Information Systems Engineering Supported by Cognitive Matchmaking

S.J. Overbeek[1], P. van Bommel[2], and H.A. (Erik) Proper[2,3]

[1] e-office B.V., Duwboot 20, 3991 CD Houten, The Netherlands, EU
Sietse.Overbeek@e-office.com
[2] Institute for Computing and Information Sciences, Radboud University Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, EU
P.vanBommel@cs.ru.nl
[3] Capgemini Nederland B.V., Papendorpseweg 100, 3528 BJ Utrecht, The Netherlands, EU
E.Proper@acm.org

Abstract. In daily practice, discrepancies may exist in the suitability match of actors and the tasks that have been allocated to them. Formal theory and the prototype of a cognitive matchmaker system are introduced as a solution to improve the fit between actors and tasks. A case study has been conducted to clarify how the proposed cognitive matchmaker system can be utilized in information systems engineering. The inductive-hypothetical research strategy has been applied when performing the case study.

Keywords: cognitive characteristics, matchmaking, task allocation.

1 Introduction

Globalization, the emergence of virtual communities and organizations, and growing product complexity has an impact on how actors (i.e. a human or a computer) fulfill tasks in organizations. Notably due to these developments, an actor working on a task may experience an increase in cognitive load while task performance decreases [1,2]. The system discussed in this paper matches cognitive characteristics supplied by actors and the cognitive characteristics required to fulfill tasks. This may achieve a better fit between actors and tasks. Cognitive characteristics can be the willpower to fulfill a task or maintaining awareness of the requirements to fulfill a task for example. These characteristics are also referred to as *volition* and *sentience*, respectively in cognitive literature [2,3]. Within the enterprise, the benefits of cognitive matchmaking can be found in at least three areas: Multi-agent systems, workflow management and business process reengineering (BPR). (1) Multi-agent systems incorporate several software agents that may work together to assist humans in performing their tasks [4]. One way of providing assistance is to match tasks with human actors to understand which tasks fit best with which human actors. (2) The primary task of a workflow management system is to enact case-driven business processes by

Z. Bellahsène and M. Léonard (Eds.): CAiSE 2008, LNCS 5074, pp. 495–509, 2008.
© Springer-Verlag Berlin Heidelberg 2008

joining several perspectives [5]. One of these perspectives is the *task* perspective. This perspective describes the elementary operations performed by actors while executing a task for a specific case. An example of a case is a tax declaration. Integration of cognitive matchmaking in a workflow management system may prescribe which available actors fit best with the tasks that are part of a case. This may improve the allocation of tasks to actors while enacting a business process. (3) BPR consists of computer-aided design of processes and automatic generation of process models to improve customer service [6]. The design and creation of processes and process models may be improved if the business process modeler knows beforehand which available actors best fit the tasks that need to be fulfilled as part of a newly designed business process.

The focus of the research reported in this paper, however, is to analyze how cognitive matchmaking can provide support for a project during which an information system is engineered. Therefore, a case study has been carried out at e-office, a company specialized in providing computer-aided support for human actors to assist them during office work. First, the framework for cognitive matchmaking that has been developed in earlier work [7] is introduced in section 2. Section 3 discusses the prototype, that has been developed proceeding from the framework. The conducted case study is explained in section 4. Section 5 briefly compares our study with other approaches in the field and outlines the benefits of our approach. Section 6 concludes this paper and gives an overview of future work.

2 Framework for Cognitive Matchmaking

The main goals of this paper are to discuss the prototype and the case study. However, it is necessary to briefly introduce the framework for cognitive matchmaking as is elaborated in [7]. First, the framework is illustrated in figure 1.

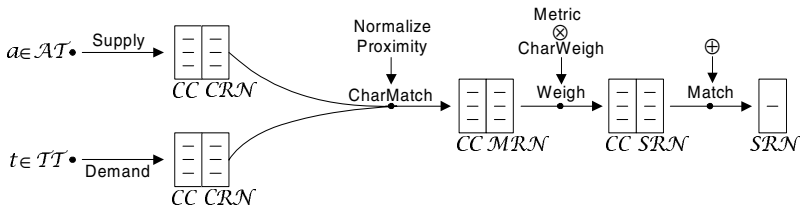

Fig. 1. Framework for cognitive matchmaking

The different concepts shown in figure 1 are functions that are necessary to calculate the eventual suitability match of an actor fulfilling a task. Even though the formal signature of these functions are not exhaustively repeated in this section, we will show some examples for clarification. First, the *supply* function shows the level on which an actor type, that is characterized by certain cognitive characteristics, offers a characteristic during task execution. The levels on which an actor type supplies a cognitive characteristic may vary over the natural numbers

from 0 up to and including 10. These levels are part of the *characteristic rank domain* indicated by the set $\mathcal{CRN}$. This ranking domain includes the rank values that can be used to indicate the level on which a characteristic can be supplied by an actor or demanded by a task. The *demand* function depicted in figure 1 shows the level on which a task of a certain type requires a certain cognitive characteristic if an actor wishes to fulfill the task.

