Expert Systems with Applications xxx (2009) xxx–xxx

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets

S.J. Overbeek [a,*], P. van Bommel [b], H.A. (Erik) Proper [b,c]

[a] Faculty of Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, 2600 GA Delft, The Netherlands
[b] Institute for Computing and Information Sciences, Radboud University, Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands
[c] Capgemini Nederland B.V., Papendorpseweg 100, 3528 BJ Utrecht, The Netherlands

ARTICLE INFO

Keywords:
Cognitive matchmaking
Knowledge exchange
Knowledge-intensive tasks
Knowledge market
Knowledge workers market

ABSTRACT

Actors require knowledge to improve or gain competencies in order to work successfully. Competencies are improved or gained by executing qualifying tasks. A knowledge market paradigm is introduced to improve the fit between supply and demand of knowledge required by actors performing such qualifying tasks. The eventual work performed by actors can be atomically divided into execution tasks. Discrepancies may exist in the suitability match of actors and the execution tasks that have been allocated to them. Therefore, a knowledge workers market paradigm is introduced as a possible solution to improve the fit between actors and execution tasks.

1. Introduction

The combination of actors and the competencies they possess are the necessary ingredients to fulfill work in organizations. An actor can be defined as a human or a computer that is able to perform a task. The tasks that can be performed by an actor can be differentiated into *qualifying* tasks and *execution* tasks. A qualifying task is executed by an actor if knowledge is required to improve competencies that have already been gained in the past or to gain new competencies. The term competence is used to mean not only possessing skills and qualifications, but also using those qualifications (Ritter & Gemünden, 2004). Note that knowledge can be regarded as 'wrapped' in information, whilst information is 'carried' by data (expressions in a symbol language) (Liang, 1994). An actor possesses competencies (gained by doing qualification tasks) to perform *execution* tasks. For the sake of this research we are specifically interested in *knowledge-intensive* execution tasks. A knowledge-intensive execution task is a task for which acquisition, application or testing of knowledge is necessary in order to successfully fulfill the task. This implies that tasks for which knowledge processing is not necessary are irrelevant for this study.

Globalization, the emergence of virtual communities and organizations, and growing product complexity have an impact on how actors fulfill execution 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 (Weir et al., 2007). When an actor performs an execution task, one or more *cognitive characteristics* are demanded by the task and supplied by the actor. For example, these characteristics can be the willpower to fulfill a task (volition) or maintaining awareness of the requirements to fulfill a task (sentience) (Kako, 2006; Weir et al., 2007). For this research we are concerned with *cognitive matchmaking* of actors and execution tasks. The characteristics demanded by execution tasks are matched with the characteristics supplied by the actors performing these tasks. Cognitive matchmaking may achieve a better fit between actors and tasks. Thus, we are interested in studying actors that perform qualifying tasks to improve their competencies or gain new competencies on the one hand. This part of the study is related with knowledge exchange in the knowledge market. Besides that, we are also concerned with cognitive matchmaking of these actors and the knowledge-intensive (execution) tasks that they perform. This part of the study is related with cognitive matchmaking in the knowledge workers market. This market dichotomy is modeled in Fig. 1. Fig. 1 can be explained by a practical example. Suppose that a computer science student requires to attend a requirements engineering course. In other words, the student demands knowledge about requirements engineering. In the context of Fig. 1 the student acts as a potential knowledge utilizer. The teacher that teaches the requirements engineering course is able to supply the student with relevant knowledge and acts as a knowledge supplier. At the end of the course, the student has to perform an exam to determine if he or she has obtained the required competencies to pass the course. The performance of such an exam can be seen as an execution task that has to be fulfilled by the student. Matching the cognitive characteristics demanded by the task and supplied

* Corresponding author. Tel.: +31 15 2785526; fax: +31 15 2783741.
E-mail addresses: S.J.Overbeek@tudelft.nl (S.J. Overbeek), P.VanBommel@cs.ru.nl (P. van Bommel), E.Proper@acm.org (H.A. (Erik) Proper).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.04.071

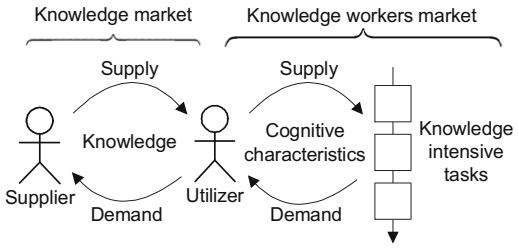

Fig. 1. Two market paradigms combined.

by the student can determine if the student can successfully fulfill the exam.

The paper is structured as follows. The knowledge market paradigm is discussed in Section 2 to reason about supply and demand of knowledge from a market perspective. The fundamentals of knowledge exchange in the knowledge market are explained in Section 3. The prototypical Web application discussed in Overbeek, van Bommel, Proper, and Rijsenbrij (2007) is modified and extended in Section 4 to operationalize the knowledge market part of this study. The knowledge workers market is elaborated in Section 5 to discover how the supply of cognitive characteristics by actors and the demand of characteristics by execution tasks can be matched. In Overbeek, van Bommel, Proper, and Rijsenbrij (2007) we have already provided a preliminary discussion about several types of knowledge-intensive tasks. This task type characterization is used as input for the theoretical framework presented in Section 5.2. The results from our initial work about matchmaking discussed in Overbeek, van Bommel, Proper, and Rijsenbrij (2007) are also integrated, extended and evaluated in Sections 5.1, 5.3 and 6. Section 7 briefly compares our models with other approaches in the field and outlines the benefits of our approach compared to others. Section 8 concludes this paper.

2. Knowledge market paradigm

Fig. 1 showed the concepts involved when an actor performs qualifying tasks. This situation in which knowledge is supplied, acquired and utilized is regarded as a *knowledge market*. Just as economic markets can be considered as a specific class of markets dealing with the trading of goods, services and money, the knowledge market deals with the trading of knowledge *assets*.

2.1. Knowledge fundamentals

Exploring the fundamentals of knowledge is necessary to gain a better understanding of that what is traded in a knowledge market. To determine possible knowledge types which can be traded, implicit knowledge and explicit knowledge can be elaborated at first (Nonaka & Takeuchi, 1995). *Implicit knowledge* comprises knowledge which is implicitly present in people's heads, such as skills which are difficult to make explicit. Implicit knowledge is closely related to what is generally experienced as intuition. *Explicit knowledge* comprises knowledge which can be expressed in terms of facts, rules, specifications or textual descriptions. Besides discerning implicit and explicit knowledge another relevant distinction can be made. Sometimes knowledge is present while one is not aware of that knowledge. This varies from hidden skills of actors via knowledge which is present in documents but not properly indexed, to knowledge which is hidden in undiscovered patterns in data collections (Hoppenbrouwers & Proper, 1999). In contrast to implicit and explicit knowledge the knowledge *status* is considered instead of the fundamental knowledge *type*. The following four knowledge types are then possible: (1) Implicit & concealed knowledge: e.g., competencies or expertise of an actor unknown to the organization. (2) Explicit & concealed knowledge: e.g., valuable insights concealed in available data collections (to be discovered by data mining). (3) Implicit & revealed knowledge: e.g., known expertise of an actor which can be appealed to. (4) Explicit & revealed knowledge: e.g., best-practice documentation, knowledge bases, scientific papers, etcetera. This implies that candidate knowledge assets for supply and exchange in the knowledge market can be typified as one of these combinations. The exchange of explicit & revealed knowledge is relatively straightforward compared to the exchange of implicit & concealed knowledge. In the latter case, concealed knowledge should be made revealed first before it can be discovered and it should be made explicit to make it exchangeable.

2.2. Knowledge market basics

In practice the difference between concealed and revealed knowledge is especially of importance. Revealed knowledge can be localized, but concealed knowledge can not be localized. Fig. 2 shows which concepts play a role in the knowledge market paradigm. The 'merchandize' within the paradigm consists of knowledge assets. In the context of the knowledge market, an asset can be defined as an entity that is accessible for the supplier which can provide knowledge to the utilizer. These *assets* are tradeable forms of revealed knowledge, which are transported physically by the transporter. If the transporter role is enacted by a computerized actor, this actor may equal an intelligent Web application that is able to mediate between the supplier and the broker. The transporter not only delivers the assets but can also check if these assets match the utilizer's demand. Implicit knowledge is also tradeable, because one can take its implicit knowledge to a

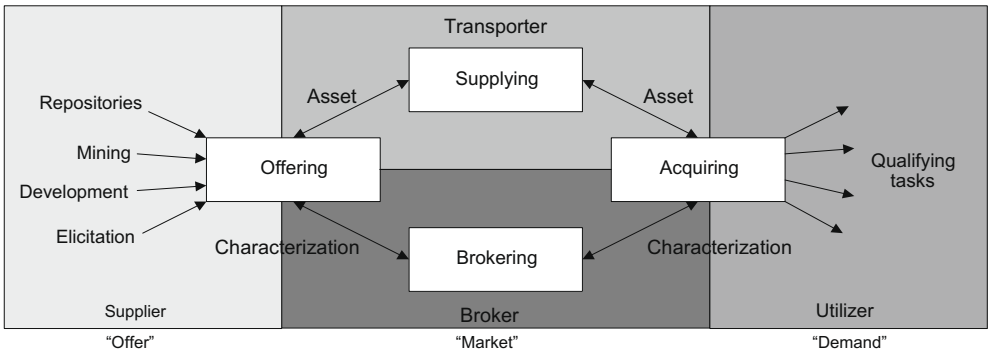

Fig. 2. A knowledge market paradigm.

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

situation where that implicit knowledge is wanted and made explicit (this is what e.g., physicians do when explaining a patient's status to a colleague).

The goal of a supplier is to deliver knowledge, which requires a 'client' who would like to utilize it. This is only possible if the supplier is able to make clear what is on offer. It is vital that the knowledge is correctly *characterized*. This is not always an easy job because terminology issues can throw a spanner in the works. Poor characterizations can inevitably lead to supplying irrelevant knowledge. On the supply side of the market various resources can be accessed: Repositories, data collections and warehouses, knowledge that is actively developed or experts that can be questioned (elicitation). A reliable supplier offers *revealed knowledge* which is localizable and available. It is possible to offer implicit knowledge, as long as is assured that it can be applied by the utilizer (albeit a certain *competence*). The sketched knowledge market model contains the following roles: a *broker* role, a *supplier* role, a *transporter* role and a *utilizer* role. These roles are further elaborated in Section 2.3. The potential utilizer is searching for knowledge, but does not know if that knowledge can be found. Often, a utilizer does not even know which knowledge can fulfill the need. The knowledge is concealed for the potential utilizer, but does certainly not have to be concealed for the supplier. Characterization is key here, which matches the knowledge demand with the knowledge to be supplied. The broker plays a very important role in matching supply and demand. The broker comes into play when potential utilisers do not know beforehand which knowledge is required to fulfill their needs.

2.3. Roles in the knowledge market

The set of roles as mentioned in Section 2.2 can formally be represented as: $\mathscr{RO} \triangleq \{\texttt{broker}, \texttt{supplier}, \texttt{transporter}, \texttt{utilizer}\}$. The roles in the knowledge market wish to achieve specific objectives and enable to abstract from the actors that enact the role. Note that knowledge markets exist that include less or more roles. This notion is also discussed in Section 7. A role enactment is a fulfillment of such a role by an actor, expressed by the function $\mathsf{Enact} : \mathscr{RE} \rightarrow \mathscr{RO}$, where $\mathscr{RE}$ is the set of all role enactments within the market. Given the role enactment e of a role $\mathsf{Enact}(e)$, we can view the actor that enacts a role as a function $\mathsf{Player} : \mathscr{RE} \rightarrow \mathscr{AC}$. Here, $\mathscr{AC}$ represents the specific set of actor instances. An actor and a role combination uniquely determines an enactment: $\mathsf{Player}(e_1) = \mathsf{Player}(e_2) \wedge \mathsf{Enact}(e_1) = \mathsf{Enact}(e_2) \Rightarrow e_1 = e_2$. For an enactment $e \in \mathscr{RE}$ the following function is introduced: $\_ \tilde{\rightsquigarrow} \_ \subseteq \mathscr{AC} \times \mathscr{RE} \times \mathscr{RO}$. This function can be defined as $a \overset{e}{\tilde{\rightsquigarrow}} r \triangleq \mathsf{Player}(e) = a \wedge \mathsf{Enact}(e) = r$ and subsequently $a \rightsquigarrow r \triangleq \exists_{e \in \mathscr{RE}}[a \overset{e}{\tilde{\rightsquigarrow}} r]$. This can be illustrated by the following example. Let an *associate professor* denoted by a be an actor that can play two roles. He either plays the role of type *broker* denoted by r_1, or the role of type *utilizer* denoted by r_2. Both $a \overset{e_1}{\tilde{\rightsquigarrow}} r_1$ and $a \overset{e_2}{\tilde{\rightsquigarrow}} r_2$ are enactments such that $\mathsf{Player}(e_1) = a, \mathsf{Player}(e_2) = a, \mathsf{Enact}(e_1) = r_1$ and $\mathsf{Enact}(e_1) = r_2$. Finally, a set of actors all enacting a certain role can be defined as follows: $\mathscr{AC}_r \triangleq \{a \in \mathscr{AC} | a \rightsquigarrow r\}$. If actor x enacts the utilizer role in the knowledge market this can be denoted as follows: $x \in \mathscr{AC}_{\texttt{utilizer}}$. Besides introducing the roles in the knowledge market, the possible qualifying tasks that can be performed by the utilizer in the market can now be introduced.

