having an empirical basis. Thus, in the following, the two approaches will be presented in more detail.

3.3 The task distribution model by Bokhari

In the model of Bokhari [39], tasks are described as modules that communicate with each other and can be assigned

Table 2 Comparison of the different distribution models and their fulfillment of the requirements: not (−), partly (○), mostly (+) or totally (++) fulfilled (none: not comparable)

Domain	Approach	1 Evaluation	1.1 Multiple criteria, factors	1.2 Empirical basis	1.3 Site-specific	1.4 Distributed overhead	2 Causal relations	3 Suggestions	4 Adaptability of goals	4 Adaptability of factors
Distributed SW Development	Empirical Studies	−	○	++	+	+	○	−		
	Modification Requests [21]	−	−	○	○	○	−	++	−	−
	Global Studio Project [28, 29]	−		+	+	○	○	○	−	−
	Distributed CoCoMo [30]	++	−	−	++	−	○	−	−	+
	Coordination Index [32]	+	○	−	○	+	−	−	−	○
	Simulation Model [33, 34]	++	+	−	○	○	+	−	−	+
	Bayesian Networks [35]	++	++	−	++	○	++	−	++	++
Distributed Production	Linear Programming [36]	−	−		○	−	○	+	−	−
	Plant Distribution [37]	−	−		+	−	○	+	−	−
	BMW Production [38]	−	−		+	−	○	+	−	−
Distributed Systems	Processor Allocation [39]	−	○		+	+	○	+	○	○
	Opportunity Costs [40]	−	++		+	−	○	○	○	○
	File Allocation [41]	−	+		+	○	○	+	−	−

to different processors. It is assumed that communication between the modules follows a tree structure, the so-called invocation tree.

The model considers two kinds of costs: the costs e_{ip} of executing module i on processor p and the costs $s_{pq}(d_{ij})$ of transmitting data d_{ij} between module i and module j with i assigned to processor p and j assigned to q.

With the invocation tree and these cost functions as input, another graph, the so called assignment graph, is built. This graph has one node for every combination of modules and nodes, representing the assignment of modules to nodes. The edges in the graph are labeled with the sum of execution and transmission costs. The problem of finding a cost-minimal assignment is thus reduced to the problem of finding a minimal subset of the assignment graph. Bokhari proposes an algorithm that solves this problem in polynomial time.

The model can be easily transferred to handling task assignment in GSD: Modules and processors represent tasks and sites. The costs of executing module i on processor p represents the effort of doing task i of a software development project at site p without regarding any communication overhead. This overhead is then represented by the costs of transmitting data between module i and module j. As in the model of Bokhari, this overhead does not only depend on the communication need between the two tasks but also on the communication efficiency between two sites.

However, adaptations are needed: First, the algorithm can only handle tasks that are connected in a tree structure. However, tasks in a development project can have arbitrary connections. Second, costs are the only criterion for comparing different assignments. Therefore, the different conflicting goals that may exist in global software development have to be aggregated into one cost function. Finally, all costs are described by a single, distinct number, which does not represent the reality of human development, which contains a large amount of uncertainty. The first problem was solved by developing an extension of Bokhari's algorithm that contains an additional first step of transferring arbitrary graphs into a set of trees (however, with reduced efficiency). The other two problems were solved by describing the cost functions not with single numbers but by using Bayesian networks instead.

3.4 Bayesian networks for effort and cost modeling

Bayesian networks represent a method of formulating causal relationships under conditions of uncertainty. This is done by using statistical methods and probabilistic functions. A Bayesian network (BN) consists of a directed acyclic graph and a set of probability tables, one for each node in the graph [42]. Each node represents a discrete uncertain variable, whose value is usually not known exactly but which has a known probability for every value. Edges between nodes are directed and indicate causal relationships between variables. The tables describe for each node the conditional probability for its states for each combination of its parent states.

Statistical reasoning makes it possible to infer through a Bayesian network and make statements on the probability of the values for one node with respect to other nodes' values. In particular, it is possible to model a causal network with input and output nodes and calculate the probabilistic distribution over the values of the output nodes with respect to the states of some of the input nodes.

In software engineering, BNs have been used several times to model development projects and describe causal relationships between project characteristics, effort, and software quality [35, 43, 44]. Due to the stochastic nature of BNs, the effort estimations do not result in a single number but in a probabilistic distribution over ranges of values together with a median value. This explicitly reflects the uncertainty in modeling and predicting human behavior.

4 The multi-criteria distribution model

In the following section, the multi-criteria distribution model TAMRI is introduced in more detail. It mainly consists of an approach for modeling the cost functions for execution and transmission costs of the Bokhari model.

4.1 Model overview

Figure 2 gives a conceptual overview of the TAMRI model. Its basic elements are the distributed system algorithm by Bokhari (adapted in order to handle arbitrarily connected tasks) and Bayesian networks for describing task execution and transmission cost. However, as the distributed system algorithm requires distinct single numbers for execution and transmission cost and the Bayesian networks work with probabilistic distributions over discrete values, a simulation algorithm was developed for selecting single numbers from the distributions.

Figure 3 shows a Bayesian network for the cost of executing a task at a site. It contains several input nodes that model certain characteristics of a task and a site that can influence the outcomes of a development task. These are modeled as output nodes representing abstract costs in terms of money, time, and quality. Each is influenced individually by the input nodes. If values are selected for these input variables, the network makes it possible to describe the outcomes of each possible assignment combination of a task and a site. Thus, the model represents the execution cost function e_{ip}. However, it does not return one single cost value, but rather

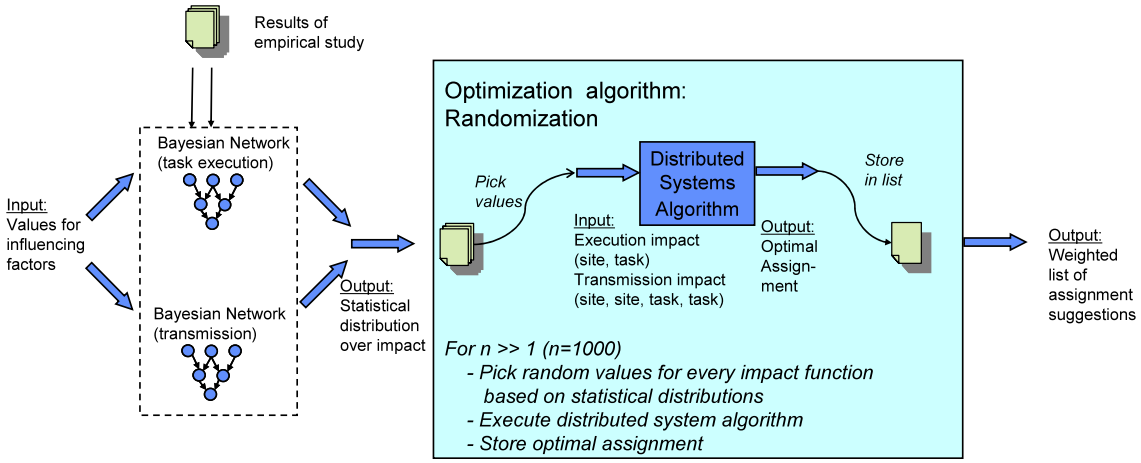
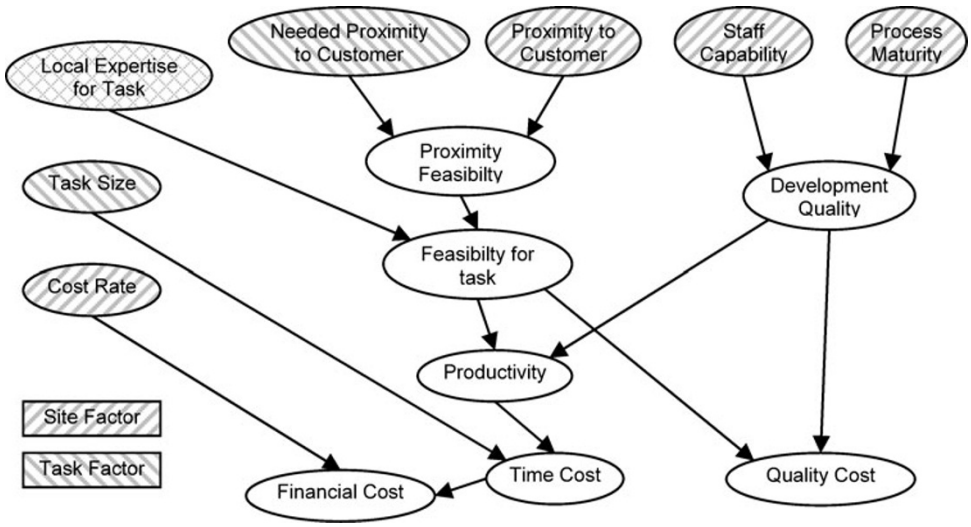