The characteristic match or *CharMatch* function shown in figure 1 matches supply and demand of a specific characteristic. There is an optimal characteristic match if an actor offers a cognitive characteristic at the same level as a certain task requires the characteristic. A characteristic match is calculated for every cognitive characteristic that is supplied by an actor type and demanded by a task type. The result is part of the *match rank domain*, which may vary over the real values from 0 up to and including 1. An optimal characteristic match is indicated by the match rank value 0.5. This is because 0 indicates complete underqualification (an actor is not able to supply a certain characteristic at all) and 1 indicates complete overqualification (the supply of a certain characteristic is not necessary at all for a task whilst an actor supplies that certain characteristic at the highest level).

The weighed characteristic match function or *Weigh* function weighes the result of the characteristic match function. The user of the system may provide a weigh value to give more importance to a characteristic match result than another. The result is part of the *suitability rank domain*, which may vary over the real values from 0 up to and including 10. The results of the weigh function are then summated by the *Match* function which shows the *suitability match*. This suitability match is also expressed by a value from the suitability rank domain. To show how we have formalized the functions of the framework, the formal signature of e.g. the match function is modeled as follows [7]: $\mathsf{Match} : \mathcal{AT} \times \mathcal{TT} \rightarrow \mathcal{SRN}$. Note that the set $\mathcal{AT}$ contains actor types, the set $\mathcal{TT}$ contains task types and the set $\mathcal{SRN}$ contains suitability rank values. This function can be defined using the aforementioned functions:

$$\mathsf{Match}(\texttt{expert}, \texttt{synthesis}) \triangleq \bigoplus_{c \in \mathcal{CC}} \mathsf{Weigh}(c, \mathsf{CharMatch}(\texttt{expert}, \texttt{synthesis}))$$

For this example the suitability match of the *expert* actor type [7] and the *synthesis* task type [8] has been calculated. The expert uses his own knowledge to solve a problem. The expert is also able to combine and modify knowledge while solving a problem and is able to learn from that. A researcher often acts as an expert. A synthesis task is related with the actual utilization of acquired knowledge. An example is a student who utilizes knowledge (acquired by reading a book) while performing an exam. The definition of the match function shows that for every characteristic the weighed characteristic match function is executed and the results are then summated. The latter is shown by the $\oplus$ operator. This operator is used instead of the large Sigma because soft (linguistic) suitability rank values can also be used instead of hard (numerical) values. The match function can be expressed as follows: $\mathsf{Match}(\texttt{expert}, \texttt{synthesis}) = \mathbf{4.25}$, which shows that the numerical suitability match of the expert fulfilling the synthesis task is 4.25. This is a fairly good result, knowing that 5 is the best suitability match that

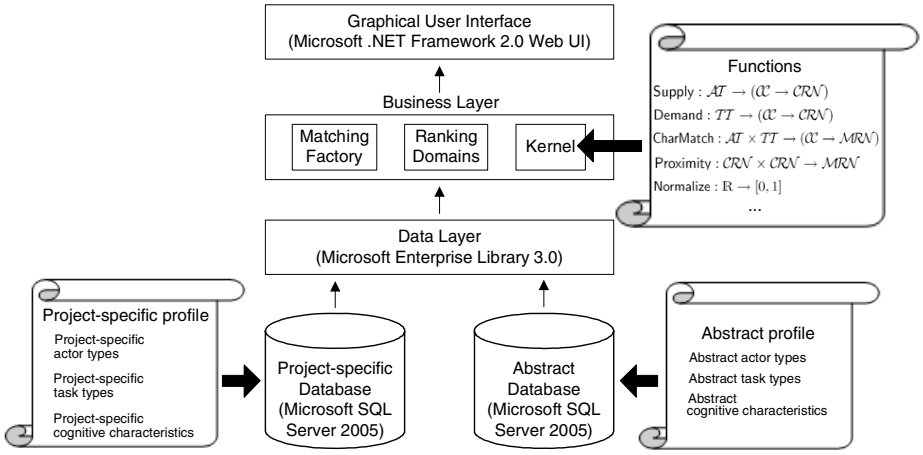

Fig. 2. Cognitive matchmaker system architecture

can be achieved. Section 3 contains a screen shot of the prototype in which this match result is shown. An implementation of the match function is shown as a code snippet in section 3. This is to illustrate that the formalisms mentioned in figure 1 are implemented in program code. The program code has been tested by means of unit tests after the implementation of all the formalisms included in the framework. This is to make sure that individual methods in the code are working properly.