2.4. Tasks in the knowledge market

Recall that the fulfillment of qualifying tasks contribute to an actor's competence development. When performing such a task, an actor exchanges knowledge in the knowledge market until no more knowledge is needed. Four qualifying task types can be described to develop competencies by means of knowledge exchange (Wieser, Houdek, & Schneider, 2000): (1) An actor's ability to convey knowledge should improve by performing an *activation task*. Even actors that have expert knowledge of a subject may be unable to convey their knowledge. A typical reason is that actors do not know what kind of knowledge others need. Another reason is that actors often do not even know what they know. This is knowledge of the implicit & concealed type. Unconscious knowledge of this kind needs active help to become surfaced and voiced. This can be done by e.g., applying interview techniques, conducting workshops and by proper questioning. (2) There are different opportunities to capture knowledge as it surfaces. An actor's ability to capture knowledge should improve by performing a *collection task*. Knowledge may be verbalized by an actor so that it can be captured. In more complex cases, an actor's knowledge may be activated in daily work, but then there must be an easy way of capturing it. This requires some means of storing the knowledge and the actor should be motivated to capture it. (3) An actor's ability to store knowledge (non-electronically as well as electronically) should improve by performing a *storage task*. Everything from databases to the Internet is available to store gained knowledge. In practice, however, storing becomes a problem of prioritization and decision. Electronic storage is not always possible too and an actor has to depend on his or her own memory. Limited resources, and more especially limited motivation, may force an actor to develop a pragmatic and feasible way to document and store. (4) Pure delivery of knowledge is far from sufficient from a cognitive perspective. A lot of emphasis must be put on making this knowledge helpful or relevant. This is dubbed *reinfusion* of knowledge. An actor's ability to reinfuse knowledge should improve by performing a *reinfusion task*. Plain provision of knowledge is rarely helpful in general. It can be more beneficial when it is known for which task the knowledge is needed. Collecting and processing is fueled by the understanding for what task the knowledge is needed.

Activating, collecting, storing and reinfusing knowledge can be considered as important qualities of actors enacting the utilizer role in the knowledge market. After all, the utilizer performs the knowledge-intensive execution tasks in the knowledge workers market as can be seen in Fig. 1. To successfully fulfill these execution tasks, the competencies of the utilizer should be shaped in such a way that, if necessary, knowledge can be activated, collected, stored and reinfused. Now that the possible roles and qualifying tasks in the knowledge market paradigm are introduced, the fundamentals of knowledge exchange from a market perspective can be elaborated.

3. Knowledge exchange in the knowledge market

Knowledge is exchanged between the roles in the knowledge market. A fundamental model for knowledge exchange can now be introduced to understand how knowledge can be exchanged. In a scientific context, knowledge exchange has been defined as 'collaborative problem-solving between researchers and decision-makers', and should take place through the processes of prioritizing, planning, conducting and disseminating new research (Graham et al., 2006). On the level of the knowledge market, such a definition can be generalized as: 'Collaborative problem-solving between actors in the knowledge market'. However, what we intend by means of knowledge exchange is to diminish the knowledge need of an actor asking for knowledge. This means that we should construct a definition of knowledge exchange that fits the knowledge market. In order to define such a view on knowledge exchange, which includes the roles as part of the knowledge market, we propose that knowledge assets flow from: (1) The utilizer to the broker and vice versa. (2) The broker to the supplier and vice versa. (3) The supplier to the transporter and vice versa. (4) The

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

transporter to the utilizer and vice versa. The actor that wishes to receive knowledge benefits from a knowledge exchange event if the need for knowledge diminishes. That need for knowledge is influenced by what the actor already has retrieved in the past. The following function measures one's need for knowledge (van der Weide & van Bommel, 2006): $\mathsf{Need} : \wp(\mathscr{KA}) \times \mathscr{KA} \rightarrow [0,1]$. $\mathsf{Need}(S,k)$ is interpreted as the residual need for knowledge $k \in \mathscr{KA}$ after the set S has been presented to the potential utilizer, where $S \subseteq \mathscr{KA}$. Thus, knowledge exchange consists of the broadcasting and subsequent reception of knowledge between the roles in the knowledge market, until the utilizer has no more need for assets. Formally, no more knowledge exchange is necessary if $\mathsf{Need}(S,k) = 0$. Knowledge exchange can manifest on two levels in the market which require further explanation.

3.1. Knowledge levels

First, knowledge assets on the *instance level* contain value for an actor so that required knowledge is added to the actor's own knowledge profile. For a human actor, this knowledge profile (i.e., the knowledge that is *possessed* by an actor) is stored in the brains. For a computerized actor, the knowledge profile is stored in e.g., a database. For instance, when a software consultant requires knowledge about how to model workflows then this knowledge can be acquired from a colleague by means of instant messaging. The knowledge that is transported about workflow modeling can be viewed as instance level knowledge. Instance level knowledge can be exchanged between the supplier, transporter and utilizer roles. Second, knowledge assets on the *meta level* contain value so that an actor is able to understand which *instance level* assets are required for an actor that is asking for instance level assets. Hence, knowledge that is exchanged on the meta level reasons about instance level knowledge assets. Discussing about e.g., which modeling language the software consultant wishes to know more about can be viewed as meta level knowledge. A meta level of knowledge exchange always contains a formulation in terms of a question or a query which reasons about knowledge that an actor wants to receive. Meta level knowledge comprises the knowledge which is exchanged between the utilizer, the broker and the supplier in the process of *matching supply and demand*. The set of knowledge levels can be defined as follows: $\mathscr{KL} \triangleq \{\mathtt{instance}, \mathtt{meta}\}$. The formal foundations of knowledge exchange, including on which knowledge level an exchange event takes place, are discussed in the following section.

3.2. Formal foundations of knowledge exchange

Formally, the possible knowledge exchange events in the knowledge market are represented by the set $\mathscr{KE}$. An actor pair should participate in a knowledge exchange event. Otherwise, no knowledge can be exchanged. The participation of an actor pair in a knowledge exchange event is expressed by the following function: $\mathsf{Part} : \mathscr{KE} \rightarrow \wp(\mathscr{AC} \times \mathscr{AC})$. The expression $\mathsf{Part}(k) = \langle x, y \rangle$ denotes that an actor pair x and y participate in knowledge exchange event k. The next function determines the knowledge level of a knowledge exchange event in the market, where $\mathscr{KL}$ is the set of knowledge levels: $\mathsf{Level} : \mathscr{KE} \rightarrow \mathscr{KL}$. Two different notations can be introduced to indicate on which level knowledge is exchanged between an actor pair. Knowledge exchange between an actor pair $x, y \in \mathscr{AC}$ on the instance level can be formulated as follows: $\_ \leftrightsquigarrow \_ \subseteq \mathscr{AC} \times \mathscr{KE} \times \mathscr{AC}$. The definition of this function can be given as follows: $x \overset{k}{\leftrightsquigarrow} y \triangleq \mathsf{Part}(k) = \langle x, y \rangle \wedge \mathsf{Level}(k) = \mathtt{instance}$. The expression $x \overset{k}{\leftrightsquigarrow} y$ indicates that knowledge is exchanged in a knowledge exchange event k on the instance level between an actor pair x and y who operate in the knowledge market. Similarly, the following notation is used in case knowledge is exchanged on the meta level: $\_ \overset{\wedge}{\leftrightsquigarrow} \_ \subseteq \mathscr{AC} \times \mathscr{KE} \times \mathscr{AC}$. The definition resembles the previous knowledge exchange function: $x \overset{k}{\overset{\wedge}{\leftrightsquigarrow}} y \triangleq \mathsf{Part}(k) = \langle x, y \rangle \wedge \mathsf{Level}(k) = \mathtt{meta}$. A question and answer mechanism is introduced in the following section to enable knowledge exchange between an actor pair on both levels.

3.3. Question and answer

In a knowledge exchange event, an actor may ask a question (or write a query) and receive an answer on both knowledge levels i.e., on the meta level, an actor may ask who is able to supply certain instance level assets. For example, an actor can ask the following question: 'Who has knowledge about fuzzy logic?'. An answer can be uttered as follows: 'Jane Doe from the Soft Computing Research Center has knowledge about fuzzy logic'. On the instance level, an actor may have received a link to a Wikipedia entry about Jane Doe as an answer to a request for a Web page. After reading the Wiki, the actor may request to update the Wiki with additional knowledge about Jane Doe. Note that in case a question is not immediately answered in some knowledge exchange event (but in a later stage), an answer can be considered empty for that event. It may also be possible that a knowledge exchange event contains an answer but not a question. An example of such a case occurs if knowledge is transferred to an actor if a request for that knowledge has been uttered in an earlier knowledge exchange event. The functions for describing an answer and a question as part of a knowledge exchange event can be modeled as follows: $\mathsf{Anw}, \mathsf{Qst} : \mathscr{KE} \rightarrow \wp(\mathscr{KA})$. The expression $\mathsf{Anw}(k) = A$ shows that there is an answer $A \subseteq \mathscr{KA}$ that is part of a knowledge exchange event $k \in \mathscr{KE}$. This approach can be applied identically when describing a question. Questions and answers are part of the knowledge input and output that actors receive respectively produce. Therefore, knowledge input and output need to be explored.

3.4. Knowledge input and output provided on carriers

The knowledge input and output that an actor playing a role in the knowledge market consumes respectively generates in the process of knowledge exchange can be depicted as: $\mathsf{In} : \mathscr{AS} \rightarrow (\mathscr{AC} \rightarrow \wp(\mathscr{KA}))$ and $\mathsf{Out} : \mathscr{AC} \rightarrow \wp(\mathscr{KA})$. Here, $\mathscr{AS}$ is the set of actor states (which differ from each other over time). When an actor experiences knowledge, then this will lead to a change in the actor's knowledge or in a state change. In this paper we restrict ourselves to state changes caused by experiencing knowledge. We do not consider other state changes. For example, forgetting knowledge may be seen as a special change of state. An actor can only experience knowledge when receiving knowledge, so states are not included in the output function. The expression $\mathsf{In}_t(a) = \{k_1, k_2\}$ for instance shows that actor a receives assets k_1 and k_2 as input in state t. For notation simplicity, knowledge input is indicated by the expression $\mathsf{In}_t(a)$ if the actor state is relevant (indicating state t of actor a). The notation $\mathsf{In}(a)$ is used if actor states are not relevant.

The knowledge that can be experienced by actors is provided on knowledge carriers. A knowledge carrier can be defined as any entity that is accessible for any actor, and which can provide knowledge to other actors (Proper, 1999). Examples of knowledge carriers are: Web pages, databases, a human brain and aggregations/groupings of knowledge carriers. Formally, knowledge carriers are introduced as the set $\mathscr{KC}$, which is presumed to be closed under carrier composition (so any combination of given knowledge carriers is another knowledge carrier). When an actor experiences a knowledge carrier, then this actor will end up in a new state. This can be expressed by means of the experience function: $\ltimes : \mathscr{AS} \times \mathscr{KC} \rightarrow \mathscr{AS}$. When an actor who is in state $t \in \mathscr{AS}$ experiences a knowledge carrier $i \in \mathscr{KC}$, then this actor will end up in a new

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

state denoted as $\ltimes(t,i)$. When applying the infix notation this would result in: $t \ltimes i$. The latter notation is used in the remainder of this paper. In combination with the knowledge input function it is possible to express an actor's state change after experiencing knowledge included in the input. For instance, a state t of an actor $x \in \mathscr{AC}$ can change to state $t \ltimes i$ when experiencing knowledge carrier i. Suppose that this knowledge carrier 'carries' knowledge assets K. The knowledge input of actor x can be expressed as follows: $\mathsf{In}_{t \ltimes i}(x) = K$. A final property of the knowledge market that is discussed is related to knowledge similarities. Knowledge input and output may overlap or not. In a knowledge exchange event, for instance, the question and the eventual answer to that question must overlap in some way to please the actor posing the question.

3.5. Knowledge similarities

Firstly, the input or output of an actor pair participating in the knowledge market may be *similar* in some way. For example, such a situation can be represented by $\mathsf{In}(x) \cap \mathsf{Out}(y) \neq \emptyset$. In this case, it seems that the input of actor x is in some way similar to the output of actor y. To actually measure the similarities between assets of actors, Jaccard's similarity coefficient can be introduced (see e.g., van der Weide & van Bommel (2006)): $\mathsf{Jacc} : \wp(\mathscr{KA}) \times \wp(\mathscr{KA}) \rightarrow [0,1]$. To measure the differences respectively similarities between a pair of knowledge assets, the 'min' and 'max' functions are necessary: $\mathsf{Min}, \mathsf{Max} : \mathscr{KA} \times \mathscr{KA} \rightarrow [0,1]$. The expression $\mathsf{Min}(i,j) = 0$ shows that there are no differences between assets i and j i.e., they are equal. The expression $\mathsf{Min}(i,j) = 1$ shows that i and j are completely different and $\mathsf{Max}(i,j) = 0$ shows that there is no overlap. If the latter expression results in 1 then i and j are equal. Assume that the sets $X, Y \subseteq \mathscr{KA}$ are assets that are part of knowledge input respectively knowledge output. Similarities between X and Y can be measured as follows:

$$\mathsf{Jacc}(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{\sum_n \mathsf{Min}(i_n, j_n)}{\sum_n \mathsf{Max}(i_n, j_n)} \tag{1}$$

Now assume that an actor pair has acquired knowledge. Jaccard's coefficient normalizes intersection $\mathsf{In}(x) \cap \mathsf{In}(y) \neq \emptyset$ with the corresponding union in case both $\mathsf{In}(x)$ and $\mathsf{In}(y)$ are non-empty:

$$\mathsf{Jacc}(\mathsf{In}(x), \mathsf{In}(y)) = \frac{|\mathsf{In}(x) \cap \mathsf{In}(y)|}{|\mathsf{In}(x) \cup \mathsf{In}(y)|} = \frac{\sum_n \mathsf{Min}(i_n, j_n)}{\sum_n \mathsf{Max}(i_n, j_n)}$$

The coefficient expresses the degree in which $\mathsf{In}(x)$ and $\mathsf{In}(y)$ are similar on a $[0,1]$ scale. Overlap between output related knowledge assets can also be measured equally. If either $\mathsf{In}(x)$ or $\mathsf{In}(y)$ is empty, we have $\mathsf{Jacc}(\mathsf{In}(x), \mathsf{In}(y)) = 0$. Finally, $\mathsf{Jacc}(\emptyset, \emptyset) = 1$. Figs. 3 and 4 show that there are four possible situations of related assets that can be discerned during the process of knowledge exchange. Jaccard's coefficient will be used in Sections 3.6 and 3.7 to determine knowledge similarities for regular exchange events in the market. In order to have a visual representation of the discussed definitions regarding knowledge exchange in the knowledge market, an Object-Role Modeling (ORM) model is presented in Fig. 5. In such a model, ovals represent object types (which are counterparts of classes), whereas boxes represent relations between object types. For more details on Object-Role Modeling, see e.g., Halpin (2001). To understand how knowledge can be exchanged in the knowledge market when all four roles are enacted by actors a basic model of knowledge exchange and an advanced model of knowledge exchange can now be elaborated.