Fig. 2 Conceptual overview

Fig. 3 Bayesian network for execution cost [24]



probabilistic distributions for the three output nodes. Similarly, Fig. 4 shows the Bayesian network modeling the transmission cost for each combination of two tasks assigned to two sites. In this case, the input nodes mainly reflect the relationships between two nodes and two sites.

BNs use ordinal scales for all variables. In this model (similarly to [35]), every variable operates on a five-level scale (from "very low" to "very high" or using numeric ranges), which makes calculation and inference between variables easy. The output nodes (i.e., the different types of cost) are normalized to five levels between 0 and 1 in order to make them comparable. They can thus be aggregated into one cost value using project-specific weights.

For the BN modeling the transmission costs, the levels of the output nodes describe relative increases of the execution costs (in percent). The increase can sometimes also be negative: In a perfect environment, a time zone difference between two sites can result in a reduction of the total development time due to the possibility of "follow-the-sun" [45]. Therefore, the lowest state in the "Time costs" node of the transmission cost BN is labeled with a negative number.

The link between the Bayesian networks results and the (adapted) algorithm of Bokhari is provided by a randomization algorithm similar to Monte–Carlo simulation. It basically consists of three steps [24]:

- Collect the probabilistic distributions by executing the BNs for every combination of tasks and sites.
- Repeat for a large number of runs: Randomly pick one number out of every probabilistic distribution and aggregate all types of costs into one cost value. The probabilities for every random pick are provided by the probabilistic distributions. Store the numbers as cost functions for the distributed systems algorithm. Execute the distributed systems algorithm and store the returned assignment.
- Return the stored assignments in an ordered list with a decreasing number of occurrences.

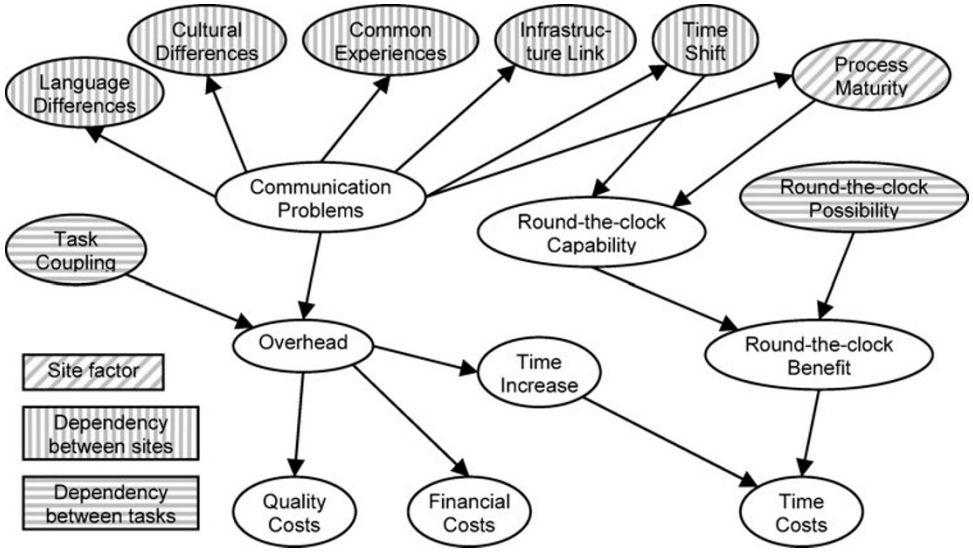


Fig. 4 Bayesian network for transmission cost [24]

In short, the algorithm simulates a number of scenarios with randomly chosen numbers for the individual cost functions, based on their probabilistic distributions. This ensures, on the one hand, that across all scenarios, the costs reflect the predictions of the Bayesian networks. On the other hand, within each individual scenario, costs are represented by distinct numbers, which enables the execution of Bokhari's algorithm. Thus, for every scenario, an individual optimal assignment is identified by the distributed systems algorithm.

As a result, the algorithm returns not one but several assignments together with information on the number of scenarios in which each distribution was optimal. This makes the uncertainty in predicting human behavior explicit and gives the project manager the opportunity to choose from an ordered set of assignments.

An additional advantage of BNs used in TAMRI is the fact that they can handle missing data: Not all input variables have to be set to distinct states in order to get an output; if a variable is not set to a specific value, it is assumed to have equal distribution across all states. Thus, it is also possible to execute the model without specifically setting all input parameters.

The separation between Bayesian network cost models and the algorithms for identifying assignment suggestions makes the distribution model very generic: New goals and influencing factors can be added easily, causal relationships can be altered, and the weights on the goals and the relationships can be changed by simply modifying the networks while the rest of the model remains untouched.

4.2 Model implementation

TAMRI was implemented as a Java prototype with a basic graphical user interface. This implementation was kept very generic in order to make it possible to easily add customized models with specific Bayesian networks. Thus, the code consists, on the one hand, of the generic implementation of the model algorithms together with abstract classes for BNs and the user interface. On the other hand, it contains specific instantiations of the Bayesian networks for execution and transmission cost and the corresponding user interface.

The generic part provides a framework for the Bayesian networks and user interfaces as well as mathematical functions and the implementation of the algorithm for suggesting assignments. It uses external libraries for inferring through the BNs [46] and basic statistical calculations [47].

The custom part of the implementation contains organization-specific Bayesian networks and the user interfaces for setting project-specific values to the custom BNs. If an organization wants to describe custom causal relationships with organization-specific influencing factors and criteria, it only has to adapt this custom part. This makes it relatively easy to adapt the underlying model to individual environments.

Figure 5 shows an example of the user interface of TAMRI, where project-specific values can be set to the influencing factors. When all known input values are set, the calculation can be started by the user, executing the algorithms as described in Sect. 4.1.

