Matching Cognitive Characteristics of Actors and Tasks

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

[1] e-office B.V., Duwboot 20, 3991 CD Houten, The Netherlands, EU
Sietse.Overbeek@e-office.com
[2] Institute for Computing and Information Sciences, Radboud University Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, EU
{P.vanBommel,E.Proper,D.Rijsenbrij}@cs.ru.nl

Abstract. Acquisition, application and testing of knowledge by actors trying to fulfill knowledge intensive tasks is becoming increasingly important for organizations due to trends such as globalization, the emergence of virtual organizations and growing product complexity. An actor's management of basic cognitive functions, however, is at stake because of this increase in the need to acquire, apply and test knowledge during daily work. This paper specifically focusses on matchmaking between the cognitive characteristics supplied by an actor and the cognitive characteristics required to fulfill a certain knowledge intensive task. This is based on a categorization and characterization of actors and knowledge intensive tasks. A framework for a cognitive matchmaker system is introduced to compute actual match values and to be able to reason about the suitability of a specific actor to fulfill a task of a certain type.

1 Introduction

The importance of an actor's (i.e. a human or a computer) abilities to acquire, apply and test already applied knowledge increases due to e.g. growing product complexity, the move toward globalization, the emergence of virtual organizations, and the increase in focus on customer orientation [1]. A knowledge intensive task is a task for which acquisition, application or testing of already applied knowledge is necessary in order to successfully fulfill the task. When the pressure to acquire, apply and test more knowledge increases, actors struggle to manage their basic cognitive functions like e.g. the willpower to fulfill a task or maintaining awareness of the requirements to fulfill a task. These cognitive functions are also referred to as *volition* and *sentience* respectively in cognitive literature [2, 3]. Difficulties to control basic cognitive functions influences practice and potentially threatens the success of task fulfillment [4]. Research in cognitive psychology has demonstrated that individual knowledge processing is negatively influenced when experiencing an overload of knowledge that needs to be processed. For example, a burden of knowledge processing events may cause actors to underestimate the rate of events [5] and to be overconfident [6].

In [7] we have discussed several types of knowledge intensive tasks, each characterized by their characteristics. These task types consist of an *acquisition* task, a *synthesis* task, and a *testing* task. An acquisition task is related with the elicitation of knowledge. A synthesis task is related with the actual utilization of the acquired knowledge. Lastly, a testing task is related with the identification and application of knowledge in practice inducing an improvement of the specific knowledge applied. The characteristics belonging to each task type indicate the cognitive requirements necessary for an actor to successfully fulfill an instance of a specific task type. Based on this earlier work, the research reported in this paper is specifically concerned with the *matching* of cognitive characteristics required to fulfill a certain task instance with the cognitive characteristics actually *possessed* by an actor. The ambition of this paper, however, is not to come up with a tool that will be concerned with cognitive matchmaking. We will merely concentrate on determining which aspects play a role in such a matchmaking process and how these aspects could be tackled. A global matchmaker architecture illustrated in figure 1 provides a first overview of the aspects that are taken into consideration.