Finally, a function has been introduced to determine the degree of certainty that an actor is suitable to fulfill a task [7]: $\mu : \mathbb{R} \rightarrow [0, 1]$. A linear certainty degree function can be defined as follows:

$$\mu(u) \triangleq \begin{cases} \frac{2}{\texttt{min}+\texttt{max}} \cdot u & \texttt{min} \leq u \leq \frac{\texttt{min}+\texttt{max}}{2} \\ \frac{-2}{\texttt{min}+\texttt{max}} \cdot u + 2 & \frac{\texttt{min}+\texttt{max}}{2} \leq u \leq \texttt{max} \end{cases}$$

In the implementation of the prototype the minimum and maximum values of a suitability match are equated to 0 respectively 10. Thus, $\texttt{min} = 0$ and $\texttt{max} = 10$. The degree of certainty that the expert is suitable to fulfill the synthesis task is: $\mu(4.25) = \frac{2}{0+10} \cdot 4.25 = 0.85$. This can be interpreted as being 85% sure that the expert is suitable enough to fulfill the synthesis task. It might be a good choice to let this actor fulfill the task, unless an available actor provides a better match.

3 Prototype of the Cognitive Matchmaker System

The prototype of the cognitive matchmaker system has been designed as a Web application according to the three tier software architecture depicted in figure 2. The graphical user interface is based on the Microsoft .NET Framework 2.0 Web UI namespace that provides classes and interfaces to create user interface elements. The business layer includes the main components of the application. The most important one is the kernel, which is an implementation of

the formal functions shown in figure 1 and in [7]. Furthermore, the 'matching factory' instantiates all the objects involved when a suitability match should be calculated and enables the application to follow the flow of the matchmaking process as depicted in figure 1. The business layer also includes an implementation of the possible ranking domains that can include characteristic ranks, match ranks and suitability ranks. The data layer includes code to interact with connected databases. The architecture shows that it is possible to include a project-specific database as well as a database including abstract types and characteristics. This signifies that the cognitive matchmaker system can compute matches between project-specific actor types and task types as well as between the abstract actor types and task types we have defined in [7]. Project-specific actor types and task types can be defined to categorize all the actors and tasks that are part of a specific project. For instance, a person called 'John Doe' can be categorized as a project-specific actor type 'developer', meaning that he acts as a software developer in a specific project. Section 4.1 includes the project-specific actor types and task types as part of the elaborated case study. On the contrary, the abstract types categorize actors and tasks based on the supplied respectively demanded cognitive characteristics. These types are explained in section 4.2. Dependent of the choice the user of the cognitive matchmaker system makes, the system communicates with one of the available databases to calculate matches. The data layer is based on the Microsoft Enterprise Library 3.0 that contains chunks of source code for e.g. data access.

The user of the system has to walk through six steps to calculate a suitability match. In the first step, the user should select an actor type and a task type for which a suitability match should be calculated. Suppose that the user selects the *expert* actor type and the *synthesis* task type. This causes the application to generate a list of all the cognitive characteristics that have been used to characterize the expert actor type and the synthesis task type. In the following step, the application displays on which level the expert supplies the involved characteristics and on which level the synthesis task demands the characteristics for successful task fulfillment. The next part shows the characteristic match results for all cognitive characteristics. Then, the user can provide the weigh values for the cognitive characteristics by entering them for each characteristic involved. Figure 3 shows the eventual suitability match result with the corresponding graph. Due to space limitations a screen shot of the suitability match screen is shown only. The resulting graph shows that in this case the suitability match of the expert fulfilling the synthesis task is 4.25. The certainty that the expert is able to fulfill the synthesis task is 85%. The implementation of the prototype is based on the framework shown in figure 1. For example, the code implementation of the suitability match function depicted in section 2 is shown in figure 4. The code implementation obviously shows that the match function takes an actor type and a task type as input parameters and a suitability rank value as output parameter just like the formal match function in our framework prescribed. Then, the results of the weighed characteristic match function are summated for each cognitive characteristic involved in the process of computing