5 An empirically based instantiation of the distribution model

The previous section provided an overview of the basic TAMRI model and the algorithm for deriving assignment suggestions from a causal model (described in Bayesian networks) and project-specific values for the influencing factors. However, this basic model does not come with specific

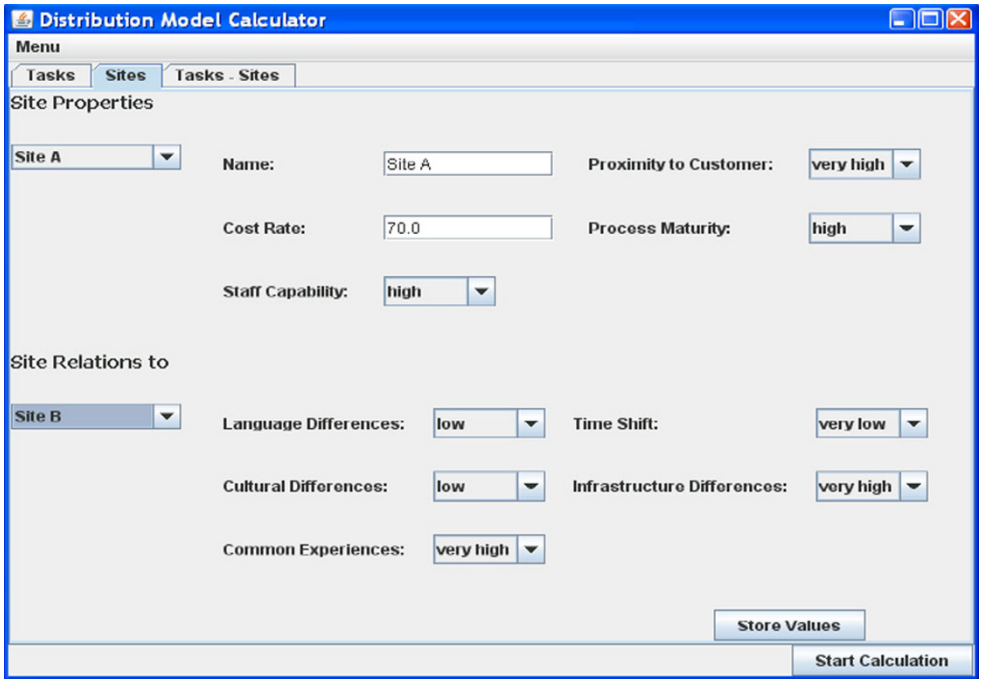


Fig. 5 Parameter window

influencing factors, criteria, and causal relationships. This is due to the fact that in our opinion, the underlying causalities are organization-specific and have to be identified individually for every environment.

In order to demonstrate the use of the model, we developed a first instance by identifying goals, influencing factors, and their relationships in an empirical study. The resulting model has to be further adapted to specific environments. However, it describes typical characteristics of distributed development and will be used in this article for several feasibility studies.

The main research questions of the empirical study were:

Question 1: What are the goals of distributed development projects?
Question 2: What characteristics of distributed development should be regarded during task assignment?
Question 3: What are the relationships between the characteristics of distributed development and project goals?

The study consisted of a literature review and an interview study, which will be explained in the following.

5.1 Literature study

A literature review was conducted in order to gather practical experiences and extract information on goals, influencing factors, and their relationships. For this, the databases ProQuest and Web of Science were searched using the keywords "Global Software Development" and "Distributed Software Development", as were special issues of IEEE Software and the proceedings of the "International Conference on Global Software Engineering" and the "International Workshop on Global Software Development for the Practitioner".

Based on the title and abstract, the findings were narrowed down to practical experience reports and descriptions of problems in practical applications of GSD. Other kinds of publications such as literature surveys, reports on tactics for alleviating the problems of distance, or presentations of technical solutions for collaborative work, were filtered out. In the end, we identified 26 publications, which were classified into case studies, empirical studies, and others. Case studies presented the experiences of singular software projects, while empirical studies reported aggregated experiences from multiple projects. Other types of literature mainly included personal experiences of highly experienced practitioners or researchers. Table 3 gives an overview of the included literature.

As a result, six goals of GSD projects were identified. These were influenced by 15 influencing factors. However, most of the influences were not direct but via some intermediate factors (e.g., cultural differences causing communication overhead, which impacts productivity and thus development time). We thus included "intermediate factors" as an additional layer between influencing factors and goals. Table 4 shows the results in the three different categories.

5.2 Interview study

In order to not only get a theoretical but also a practical view on the causal relationships and to identify relative weights

Table 3 Included literature

Case Studies	Ebert & De Neve (2001) [48], Casey & Richardson (2006) [9], Treinen & Miller-Frost (2006) [45], Battin et al. (2001) [49], Lindqvist et al. (2006) [50], Heeks et al. (2001) [51], Kobitzsch et al. (2001) [52], Mullick et al. (2006) [28]
Empirical Studies	Alami et al. (2008) [53], Gareiss (2002) [54], Kommeren & Parviainan (2007) [55], DeLone et al. (2005) [56], Pilatti et al. (2006) [57], Smite & Moe (2007) [58], Ramasubbu & Balan (2007) [59], Smite (2004) [60], Oza & Hall (2007) [61], Komi-Sirvio & Tihinen (2005) [62], Herbsleb et al. (2005) [63], Espinosa et al. (2007) [64], Herbsleb & Mockus (2003) [65], Espinosa et al. (2007) [66]
Other	Sakthivel (2007) [7], Carmel (1997) [67], Gurung & Prater (2006) [68]

Table 4 Literature results

Influencing factors	Labor costs, Expertise and Knowledge, Local government, process maturity, IP security, available resources, physical distance, language difference, cultural difference, organizational difference, infrastructure distance, time shift, common working experiences, needed knowledge for task, coupling
Intermediate factors	Cost overhead, lack of trust, productivity, fit between knowledge needed for task and available at site, problems in communication, coordination, and control
Goals	Costs, proximity to customer, time, quality, IP protection, staffing

for the identified factors, the literature findings were validated in an interview study. Thus, interviews with experienced practitioners were conducted and the literature findings were presented to them. The interviewees were given the overall model as well as each causal relationship individually and were asked to comment on them. In addition, they were asked to weight the existing influencing factors and—if necessary—add or remove factors or relationships.

Ten of the participants of the interview study presented in Sect. 2 were interviewed. The interviews were recorded and the transcriptions were analyzed.

Based on the analysis results, several modifications were made to the findings of the literature study. The main changes were:

(1) The goals of "Resource utilization" and "Proximity to customer" did not appear to be official project goals in most cases. Resource utilization seemed to be more a constraint (i.e., work can only be assigned to sites with available resources) and proximity to customer a factor influencing effort and quality. They were thus removed as goals.
(2) "IP protection", on the other hand, could presumably be an issue and goal in GSD projects. However, the practitioners for which this was the case refused to talk about this in detail. Since no information could be gathered, it was also removed as a goal.
(3) The benefit of round-the-clock development (or "follow-the-sun") is mentioned in several publications [45, 66, 69]. In the interviews, it turned out that the benefit in round-the-clock development can only be achieved with certain types of tasks and with very high process maturity.
(4) While some literature reports explicitly mentioned the additional costs due to traveling between distant sites, most practitioners found this cost factor negligible compared to other cost drivers in GSD such as labor cost rates.
(5) "Lack of trust" as a factor mentioned was confirmed as relevant by many of the practitioners. However, it is very hard to specify and was interpreted differently by the interviewees. It was therefore not included in the model.

Figure 6 shows the final causal model with goals, influencing factors, and causal relationships. These relationships describe how certain factors increase (+) or decrease (−) others based on the findings in the literature and the interviews (e.g., "high process maturity increases the benefit of follow-the-sun", "the benefit of follow-the-sun decreases development time"). The figure also contains weights for the causal relationships according to the answers of the interviewees. However, the weighs were hard to gather: Only for few influences (marked with "+++" or "− − −") did most practitioners agree on their importance. Others were only seen important by some of the interviewees ("++", "− −") or were subject to very different views ("+", "−"). This shows again that many influencing factors and their weights depend on the environment and emphasizes the need for a distribution model to be adaptable.