3.6. Basic knowledge exchange

The basic knowledge exchange model in the knowledge market consists of four knowledge exchange events. The first exchange event involves the utilizer and the supplier. Suppose $x \in \mathscr{AC}_{\mathtt{utilizer}}$ and $y \in \mathscr{AC}_{\mathtt{broker}}$. The first exchange event can then be depicted by the following equation:

$$\exists_{t \in \mathscr{T}} \exists_{i \in \mathscr{KA}} [x \overset{\wedge\wedge}{\underset{k}{\rightsquigarrow}} y \wedge \mathsf{In}_{t \ltimes i}(y) \subseteq \mathsf{Out}(x) \subseteq \mathsf{Qst}(k) \wedge \mathsf{Anw}(k) = \emptyset] \tag{2}$$

What can be derived from the equation is that the utilizer poses a question and the broker is able to answer it. This question contains knowledge assets on the meta level and is contained in the input of the broker and the output of the utilizer. In the basic knowledge exchange model, however, no direct answer is provided by the broker so the answer is empty in this case. Finally, the broker's state changes to $t \ltimes i$ after experiencing the question. Suppose $x \in \mathscr{AC}_{\mathtt{broker}}$ and $y \in \mathscr{AC}_{\mathtt{supplier}}$. With this in mind, knowledge exchange between the broker and the supplier can be modeled as follows:

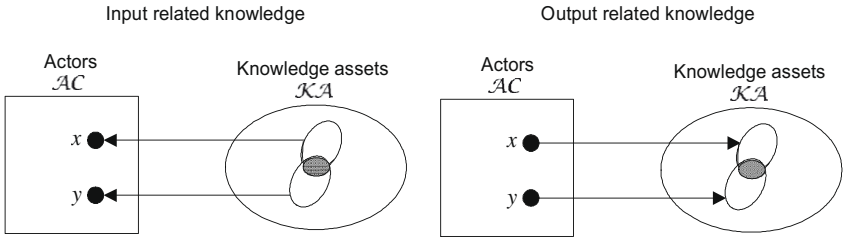

Fig. 3. Input respectively output related knowledge.

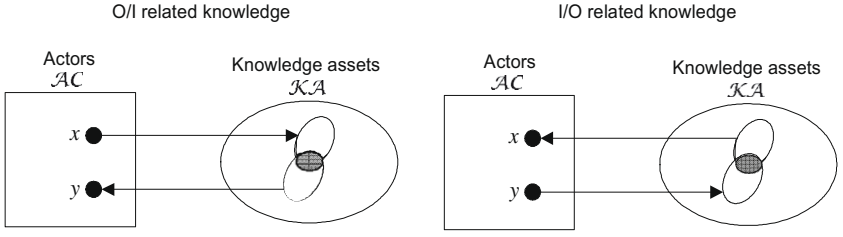

Fig. 4. O/I respectively I/O related knowledge.

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

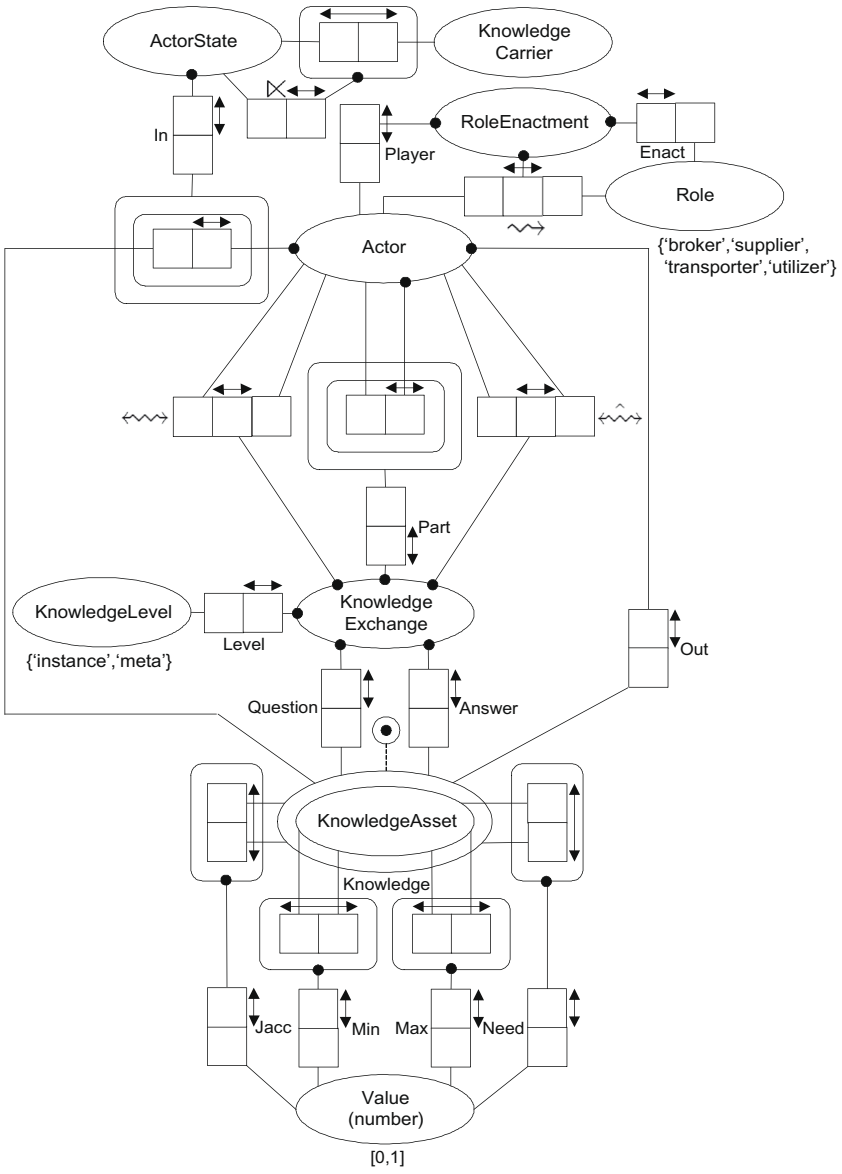

Fig. 5. Object-Role Modeling (ORM) model of knowledge exchange in the knowledge market.

$$\exists_{t\in\mathscr{AG}}\exists_{i\in\mathscr{KC}}[x \underset{k}{\hat{\rightsquigarrow}} y \wedge \mathsf{In}_{t\times i}(y) \subseteq \mathsf{Out}(x) \subseteq \mathsf{Qst}(k) \wedge \mathsf{Anw}(k) = \emptyset] \quad (3)$$

After the supplier has received this question from the broker, it may provide knowledge assets on the instance level to the transporter. This knowledge is in fact the answer on the question that was first posed by the utilizer. The knowledge exchange between the supplier and the transporter can be modeled as follows if $x \in \mathscr{AC}_{\mathtt{transporter}}$ and $y \in \mathscr{AC}_{\mathtt{supplier}}$:

$$\exists_{t\in\mathscr{AG}}\exists_{i\in\mathscr{KC}}[x \underset{k}{\rightsquigarrow} y \wedge \mathsf{In}_{t\times i}(x) \subseteq \mathsf{Out}(y) \subseteq \mathsf{Anw}(k) \wedge \mathsf{Qst}(k) = \emptyset] \quad (4)$$

Suppose $x \in \mathscr{AC}_{\mathtt{utilizer}}$ and $y \in \mathscr{AC}_{\mathtt{transporter}}$, then the final knowledge exchange event can be depicted as follows:

$$\exists_{t\in\mathscr{AG}}\exists_{i\in\mathscr{KC}}[x \underset{k}{\rightsquigarrow} y \wedge \mathsf{In}_{t\times i}(x) \subseteq \mathsf{Out}(y) \subseteq \mathsf{Anw}(k) \wedge \mathsf{Qst} = \emptyset] \quad (5)$$

Using Jaccard's similarity coefficient, it is straightforward that knowledge similarity is maximum for e.g., the knowledge output of the utilizer and the knowledge input of the broker. In case $x \in \mathscr{AC}_{\mathtt{utilizer}}$ and $y \in \mathscr{AC}_{\mathtt{broker}}$ this can be calculated as follows:

$$\mathsf{Jacc}(\mathsf{Out}(x), \mathsf{In}(y)) = \frac{|\mathsf{Out}(x) \cap \mathsf{In}(y)|}{|\mathsf{Out}(x) \cup \mathsf{In}(y)|} = 1$$

Jaccard's coefficient will never result in 0 when using the coefficient for the above instantiation of the basic model because all knowledge input and output are related to each other in some way. The coefficient results in a value greater than 0 but smaller than 1 when measuring the similarities between an actor's input (or output) of instance level knowledge and an actor's input (or output) of meta level knowledge. This is the case if e.g., $x \in \mathscr{AC}_{\mathtt{broker}}$ and $y \in \mathscr{AC}_{\mathtt{transporter}}$: $\mathsf{Jacc}(\mathsf{In}(x), \mathsf{In}(y)) > 0 \wedge \mathsf{Jacc}(\mathsf{In}(x), \mathsf{In}(y)) < 1$. This is because the input or output is not exactly the same on the two levels, but they are related with respect to content. Knowledge similarities can be calculated for $8^2 - 8 = 56$ different input and output comparisons in the basic model.

3.7. Advanced knowledge exchange

In the basic model, questions were only posed on the meta level and the set of answers in the exchange events were empty.

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

Furthermore, answers were only provided on the instance level exchange events and the set of questions in those events were empty. Advanced knowledge exchange extends the basic model by completing the Q&A cycles on both knowledge levels. The first knowledge exchange event in the advanced model concerns meta level knowledge exchange between the utilizer and the broker, where $x \in \mathscr{AC}_{\texttt{utilizer}}$ and $y \in \mathscr{AC}_{\texttt{broker}}$:

$$\exists_{t_1,t_2\in\mathscr{AS}}\exists_{i_1,i_2\in\mathscr{KC}}[x \overset{\wedge}{\underset{k}{\sim}} y \wedge \mathsf{In}_{t_1\times i_1}(y) \subseteq \mathsf{Out}(x) \subseteq \mathsf{Qst}(k) \wedge \mathsf{In}_{t_2\times i_2}(x) \subseteq \mathsf{Out}(y) \subseteq \mathsf{Anw}(k)] \quad (6)$$

A possible situation that instantiates this first exchange event in the advanced model may occur after the utilizer has characterized the need for those knowledge assets. Firstly, the utilizer provides the broker with a characterization of this knowledge need. This can be done by asking a question in case the broker is a human actor or by sending a query in case the broker is a computerized actor. Before the broker can send an answer, it is determined whether or not the potential utilizer has asked the right question or has typed the right query to fulfill the need. Any suggestions that may improve the characterization are then returned to the utilizer. Suppose $x \in \mathscr{AC}_{\texttt{broker}}$ and $y \in \mathscr{AC}_{\texttt{supplier}}$. Knowledge exchange between the broker and the supplier can now be modeled as follows in the advanced model:

$$\exists_{t_1,t_2\in\mathscr{AS}}\exists_{i_1,i_2\in\mathscr{KC}}[x \overset{\wedge}{\underset{k}{\sim}} y \wedge \mathsf{In}_{t_1\times i_1}(y) \subseteq \mathsf{Out}(x) \subseteq \mathsf{Qst}(k) \wedge \mathsf{In}_{t_2\times i_2}(x) \subseteq \mathsf{Out}(y) \subseteq \mathsf{Anw}(k)] \quad (7)$$

Assume that the broker has acquired a characterization of the knowledge need from the utilizer. The broker can use this characterization to find relevant suppliers that are able to supply the assets needed by the utilizer. The broker can acquire information about relevant suppliers and candidate knowledge assets to be supplied by asking the supplier for this information. This may complete a knowledge exchange event between the broker and the supplier. Suppose $x \in \mathscr{AC}_{\texttt{supplier}}$ and $y \in \mathscr{AC}_{\texttt{transporter}}$. Knowledge exchange between the supplier and the transporter is depicted as follows in the advanced model:

$$\exists_{t_1,t_2\in\mathscr{AS}}\exists_{i_1,i_2\in\mathscr{KC}}[x \overset{\sim}{\underset{k}{\sim}} y \wedge \mathsf{In}_{t_1\times i_1}(y) \subseteq \mathsf{Out}(x) \subseteq \mathsf{Qst}(k) \wedge \mathsf{In}_{t_2\times i_2}(x) \subseteq \mathsf{Out}(y) \subseteq \mathsf{Anw}(k)] \quad (8)$$

Subsequently, the supplier can send the required assets to the utilizer via the transporter. This involves the aforementioned third knowledge exchange equation in the advanced model. The final knowledge exchange equation is necessary to express the communication between the transporter and the utilizer, where $x \in \mathscr{AC}_{\texttt{transporter}}$ and $y \in \mathscr{AC}_{\texttt{utilizer}}$:

$$\exists_{t_1,t_2\in\mathscr{AS}}\exists_{i_1,i_2\in\mathscr{KC}}[x \overset{\sim}{\underset{k}{\sim}} y \wedge \mathsf{In}_{t_1\times i_1}(y) \subseteq \mathsf{Out}(x) \subseteq \mathsf{Qst}(k) \wedge \mathsf{In}_{t_2\times i_2}(x) \subseteq \mathsf{Out}(y) \subseteq \mathsf{Anw}(k)] \quad (9)$$

It would be too trivial if simply transporting assets from the supplier to the broker was the only activity of the transporter in the advanced model. It is also possible to exchange knowledge from the utilizer to the supplier via the transporter on the instance level. This is typically the case if the knowledge that is supplied is *paraphrased*. Suppose that the utilizer has acquired images from the transporter showing red tomatoes. The transporter asks if the images indeed contain the red tomatoes that the utilizer had in mind. Subsequently, the utilizer may answer with e.g., 'Yes, those images depict the tomatoes I was looking for', or, 'No, I meant younger tomatoes that are more greenly colored'. Full knowledge similarities can be identified in the knowledge exchange events between actors that enact neighboring roles. If $x \in \mathscr{AC}_{\texttt{utilizer}}$ and $y \in \mathscr{AC}_{\texttt{broker}}$, an example of such a calculation when using Jaccard's coefficient can be given as follows: $\mathsf{Jacc}(\mathsf{Out}(x), \mathsf{In}(y)) = 1$. Knowledge similarities can be calculated for $16^2 - 16 = 240$ different input and output comparisons in the advanced knowledge exchange model. A 'knowledge exchange cycle' can now be introduced to discover the full potential of the knowledge market paradigm.