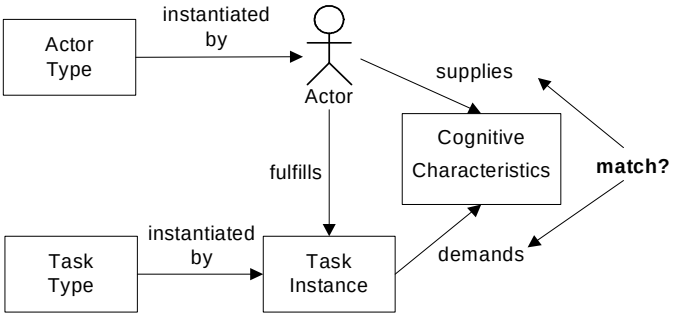

Fig. 1. Cognitive matchmaker architecture

The paper is structured as follows. Several cognitive settings of actors are discussed in section 2 to be able to characterize the different actors fulfilling a task instance. A framework of a cognitive matchmaker system is introduced in section 3 to be able to compute a match between the supply of certain cognitive characteristics by an actor and the demanded cognitive characteristics to fulfill a task instance. During section 3, the theory is materialized in a running example by matching an actor type from the theory of section 2 with a task type from theory as discussed in earlier work [7]. Section 4 briefly compares our models with other approaches in the field and outlines the benefits of our approach compared to others. Section 5 concludes this paper and gives an overview of future research plans.

2 Cognitive Actor Settings

Before elaborating on matching cognitive characteristics possessed by an actor with the cognitive characteristics required when fulfilling a task instance, a characterization of possible actor types is needed.

2.1 Actor Types

Actor types may draw from a pool of basic cognitive characteristics an actor might possess, such as sentience, volition, and causability [8]. No one actor type necessarily has all of these characteristics, and some have more than others. Using a series of linguistic diagnostics, Dowty [8] has shown that each of these characteristics can be isolated from the others, and so should be treated as distinct. The following characteristics can thus be distinguished that can be utilized to generate a framework for cognitive settings of possible different actor types:

- The *volition* characteristic is concerned with an actor's willpower to fulfill some knowledge intensive task instance.
- *Sentience* expresses that an actor has complete awareness of required knowledge to fulfill some task instance.
- 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 instance.
- During fulfillment of certain knowledge intensive task instances an actor should be able to improve its own cognitive abilities. This is indicated by the *improvability* characteristic.
- The *independency* characteristic is necessary to be able to determine if an actor is able to fulfill a task instance on its own or not.

Having determined possible cognitive characteristics an actor may have it is now appropriate to distinguish several actor types. The combination of an actor type with the cognitive characteristics belonging to a type forms a *cognitive actor setting*. This characterization is shown in table 1. The five distinguished actor types are based on a classification of knowledge worker types [9] and on linguistic literature [2]. The knowledge worker classification is more practically oriented than the ideas found in the linguistic literature. Practical as well as theoretical ideas now intermingle when developing a framework of cognitive actor settings. Now the set of actor types can be represented as:

$$\{\texttt{experiencer}, \texttt{collaborator}, \texttt{expert}, \texttt{integrator}, \texttt{transactor}\} \subseteq \mathcal{AT} \quad (1)$$

The set of cognitive characteristics can be represented as:

$$\{\texttt{volition}, \texttt{sentience}, \texttt{causability}, \texttt{improvability}, \texttt{independency}\} \subseteq \mathcal{CC} \quad (2)$$

An important remark to make here is that the possible actor types as well as the possible cognitive characteristics are not limited to five actor types and five cognitive characteristics. However, in this paper we restrict ourselves to the above mutually independent cognitive actor settings. The actor types as shown in table 1 can now be introduced:

Table 1. Cognitive actor settings characterized

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

The experiencer The *experiencer* actor type has the sentience characteristic only. An experiencer is thus only aware of all the knowledge requirements to fulfill some task instance. Consider for example the following sentence: *John thoroughly reads an article about balanced scorecards before joining a meeting about balanced scorecards.* This indicates that John, as an experiencer, probably understands that reading an article about balanced scorecards is enough to successfully prepare himself for a meeting about that topic.

The collaborator This actor type possesses the volition, causability, and improvability characteristics. A collaborator has the ability to exert an influence on state changes of knowledge involved during fulfillment of a task instance. During fulfillment of a knowledge intensive task instance 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 a task instance and requires others to fulfill a task instance. 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, as a collaborator, understands that in order to acquire knowledge about a patient's history he must collaborate with another actor. After that John is able to update the patient's log with recent changes.

The expert An expert possesses all characteristics depicted in table 1. Suppose that John is an assistant 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.