From the results of the study, the initial Bayesian networks for execution and transmission costs shown in Figs. 3 and 4 were derived as follows:

First, the causal model was split into two parts, with one containing the characteristics of tasks and sites and the other one containing the relationships between tasks and sites. Afterward, a Bayesian network was constructed for each model.

As the probabilistic tables for a node in a BN grow exponentially with the number of input nodes, new intermediate nodes were created in order to reduce the number of

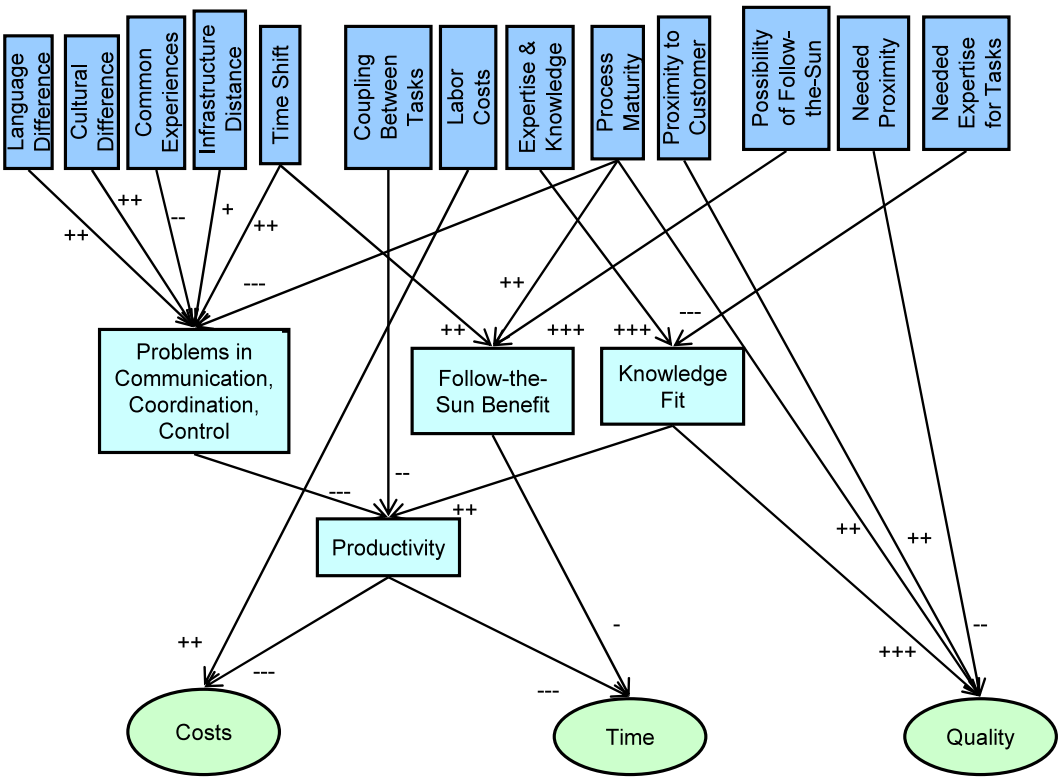
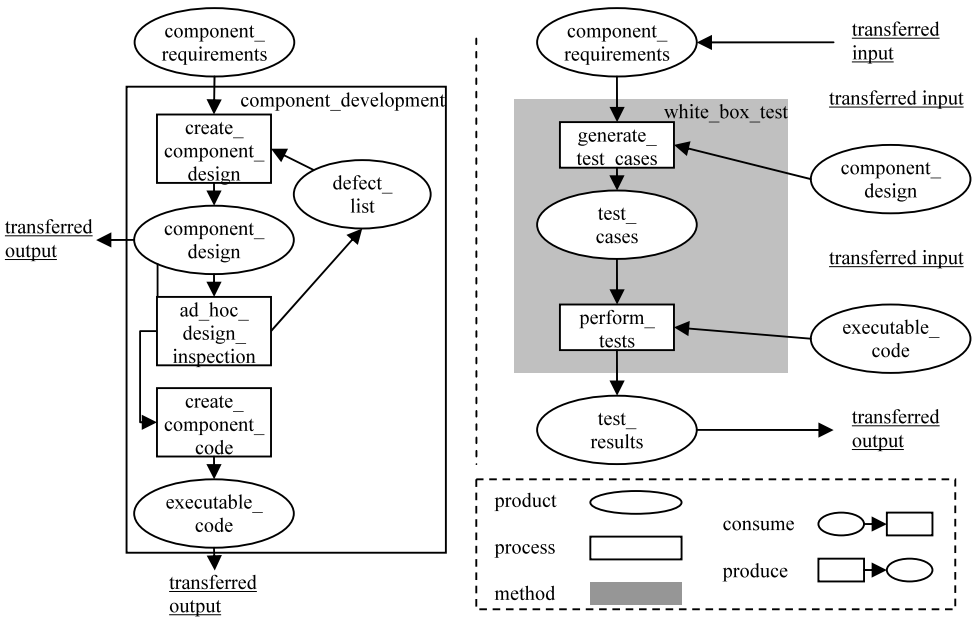


Fig. 6 Causal model after interview study

Fig. 7 Development (*left*) and test process (*right*) [71]



inputs per node. For the node "communication problems", the causal relationship to its input nodes was reversed, with the influencing factors being indicators for the problems in communication. This was done because in BNs, both deductive and inductive reasoning is possible [35] and because complexity is reduced largely.

Many of the resulting model's influencing factors are in accordance with the criteria for task allocation we identified in practice (see Table 1). However, there are also several differences that are mainly due to two reasons: First, many of the criteria that were named as being applied in practice were very fuzzy and could not be included in a systematic

model (e.g., "political reasons" or "personal reasons"). Second, there were some factors that were not named as criteria for work allocation but were included as influencing factors (e.g., "language differences"). These factors were named by both literature and practitioners as having an influence on project outcomes (e.g., by impacting productivity) and we therefore believe that it is important to include them in a systematic task allocation decision process.

6 Model application

In the following, we will now demonstrate how the model can be applied in typical project planning situations. Thus, we will use the instantiation of the model described in Sect. 5, set the values of the influencing factors according to different scenarios, and apply the algorithms described in Sect. 4 using the TAMRI tool. In order to do so, we implemented the causal model derived from the empirical study (see Fig. 6) as Bayesian networks (see Figs. 3 and 4) in the custom part of the TAMRI tool (see Sect. 4.2).

The application was done in two environments. First, a previously defined scenario for distributed process planning [20] is presented and extended. The other application replicates a real-world scenario [3, 8, 70] that already was subject to another optimization approach for task distribution [21].

In all scenarios (unless explicitly stated otherwise), we used the causal model derived from the empirical study (Figs. 3, 4, 6). However, in order to reduce the complexity of the examples, not all variables of the model were used in each scenario: Variables that were not described in a scenario were set to medium values for all involved tasks and sites.

6.1 Distributed process planning

The following application uses a scenario for distributed process planning of development and testing activities. It stems from an application of an approach for integrated design support developed by Goldman et al. [71]. The scenario consists of two activities, component development and component testing, whose specifications are shown in Fig. 7.

In the original scenario, the focus is on the assignment of activities to individuals at already determined sites and on process management. This is extended by adding the step of choosing a development or testing site and is demonstrated in three cases [20]. In the scenario, three sites (A, B, C) are available, with Fig. 8 showing some of their characteristics.

Case 1: In the basic scenario, development and testing of one single component are regarded. It was already decided to assign the development to site A. Testing can now be assigned either to site B or C. As component development and testing are complementary activities, round-the-clock development is possible between the two activities. The time shift between A and C would make round-the-clock development possible if testing was assigned to C. However, at B, people are more familiar with the component and thus have higher testing skills. Thus, the decision maker in task assignment has to weigh the benefits of round-the-clock development against the impact of specific knowledge on effort and time.