3.8. Knowledge exchange cycle in the knowledge market

The knowledge exchange cycle is a specific instantiation of the advanced model consisting of sixteen steps. Considering the knowledge need function from Section 3, the initial need for an asset at the start of a knowledge exchange cycle by an actor playing the role of utilizer is denoted as $\mathsf{Need}(S,k) > 0$, where $k \in \mathscr{K}\mathscr{A}$ is (part of) the input that the utilizer wishes to receive from the transporter. Here, S is the current knowledge profile of the utilizer. The retrieved assets after a cycle has been completed should diminish the utilizer's knowledge need compared to the need that the utilizer had at the start of a knowledge exchange cycle. This can be expressed by the following equation for a single knowledge asset $k : S \subseteq T \Rightarrow \mathsf{Need}(S,k) \geqslant \mathsf{Need}(T,k)$. Here, S is the knowledge profile of the utilizer before receiving an asset from the transporter and $T \subseteq \mathscr{K}\mathscr{A}$ is the knowledge profile of the utilizer after receiving an instance level knowledge asset. The steps of the cycle can be described as follows:

(1) The potential utilizer characterizes the need for those knowledge assets that he or she wishes to receive.
(2) The potential utilizer provides the broker with a characterization of this knowledge need. This can be done by:
   (a) Asking a question in case any human actors are involved. Continue to step 3 or to step 7.
   (b) Sending a query if at least the broker is a computerized actor. Continue to step 3 or to step 7.
(3) Using this characterization, the broker then tries to find out if the potential utilizer has asked the right question or has typed the right query to fulfill the knowledge need.
(4) Any suggestions that may improve the characterization are returned to the potential utilizer.
(5) Based on these results, the potential utilizer may:
   (a) Revise his question or query. Return to step 2a or to step 2b.
   (b) Acknowledge the characterization.
(6) The potential utilizer then sends the final characterization to the broker.
(7) The broker uses the characterization to find relevant suppliers that are able to supply the knowledge assets needed by the utilizer.
(8) The broker acquires the following information from the supplier:
   (a) Information about relevant suppliers.
   (b) Insight in candidate knowledge assets that are suitable for supply.
(9) Information about the suppliers that have been identified, together with information about the knowledge assets that can be supplied, are then submitted to the potential utilizer.
(10) The potential utilizer determines the knowledge assets that he or she wishes to receive from a certain supplier.
(11) The potential utilizer provides the broker with a request to obtain knowledge assets from a certain supplier. This can be done by:
   (a) Asking a question in case any human actors are involved.
   (b) Sending a query if at least the broker is a computerized actor.

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

(12) The broker passes this request to the relevant supplier.
(13) The transporter acquires the requested knowledge assets from the supplier.
(14) The utilizer acquires the following from the transporter:
  (a) The requested knowledge assets.
  (b) A remark to find out if the transmitted assets were also intended by the utilizer.
(15) Based on these results, the utilizer may:
  (a) Request to receive an alteration of the assets. Continue to step 16.
  (b) Accept the received assets. This ends the cycle.
(16) The transporter passes this request to the relevant supplier. Return to step 13.

In a knowledge cycle, steps 1 up to and including 12 consist of the exchange of meta knowledge. Steps 13 up to and including 16 include the actual transport of instance level knowledge. Thus, knowledge exchange in the knowledge market largely consists of the exchange of meta knowledge. This stipulates the importance of correctly characterizing a knowledge need and matching supply and demand. The knowledge exchange cycle can be visualized by means of a workflow diagram. Workflow diagrams describe case-driven business processes by joining several perspectives (van der Aalst & ter Hofstede, 2000). One of these perspectives is the *control-flow* perspective. In this perspective, workflow schemas are defined to specify which tasks need to be executed and in what order. The control-flow perspective can be used when modeling the cycle in a workflow diagram. The tasks in such a diagram constitute of the sixteen steps that have to be carried out. The order of performing a step in the cycle can also be determined. The steps of the knowledge exchange cycle can be aggregated to four composite steps. A composite step contains underlying atomic steps that are part of the higher-level composite one. The main workflow diagram containing the composite steps of the cycle are shown in Fig. 6. The workflow modeling language YAWL (Yet Another Workflow Language) has been used to create the different diagrams. YAWL has a formal foundation, based upon Petri-nets. A condition is depicted by a circle. A composite step or task is depicted by a square with a double border. An atomic step or task is depicted by a square with a single border. An arrow indicates the flow of the diagram. For more details on YAWL, see e.g., van der Aalst and ter Hofstede (2005). The first composite step boils down to the atomic steps that are part of the process to characterize the utilizer's knowledge need. This results in a second workflow diagram as is shown in Fig. 7. Note that there are several OR-splits and OR-joins in the diagram. An OR-split and an OR-join are indicated by a diamond. An OR-split means that one can choose which step needs to be performed next. After performing the 'provide characterization' step, for instance, one can choose to ask a question or send a query as a next step. Therefore, the 'provide characterization' step in the workflow diagram contains an OR-split. After choosing a step to perform the regular control-flow is resumed again by an OR-join. Another peculiar OR-split is situated on the 'ask question' and 'send query' steps. To increase the chance that the supplied knowledge assets are indeed what the potential utilizer needs one may choose to let the broker check the utilizer's characterization for correctness. However, if e.g., time is scarce, or if the broker is just not capable of checking such a characterization this step can

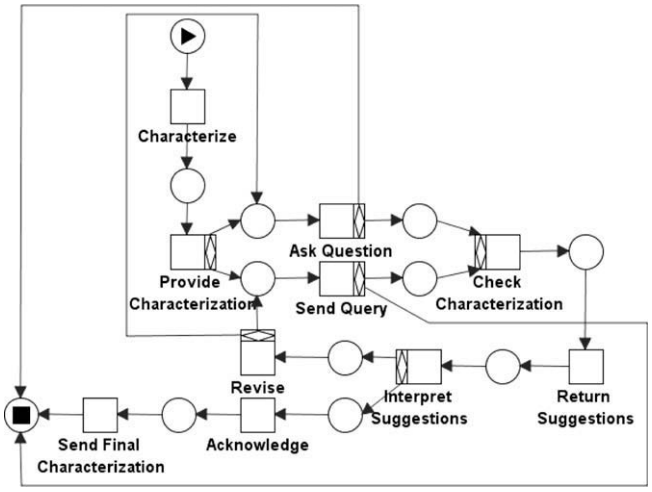

Fig. 7. Workflow diagram concerning characterization of knowledge need.

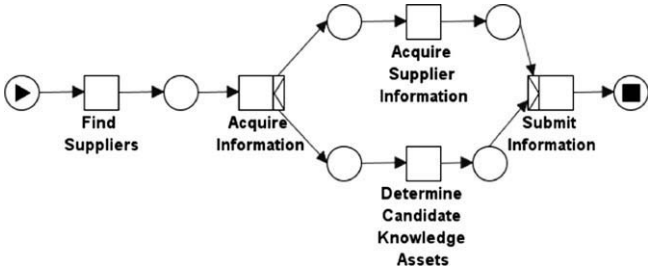

Fig. 8. Workflow diagram concerning identification of suppliers.

be skipped. *Google*, for instance, is not capable of interpreting whether or not the user's search query is correctly uttered to acquire the needed knowledge assets. If this check is skipped, the composite step 'identify suppliers' will be executed. The second composite step includes the atomic steps to identify and select relevant suppliers that can deliver the needed knowledge assets for the utilizer. This results in a third workflow diagram as is shown by Fig. 8. Note that the diagram contains an AND-split and an AND-join. An AND-split signifies that all the tasks following the split need to be fulfilled. In the case of this diagram the broker needs to acquire supplier information and the broker also needs to determine candidate knowledge assets that are suitable to supply. Fig. 9 contains the workflow diagram showing the atomic steps that need to be performed to indicate which assets have to be transported to the utilizer. The final workflow diagram that completes the knowledge exchange cycle shows the atomic steps to physically *transport* the requested assets to the utilizer and check whether or not the utilizer is satisfied. This diagram is shown in Fig. 10. If $\mathrm{Need}(S,k) > 0$, then another knowledge exchange cycle might be desirable. If the utilizer has no more need for knowledge, then required knowledge exchange cycles end.

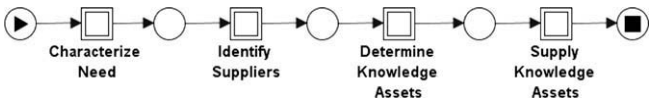

Fig. 6. Main workflow diagram of the knowledge exchange cycle.

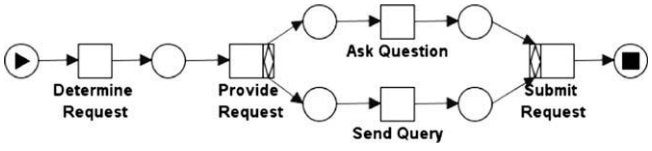

Fig. 9. Workflow diagram concerning the request of knowledge assets.

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

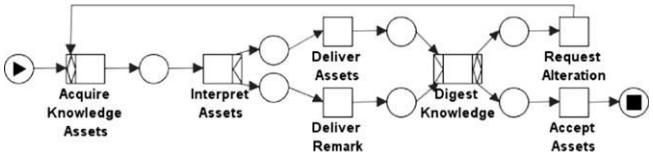

Fig. 10. Workflow diagram concerning the transportation of knowledge assets.

4. Application of the knowledge market

An application of the knowledge market can be materialized by a practical example of the knowledge exchange cycle. Suppose that a math teacher named 'John Doe' wishes to acquire specific knowledge about the Fourier series. Therefore, he characterizes his knowledge need as follows: 'I would like to acquire an overview of the network of Web pages that contain knowledge about the Fourier series'. The math teacher decides to use the DEXAR: *Discovery and eXchange of Revealed knowledge* Web application as a broker. This application assists the potential utilizer on the meta level by matching supply and demand of knowledge assets. On the instance level, Dexar can act as a *transporter* by interacting with the supplier and the utilizer. For more details about the Dexar prototype, see Overbeek et al. (2007). For this specific instantiation of the knowledge exchange cycle it can be determined that $w, z \rightsquigarrow$DEXAR and $x \rightsquigarrow$John_Doe. It remains to be seen which actor will enact the supplier role. To actually initiate the cycle, the following query is submitted to the broker: 'Network of Web pages about Fourier series'. Fig. 11 shows how Dexar makes an inventory of the teacher's knowledge need by determining his intention.

These first steps initiate Eq. (6) of Section 3.7. In this case, the knowledge *question* is postulated by the utilizer by means of the Fourier series query and the knowledge *answer* is provided by the gathered inventory. Obviously, the teacher wishes to acquire a graph of interconnected Web pages that are related with the Fourier series. Dexar has acquired enough meta knowledge at this point from the teacher. Therefore, relevant suppliers can now be sought that are able to provide a graph of interconnected Web pages about the Fourier series. Possible suppliers that are found by Dexar are shown in Fig. 12.

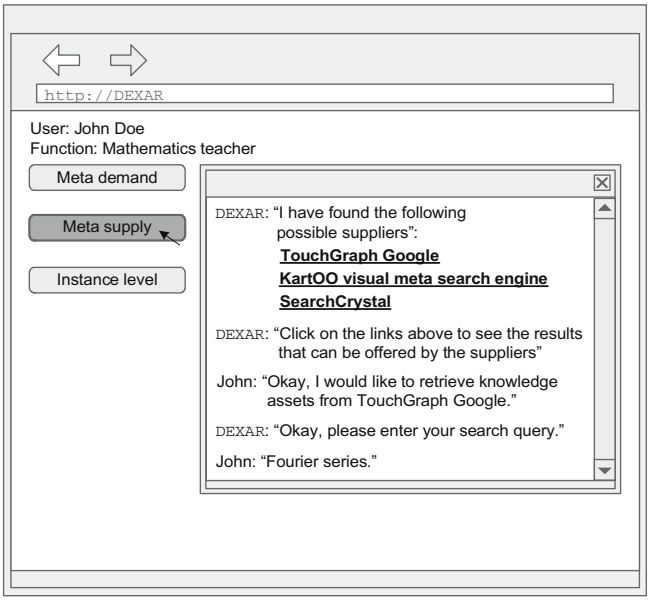

Fig. 12. Identification of knowledge suppliers.