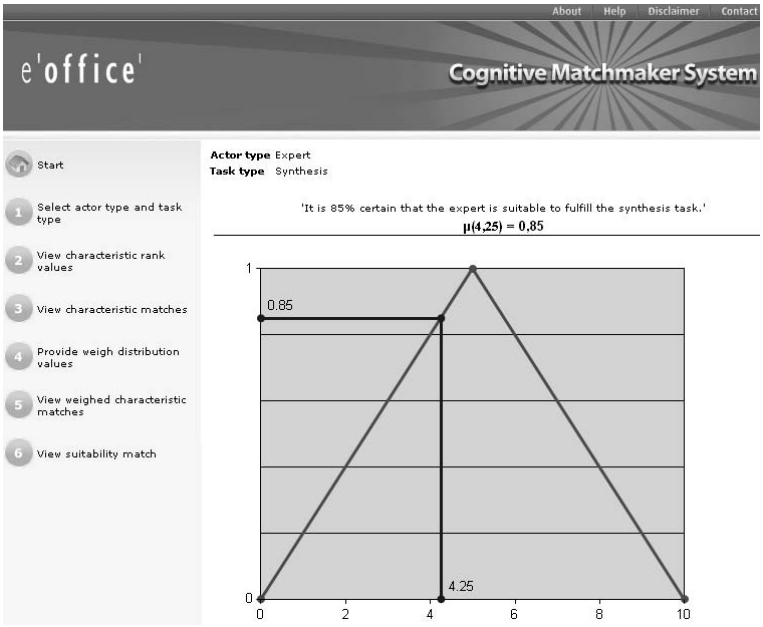

Fig. 3. Suitability match screen

```
public static SuitabilityRank Match(TaskType taskTypeObject, ActorType actorTypeObject) {

  SuitabilityRank SuitabilityRankObject = new SuitabilityRank();

  foreach (Characteristic CharacteristicObj in _matching.RetrieveCharacteristics()) {
    SuitabilityRankObject.RankValue += Weigh(CharacteristicObj,
    CharMatch(actorTypeObject, taskTypeObject,
    CharacteristicObj)).RankValue;
  }

  return SuitabilityRankObject;
}
```

Fig. 4. Source code of the suitability match function

the suitability match. This also corresponds with the definition of the suitability match function as shown in section 2.

4 Case Study and Evaluation

The case study that has been conducted is related with a recently completed information systems engineering (ISE) project at 'e-office'. This project has been concerned with the development of an 'Action Reporting Tool' (ART for short) for an international provider of banking and insurance services. ART is a software application that can generate risk reports for the user. This tool should assist risk management to better monitor and control insurance risks. The research strategy

for the case study has been derived from the inductive-hypothetical research strategy [9], which consists of five phases. Empirical knowledge of the problem domain is elicited in the *initiation* phase. Elicited empirical knowledge is applied in a descriptive conceptual model in the *abstraction* phase. The *theory formulation* phase is necessary to make the descriptive conceptual model prescriptive. The prescriptive conceptual model is empirically tested in the *implementation* phase. A comparison of phase 1 with the prescriptive empirical model of phase 4 is needed to fulfill the *evaluation* phase. The derived research strategy includes the following steps. (1) Description of the project phases in which the ISE project has been divided. The description includes project-specific actor types and task types and relations between them. (2) Abstraction of the results of phase 1 of the research strategy to our abstract cognitive model of actor types and task types. (3) Formulation of how the cognitive matchmaker system can be utilized in every project phase related to the actor types and task types involved in the project. (4) Analysis to identify the benefits if the system had been applied in the studied ISE project. (5) Evaluation by comparing phase 1 with phase 4.

4.1 Initiation

The ART project is based on the Microsoft Solutions Framework (MSF) information systems engineering method. The resulting tool is a Web application running on the Microsoft Office SharePoint Server 2007 platform (MOSS 2007 for short). Applications based on this platform are aimed to facilitate organizational collaboration, content management and business process management. The following project phases are determined as part of the ART project: The definition phase, the development phase, the acceptance phase and the implementation phase. The *definition* phase incorporates requirements engineering by means of interviews with the future users of the tool and conducting interactive workshops. The requirements engineering process is proceeded by the creation of several use cases to determine the interactions between the users and the tool. These use cases can be used as input to create screen mockups. The tool is developed in an iterative way during the *development* phase. The results after every iteration are tested before proceeding to the next iteration. The tool is integrally tested during the *acceptance* phase in conformity with a test plan. The acceptance test has been carried out by the banking and insurance service provider. Eventually, the final version of the tool is implemented at the service provider during the *implementation* phase. The actors participating in the project have been categorized into several project-specific actor types based on the MSF method. First, an *integrated program management (IPM) officer* can be identified. The IPM officer is responsible for organizational scheduling, planning and resource allocation. The *project manager* develops project plans, iteration plans and status reports. The project manager also mitigates risks. The *product manager* monitors the budget and the realization of the business case. The *infrastructure architect* focuses on server deployment and the services which run on them. Furthermore, the *solution architect* is responsible for defining both the organizational and physical structure of the application. Finally, the *lead developer* and the *developer* actor

types can be identified. The lead developer lends experience and skills to fellow developers. The developer is mainly responsible for building the product. The project manager of the ART project has created breakdowns of the tasks to be fulfilled. Using this documentation a project-specific task type categorization can be described together with the fulfilled tasks. Analysis of the project documentation also reveals which project-specific actor type performs a project-specific task type. The results of this analysis are shown in table 1.