The integrator An integrator is able to fulfill a knowledge intensive task instance by working together and is able to initiate state changes of knowledge involved during task instance fulfillment. An integrator primarily wishes to acquire and apply knowledge of the highest possible quality. An engineer contributing to the construction of a flood barrier is an example of an integrator.

The transactor Volition, sentience, and independency are the characteristics belonging to the transactor actor type. A transactor can fulfill a task instance 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.

A specific instantiation of an actor type is expressed by $\mathsf{AType} : \mathcal{AC} \rightarrow \mathcal{AT}$, where $\mathcal{AC}$ is a set of *actor instances* that can be classified by a specific type. The example $\mathsf{AType}(a) = \texttt{experiencer}$ for instance expresses that an actor a can be

classified as an experiencer. We can view the actor that is specifically fulfilling a task instance $i \in \mathcal{TI}$ as a function $\mathsf{Fulfillment} : \mathcal{AC} \rightarrow \wp(\mathcal{TI})$. Here, $\mathcal{TI}$ is a set of task instances which are fulfilled by an actor. An actor a that fulfills a task instance i can be expressed as $\mathsf{Fulfillment}(a) = \{i\}$. A specific instantiation of a task type is expressed by $\mathsf{TType} : \mathcal{TI} \rightarrow \mathcal{TT}$, where $\mathcal{TT}$ is a set of *task types* that can be instantiated by a specific task instance. The expression $\mathsf{TType}(i) = \texttt{acquisition}$ can be used to assert that a task instance i is characterized as an acquisition task.

Now that a characterization of different actor types has been introduced (resulting in several cognitive actor settings), the different cognitive characteristics mentioned in table 1 need to be explored.

2.2 Definitions of Cognitive Characteristics

Volition An actor has the *volition* characteristic, if an actor has the willpower to fulfill some knowledge intensive task instance. It can be said that an actor has a *strong* motivation to fulfill a task instance. It is important to note that for each of the cognitive characteristics an actor might have, an actor may possess it at a certain level. Once an actor has the volition characteristic, however, the level of willing to fulfill a task is high because of the present strongness of the motivation. The introduction of a motivation function is necessary to determine an actor's motivation while fulfilling a task instance:

$$\mathsf{Motivation} : \mathcal{AS} \rightarrow (\mathcal{AC} \times \mathcal{TI} \rightarrow \mathcal{MO}) \tag{3}$$

The set $\mathcal{AS}$ contains *actor states*. An actor state is necessary because an actor's motivation might change over time. For example, an actor might be strongly motivated in one state, while an actor might be weakly motivated in another state. Assume $\{\texttt{weak}, \texttt{moderate}, \texttt{neutral}, \texttt{strong}\} \subseteq \mathcal{MO}$. The set $\mathcal{MO}$ includes possible motivation types of an actor. An actor a in a state $t \in \mathcal{AS}$ with a volition characteristic, however, has a strong motivation. This is denoted as: $\mathsf{Motivation}_t(a, i) = \texttt{strong}$. The volition characteristic can now be modeled as follows. An actor $a \in \mathcal{AC}$ has the volition characteristic, denoted as $\mathsf{Volition}(a)$, if that actor has a state $t \in \mathcal{AS}$ in which that actor has a strong motivation for some task instance to be fulfilled:

$$\exists_{t \in \mathcal{AS}} \exists_{i \in \mathsf{Fulfillment}(a)} [\mathsf{Motivation}_t(a, i) = \texttt{strong}] \tag{4}$$

Sentience An actor has the *sentience* characteristic, if that actor has complete awareness of required knowledge to fulfill some task instance. The level of knowing which knowledge is required for task fulfillment is high, because of the complete awareness once an actor possesses the sentience characteristic. In [7] a function has been introduced to understand to what extent a *knowledge asset* (as part of the set $\mathcal{KA}$) is applicable for a task, i.e. has a useful effect for completing the task:

$$\mathsf{Applicable} : \mathcal{TI} \times \mathcal{KA} \mapsto [0, 1] \tag{5}$$

These *assets* are tradeable forms of knowledge, i.e. knowledge that is exchangeable between actors. This may include knowledge obtained by viewing a Web site

or a document or by conversing with a colleague. When an instructor explains a learner how to drive a car for instance, the explanation may contain valuable knowledge assets for the learner.