The meta knowledge exchanged between the broker and the supplier so far is typically an initiation of Eq. (7) of Section 3.7. The question provided by the broker is concerned with which suppliers are able to offer the intended graph. The answer includes possible suppliers and possible assets they can supply. The teacher can find out which supplier is suitable to provide knowledge about the Fourier series by clicking on the links shown in Fig. 12. The suppliers are materialized by means of the Web applications *TouchGraph*, *KartOO* and *SearchCrystal*. After viewing the Web applications, the teacher decides that TouchGraph (URL: http://www.touchgraph.com) may be able to provide the most suitable results. This implies $y \rightsquigarrow$TouchGraph. The teacher also instructs Dexar that the query 'Fourier series' should be used to retrieve assets from TouchGraph. These steps force a call to Eq. (6) again, because exchange of meta knowledge is necessary between the broker and the utilizer. Broker Dexar transmits this query to

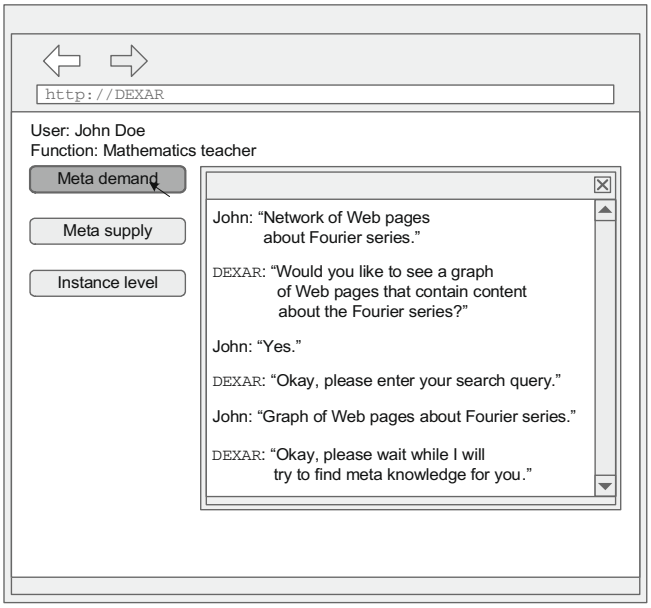

Fig. 11. Making an inventory of the knowledge need by DEXAR.

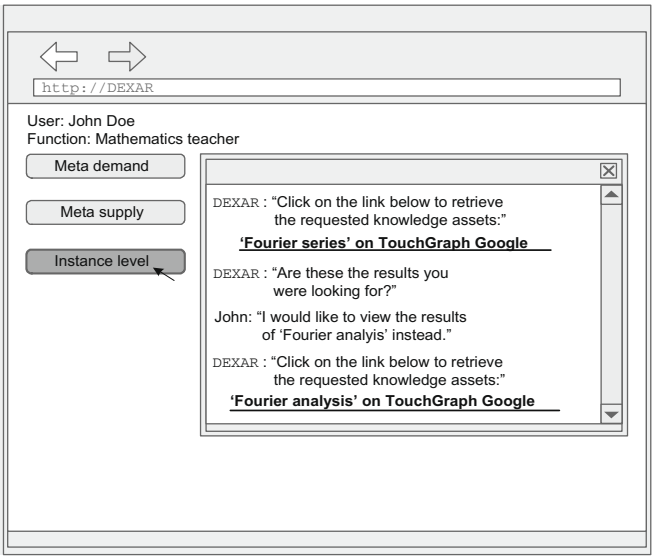

Fig. 13. Eventual supply of instance level knowledge assets.

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

TouchGraph, therefore initiating Eq. (7) again, and the teacher can view the results by clicking on the link shown in Fig. 13. The teacher can only view these results if Eq. (6) is called for the occasion.

At this point, Dexar enacts the transporter role to transmit the assets provided by TouchGraph to the teacher. The transporter also sends a remark to the teacher if these were the results that he was looking for. This involves the initiation of Eq. (8). After Eq. (9) has been called, the teacher can now *utilize* the provided assets and he can find out if he is pleased with the results. As can be seen in Fig. 13 he is obviously not quite pleased and wishes to retrieve knowledge about Fourier analysis from the supplier TouchGraph via the transporter Dexar. This involves Eqs. (8) and (9) again. The final results consist of a displayed network of Web pages about Fourier analysis provided by TouchGraph. The knowledge exchange procedure discussed so far materializes the knowledge exchange cycle of Section 3.8. A detailed description of this materialization can be given as follows:

(1) The math teacher would like to acquire an overview of the network of Web pages that contain knowledge about the Fourier series.
(2) The math teacher sends the query 'Network of Web pages about Fourier series' to broker Dexar.
(3) Dexar wonders if the teacher would like to see a graph of Web pages or perhaps something else.
(4) Dexar asks the teacher if he would like to see a graph of Web pages that contain content about the Fourier series.
(5) The teacher answers affirmatively.
(6) The query 'Graph of Web pages about Fourier series' is then submitted to Dexar.
(7) Dexar searches its knowledge resources to find suppliers that can visualize a graph of Web pages about the Fourier series.
(8) Dexar determines that TouchGraph Google, KartOO and SearchCrystal are Web applications that may provide a graph of Web pages that include content about the Fourier series.
(9) Dexar shows these suppliers to the teacher with hyperlinks to the Web applications.
(10) The teacher determines that he would like to retrieve knowledge assets from TouchGraph Google.
(11) The query 'Fourier series' is then submitted to Dexar.
(12) Dexar enters the query 'Fourier series' in TouchGraph Google.
(13) TouchGraph Google sends the graph of Web pages about the Fourier series to Dexar.
(14) The teacher can view these results by clicking on the hyperlink provided by Dexar. Dexar also provides a remark to find out if the teacher is pleased with the results.
(15) The teacher requests an alteration by providing the 'Fourier analysis' query. This implies that it is required to return to step 12 in the cycle until the teacher is pleased with the results.

So far, the framework and an application of the knowledge market paradigm have been discussed. What remains to be devised is the *knowledge workers market* paradigm.

5. Knowledge workers market paradigm

Fig. 1 of Section 1 showed the concepts involved when actors and execution tasks are matched. This situation in which supply and demand of cognitive characteristics are matched is regarded as a *knowledge workers market*.

5.1. Actors in the knowledge workers market

In the knowledge workers market, an actor may instantiate an *actor type* that is characterized by cognitive characteristics. Linguistic literature (Kako, 2006) and literature on the notion of knowledge workers (Davenport, 2005) form the basis for the five actor types. More examples of possible actor types may be introduced, but in this paper we choose to restrict ourselves to the types below because we do not strive for completeness. The five actor types as shown in Table 1 can be explained as follows: (1) The *experiencer* is only aware of knowledge requirements to fulfill some task. Consider for example the following sentence: *John thoroughly reads an article about data warehouses before joining a meeting about implementing a data warehouse*. This indicates that John probably understands that reading an article is enough to prepare himself for a meeting about that topic. (2) The *collaborator* has the ability to exert an influence on state changes of knowledge involved during fulfillment of an execution task. During fulfillment of an execution task a collaborator is also able to improve its own cognitive abilities. However, a collaborator does not have complete awareness of all required knowledge to fulfill an execution task and requires others to be able to complete it. Consider the following example: *John works at a hospital and requires knowledge about a patient's history. Therefore, he acquires the most recent patient log from a colleague*. This indicates that John understands that in order to acquire knowledge about a patient's history he must collaborate with another actor. (3) All characteristics are possessed by the *expert*. Suppose that John is an associate professor working at a university and he would like to solve a difficult mathematical problem when developing a theory. He then uses his own knowledge about mathematics to solve the problem. John is also able to combine and modify his own knowledge while solving the problem and he can also learn from that. (4) The *integrator* is able to fulfill an execution 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 requirements engineer that is testing a set of requirements for the final time is an example of an actor that can be characterized by the integrator type. (5) The *transactor* can fulfill an execution task without collaborating with others and is not required to cause modifications in the knowledge acquired and applied during task fulfillment. A customer support employee working at a software company is an example of a transactor.

When regarding cognitive literature (Cruse, 1973; Dowty, 1991), the following five characteristics can be distinguished that can be utilized to generate a framework for cognitive settings of possible different actor types: (1) The *volition* characteristic is concerned with an actor's willpower to fulfill some execution task. For instance, a skilled software developer may have more willpower to implement an intelligent search algorithm than implementing source code to access a database. (2) *Sentience* expresses that an actor has much awareness of required knowledge to fulfill some task. When a project manager creates a project plan he may have all the necessary knowledge to create such a plan. This may be due to

Table 1
Cognitive actor settings characterized.

$\mathcal{AT}$	$\mathcal{CC}$				
	Volition	Sentience	Causability	Improvability	Independency
Experiencer	–	×	–	–	–
Collaborator	×	–	×	×	–
Expert	×	×	×	×	×
Integrator	×	–	×	–	–
Transactor	×	×	–	–	×

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

earlier planning experiences of the project manager or by education. (3) The *causability* characteristic expresses that an actor has the ability to exert an influence on state changes of knowledge involved during fulfillment of a task. Suppose that a business consultant facilitates a brainstorm session in which he or she writes models on a whiteboard. In this case, the consultant causes knowledge that is implicitly present in his or her head to be made explicit on the whiteboard (Nonaka & Takeuchi, 1995). (4) During fulfillment of certain knowledge-intensive tasks an actor should be able to improve its own cognitive abilities. This is indicated by the *improvability* characteristic. For instance, a manager may have recently completed a course about cybernetics. During his work he or she successfully applies several principles that the manager has learned in the course. Participating in the course may thus have improved his or her abilities. (5) The *independency* characteristic is necessary to be able to determine if an actor is able to fulfill a task on his own. An example is a journalist who may successfully write a news article without having to collaborate with others. By determining possible characteristics an actor may have it is now appropriate to discern several actor types. The combination of an actor type with the characteristics belonging to a type is dubbed a *cognitive actor setting*. These settings are shown in Table 1.

5.2. Tasks in the knowledge workers market

Now that several actor types have been described together with the cognitive characteristics that can be supplied by actors that instantiate these types it is a logical step to focus on execution task types. In the knowledge workers market, a knowledge-intensive execution task is a task for which acquisition, application or testing of knowledge is necessary in order to successfully fulfill the task. As can be seen in Fig. 1, only actors that enact the utilizer role perform execution tasks. As is elaborated in earlier work, possible execution tasks that can be fulfilled can be abstracted to a pattern of three types (Overbeek et al., 2007): (1) Acquisition tasks, which are related with the *acquisition* of knowledge. This can be illustrated by a student reading a book in order to prepare himself for an exam. (2) Synthesis tasks, which are related with the actual utilization of the acquired knowledge. An example is a student who utilizes knowledge (acquired by reading a book) while performing an exam. (3) Testing tasks, which are related with the identification and application of knowledge in practice inducing an improvement of the specific knowledge applied e.g., a student who failed an exam studies a teacher's feedback on his exam. Then a re-examination attempt follows to improve his previously acquired and utilized knowledge.

The following six cognitive characteristics characterize these task types: (1) The *satisfaction* characteristic is related with a need for knowledge during a task's fulfillment and the eventual disappearance of that need. Suppose that a salesman requires insight in future developments of a certain market. Therefore, he asks a colleague to provide a forecast of these developments. After interpreting the forecast, the salesman's need for this knowledge may have decreased. (2) *Relevance* is concerned with whether or not knowledge acquired is deemed appropriate during task fulfillment. This is the case if e.g., the salesman is not able to acquire the necessary knowledge by interpreting the aforementioned forecast. (3) The *applicability* characteristic expresses to what extent knowledge is applicable in a task. For instance, a requirements engineer interviews a customer to acquire certain requirements for an information system to be build. After the interview has been conducted the engineer has acquired a lot of knowledge about the customer's organization but not about system requirements. In this example, the acquired knowledge is not very applicable for the task at hand. (4) When knowledge is applied it should meet its requirements. This is indicated by the *correctness* characteristic. For instance, when a software developer writes code it should meet the requirements to be able to compile the code and to achieve a system that is working correctly. (5) The *faultiness* characteristic is necessary to be able to determine whether or not applied knowledge contains flaws. For example, a software tester should be able to find bugs in software. (6) To correct already applied knowledge containing flaws, the *rectification* characteristic can be determined. This may be the case when a software developer fixes a bug found by a software tester. Table 2 shows how the task types are characterized. An acquisition task, for instance, demands the satisfaction and relevance characteristics. The formal definitions of these characteristics have already been discussed in Overbeek et al. (2007) and will therefore not be repeated here. An important remark to make here is that the possible task types as well as the possible cognitive characteristics are not limited to three task types and six cognitive characteristics. The three defined task types together with the characteristics serve as examples for the cognitive matchmaking framework.

5.3. Framework for cognitive matchmaking

The framework for cognitive matchmaking will be briefly discussed here because it is already elaborated in Overbeek et al. (2007). First, the framework is illustrated on a conceptual level in Fig. 14. The different concepts shown in Fig. 14 are functions that are necessary to calculate the eventual suitability match of an actor fulfilling a task. In this section only the main functions of the framework will be explained. Even though the formal signature of these functions is not exhaustively repeated here, we will show some examples for clarification. First, the *supply* function shows the level on which an actor type offers a cognitive characteristic during task execution. The levels on which an actor type supplies a 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 $\mathscr{CRN}$. 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 Fig. 14 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 Fig. 14 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

Table 2
Knowledge-intensive task types characterized.

$\mathscr{TA}$	$\mathscr{CC}$					
	Satisfaction	Relevance	Applicability	Correctness	Faultiness	Rectification
Acquisition	×	×	–	–	–	–
Synthesis	–	–	×	×	–	–
Testing	×	–	×	–	×	×

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

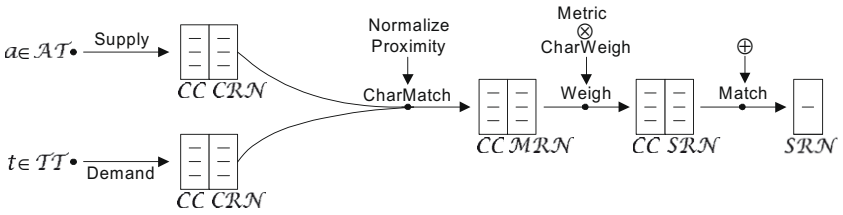

Fig. 14. Framework of the cognitive matchmaker system.