Table 1. Project-specific actor types and task types

Task instance	Project phase	Project-specific task type	Project-specific actor type
Conduct interview with stakeholder	Definition	Elicitation task	Project manager
	Development		Product manager
			Solution architect
Conduct workshop with stakeholders	Definition	Elicitation task	Project manager
	Development		Product manager
			Solution architect
Design use case	Definition	Design task	Solution architect
	Development		Developer
Design mockup	Definition	Design task	Developer
Design risk report	Definition	Design task	Lead developer
			Developer
Write technical tool description	Definition	Documentation task	Developer
	Development		
Write project initiation document	Definition	Documentation task	Project manager
			Product manager
Determine hardware requirements	Definition	Documentation task	Infrastructure architect
Write security plan	Definition	Documentation task	Infrastructure architect
	Development		
Write project plan	Definition	Documentation task	Project manager
	Development		
Attend project meeting	All phases	Meeting task	All actor types
Attend steering committee meeting	All phases	Meeting task	IPM officer
Set up MOSS 2007 environment	Development	Code development task	Infrastructure architect
Build custom Web part	Development	Code development task	Developer
Configure Web part	Development	Code development task	Developer
Create risk report	Development	Code development task	Lead developer
			Developer
Implement security for tool	Development	Code development task	Infrastructure architect
Commit partial system test	Acceptance	System test task	Lead developer
			Developer
Commit integral system test	Acceptance	System test task	Lead developer
			Developer
Deploy completed tool	Implementation	Deployment task	Lead developer
			Developer

4.2 Abstraction

When performing the second phase of the inductive-hypothetical research strategy, it is possible to abstract the project-specific actor types and task types. First, it is shown how the project-specific task types can be abstracted to the abstract task types mentioned in [8]. Second, this section discusses how the project-specific actor types can be abstracted to the actor types mentioned in [7].

The distinguished abstract task types are the acquisition, synthesis and testing types. The project-specific task types depicted in table 1 can be abstracted to these types as follows. The mentioned elicitation tasks are typical knowledge *acquisition* tasks. The actors executing an elicitation task acquire knowledge by means of interviews or workshops. Design tasks can be abstracted as *synthesis* tasks. In a design task, the actor applies already acquired knowledge when designing a use case, mockup or risk report. Documentation tasks can also be classified as synthesis tasks. The documentation tasks mentioned in table 1

are related with the application of knowledge when writing a technical tool description, project initiation document, hardware requirements report, security plan and project plan. Next, meeting tasks are abstracted to acquisition tasks. During project meetings and steering committee meetings it is intended to acquire knowledge about e.g. project planning, project status and the remaining budget. Code development tasks can be viewed as synthesis tasks. These tasks are necessary to build the action reporting tool itself. The build process consisted of setting up the programming environment, and the creation of Web parts and risk reports. Web parts are the visual components that are part of a Microsoft SharePoint application which include functionality, such as: Listed announcements, a calendar, a discussion part, etc. *Testing* tasks are related with the project-specific system test tasks. In a testing task, earlier applied knowledge is thoroughly examined inducing an improvement of the specific knowledge applied. The partial and integral system tests are needed to identify and correct flaws in the action reporting tool. Finally, the deployment task can be abstracted to a synthesis task. Here, all relevant knowledge that is applied is related with a successful deployment of the system.

The distinguished abstract actor types are the collaborator, experiencer, expert, integrator and the transactor. Only the expert type has been mentioned up till now. The other types are explained as follows. The *collaborator* has the ability to exert an influence on state changes of knowledge involved during task fulfillment. During task fulfillment the collaborator is also able to improve its own cognitive abilities. However, a collaborator does not have complete awareness of all required knowledge to fulfill a task and requires others. An *experiencer* is aware of all the knowledge requirements to fulfill some task. An *integrator* is able to fulfill a task by working together and is able to initiate state changes of knowledge involved during task fulfillment. An integrator primarily wishes to acquire and apply knowledge of the highest possible quality. A *transactor* can fulfill a task without collaborating with others and is not required to cause modifications in the knowledge acquired and applied during task fulfillment. A project-specific actor can be classified as a certain abstract type while performing an abstract task. For instance, a project manager may be classified as a collaborator when fulfilling an acquisition task. However, if a project manager executes a synthesis task he may act differently and may be classified as a transactor instead. These abstractions materialize in the following phase of the research strategy.

4.3 Theory Formulation

The results of the previous phase of the research strategy are discussed throughout this section. The cognitive matchmaker system can be utilized in the four phases of the ART project. However, only the definition phase is elaborated in this section to understand how ISE can be supported by cognitive matchmaking. The approach can be used for the remaining phases in an identical manner. Based on sections 4.1 and 4.2 it is now possible to calculate the certainty that the actors involved in the definition phase of the ART project can successfully