So, $\mathsf{Applicable}(i,k) > 0$ expresses that knowledge asset k is somehow applicable for a task instance i. Another function denotes the need for knowledge of an actor during fulfillment of a task instance [7]:

$$\mathsf{Need} : \mathcal{AS} \rightarrow (\wp(\mathcal{KA}) \times \mathcal{KA} \mapsto [0,1]) \tag{6}$$

The expression $\mathsf{Need}_t(S,k)$ is interpreted as the residual need for knowledge k of an actor in state t after the set $\mathcal{S}$ has been presented to an actor, where $t \in \mathcal{AS}$, $k \in \mathcal{KA}$ and $\mathcal{S} \subseteq \mathcal{KA}$. The set $\mathcal{S}$ can be interpreted as the personal knowledge of an actor (also called a knowledge profile). At this point the sentience characteristic can be modeled:

$$\exists_{i \in \mathsf{Fulfillment}(a)} \exists_{k \in \mathcal{KA}} \exists_{S \subseteq \mathcal{KA}} [\mathsf{Applicable}(i,k) > 0 \wedge \mathsf{Need}(\mathcal{S},k) \geqslant 0] \tag{7}$$

In other words, an actor $a \in \mathcal{AC}$ has the sentience characteristic, denoted as $\mathsf{Sentience}(a)$, if that actor fulfills some task instance and if there exists a knowledge asset $k \in \mathcal{KA}$ that is applicable in a task instance and already possessed by actor a (i.e. part of that actor's knowledge profile $S \subseteq \mathcal{KA}$) or otherwise required by actor a. The actor's state has been omitted because it is not of particular relevance in the sentience characteristic.

Causability An actor has the *causability* characteristic, if an actor has the ability to exert an influence on changes of the knowledge *type* involved during fulfillment of a task instance. The level of this influence is dependent of to what extent an actor masters this characteristic. Four knowledge types are distinguished [10, 11]:

- Implicit & concealed knowledge: e.g. competencies or expertise of a worker unknown to the organization;
- Explicit & concealed knowledge: e.g. valuable insights concealed in available data collections (to be discovered by data mining);
- Implicit & revealed knowledge: e.g. known expertise of a worker which can be appealed to;
- Explicit & revealed knowledge: e.g. best-practice documentation, knowledge bases, scientific papers, etcetera.

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 workers (for an individual or for the organization) to knowledge which is hidden in undiscovered patterns in data collections (the basis for data mining). This results in *revealed* and *concealed* knowledge. To be specific, it can be said that an actor having the causability

characteristic can change knowledge from one type to another type. This can be modeled as a function:

$$\ltimes : \mathcal{AC} \rightarrow (\mathcal{KA} \times \mathcal{KT} \rightarrow \mathcal{KT}) \tag{8}$$

The set $\mathcal{KT}$ comprises the possible knowledge types. The four discussed knowledge types can formally be depicted as:

$$\{\texttt{implicit—concealed}, \texttt{implicit—revealed}, \texttt{explicit—concealed}, \texttt{explicit—revealed}\} \subseteq \mathcal{KT} \tag{9}$$