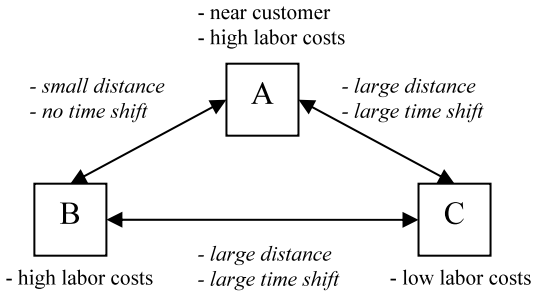


Fig. 8 Available sites [20]

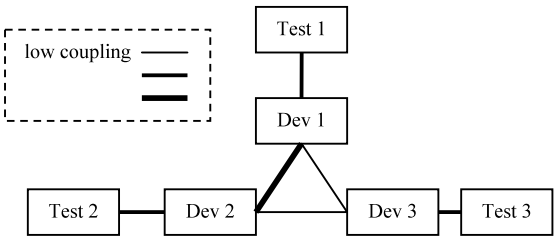


Fig. 9 Tasks in Scenario 2 [20]

The assignment suggestions made by TAMRI for this case depend on the project weights regarding different goals time, cost, and quality. Table 5 shows the suggestions for two different weights. First, the strongest weight is put on quality, which results in the model strongly (in 74% of the simulation runs) suggesting assigning testing to B due to the higher expertise at this site. If more emphasis is put on development time, assigning testing to C is suggested. However, this suggestion is much weaker, as it was only optimal in 52% of the runs.

Case 2: The basic scenario is extended to three components that are to be developed and tested. The components are coupled differently, thus requiring different amounts of communication between the teams developing them. Figure 9 gives an overview of the tasks. From the viewpoint of expertise and skills, the best abilities are located at A for developing the components and at B for testing them. However, labor costs are much lower at C and the time shift to C would enable round-the-clock development.

Table 6 shows the result for case 2. Here, the focus was put on quality (however, with smaller weights on cost and time as well). It can be seen that due to the high number

Table 5 Results for scenario 1 with focus on quality (top) and development time (bottom)

Focus on quality							
	1.: 74%				2.: 26%		
	Site A	Site B	Site C		Site A	Site B	Site C
Development	X			Development	X		
Testing		X		Testing			X
Focus on development time							
	1.: 52%				2.: 47%		
	Site A	Site B	Site C		Site A	Site B	Site C
Development	X			Development	X		
Testing			X	Testing		X	

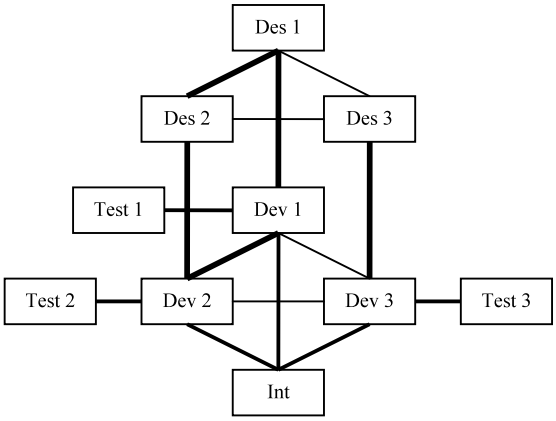


Fig. 10 Tasks at Scenario 3 [20]

of possible assignments, the suggestions are less clear: The best-rated assignment was optimal in only 17% of the simulations. However, the results give some implicit suggestions: Due to the emphasis on quality, work should, in principle, be assigned to the sites with the best skills: Development should be done at A and testing at B. Nevertheless, single test tasks could also be done at C in order to profit from the low labor rates and the possibility of round-the-clock development.

Case 3: Finally, the development step of Fig. 7 was split into design and (code) development. The same three components are to be designed, developed, and tested as in the previous scenario (see Fig. 10). Additionally, an integration step was added, which can only be done at customer site A. Design would also benefit from being at the customer site, but the people at C have the most expertise in design. Due to staff availability, development can only be done at B or C.

Table 7 shows the model suggestions with an emphasis on quality and cost. It can be seen that due to the strong coupling between the different tasks, it is suggested keeping most tasks together at one site. With quality in focus, no assignment was suggested strongly, but again some rules can be identified (e.g., that in most cases it is better to do design at A or that either very little or all work should be assigned to B in order to reduce the communication overhead). If more emphasis is put on cost, then the model suggests assigning large chunks of work to C, as it is the site with the lowest labor rates.

6.2 The Lucent scenario

Herbsleb et al. [3, 8, 70] conducted empirical studies on distributed development at Lucent Technologies that are reported in multiple publications. They describe the impact of work distribution on the length of work intervals and conclude that remote collaboration significantly increases the amount of time needed for single work packages [70]. Based on these findings, Mockus and Weiss [21] developed a model for selecting candidates of code files that should be assigned to one site in order to minimize the number of multi-site modification requests (i.e., work packages that include changes to files assigned to multiple sites). However, this approach only considers the needed communication and does not regard other influencing factors as identified in Sect. 5. In the following, we will thus show how the distribution model could be applied in this scenario.

The sites involved in the scenario are described in detail in the original publications. Four sites are reported, two in India, one in Germany, and one in the UK. Based on the descriptions of the sites, assumptions about the site characteristics and their dependencies can be made as illustrated in Tables 8 and 9.

In order to keep consistent the model presented by Mockus and Weiss [21], we try to optimize the assignment of code chunks (or subsystems) to sites and regard the number of modification requests between two code chunks as a measure of the needed communication. However, we additionally assume that each site is specialized in different subsystems and that the code chunks require different levels of interaction with the customer.

We assume that six code chunks A—F are to be assigned. Table 10 shows the coupling between these chunks based on the number of modification requests that include each pair of chunks. It shows that the system consists of two closely coupled sub-components ACD and BEF.

Table 6 Results for Scenario 2: first three suggested assignments

	1. 17%			2. 5%			3. 5%		
	Site A	Site B	Site C	Site A	Site B	Site C	Site A	Site B	Site C
Dev Comp 1	X			X			X		
Dev Comp 2	X			X			X		
Dev Comp 3	X			X			X		
Test Comp 1		X				X		X	
Test Comp 2		X			X				X
Test Comp 3		X			X			X	

Table 7 Results for scenario 3 with focus on quality (top) and cost (bottom)

Focus on quality									
	1.: 2%			2.: 1%			3.: 1%		
	Site A	Site B	Site C	Site A	Site B	Site C	Site A	Site B	Site C
Des Comp 1	X			X				X	
Des Comp 2	X			X				X	
Des Comp 3	X			X				X	
Dev Comp 1			X			X		X	
Dev Comp 2			X			X		X	
Dev Comp 3			X		X			X	
Test Comp 1			X			X		X	
Test Comp 2			X			X		X	
Test Comp 3			X		X			X	
Integration	X			X			X		
Focus on cost									
	1.: 37%			2.: 4%			3.: 2%		
	Site A	Site B	Site C	Site A	Site B	Site C	Site A	Site B	Site C
Des Comp 1			X	X					X
Des Comp 2			X	X					X
Des Comp 3			X	X					X
Dev Comp 1			X			X			X
Dev Comp 2			X			X			X
Dev Comp 3			X			X			X
Test Comp 1			X			X	X		
Test Comp 2			X			X			X
Test Comp 3			X			X			X
Integration	X			X			X		