Table 2. Cognitive matchmaking in the definition phase

Project-specific actor type	Task type	Actor type	Weigh values	Suitability match	Certainty
IPM officer	Acquisition	Collaborator	2, 1.5, 0.5, 3, 3	4.3	86%
Project manager	Acquisition	Collaborator	2, 1, 1, 3, 3	4.4	88%
	Synthesis	Transactor	1, 1.5, 1.5, 3, 3	4.85	97%
Product manager	Acquisition	Collaborator	2, 1, 1, 3, 3	4.4	88%
	Synthesis	Transactor	1, 1.5, 1.5, 3, 3	4.85	97%
Infrastructure architect	Acquisition	Experiencer	4, 3, 3	3.5	70%
	Synthesis	Expert	1.5, 1, 1, 1, 1.5, 2, 2	4.575	91.5%
Solution architect	Acquisition	Collaborator	1.5, 2, 1.5, 3, 2	4.4	88%
	Synthesis	Expert	1, 1.5, 1, 0.5, 1, 2, 3	4.8	96%
Lead developer	Acquisition	Experiencer	4, 3, 3	3.5	70%
	Synthesis	Expert	1, 2, 1, 1.5, 1, 1.5, 2	4.6	92%
Developer	Acquisition	Experiencer	3, 4, 3	3.3	66%
	Synthesis	Collaborator	2, 1.5, 2.5, 2, 2	4.4	88%

fulfill the tasks allocated to them. The results are depicted in table 2. The system can be utilized to calculate the matches between the actor types and task types involved in the definition phase. For this purpose, the way to abstract the project-specific actor types and task types as discussed in section 4.2 should be applied. The results of table 2 can be explained as follows. The solution architect, for instance, acts as a collaborator when working on an acquisition task and acts as an expert when working on a synthesis task respectively. In the definition phase, the solution architect conducts interviews and workshops, and attends project meetings. These tasks can be regarded as knowledge acquisition tasks. Five weigh values have to be provided by the user of the cognitive matchmaker system when calculating the suitability match of the collaborator fulfilling an acquisition task. The weigh values express the importance of the involved cognitive characteristics. The following weigh values are provided for the five characteristics involved: 1.5, 2, 1.5, 3 respectively 2. The cognitive characteristics used to characterize actor types and task types are further explained in [7,8]. At the moment, the weigh values have to be provided manually by the user. However, the next version of the prototype should include an algorithm that determines these weigh values dependent of how important a cognitive characteristic is in a certain combination of an actor type and a task type. The highest weigh value of 3 has been applied to the *satisfaction* characteristic. This is to make absolutely sure that the solution architect is pleased with the knowledge acquired and that no additional need for knowledge remains [8]. The cognitive matchmaker system then sums up the resulting weighed characteristic matches resulting in a suitability match of 4.4. The certainty that the solution architect acting as a collaborator can successfully fulfill an acquisition task is: $\mu(4.4) = \frac{2}{0+10} \cdot 4.4 = 0.88$ or $0.88 \cdot 100\% = 88\%$. The solution architect acts as an expert when working on a synthesis task during the definition phase. These synthesis tasks are related with the design of use cases. The solution architect should be able to use his own knowledge about use cases to correctly design them. The architect should also be able to combine and modify his own knowledge while designing use cases and he should also be able to learn from that process. The expert actor type matches very well with the synthesis task, because the result of the suitability match calculation is 4.8. This results in a certainty of 96%.

4.4 Implementation

The results from the theory formulation phase are now utilized to describe how ISE can be supported by cognitive matchmaking. The utilization of the system in ISE is described using three viewpoints. (1) The *design time* viewpoint embraces the situation before the project is initiated (before the definition phase starts). First, the project-specific actor types and the project-specific task types need to be conceived. If this is done, there are two options to choose from: Use the project-specific profile as a starting point or the abstract profile. The latter has been done in the case study as is elaborated in section 4.2. When using a project-specific profile as input for the system, a project-specific profile of actors and tasks should be generated. This has also been done in section 4.1. If not already entered in the project-specific database as is shown in figure 2, the actor and task data should be provided as a next step. The person who needs to allocate tasks to actors, the project manager for instance, can now calculate the suitability matches. Based on these results he can allocate tasks to actors before starting the project. (2) The *runtime* viewpoint is related to the recalculation of suitability matches if changes to task allocations are necessary during the enactment of an ISE project. This may be the case if a different actor needs to work on a task than the one specified in the project plan. The cognitive matchmaker system can then be used again to recalculate the suitability match. New tasks may also be introduced during the project that need to be allocated to actors. This may entail the need to calculate additional suitability matches during project enactment. (3) From a *post-mortem* point of view, task allocations in the project as a whole can be analyzed. The suitability matches for every actor / task combination in the ISE project may be compared to the actual results brought forward by the actors. Lessons learned should be recorded for future projects. This may help to better decide which actor types are suitable to work with which types of tasks.

4.5 Evaluation

In this section, the results of the initiation phase are compared with the results of the implementation phase. The evaluation of the initiation phase is related to the three viewpoints of the implementation phase. At *design time*, the choices leading to the project-specific actor types as shown in table 1 have not been argued in the ART project documentation. Recall that the project-specific actor types originate from the Microsoft Solutions Framework ISE method. Entering the actor types that MSF distinguishes in the project-specific database of the cognitive matchmaker system enables a better argued decision of which actor types to use in a project. For instance, the system test tasks shown in table 1 are performed by the (lead) developer actors. However, the MSF method also distinguishes the tester and test manager types. Including these actor types in the ART project may have improved the suitability matches related with the system test tasks. A difficulty is that the MSF method does not provide a clear description of the cognitive characteristics that characterize an actor type. The MSF method, however, provides a natural language description of each actor type included in the

method. Proceeding from these descriptions the administrator of the cognitive matchmaker system should be able to characterize the project-specific actor types by adding or reusing cognitive characteristics to the project-specific database.