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 an example of how we have formalized the functions of the framework, the formal signature of e.g., the match function is modeled as follows (Overbeek et al., 2007): $\mathsf{Match} : \mathcal{AT} \times \mathcal{TT} \to \mathcal{SRN}$. 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:

$$\mathsf{Match}(\mathtt{transactor}, \mathtt{synthesis}) \triangleq \bigoplus_{c \in \mathscr{C}} \mathsf{Weigh}(c, \mathsf{CharMatch}(\mathtt{transactor}, \mathtt{synthesis}))$$

For this example the suitability match of the *transactor* and the *synthesis* task has been calculated. 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. The match function can be expressed as follows: $\mathsf{Match}(\mathtt{transactor}, \mathtt{synthesis}) = 4.65$, which shows that the suitability match of the transactor fulfilling the synthesis task is 4.65. This is a fairly good result, knowing that 5 is the best suitability match that can be achieved.

Finally, a certainty function has been introduced to make sure how certain it is that an actor is suitable to fulfill a task (Overbeek et al., 2007): $\mu : \mathbb{R} \to [0, 1]$. A linear certainty function can be defined as follows:

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

In the implementation of the prototype the minimum and maximum values of a suitability match are equated to 0 respectively 10. Thus, $\mathtt{min} = 0$ and $\mathtt{max} = 10$. The certainty that the transactor is suitable to fulfill the synthesis task is: $\mu(4.65) = \frac{2}{0+10} \cdot 4.65 = 0.93$. This can be interpreted as being 93% sure that the transactor 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.

5.4. 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 Fig. 15. 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 Section 5.3 and Overbeek et al. (2007). 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 Fig. 14. The business layer also includes an implementation of the possible ranking domains that can include

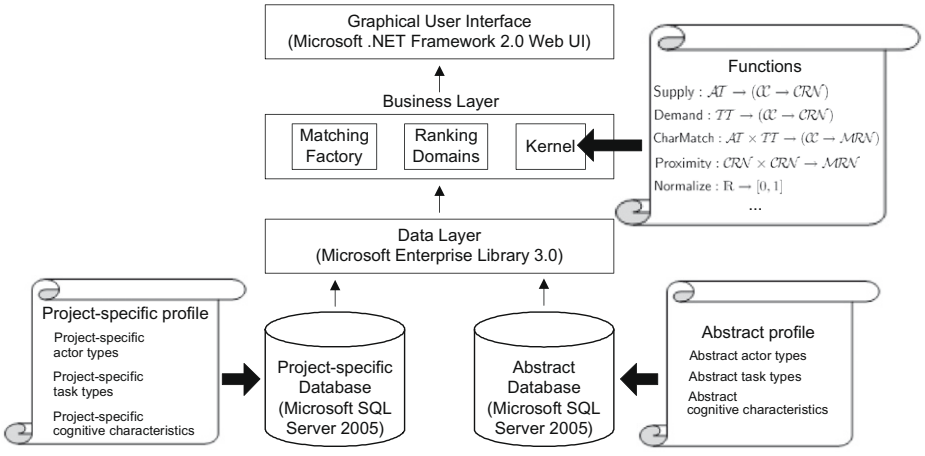

Fig. 15. Cognitive matchmaker system architecture.

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

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 Sections 5.1 and 5.2. 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 software project. Section 6.2 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. This categorization is shown in Section 6.3. 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 let the system 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 *transactor* 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 transactor actor type and the synthesis task type. In the following step, the application displays on which level the transactor supplies the involved characteristics and on which level the synthesis task demands the characteristics for successful fulfillment of the task. The next part shows the characteristic match results for all cognitive characteristics. Subsequently, the user can provide the weigh values for the cognitive characteristics by entering them for each characteristic involved. Fig. 16 shows the eventual suitability match result with the corresponding graph. The resulting graph shows that the suitability match of the transactor fulfilling the synthesis task is 4.65. The certainty that the transactor is able to fulfill the synthesis task is 93%. The implementation of the prototype is based on the cognitive matchmaking framework. For example, the code implementation of the suitability match function depicted in Section 5.3 is shown in Fig. 17. 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, for each cognitive characteristic involved in the process of computing the suitability match the results of the weighed characteristic match function are summated. This also corresponds with the definition of the suitability match function. Now that the framework for cognitive matchmaking and the cognitive matchmaker system have been elaborated, they can be evaluated by means of a case study.