Table 8 Characteristics of sites in Lucent scenario

	Germany	UK	India 1 (Luc)	India 2 (Contract)
Cost Factor	High	Very high	Low	Low
Proximity to Customer	High	Very High	Very Low	Very Low
Staff Capability	High	High	Medium	Medium
Process Maturity	Medium	Medium	High	High

The Indian sites have high knowledge about the code chunks E and F. The UK site has high knowledge about all code parts, while the German site is specialized in A and C. The chunks A, B, and C contain user-specific models and

Table 9 Dependencies between sites in Lucent scenario

	Germany–UK	Germany–India 1	Germany–India 2	UK–India 1	UK–India 2	India 1–India 2
Language Diff	Medium	High	High	Low	Low	Very low
Cultural Diff	Very low	High	Very high	Medium	High	Low
Common Exp	Low	Medium	Very high	Low	Very high	Medium
Infrastructure Link	Very high	Very high	Medium	Very high	Medium	Medium
Time Shift	Low	High	High	High	High	Very low

Table 10 Coupling between code chunks

	A	B	C	D	E	F
A		Low	High	High	Low	Low
B	Low		Low	Low	High	High
C	High	Low		High	Low	Low
D	High	Low	High		Low	Low
E	Low	High	Low	Low		High
F	Low	High	Low	Low	High	

thus require high interaction with the customer. Due to availability constraints, A can only be done at the German site and F can only be done at the Lucent Indian site.

Table 11 shows the resulting suggestions of the model in two different executions: One execution is consistent to the model by Mockus and Weiss [21] and the second includes various other influencing factors. First, the model only considers the communication overhead between sites, disregarding the different characteristics and abilities at the sites (here we did not use all influencing factors described in Fig. 6). Thus, it tries to minimize the number of multi-site requests (as do Mockus and Weiss). It can be seen that the model suggests keeping as much work as possible together at one site, assigning coherent components to one site, or using the UK site as a bridge between the German and Indian sites.

However, if the various influencing factors of our distribution model are considered, the model suggests very different assignments: It now suggests assigning most of the work to the UK site due to the high staff capability and proximity to customer there. In this case, the focus was set on quality rather than on development costs. If the focus was set more on cost minimization, the model would assign much less work to the UK site due to the high labor cost rates there.

The application of the model to the Lucent scenario shows that whether an assignment is "optimal" depends very much on the included influencing factors and on their specific weights. Thus, a decision support model should include various different factors and include the possibility to adjust their weights to project-specific preferences.

7 Discussion

In this final section, we will discuss the applicability and practical use of the model by first evaluating the model against the previously stated requirements and then sketching a process for applying the model in new environments.

7.1 Evaluation of the Model

In Sect. 2.2, a set of requirements for a task distribution model was formulated. The TAMRI model addresses them in the following manner:

REQ 1. Evaluation of assignments: The Bayesian networks are able to evaluate task assignments with respect to individual goals and influencing factors. The assignment suggestions made by TAMRI are based on this evaluation. However, the evaluation is not very transparent to the user, since it is based on abstract aggregated numbers. The TAMRI tool also does not present the evaluation to the user, working as a black box that presents only the final assignment suggestions to the user. The model can therefore not be used to transparently evaluate different assignment alternatives.

REQ 1.1. Use of multiple criteria and factors: The Bayesian networks allow for using multiple nodes as influencing factors or assignment criteria.

REQ 1.2. Empirical basis: The basic model consists only of abstract concepts and algorithms. However, the instantiation of the model presented in this article is based on an empirical literature and interview study.

REQ 1.3. Consideration of site-specific characteristics: The BN for execution costs can handle various task- and site-specific characteristics.

REQ 1.4. Consideration of distributed overhead: The BN for transmission costs is able to model the overhead of distributed collaboration with respect to multiple influencing factors.

REQ 2. Causal relationships: Causal relationships are described by the BNs.

REQ 3. Suggestion ability: The algorithms of the Bokhari model provide the ability to return an ordered list of assignment suggestions.

Table 11 Results for Lucent Scenario with focus on task and site dependencies only (top) and on all influencing factors (bottom)

Only task dependencies and site relations considered												
	1.: 40%				2.: 5%				3.: 4%			
	Ger	UK	Ind1	Ind2	Ger	UK	Ind1	Ind2	Ger	UK	Ind1	Ind2
A	X				X				X			
B	X						X		X			
C	X				X					X		
D	X				X				X			
E	X						X		X			
F			X				X				X	

Complete model including various influencing factors												
	1.: 16%				2.: 14%				3.: 8%			
	Ger	UK	Ind1	Ind2	Ger	UK	Ind1	Ind2	Ger	UK	Ind1	Ind2
A	X				X				X			
B		X				X				X		
C		X			X					X		
D		X				X				X		
E		X				X						X
F			X				X				X	

REQ 4. Adaptability: As the Bayesian networks can be exchanged or adapted to organization-specific environments, goals and influencing factors can be altered freely.

The evaluation of the model shows that all requirements are completely fulfilled with the exception of the ability to evaluate task assignment alternatives. Further limitations of the model are:

- The model does not provide a process for identifying organization-specific Bayesian networks for custom environments.
- The model assumes that it is possible to identify and characterize distinct tasks at the time the decision is made (e.g., after architecture definition). In some cases, this might not be possible.
- A certain degree of formality and quantifiability of the underlying causal model is required in order to apply the model. It is, for example, necessary to explicitly weight the relative impact of cultural and language differences on communication overhead. Even though this problem is reduced by the use of Bayesian networks, it is sometimes hard to explicitly weight all causal relationships.

In order to address these limitations, we are developing a process for an organization-specific application of a model-based task assignment decision that includes both the TAMRI model and models for evaluating task assignment alternatives. The following section will give an overview of this process.

7.2 Applying systematic task assignment to organization-specific contexts

In specific GSD contexts (i.e., a software development organization), the knowledge on causal relationships between influencing factors and goals of distributed development is typically not available explicitly. Hence, there thus must be a process for gathering this knowledge and transforming it into the Bayesian networks needed for the TAMRI model. In addition, further models are needed to transparently evaluate different assignment scenarios. We suggest the following process steps:

(1) Collect lessons learned: By interviewing project managers or analyzing data from past projects, the organization-specific experiences in distributed development are gathered as a set of lessons learned or rules that describe simple relationships (e.g., “cultural differences decrease productivity”).

(2) Derive causal relationships: The lessons learned are formalized as a set of causal relationships between influencing factors (e.g., “cultural differences”), intermediate factors (e.g., “productivity”), and goals (e.g., “development time”).

(3) Enhance causal model: In incremental rounds of discussion with practitioners, the causal rules are aggregated into a causal model. The model is enhanced by further causal relationships until finally all output nodes describe project goals and all relevant influences are covered by the model (see Fig. 6).

(4) Develop BNs from causal model: The causal model is split up into models describing the characteristics of the tasks and sites (i.e., the BN for execution costs, see Fig. 3)

and the dependencies between tasks and sites (i.e., the BN for transmission cost, see Fig. 4). For every sub-model, the number of inputs for every node is reduced by adding new nodes and basic functions, and weights are selected for defining the probabilistic tables, resulting in two Bayesian networks. These networks are implemented in the custom part of the TAMRI tool.

(5) Develop cost model: In addition, the influencing factors and causal relationships identified in step 3 are used for deriving effort drivers in a cost model. In a different publication [72], we describe how such a cost model for evaluating task allocation alternatives can be derived from a causal model.

(6) Characterize project: If a task allocation decision has to be made for a new project, the project is split up into distinct tasks and the available sites are identified. They are then characterized according to the previously identified influencing factors.