At *runtime*, the results of the theory formulation phase included suitability matches for every actor / task combination differentiated to a specific project phase. These suitability matches may be reviewed after every project phase. The lowest certainty percentages shown in table 2 deserve special attention to discover the reasons of the lowest match results. For instance, table 2 shows that the developer acting as an experiencer has a certainty of 66% to successfully fulfill an acquisition task. When viewing table 1 it can be interpreted that the acquisition task performed by the developer in the definition phase is related with the attendance of project meetings. This may be caused in case a meeting is not very relevant for a developer. For instance, when a large part of a certain meeting is about project management issues a developer may not have a satisfied feeling after the meeting. Letting developers attend the most relevant meetings may increase the suitability matches for these acquisition tasks. In the same way, the other calculated matches can be analyzed for every project phase.

For instance, the testing tasks shown in table 1 deserve attention when comparing the actual project results with the suitability matches from a *post-mortem* viewpoint. According to table 1 the partial and integral system tests are conducted by the developer and lead developer types. Assume that it is 78.5% certain that the developer can successfully fulfill these testing tasks. The MSF method includes the tester actor type that may be more suitable to fulfill testing tasks in general. According to the MSF, a key goal for the tester is to find and report the significant bugs in the product by testing the product. Obviously, more bugs could have been found and solved after testing each iteration and the overall product by the tester actor type. In the current project situation, the developer has the responsibility for code development and testing as well. Usability issues also arose during the system test tasks. What can be seen in table 1 is that the developer is also responsible for designing the mockups. The responsibility of the developer to design, develop as well as test the system may have contributed to the existence of some usability problems. The MSF advocates the addition of a user experience architect in the project to increase the usability of the tool. According to the MSF, the user experience architect is responsible for the form and function of the user interface, its aesthetics and the overall product usability. Recall that designing mockups is a synthesis task. Assume the certainty that the developer can successfully fulfill a synthesis task in the definition phase is 88%. This is not a low percentage, but may further increase when the main focus of a developer is on developing code. For future projects it may be a smart idea to introduce a tester and a user experience architect.

5 Discussion

Literature indicates that cognitive matchmaking can be found in several areas of computer science. One of these initiatives is Cognitive Match Interface Design

(COMIND) [10]. COMIND is the designing of system processes so that they proceed and interact with the user in a manner that parallels the flow of the user's own thought processes. It consists of several principles, such as: The user should be able to express his needs to the computer with constructs which mirror the user's own thought processes. Another principle is the readiness of a computer to solve problems of the user in his / her area of need. Also, the computer should sanction flexibility just like the mind. The mind is regarded as a versatile and flexible problem solver. The authors tried to apply these principles when designing a medical information system. Unfortunately, a method for interface design that incorporates COMIND is not introduced. Only the medical information system case is elaborated. Creation of a COMIND framework including the proposed cognitive principles for user interface design would have possibly enabled reuse of COMIND in different areas. The existence of our cognitive matchmaker system framework does enable its specific application in many different areas.

Another interesting study is the cognitive matchmaking of students with e-learning system functionality [11]. A way of working is presented to design e-learning systems that better adapt to the cognitive characteristics of students. First, a taxonomy of learning styles is selected to classify the user. Next, techniques should be developed to introduce the adaptation into the system that fits the learning styles. The designed adaptation is then implemented on a computer. Finally, a selection of the technologies is made that are adequate for the adaptation. Besides this described way of working, a cognitive method or a system to match students and e-learning systems is not proposed. The mentioned concept of reflection can be very useful for our own work, though. Reflection is defined as the capability of a computational system to adjust itself to changing conditions. This can be seen on e.g. `http://maps.google.com`. The process of adaptation is made stronger since it is possible to create specific code depending on the supplied characteristics of the user when using the system. Adding reflection to our system may take situational elements into account when determining a match, for instance. Concretely, the actual availability of actors during the ART project may be included when allocating tasks to actors.

Jaspers et al. [12] argue that early involvement of cognitive matchmaking in ISE may be of importance to design systems that fully support the user's work practices. From this perspective, cognitive matchmaking is used for requirements engineering to match system requirements with the user's task behavior. To understand the task behavior of future users of a clinical system, the think aloud method has been applied [12]. Think aloud is a method that requires subjects to talk aloud while performing a task. This stimulates understanding of the supplied cognitive characteristics when performing a task. Unfortunately, the method has only been utilized to design a user interface for a clinical information system. The study lacks a more abstract framework that can be reused to design interfaces in general that better match task behavior of its users. Task-analysis methods such as the think aloud method can be useful when refining our research. For instance, these methods may be very valuable to improve the way we have characterized the abstract actor types and task types based on

cognitive characteristics. Jaspers et al. [12] also included a simplified model of the human cognitive system. Studying that model may further improve the way we interpret cognitive matchmaking processes.