The knowledge type of a specific knowledge item k can easily be found by using the function $\mathsf{KType} : \mathcal{KA} \rightarrow \mathcal{KT}$. For example, $\mathsf{KType}(k) = s$ expresses that knowledge $k \in \mathcal{KA}$ is of the type $s \in \mathcal{KT}$. When knowledge asset k of type s is changed to another type by actor $a \in \mathcal{AC}$, this type change is denoted as: $\ltimes_a(k, s)$. When applying the infix notation this would result in: $\ltimes_a(k, s) \equiv k \ltimes_a s$. At this point the causability characteristic can be modeled:

$$\exists_{i \in \mathsf{Fulfillment}(a)} \exists_{k \in \mathcal{KA}} \exists_{s \in \mathcal{KT}} [\mathsf{Applicable}(i, k) > 0 \wedge k \ltimes_a s] \tag{10}$$

In other words, an actor $a \in \mathcal{AC}$ has the causability characteristic, denoted as $\mathsf{Causability}(a)$, if that actor fulfills some task instance and if there exists a knowledge asset $k \in \mathcal{KA}$ of some type $s \in \mathcal{KT}$ that is changed to some other type $k \ltimes_a s \in \mathcal{KT}$ by actor a.

Improvability An actor has the *improvability* characteristic, if that actor is able to improve its own cognitive capabilities while fulfilling some task instance. An actor may have a certain level to improve its own capabilities, ranging from e.g. a low level to a high level. First, it is necessary to introduce a $\twoheadrightarrow^*$ operator that expresses an actor's state change after fulfilling some task instance:

$$\twoheadrightarrow^* : \mathcal{AS} \times \mathcal{TI} \rightarrow \mathcal{AS} \tag{11}$$

Thus, an actor state t changes to state $t \twoheadrightarrow^* i$ after fulfilling task instance i. However, to construct the improvability characteristic the actual improvement of cognitive characteristics should also be tackled. The following actor characteristics function can be utilized to solve this issue:

$$\mathsf{AChar} : \mathcal{AS} \rightarrow (\mathcal{AC} \rightarrow \wp(\mathcal{CC})) \tag{12}$$

This function specifies which cognitive characteristics as part of the set $\mathcal{CC}$ belong to a certain actor instance (that is classified by an actor type). The set $\mathcal{AS}$ contains *actor states*. An actor state is necessary because the characterization of an actor might change over time. An actor $a \in \mathcal{AC}$ possessing cognitive characteristics included in a set of cognitive characteristics C while in state $t \in \mathcal{AS}$ can be denoted as: $\mathsf{AChar}_t(a) = C$. The improvability characteristic can be modeled subsequently:

$$\exists_{i \in \mathsf{Fulfillment}(a)} \exists_{t \in \mathcal{AS}} [\mathsf{AChar}_t(a) \subseteq \mathsf{AChar}_{t \twoheadrightarrow^* i}(a)] \tag{13}$$

An actor $a \in \mathcal{AC}$ has the improvability characteristic, denoted as $\mathsf{Improvability}(a)$, if the set of cognitive characteristics $\mathsf{AChar}_t(a) = C$ can be complemented with additional characteristics after fulfilling some task instance while being in some state $t \in \mathcal{AS}$.

Independency An actor has the *independency* characteristic, if that actor is able to fulfill some task instance on its own. If an actor is fully able to fulfill a task instance on its own, then it can be said that an actor has the characteristic at a (very) high level and vice versa. A fulfiller function is necessary to reason specifically about actors that are fulfilling some task instance:

$$\mathsf{Fulfiller} : \mathcal{TI} \rightarrow \wp(\mathcal{AC}) \tag{14}$$

If it is necessary to determine *fulfillers* of a task instance i, the fulfiller function returns actors responsible for the fulfillment of some task. The independency characteristic can be modeled as follows:

$$\exists_{i \in \mathcal{TI}}[\mathsf{Fulfiller}(i) = \{a\}] \tag{15}$$

An actor a has the independency characteristic, denoted as $\mathsf{Independency(a)}$, if for task instance i the only fulfiller is actor a.