(7) Apply TAMRI on project: The project characteristics are inserted into the customized TAMRI tool. An execution of the tool results in a list of assignment suggestions.

(8) Apply cost model on project: The assignment suggestions are using the cost model developed in step 5. This allows for a transparent evaluation of task assignment alternatives.

(9) Select task assignment: Based on the results of the TAMRI application and the evaluation of the suggested alternatives, an assignment decision is made.

7.3 Conclusion

In this article, we described a model for making systematic task assignment decisions in GSD projects based on multiple criteria. Its main contributions are that the model is able to handle various, customizable influencing factors and goals and can make project-specific assignment suggestions that are based on these criteria. An analysis of the Lucent scenario shows that this inclusion of multiple criteria yields very different results compared to models that only focus on selected aspects.

In addition, the model is able to handle the inherent uncertainty by applying stochastic methods and weighting the suggestions with the probability (in percent) of every suggestion being optimal. In many cases, this results in very low numbers for the suggested assignments (e.g., scenario 3, Table 7), which raises the question of whether these suggestions can be seen as trustworthy. However, our experience with applying the model shows that in these cases, the assignment suggestions typically differ only in selected aspects and general assignment rules can be extracted with higher trustworthiness (as also in scenario 3). In practice, it will depend on the organizational environment and the experience of the user which percentage of the suggestions can be seen as trustworthy.

The model still has some limitations such as not making the evaluation of assignments transparent. We are addressing some of these limitations by embedding the model into a larger process for systematic task assignment that includes other evaluation models. Others, such as the ability to identify and characterize influencing factors before the task assignment decision, may still hinder the application of the approach in every context. However, we see the principle of having various influencing factors that determine the applicability of a certain task allocation alternative as a general basis for systematic task allocation and for planning and managing global software development projects.

Acknowledgements The authors would like to thank all participants of the interview study for giving their time and for providing insights into the practices of distributed software development. Most 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 this paper.

References

1. Herbsleb JD, Moitra D (2001) Guest editors' introduction: Global software development. IEEE Softw 18(2):16–20. doi:10.1109/52.914732
2. Damian D, Moitra D (2006) Global software development: How far have we come? IEEE Softw 23(5):17–19. doi:10.1109/MS.2006.126
3. Herbsleb JD, Grinter RE (1999) Splitting the organization and integrating the code: Conway's law revisited. In: Proceedings of the 21st international conference on software engineering, pp 85–95
4. Krishna S, Sahay S, Walsham G (2004) Managing cross-cultural issues in global software outsourcing. Commun ACM 47(4):62–66. doi:10.1145/975817.975818
5. Casey V (2009) Leveraging or Exploiting Cultural Difference? In: Proceedings of the fourth IEEE international conference on global software engineering, pp 8–17. doi:10.1109/ICGSE.2009.9
6. Chua AL, Pan S (2006) Knowledge transfer in offshore insourcing. In: Proceedings of the 27th international conference on information systems, pp 1039–1053
7. Sakthivel S (2007) Managing risks in offshore systems development. Commun ACM 50(4):69–75. doi:10.1145/1232743.1232750
8. Herbsleb JD, Mockus A, Finholt TA, Grinter RE (2001) An empirical study of global software development: distance and speed. In: Proceedings of the 23rd international conference on software engineering, pp 81–90. doi:10.1109/ICSE.2001.919083
9. Casey V, Richardson I (2006) Uncovering the reality within virtual software teams. In: International conference on software engineering, Proc of the international workshop on global software development for the practitioner, pp 66-72. doi:10.1145/1138506.1138523
10. Fabriek M, Brand M, Brinkkemper S, Harmsen F, Helms RW (2008) Reasons for success and failure in offshore software development projects. In: European conference on information systems, pp 446–457
11. Carmel E, Abbott P (2007) Why 'nearshore' means that distance matters. Commun ACM 50(10):40–46. doi:10.1145/1290958.1290959
12. Seshagiri G (2006) GSD: Not a business necessity, but a march of folly. IEEE Softw 23(5):63f. doi:10.1109/MS.2006.138

13. Betz S, Mäkiö J (2007) 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, pp 33–46
14. Herbsleb JD (2007) Global software engineering: The future of socio-technical coordination. In: Proceedings of the future of software engineering, FOSE 2007, pp 188–198. doi:10.1109/FOSE.2007.11
15. Prikladnicki R, Yamaguti MH (2004) Risk management in global software development: a position paper. In: International conference on software engineering, Proc of the third international workshop on global software development, pp 18–20
16. Edwards HK, Kim JH, Park S, Al-Ani B (2008) Global software development: project decomposition and task allocation. In: International conference on business and information (BAI2008) Academy of Taiwan Information Systems Research. ISSN:1729-9322
17. Raffo D, Setamanit S (2005) A simulation model for global software development project. In: The international workshop on software process simulation and modeling
18. Bass M, Paulish D (2004) Global software development process research at Siemens. In: International conference on software engineering, Proc third International workshop on global software development, pp 8–11
19. Keil P, Paulish DJ, Sangwan R (2006) Cost estimation for global software development. In: International workshop on economics driven software engineering, pp 7–10. doi:10.1145/1139113.1139117
20. Lamersdorf A, Münch J (2009) 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, Proceedings of the fourth international conference on global software engineering, pp 322–327. doi:10.1109/ICGSE.2009.50
21. Mockus A, Weiss DM (2001) Globalization by chunking: a quantitative approach. IEEE Softw 18(2):30–37. doi:10.1109/52.914737
22. Lamersdorf A, Münch J, Rombach D (2008) Towards a multi-criteria development distribution Model: an analysis of existing task distribution approaches. In: Proceedings of the third international conference on global software development, pp 109–118. doi:10.1109/ICGSE.2008.15
23. Lamersdorf A, Münch J, Rombach D (2009) A survey on the state of the practice in distributed software development: criteria for task allocation. In: Proceedings of the fourth international conference on global software engineering, pp 41–50. doi:10.1109/ICGSE.2009.12
24. Lamersdorf A, Münch J, Rombach D (2009) A decision model for supporting task allocation processes in global software development. In: Proceedings of the 10th international conference on product focused software development and process improvement, pp 332–346. doi:10.1007/978-3-642-02152-7_25
25. Seaman C (2008) Qualitative methods. In: Shull F et al (eds) Guide to advanced empirical software engineering. Springer, Heidelberg, pp 35–62
26. Grinter RE, Herbsleb JD, Perry DE (1999) The geography of coordination: dealing with distance in R&D work. In: Proceedings ACM conference on supporting group work, pp 306–315. doi:10.1145/320297.320333
27. Westner MK, Strahringer S (2008) Evaluation criteria for selecting offshoring candidates: an analysis of practices in German businesses. J Inf Technol Manage 19(4):16–34
28. Mullick N, Bass M, Houda Z, Paulish DJ, Cataldo M, Herbsleb JD, Bass, L (2008) Siemens global studio project: experiences adopting an integrated GSD infrastructure. In: Proceedings first international conference on global software engineering, pp 203–212. doi:10.1109/ICGSE.2006.261234
29. Raghvinder S, Bass M, Mullick N, Paulish DJ, Kazmeier J (2006) Global software development handbook. Auerbach Publications, London
30. Madachy R (2007) Distributed global development parametric cost modeling. In: Proceedings international conference on software process, pp 159–168. doi:10.1007/978-3-540-72426-1_14
31. Boehm B, Abts C, Brown A, Chulani S, Clark B, Horowitz E, Madachy R, Reifer D, Steece B (2000) Software cost estimation with COCOMO II. Prentice-Hall, New Jersey
32. Sooraj P, Mohapatra PKJ (2008) Developing an Inter-site Coordination Index for Global Software Development. In: Proceedings of the third international conference on global software development, pp 119–128. doi:10.1109/ICGSE.2008.30
33. Setamanit S, Raffo D (2008) Identifying key success factors for globally distributed software development project using simulation: a case study. In: International proceedings conference on software process, pp 320–332. doi:10.1007/978-3-540-79588-9_28
34. Setamanit S, Wakeland WW, Raffo D (2007) Using simulation to evaluate global software development task allocation strategies. Softw Process Improv Pract 12(5):491–503. doi:10.1002/spip.v12:5
35. Fenton N, Marsh W, Neil M, Cates P, Forey S, Tailor M (2004) Making resource decisions for software projects. In: Proceedings of the 26th international conference on software engineering, pp 397–406
36. Cheng M, Wang W (1997) A linear programming model for integrated steel production and distribution planning. Int J Oper Prod Manag 17(6):592–610
37. MA Cohen, Moon S (1991) An integrated plant loading model with economies of scale and scope. Eur J Oper Res 50(3):266–279
38. Fleischmann B, Ferber S, Henrich P (2006) Strategic planning of BMW's global production network. Interfaces 36(3):194–208
39. Bokhari SH (1981) A shortest tree algorithm for optimal assignments across space and time in a distributed processor system. IEEE Trans Softw Eng 7(6):583–589. doi:10.1109/TSE.1981.226469
40. Amir Y, Awerbuch B, Barak A, Borgstrom RS, Keren A (2000) An opportunity cost approach for job assignment in a scalable computing cluster. IEEE Trans Parallel Distrib Syst 11(7):760–768. doi:10.1109/71.877834
41. Chu WW (1969) Optimal file allocation in a multiple computer system. IEEE Trans Comput 18(10):885–889. doi:10.1109/T-C.1969.222542
42. Ben-Gal I (2007) Bayesian networks. In: Ruggeri F, Kenett R, Faltin F (eds) Encyclopedia of statistics in quality and reliability. Wiley, New York
43. Fenton N, Hearty P, Neil M, Radliński Ł (2009) Software project and quality modelling using Bayesian networks. In: Meziane F, Vadera S (eds) Artificial intelligence applications for improved software engineering development: new prospects. Information Science Reference, pp 223–231
44. Pai GJ, Dugan JB (2007) Empirical analysis of software fault content and fault proneness using Bayesian methods. IEEE Trans Softw Eng 33(10):675–686. doi:10.1109/TSE.2007.70722
45. Treinen JJ, Miller-Frost SL (2006) Following the sun: case studies in global software development. IBM Syst J 45(4):773–783
46. Cozman FG (2001) JavaBayes—Bayesian networks in Java. http://www.cs.cmu.edu/~javabayes/ Retrieved at 09-15-08
47. Flanagan MT (2008) Java Scientific Library. http://www.ee.ucl.ac.uk/~mflanaga/java/ Retrieved at 09-16-08
48. Ebert C, De Neve P (2001) Surviving global software development. IEEE Softw 18(2):62–69. doi:10.1109/52.914748
49. Battin RD, Crocker R, Kreidler J, Subramanian K (2001) Leveraging resources in global software development. IEEE Softw 18(2):70–77. doi:10.1109/52.914750