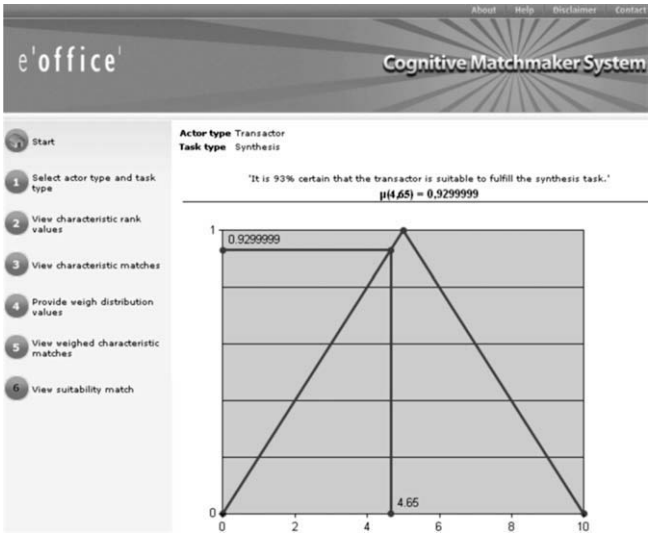

Fig. 16. Suitability match screen.

```
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. 17. Source code of the suitability match function.

6. Application of the knowledge workers market

An earlier reported case in which we have materialized the theory about the knowledge workers market can be found in Overbeek, van Bommel, and Proper (2008). This earlier reported case involved a recently completed information systems engineering (ISE) project at 'e-office'. e-office is a company specialized in providing computer-aided support for human actors to support them in their office work. ISE is related with the conceptualization, design, development and implementation of information systems to support business functions (Joshi, Sarker, & Sarker, 2007). Cognitive matchmaking can be utilized to support in allocating tasks to actors that are involved in every ISE phase. The studied ISE 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 Web application that can generate risk reports for the user.

6.1. Case study design

The 'e-office' case is part of an overall case study in which we try to evaluate the framework of cognitive matchmaking and the prototype of the cognitive matchmaker system in the context of ISE. The case study design is based on the inductive-hypothetical research strategy (see e.g., Sol, 1982; Wang, van de Kar, & Meijer, 2005). This strategy 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 case study design consists of the following phases after applying the inductive-hypothetical research strategy: (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 general 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 cognitive matchmaker system had been applied in the studied ISE project. (5) Evaluation by comparing phase 1 with phase 4.

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

A second case as part of the case study is reported throughout this section. This case is related with an ISE project at 'Everest'. This is a company specialized in the modeling and automation of business processes. The customers of Everest can be found for a large part in the mortgage and insurance domain. The studied ISE project has been concerned with the renewal of an 'international Automated Mortgage System' (iAMS for short) for an international provider of mortgage services. The upgrade of iAMS should enable a fully electronic workflow process that services the monetary transactions and the required securitization when enacting mortgage business processes. The following procedure has been applied to collect data for the Everest case: Perform explorative interviews, analyze documentation and let key informants review the draft case study report. The interviews were conducted with the project members of the iAMS project. Each employee answered the following project-related questions:

- Can you explain the meaning of your role in the project?
- Which ISE method have you been using?
- Which project phases have been distinguished?
- Which project-specific actor types and task types have been defined?
  - Have cognitive profiles been used to categorize actors in project-specific actor types?
  - If no, based on what are roles assigned to project members?
- Which project-specific task types are related to which project-specific actor type?

Next, each employee also answered the following employee-related questions:

- How have the tasks been divided among the project members?
- Did you think the tasks that were assigned to you matched the role you have enacted in the project?
- Have you been able to fulfill the tasks assigned to you in the project?

Complementary documentation was provided by access to the Internet and via e-mail. The results were also send to the informants for review.

6.2. Initiation

The iAMS project is not based on a single information systems engineering method. Instead, the development team has made use of a variety of *agile* information systems engineering methods, such as eXtreme Programming (XP), Microsoft Solutions Framework (MSF) for Agile Software Development and Scrum. By 'tailoring' existing agile methods, the development team strived to exhibit flexibility to accommodate project changes rapidly. The resulting system is a Web application based on several technologies, such as: Microsoft BizTalk Server 2006, SQL Server 2005 and Windows Server 2003.

The following project phases are determined as part of the iAMS project: The analysis phase, the design phase, the development phase and the acceptance phase. During the *analysis* phase requirements are engineered by eliciting knowledge from future users of the tool. The definitions of the requirements are tested by means of meetings with the future users. After the requirements engineering process, the mortgage provider is provided with possible solution approaches leading to fulfillment of the requirements. The knowledge gathered in the analysis phase about the system to be build is used as input to determine the project approach. The *design* phase includes the global and detailed system functionalities that are designed proceeding from the requirements. These functional models are implemented in a Web application called *Everest Studio*. The goal of using Everest Studio in an ISE project is to diminish time that is spend on code development. Various designed models can be added to Everest Studio, such as: Process models, product models and domain models of the system under development. These models can be compiled by Everest Studio to provide a prototype version of iAMS. These functional designs should be compliant to the functional architecture, which is also designed during the design phase. Next, the functional specifications and architecture are used as input to conceive the technical architecture. The technical system design should be compliant to this technical architecture. Finally, system tests are created during the design phase. The *development* phase involves the implementation of the system. The functional specifications modeled in Everest Studio are used as input for this phase, as well as the prototype application. Additional coding is needed to let the mortgage system communicate with external information systems. Project-specific exceptional system behavior which is not desirable to integrate in Everest Studio is also coded in the development phase. Finally, system tests are conducted during the development phase. The application is deployed in the *acceptance* phase. This involves acceptance tests by future users of iAMS and the implementation of the final version of iAMS at the mortgage service provider. Note that the mortgage system is developed by approaching the project phases in an iterative way e.g., it may be possible to perform a task in the design phase after performing a task in the acceptance phase.

The actors participating in the project have been categorized into several project-specific actor types based on the aforementioned agile methods. Despite the fact that many more project-specific actor types can be incorporated when tailoring agile methods, the following types were identified in the iAMS project. First, a *business engineer* can be identified. At Everest, the business engineer is tightly involved in every phase of an ISE project. The business engineer is responsible for requirements engineering and is involved in designing system functionalities and implementing them in Everest Studio. Thus, the main functionality that does not require additional coding is realized by the business engineer. Designing and fulfilling system tests are also part of the business engineer's job. In case of the iAMS project, the business engineer additionally coordinates the project members and assigns tasks to them i.e., the business engineer takes care of the *internal* project coordination. The *functional architect* conceives the functional architecture and coordinates the functional designs of the mortgage system. The functional architect also verifies if the designed functional architecture is correctly fulfilled. The *technical architect* conceives the technical architecture and coordinates the technical realization of the mortgage system. This also involves the selection of development tools and technical standards. Both the functional architect and the technical architect are involved in the requirements engineering process to assist in collecting, designing and testing requirements. The *technical engineer* realizes the communication between iAMS and external systems with which iAMS needs to interact. Additional coding of project-specific exceptional system behavior which is not desirable to integrate in Everest Studio is also realized by the technical engineer. The technical engineer also conducts system tests and the system's deployment. Next, the *project manager* is responsible to maintain a healthy balance between serving the interests of Everest and the interests of the mortgage service provider during the iAMS project. The project manager primarily coordinates the interrelationships between external project teams that are in some way involved in the iAMS project. Internal project coordination is part of the business engineer's job. Finally, the project manager writes project plans and advises about the project approach.

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

Table 3
Project-specific actor types and task types.

Task instance	Project phase	Project-specific task type	Project-specific actor type
Attend project meeting	All phases	Meeting task	All actor types
Attend steering committee meeting	All phases	Meeting task	Project manager
Coordinate project internally	All phases	Coordination task	Business engineer
Coordinate project externally	All phases	Coordination task	Project manager
Write issue report	All phases	Documentation task	Project manager
Write financial status document	All phases	Documentation task	Project manager
Write project progress document	All phases	Documentation task	Project manager
Write project initiation document	Analysis	Documentation task	Project manager
Write project plan	Analysis	Documentation task	Project manager
Write risk report	Analysis	Documentation task	Project manager
Collect requirements	Analysis	Elicitation task	Business engineer Functional architect Technical architect
Design requirements	Analysis	Design task	Business engineer Functional architect Technical architect
Test requirements	Analysis	Requirements test task	Business engineer Functional architect Technical architect
Recommend possible solutions	Analysis Design	Consultancy task	Business engineer
Recommend project approach	Analysis	Consultancy task	Project manager
Design functional architecture	Design	Design task	Functional architect
Recommend functional specifications	Design	Consultancy task	Functional architect
Determine functional standards	Design	Elicitation task	Functional architect
Verify fulfillment of functional architecture	Design Development	Elicitation task	Functional architect
Create global functional system design	Design	Design task	Business engineer
Create detailed functional system design	Design	Design task	Business engineer
Implement system design in Everest Studio	Design	Design task	Business engineer
Design test specifications	Design	Design task	Business engineer
Design technical architecture	Design	Design task	Technical architect
Recommend technical specifications	Design	Consultancy task	Technical architect
Determine technical standards	Design	Elicitation task	Technical architect
Select development tools	Design	Elicitation task	Technical architect
Verify fulfillment of technical architecture	Design Development	Elicitation task	Technical architect
Create technical system design	Design	Design task	Technical engineer
Write technical system description	Development	Documentation task	Technical engineer
Recommend technical solutions	Development	Consultancy task	Technical engineer
Create connections with external information systems	Development	Code development task	Technical engineer
Create specific iAMS functionality	Development	Code development task	Technical engineer
Commit partial system test	Development Acceptance	System test task	Business engineer Technical engineer Business engineer Technical engineer
Commit integral system test	Development Acceptance	System test task	Business engineer Technical engineer Business engineer Technical engineer
Deploy completed system	Acceptance	Deployment task	Technical engineer

The project manager of the iAMS project has created plans for every phase that include breakdowns of the tasks to be fulfilled in every phase. 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 is responsible to fulfill a project-specific task. The results of this analysis are shown in Table 3.

6.3. Abstraction

When performing the second phase of the case study strategy, it is possible to abstract from 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 Section 5.2. Second, the project-specific actor types are abstracted to the actor types mentioned in Section 5.1.

Recall that the distinguished abstract task types are the acquisition task type, synthesis task type and the testing task type. The project-specific task types depicted in Table 3 can be abstracted to these task types as follows. The mentioned 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. Coordination tasks can be abstracted to *synthesis* tasks. Knowledge about project management is applied during internal and external project coordination. Documentation tasks can also be classified as synthesis tasks. The documentation tasks mentioned in Table 3 are related with the application of knowledge when writing an issue report, a financial status document, a project progress document, a project initiation document, a project plan and a risk report. Next, elicitation tasks are typical knowledge acquisition tasks. The actors performing an elicitation task acquire and memorize knowledge by means of knowledge elicitation techniques, such as interviews and workshops. Design tasks can be viewed as synthesis tasks. In a design task, an actor applies already acquired knowledge when designing requirements, the functional and technical architecture, test specifications and technical system descriptions. The creation of functional system designs and the consecutive implementation

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

Table 4
Actor type and task type abstraction.

Project-specific actor type	Abstract task type	Abstract actor type
Business engineer	Acquisition	Collaborator or Integrator
	Synthesis	Collaborator or Expert
	Testing	Collaborator
Functional architect	Acquisition	Collaborator or Experiencer or Expert
	Synthesis	Collaborator or Expert
	Testing	Collaborator
Project manager	Acquisition	Collaborator
	Synthesis	Expert or Transactor
Technical architect	Acquisition	Collaborator or Experiencer or Expert
	Synthesis	Collaborator or Expert
	Testing	Collaborator
Technical engineer	Acquisition	Experiencer
	Synthesis	Expert or Integrator
	Testing	Collaborator

of these models in Everest Studio can also be considered as design tasks. Code development tasks can be viewed as synthesis tasks. These tasks are necessary to realize the mortgage system itself. Consultancy tasks can be categorized as synthesis tasks. These tasks are performed by actors providing advice about possible solution approaches to build iAMS, project approaches, and functional and technical specifications. *Testing* tasks are related with the project-specific requirements and system test tasks. In a testing task, earlier applied knowledge is thoroughly examined inducing an improvement of the specific knowledge applied. Finally, the deployment task can be abstracted to a synthesis task. Here, all knowledge that is applied is related with a successful deployment of the system at the customer's location.

Recall that the distinguished abstract actor types are the collaborator, experiencer, expert, integrator and the transactor. Table 4 shows the project-specific actor types, the abstract task types that a project-specific actor type can fulfill and the abstract actor types. 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 an expert or a transactor dependent of the project-specific task that is performed. Note that some project-specific actors are not related with every abstract task type, e.g., a project manager does not fulfill a testing task.

6.4. Theory formulation

The results of applying the third phase of the case study strategy are discussed throughout this section. The cognitive matchmaker system can be utilized in all four phases of the iAMS project. However, only the analysis 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 the results of Section 6.2 and 6.3 it is now possible to calculate the certainty that actors involved in the analysis phase of the iAMS project can successfully fulfill the tasks allocated to them. The results after calculating the cognitive matches are depicted in Table 5. Table 3 provides the project-specific task types and the project-specific actor types that are involved in the analysis phase. The cognitive matchmaker system can be utilized to calculate the matches between the actor types and task types involved in the analysis phase. For this purpose the abstraction of the project-specific task types and actor types in Table 4 must be used. The results of Table 5 can be explained as follows. The project manager, for instance, acts as a collaborator when working on an acquisition task. The project manager may act as an expert or a transactor when working on a synthesis task. The project manager attends project meetings and steering committee meetings in the analysis phase. 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 when the project manager needs to fulfill an acquisition task in the analysis phase. Because we were the users of the cognitive matchmaker system, we have provided the following weigh values for the volition, causability, improvability, satisfaction and relevance characteristics: 1.5, 3, 1, 2 respectively 2.5. 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 has been applied to the causability characteristic. That the project manager should supply the causability characteristic is obviously very important when fulfilling an acquisition task. This is to make sure that the project manager has a great ability to exert an influence on state changes of knowledge involved during task performance. During the meetings, for instance, the project manager must be able to transform his implicit knowledge to explicit knowledge that is exchangeable with the participants of the meetings. Next, the system sums up the weighed characteristic matches resulting in a suitability match of 4.375. The certainty that the project manager acting as a collaborator can successfully fulfill an acquisition task is: $\mu(4.375) = \frac{2}{0+10} \cdot 4.375 = 0.875$ or $0.875 \cdot 100\% = 87.5\%$. The project manager acts as an expert or a transactor when working on a synthesis task during the analysis phase. These synthesis tasks are related with the consultancy, coordination and documentation tasks. When the project manager performs consultancy and coordination tasks, he acts as an expert.

Table 5
Cognitive matchmaking in the analysis phase.

Project-specific actor type	Task type	Actor type	Weigh values	Suitability match	Certainty (%)
Business engineer	Acquisition	Collaborator	1, 2, 2, 2, 3	4.55	91
		Integrator	2, 3, 2, 3	5.55	89
	Synthesis	Collaborator	1.5, 1.5, 1.5, 2.5, 3	4.45	89
	Testing	Collaborator	1, 1, 1, 1, 1.5, 2, 2.5	4.225	84.5
Functional architect	Acquisition	Collaborator	1.5, 2.5, 1, 2, 3	4.4	88
	Synthesis	Collaborator	1, 1.5, 1.5, 3, 3	4.4	88
	Testing	Collaborator	1, 1, 1, 1, 1.5, 2, 2.5	4.225	84.5
Project manager	Acquisition	Collaborator	1.5, 3, 1, 2, 2.5	4.375	87.5
	Synthesis	Expert	1.5, 2, 0.5, 0.5, 1, 2, 2.5	4.725	94.5
		Transactor	1, 2.5, 2.5, 2, 2	5.2	96
Technical architect	Acquisition	Collaborator	1.5, 2.5, 1, 2, 3	4.4	88
	Synthesis	Collaborator	1, 1.5, 1.5, 3, 3	4.4	88
	Testing	Collaborator	1, 1, 1, 1, 1.5, 2, 2.5	4.225	84.5
Technical engineer	Acquisition	Experiencer	2, 4, 4	3	60

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

He uses his own knowledge about project management to determine the project approach and to coordinate the iAMS project. He is also able to combine and modify his own knowledge during these tasks and he can learn from that experience. The expert actor type matches very well with the synthesis task in this case, because the result of the suitability match calculation is 4.725 and the result of the certainty function is 94.5%. When the project manager performs documentation tasks, he acts as a transactor. He should be able to work independently for a large part during the documentation task and is not required to cause a lot of modifications in the knowledge applied during task fulfillment. The transactor also matches very well with the synthesis task, because the result of the suitability match calculation is 5.2 and the result of the certainty function is 96%.

6.5. 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 analysis 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 actor and task profile as a starting point or the abstract profile including the abstract actor types and task types from our framework. The latter has been done in the case study as is elaborated in Section 6.3. When using a project-specific profile as input for the cognitive matchmaker system, a project-specific profile of actors and tasks should be generated. This has also been done in Section 6.2. If not already entered in the project-specific database as is shown in Fig. 15, the actor and task data should be provided as a next step. The person that 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. 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. The cognitive matchmaker system can also be utilized to evaluate task allocations after every project phase. The suitability matches may be compared with the actual fulfillment of the tasks in a phase. An in-depth analysis may be necessary if there are striking differences between the suitability match and the results of task fulfillment by an actor. (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 then be recorded for future projects. This may help to better decide which actor types are suitable to work with which types of tasks.

6.6. Case evaluation