6 Conclusions and Future Work

This paper describes how actors and tasks can be matched based on cognitive characteristics. Therefore, the framework for cognitive matchmaking developed in earlier work is mentioned. Proceeding from this framework the prototype implementation of a cognitive matchmaker system is demonstrated. An information systems engineering project provided the breeding ground for the case study in which the system has been evaluated. The ISE project has been concerned with the development of an 'Action Reporting Tool' for an international provider of banking and insurance services. This tool is a Web application that can generate risk reports. The suitability matches of the tasks allocated to the actors in the definition project phase have been determined using the cognitive matchmaker system. It can be concluded that the system can provide support for task allocation in at least three different ways: Before project initiation (at design time), during project enactment (at runtime) and after the project has finished (post-mortem). At design time, the person that needs to allocate tasks to actors can calculate the suitability matches. Tasks can be allocated to actors before starting the project based on these results. At runtime, suitability matches can be recalculated if changes to task allocations are necessary. The system can also be utilized to evaluate task allocations after every project phase. The calculated suitability matches can be compared with actual task performance. From a post-mortem point of view, the suitability matches for every actor / task combination in the project can be compared to the actual task results. Lessons learned may help to better decide which actor types are suitable to work with which task types. In this case, the cognitive matchmaker system is related with ISE. The system may also be usable in other areas, such as: Multi-agent systems, workflow management and BPR. Future work is concentrated on improving the theory as well as the prototype and further evaluation in case studies. More efforts of how the prototype could prove the usefulness of the framework can be called for. This may include testing the prototype in real settings with real users. At this moment, it is only possible to calculate a match based on one actor type and one task type. However, there are situations imaginable that multiple actors are working together to fulfill a set of tasks. It might be interesting to determine a match based on the total amount of actors and the total amount of tasks that the actors are fulfilling as a group. Besides these additions, the future system may consider situational elements. This may include the availability of actors as well as personal preferences and goals. Finally, the notion of human knowledge can be considered to determine which aspects of human knowledge, its development and its synergy in team work can be taken into consideration in the current version of the framework and the system.

References

1. Staab, S., Studer, R., Schnurr, H., Sure, Y.: Knowledge processes and ontologies. IEEE Intelligent Systems 16(1), 26–34 (2001)
2. Weir, C., Nebeker, J., Bret, L., Campo, R., Drews, F., LeBar, B.: A cognitive task analysis of information management strategies in a computerized provider order entry environment. Journal of the American Medical Informatics Association 14(1), 65–75 (2007)
3. Kako, E.: Thematic role properties of subjects and objects. Cognition 101(1), 1–42 (2006)
4. Shakshuki, E., Prabhu, O., Tomek, I.: FCVW agent framework. Information and Software Technology 48(6), 385–392 (2006)
5. van der Aalst, W., ter Hofstede, A.: Verification of workflow task structures: A Petri-net-based approach. Information Systems 25(1), 43–69 (2000)
6. R-Moreno, M., Borrajo, D., Cesta, A., Oddi, A.: Integrating planning and scheduling in workflow domains. Expert Systems with Applications 33(2), 389–406 (2007)
7. Overbeek, S., van Bommel, P., Proper, H., Rijsenbrij, D.: Matching cognitive characteristics of actors and tasks. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 371–380. Springer, Heidelberg (2007)
8. Overbeek, S., van Bommel, P., Proper, H., Rijsenbrij, D.: Characterizing knowledge intensive tasks indicating cognitive requirements – Scenarios in methods for specific tasks. In: Ralyté, J., Brinkkemper, S., Henderson-Sellers, B. (eds.) Proceedings of the IFIP TC8 / WG8.1 Working Conference on Situational Method Engineering: Fundamentals and Experiences., Geneva, Switzerland, vol. 244, pp. 100–114. Springer, Boston, USA (2007)
9. Sol, H.: Simulation in Information Systems. PhD thesis, University of Groningen, The Netherlands, EU (1982)
10. Coll, R., Coll, J.: Cognitive match interface design, a base concept for guiding the development of user friendly computer application packages. Journal of Medical Systems 13(4), 227–235 (1989)
11. Ruiz, M., Díaz, M., Soler, F., Pérez, J.: Adaptation in current e-learning systems. Computer Standards & Interfaces 30(1–2), 62–70 (2008)
12. Jaspers, M., Steen, T., van den Bos, C., Geenen, M.: The think aloud method: A guide to user interface design. International Journal of Medical Informatics 73(11–12), 781–795 (2004)