50. Lindqvist E, Lundell B, Lings B (2006) Distributed development in an intra-national, intra-organizational context: an experience report. In: International conference on software engineering, Proc international workshop on global software development for the practitioner, pp 80–86. doi:10.1145/1138506.1138525
51. Heeks R, Krishna S, Nicholson B, Sahay S (2001) Synching or sinking: global software outsourcing relationships. IEEE Softw 18(2):54–60. doi:10.1109/52.914744
52. Kobitzsch W, Rombach HD, Feldmann RL (2001) Outsourcing in India. IEEE Softw 18(2):78–86. doi:10.1109/52.914751
53. Alami A, Wong B, McBride T (2008) Relationship issues in global software development enterprises. J Glob Inf Technol Manag 11(1):49–68
54. Gareiss R (2002) Analyzing the outsourcers. Information Week, Nov. 18, 2002
55. Kommeren R, Parviainan P (2007) Philips experiences in global distributed software development. Empir Softw Eng 12(6):647–660. doi:10.1007/s10664-007-9047-3
56. DeLone W, Espinosa JA, Lee G, Carmel E (2005) Bridging global boundaries for IS project success. In: Proceedings 38th Hawaii international conference on system sciences 48:2. doi:10.1109/HICSS.2005.126
57. Pilatti L, Audy J, Prikladnicki R (2006) Software configuration management over a global software development environment: lessons learned from a case study. In: International conference on software engineering, Proc international workshop on global software development for the practitioner, pp 45–50. doi:10.1145/1138506.1138517
58. Smite D, Moe NB (2007) Understanding lacking trust in global software teams: a multi-case study. In: Proceedings international conference on product focused software development and process improvement, pp 20–34. doi:10.1007/978-3-540-73460-4_6
59. Ramasubbu N, Balan RK (2007) Globally distributed software development project performance: an empirical analysis. In: Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, pp 125–134. doi:10.1145/1287624.1287643
60. Smite D (2004) Global software development project management—distance overcoming. In: European conference on software process improvement, pp 23–33. doi:10.1007/b102170
61. Oza NV, Hall T (2005) Difficulties in managing offshore software outsourcing relationships: an empirical analysis of 18 high maturity Indian software companies. J Inf Technol Case Appl Res 7(3):25–41
62. Komi-Sirvio S, Tihinen M (2005) Lessons learned by participants of distributed software development. Knowl Process Manag 12(2):108–122
63. Herbsleb JD, Paulish DJ, Bass M (2005) Global software development at Siemens: experience from nine projects. In: Proceedings international conference on software engineering, pp 524–533. doi:10.1145/1062455.1062550
64. Espinosa A, Slaughter SA, Kraut RE, Herbsleb JD (2007) Familiarity, complexity, and team performance in geographically distributed software development. Organ Sci 18(4):613–630. doi:10.1287/orsc.1070.0297
65. Herbsleb JD, Mockus A (2003) An empirical study of speed and communication in globally-distributed software development. IEEE Trans Softw Eng 29(6):481–494. doi:10.1109/TSE.2003.1205177
66. Espinosa JA, Nan N, Carmel E (2007) Do gradations of time zone separation make a difference in performance? A first laboratory study. In: Proceedings second international conference on global software engineering, pp 12-22. doi:10.1109/ICGSE.2007.20
67. Carmel E (1997) The explosion of global software teams. Computerworld 31(49):C6
68. Gurung A, Prater E (2006) A research framework for the impact of cultural differences on IT outsourcing. J Global Inf Technol Manag 9(1):24–43
69. Carmel E, Agarwal R (2001) Tactical approaches for alleviating distance in global software development. IEEE Softw 18(2):22–29. doi:10.1109/52.914734
70. Herbsleb JD, Mockus A, Finholt TA, Grinter RE (2000) Distance, dependencies, and delay in a global collaboration. In: Proc ACM 2000 conf computer-supported cooperative work, pp 319–328. doi:10.1145/358916.359003
71. Goldmann S, Münch J, Holz H (2000) Distributed process planning support with MILOS. Int J Softw Eng Knowl Eng 10(4):511–525
72. Münch J, Lamersdorf A (2009) Systematic task allocation evaluation in distributed software development. In: Meersman R, Herrero P, Dillon T (eds) OTM 2009 Workshops, LNCS 5872, pp 228–237. doi:10.1007/978-3-642-05290-3_34