In this section, the results of the initiation phase are compared with the results of the implementation phase. The validity of the Everest case is also judged by applying case study tactics (Yin, 2003). The evaluation of the initiation phase is related to the three viewpoints of the implementation phase. (1) At *design time*, the choices leading to the assignment of the tasks to the actors participating in the project as shown in Table 3 have not been underpinned by Everest. Constructing relations between the tasks and the actors beforehand could have increased the suitability matches of the actors fulfilling the tasks assigned to them. Second, the project-specific actor types mentioned in Table 3 have not been inferred from the agile ISE methods used. The Microsoft Solutions Framework for Agile Software Development, for instance, distinguishes eight different project-specific actor types. The tester, the release manager and the database-related actor types are additional to the five types introduced by Everest. 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 3 are performed by the business engineer and the technical engineer. However, the MSF method also distinguishes the tester actor type. Including this actor type in the iAMS 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 cognitive characteristics to the project-specific database or by reusing characteristics. (2) 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 5 deserve special attention to discover the reasons of the lowest match results. For instance, the technical engineer acting as an experiencer has a certainty of 60% to successfully fulfill an acquisition task. When viewing Table 3 it can be interpreted that the acquisition task performed by the technical engineer in the analysis phase is related with the attendance of project meetings. This may be caused in case a meeting is not very relevant for a technical engineer. For instance, when a large part of a certain meeting is about issues related to the collection of requirements a technical engineer may not have a satisfied feeling after the meeting. Letting the technical engineer attend the most relevant meetings may increase the suitability matches for these acquisition tasks. (3) The testing tasks shown in Table 3 deserve attention when comparing the actual project results with the suitability matches from a *post-mortem* viewpoint. According to Table 3 the requirements tests are conducted by the business engineer and the architect types. The system tests are conducted by the business engineer and the technical engineer. This is because testing tasks are divided across the current actor types of the iAMS project. Table 5shows that the certainty is 84.5% that the business engineer, the functional architect and the technical architect can successfully fulfill these testing tasks. The agile 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. Once a bug is found, it is also the tester's job to accurately communicate its impact and describe any solutions that could lessen its impact. Probably, more bugs could have been found and resolved after testing each iteration and the overall mortgage system by the tester actor type. In the current project situation, the business engineer has the responsibility for requirements testing and system testing as well. The business engineer is not only involved in these testing tasks, but is also responsible for internal project coordination, performs consultancy tasks and also design tasks. This occupation implies that the business engineer is tightly involved throughout the whole project and in every phase. The risk exists that the business engineer tends to become a Jack-of-all-trades. To prevent this from happening, several recommendations can be verbalized. First, design tasks, such as the creation of functional

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

system models, can be assigned to the functional architect. The functional architect is responsible for the creation of the functional architecture and, therefore, possesses a vast amount of knowledge related to all the functionalities of the system. Second, testing tasks can be performed by a separate tester. An advantage of a designated tester actor type is that the tester may be able to fully concentrate on (and may have more time for) e.g., determining test methods and techniques for the test tasks at hand. Internal project coordination and consultancy tasks can also be partly done by an actor enacting the project manager type. The aforementioned suggestions may decrease the dependencies of the business engineer and may increase the suitability matches.

To make sure that the gathered data for the Everest case is valid, the case study tactics defined by Yin (2003) have been applied. Four research design tests have been conducted to judge the quality of the research design: Construct validity, internal validity, external validity and reliability. The case study can be classified as a multiple-case study design (Yin, 2003), because two cases can be distinguished that have been based on the same design, namely the e-office case that has been reported in earlier work (Overbeek et al., 2008) and the Everest case. In addition, the case study design is a multiple-case embedded design as multiple units of analysis, in this case individual employees and ISE projects, are included. Data was gathered from documentation, Web pages, interviews and e-mail correspondence. This implies that multiple sources of evidence were used and a chain of evidence was established during data collection. The character of this case study is exploratory, because we have tried to evaluate the framework of cognitive matchmaking and the prototype of the cognitive matchmaker system in the context of real ISE projects. Therefore, the internal validity of the design is irrelevant (Yin, 2003). The validity of the construction of the data has been increased because key informants have reviewed the draft case study reports. Next, external validity can be tested by including replication logic in case of a multiple-case design. External validity is concerned with whether or not the study's findings are generalizable beyond the immediate case study. By following the inductive-hypothetical strategy mentioned in Section 6.1 we presume the findings of the case study are generalizable for other ISE projects. Other cases can be performed identically when utilizing the approach we have used. Finally, the reliability of the case study is obtained by using a formal case study protocol and developing a case study database.

7. Discussion

Literature shows that several other theories exist that reason about knowledge exchange viewed from a market perspective. Desouza and Awazu (2003) define an *internal knowledge market* as a collection of buyers and sellers who interact to determine the price of a product. There are no components in their market paradigm that can be compared to the transporter and the broker. Analysis of the interaction between buyers and sellers is necessary to be able to focus on the exchange of knowledge within a knowledge market. An advantage of our model is that those interactions are made clearer by the introduction of additional roles and the two knowledge levels. The rules mentioned in Desouza and Awazu (2003), however, do intend to cover exchange mechanisms within a knowledge market. These exchange mechanisms in a market should address what assets will be bought and sold and how they will be paid for, even though money is rarely the form of payment. The way how knowledge markets function is related with the way how traditional economic markets function (Desouza & Awazu, 2003). This economically-oriented focus results in the elaboration of pricing and quantization of knowledge. The characterization of supply and demand of knowledge is not discussed in-depth. In our view, however, this is an important issue when studying knowledge markets, especially when regarding the knowledge exchange cycle of Section 3.8. A knowledge market can also be described as an open and commercial marketplace where knowledge assets may be traded in a manner analogous to business-to-business marketplaces of goods and services (Mentzas, Kafentzis, & Georgolios, 2007). An elaborated architecture is provided for trading knowledge services which is a computer-based materialization of the knowledge market theory discussed in Mentzas et al. (2007). How this architecture can be exploited in practice is not made very clear. Such a gap can be bridged by the knowledge exchange cycle that is provided in our study which illustrates how the concepts as part of a knowledge market may support knowledge exchange in practice. Recall from Fig. 1 that we intended to complement our knowledge market theory with cognitive matchmaking. Complementing knowledge market theory with cognitive matchmaking entails an enriched overall framework compared to other theories that purely concentrate on knowledge markets. In the case of this paper, knowledge exchange can be used as a mechanism for a utilizer to gain or improve competencies followed by the application of cognitive matchmaking to match tasks and actors that enact the utilizer role in the market. Dignum (2006) has introduced a knowledge market paradigm that consists of two layers and incorporates eight roles. Coordination of the activities in the knowledge market takes place in the facilitation layer. Typical roles as part of the facilitation layer are the matchmaker, the gatekeeper, the notary and the monitoring role. The matchmaker role resembles the broker role as introduced in our model. The gatekeeper is responsible for accepting and introducing new actors to the market. The notary registers and keeps track of collaboration contracts between actors. Finally, the actor enacting the monitoring role keeps track of the execution of contracts between actors in the market. The seeker (which can be equated to the utilizer) and owner (which can be equated to the supplier) are part of the operational layer. The model in Dignum (2006) contains additional interesting roles compared to our model. However, these roles are not clearly specified i.e., it may be interesting to understand certain properties of these roles or the requirements of actors that should be able to enact those roles. Interesting evaluative results would have been discovered if a prototype or a case study would have been provided in Dignum (2006).

Apart from knowledge market theories, the concept of cognitive matchmaking can also be found in computer science literature. One of these initiatives is the cognitive matchmaking of students with e-learning system functionality (Ruiz, Díaz, Soler, & Pérez, 2008). A way of working is presented to design e-learning systems that better adapt to the cognitive characteristics of students. Besides the way of working, a cognitive method or a system to match students and e-learning systems is not proposed. The concept of reflection mentioned in Ruiz et al. (2008) 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., URL: 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 cognitive matchmaker system may take situational elements into account when determining a match. For example, the actual availability of actors during the studied iAMS project may be included when allocating tasks to actors. Jaspers, Steen, van den Bos, and Geenen (2004) use cognitive matchmaking to match information system requirements with the user's task behavior. The think-aloud method has been applied to understand task behavior. This method requires subjects to talk aloud while performing a task. This stimulates understanding of the supplied cognitive characteristics when

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

performing a task. Unfortunately, the study lacks a more abstract framework that can be reused to better match tasks and users. The think-aloud method, however, may be very valuable to improve the way we have characterized the abstract actor types and task types based on cognitive characteristics. The ideas presented in this paper somehow relate with the notion of *cognitive fit* (Vessey, 1991). The basic model of cognitive fit views task fulfillment as the outcome of the relationship between the actor's perception of how to fulfill the task and the nature of the task, which are both characterized by the type of knowledge they emphasize. When the types of knowledge emphasized in the actor and task elements match, the actor can employ processes (and formulate a mental representation) that also emphasize the same type of knowledge. Cognitive fit exists when the cognitive processes used to complete the task match. Elaborating an actor's perception related to the fulfillment of every task may be a time consuming process in practice and therefore an advantage of our approach may be that a match value can easily be determined once actors and tasks are classified by their types (by following the system's flow as presented in Fig. 14). However, an actor's cognitive capabilities may change (they may improve or deteriorate) and that may cause an actor to be classified as a different type in our model at different points in time.

8. Conclusions and future work

This paper describes a dichotomy of markets: The knowledge market and the knowledge workers market. The knowledge market paradigm is based on how traditional economic markets function and shows how knowledge assets can be exchanged. This exchange of knowledge is necessary for potential knowledge utilisers that perform qualifying tasks to gain or improve their competencies. Competencies and utilisers are necessary to execute knowledge work. First, the basic components of the knowledge market are conceived, which includes knowledge assets, roles and qualifying task types. Next, a framework for knowledge exchange in the knowledge market is elaborated. This framework includes the functions to determine how knowledge can be exchanged in a knowledge market. Proceeding from the framework an application of the knowledge market paradigm is demonstrated eventually. It can be concluded that the inclusion of the broker and the transporter roles in the knowledge market has introduced several advantages for knowledge exchange events. The broker adds the capability of characterizing supply and demand of knowledge in the market. This is important to increase the chance that there will be an actor enacting the supplier role that is able to supply knowledge assets that may fit the need of the utilizer. The transporter adds the capability to not only physically transport knowledge assets from the supplier to the utilizer, but also to paraphrase knowledge. This paraphrasing is necessary to understand if the utilizer's questions for knowledge have been answered accordingly. Exploiting the knowledge market in practice by instantiating the knowledge exchange cycle may fuel the diminishment of knowledge needs.

The knowledge workers market shows how actors that enact the utilizer role and execution tasks can be matched based on cognitive characteristics. First, several actor types and task types are categorized based on cognitive characteristics that can be supplied respectively demanded. Next, a framework for cognitive matchmaking is briefly discussed. This framework includes the functions to calculate the suitability of an actor to fulfill a task. Proceeding from the framework the prototype implementation of a cognitive matchmaker system is demonstrated. Two information systems engineering projects provided the breeding ground for the multiple-case study in which possible benefits of the cognitive matchmaker system have been evaluated. The second and final ISE project as part of the multiple-case study has been reported in this paper. This project has been concerned with the renewal of an 'international Automated Mortgage System (iAMS for short) for an international provider of mortgage services. The suitability matches of the tasks allocated to the actors in the analysis phase have been evaluated using the cognitive matchmaker system. By utilizing the actor type and task type categorizations it has been possible to calculate matches with the functions of the framework. The implementation of the match functions in a prototype shows that it is possible to build a functioning cognitive matchmaker system. When reviewing the case study, it can be concluded that the system can provide support for task allocation in an ISE project in at least three different ways, namely: Design time, runtime and post-mortem.

Future work with respect to the knowledge market part of this study can be aimed at improving the foundations as well as the Dexar prototype and further evaluation in a separate multiple-case study. Recall from Section 7 that several additional knowledge market roles have been discussed in Dignum (2006). Such roles may further enrich our knowledge market paradigm e.g., collaboration contracts for actors participating in the market can then be determined and monitored. Properties that may further characterize the roles and tasks as part of the knowledge market can also be made more explicit. The Dexar prototype may also be implemented as a functioning Web application in the future. Future work with respect to the knowledge workers market is concentrated on improving the framework as well as the prototype and further evaluation in a multiple-case study. 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. If this is the case, it might be interesting to determine a match based on the total amount of actors and the total amount of tasks the actors are fulfilling as a group. Besides this addition, the future matchmaking framework and prototype may consider situational elements. This may include personal preferences of actors, goals and actor availability. Suppose that an actor has a high match value when fulfilling a certain task but does not like to fulfill that task at all, then this may negatively influence the actor's task performance. The next version of the cognitive matchmaking framework and prototype should also take the concept of *actor contention* into account. This can be explained as follows. Assume that two actors receive the same best suitability for a task. Let one of these actors be mediocre at all required characteristics, while the other actor is really good at some and really bad at others. Somehow, the system should choose an actor to assign the task to.

References

Cruse, D. (1973). Some thoughts on agentivity. *Journal of Linguistics, 9*, 11–23.

Davenport, T. (2005). *Thinking for a living – How to get better performances and results from knowledge workers*. Boston, MA, USA: Harvard Business School Press.

Desouza, K., & Awazu, Y. (2003). Constructing internal knowledge markets: Considerations from mini cases. *International Journal of Information Management, 23*(4), 345–353.

Dignum, V. (2006). An overview of agents in knowledge management. In M. Umeda, A. Wolf, O. Bartenstein, D. Seipel, & O. Takata (Eds.), *Declarative programming for knowledge management. Lecture notes in computer science* (Vol. 4369). Springer: Berlin, Germany, EU.

Dowty, D. (1991). Thematic proto-roles and argument selection. *Language, 67*(3), 547–619.

Graham, I., Logan, J., Harrison, M., Straus, S., Tetroe, J., Caswell, W., et al. (2006). Lost in knowledge translation: Time for a map? *Journal of Continuing Education in the Health Professions, 26*(1), 13–24.

Halpin, T. (2001). *Information modeling and relational databases from conceptual analysis to logical design*. San Mateo, CA, USA: Morgan Kaufman.

Hoppenbrouwers, S., & Proper, H. (1999). Knowledge discovery: De zoektocht naar verhulde en onthulde kennis. *DB/Magazine, 10*(7), 21–25 (in Dutch).

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071

Jaspers, M., Steen, T., van den Bos, C., & Geenen, M. (2004). The think aloud method: A guide to user interface design. *International Journal of Medical Informatics, 73*(11–12), 781–795.

Joshi, K., Sarker, S., & Sarker, S. (2007). Knowledge transfer within information systems development teams: Examining the role of knowledge source attributes. *Decision Support Systems, 43*(2), 322–335.

Kako, E. (2006). Thematic role properties of subjects and objects. *Cognition, 101*(1), 1–42.

Liang, T. (1994). The basic entity model: A fundamental theoretical model of information and information processing. *Information Processing and Management, 30*(5), 647–661.

Mentzas, G., Kafentzis, K., & Georgolios, P. (2007). Knowledge services on the semantic web. *Communications of the ACM, 10*(50), 53–58.

Nonaka, I., & Takeuchi, H. (1995). *The Knowledge Creating Company*. New York, NY, USA: Oxford University Press.

Overbeek, S., van Bommel, P., Proper, H., & Rijsenbrij, D. (2007). Characterizing knowledge-intensive tasks indicating cognitive requirements – Scenarios in methods for specific tasks. In J. Ralyté, S. Brinkkemper, & B. Henderson-Sellers (Eds.), *Proceedings of the IFIP TC8/WG8.1 working conference on situational method engineering: Fundamentals and experiences, Geneva, Switzerland* (Vol. 244). Boston, USA: Springer.

Overbeek, S., van Bommel, P., Proper, H., & Rijsenbrij, D. (2007). Knowledge discovery and exchange – Towards a web-based application for discovery and exchange of revealed knowledge. In J. Filipe, J. Cordeiro, B. Encarnação, & V. Pedrosa (Eds.), *Proceedings of the third international conference on web information systems and technologies (WEBIST), Barcelona, Spain, EU*. Setúbal, Portugal, EU: INSTICC Press.

Overbeek, S., van Bommel, P., Proper, H., & Rijsenbrij, D., 2007. Matching cognitive characteristics of actors and tasks. In R. Meersman, & T. Zari, (Eds.), *Proceedings on the move to meaningful internet systems 2007: DOA, CoopIS, ODBASE, GADA, and IS, Vilamoura, Portugal, November 25–30, 2007, Part I. Lecture notes in computer science, Vilamoura, Portugal, EU (Vol. 4803)*. Berlin, Germany, EU: Springer.

Overbeek, S., van Bommel, P., & Proper, H. (2008). Information systems engineering supported by cognitive matchmaking. In M. Léonard, & Z. Bellahsène (Eds.), *Proceedings of 20th international conference on advanced information systems engineering, CAiSE 2008, Montpellier, France, June 16–20, 2008. Lecture notes in computer science, Montpellier, France, EU*. Berlin, Germany, EU: Springer.

Proper, H. (1999). What is information discovery about? *Journal of the American Society for Information Sciences, 50*(9), 737–750.

Ritter, T., & Gemünden, H. (2004). The impact of a company's business strategy on its technological competence, network competence and innovation success. *Journal of Business Research, 57*(5), 548–556.

Ruiz, M., Díaz, M., Soler, F., & Pérez, J. (2008). Adaptation in current e-learning systems. *Computer Standards and Interfaces, 30*(1–2), 62–70.

Sol, H., 1982. Simulation in information systems. Ph.D. thesis, University of Groningen, The Netherlands, EU.

van der Aalst, W., & ter Hofstede, A. (2000). Verification of workflow task structures: A petri-net-based approach. *Information Systems, 25*(1), 43–69.

van der Aalst, W., & ter Hofstede, A. (2005). YAWL: Yet another workflow language. *Information Systems, 30*(4), 245–275.

van der Weide, T. P., & van Bommel, P. (2006). Measuring the incremental information value of documents. *Information Sciences, 176*(2), 91–119.

Vessey, I. (1991). Cognitive fit: A theory-based analysis of the graphs versus tables literature. *Decision Sciences, 22*(2), 219–240.

Wang, Y., van de Kar, E., & Meijer, G., 2005. Designing mobile solutions for mobile workers: Lessons learned from a case study. In: *ICEC'05: Proceedings of the 7th international conference on electronic commerce, Xi'an, China*. New York, NY, USA: ACM Press.

Weir, C., Nebeker, J., Bret, L., Campo, R., Drews, F., & LeBar, B. (2007). 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.

Wieser, E., Houdek, F., & Schneider, K., 2000. Push or pull: Two cognitive modes of systematic experience transfer at Daimler Chrysler. In: *Learning software organizations. Lecture notes in computer science* (Vol. 1756). Berlin: Springer.

Yin, R. (2003). *Case study research: Design and methods* (3rd ed.). CA, USA: Sage Publications, Thousand Oaks.

Please cite this article in press as: Overbeek, S. J., et al. Embedding knowledge exchange and cognitive matchmaking in a dichotomy of markets. *Expert Systems with Applications* (2009), doi:10.1016/j.eswa.2009.04.071