In order to have a graphical representation of the discussed definitions throughout section 2, an Object-Role Modeling (ORM) model is presented in figure 2. For details on Object-Role Modeling, see e.g. [12]. Now that several cognitive

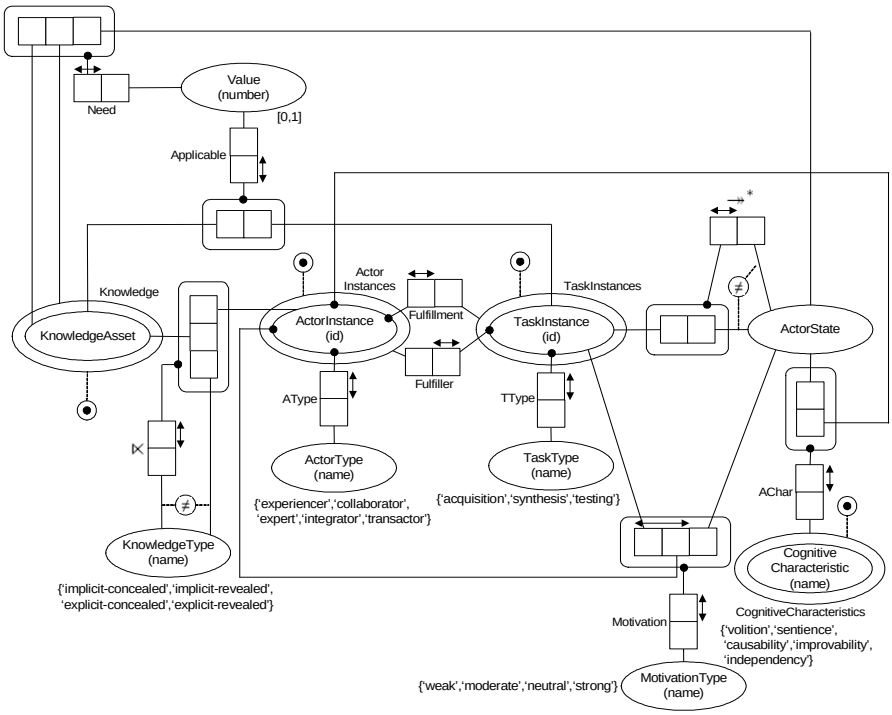

Fig. 2. Object-Role Modeling (ORM) model of cognitive actor settings

actor settings have been explored in detail, a cognitive matchmaker system can be used to match the actors with the tasks they can fulfill.

3 Cognitive Matchmaker System

In this section a framework for a cognitive matchmaker system is introduced that is able to compute a match between cognitive characteristics required for a specific task type and cognitive characteristics that are provided by a specific actor type. As a running example, we use the matchmaker system to match the cognitive characteristics offered by the *transactor* actor type with the required cognitive characteristics of a *synthesis* task. Figure 3 shows the architecture of the system on a conceptual level, which is translated into the formalisms throughout this section. In section 2, a function $\mathsf{AChar}_j(a) = C$ indicated the cognitive

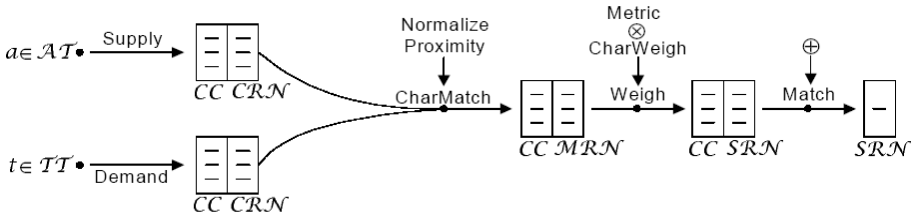

Fig. 3. Cognitive matchmaker system

characteristics that characterized an actor instance of a certain type, where j is a task type belonging to the set of task types $\mathcal{TT}$, a is an actor instance belonging to the set of actor instances $\mathcal{AC}$ and C is a set of cognitive characteristics that is a subset of or equal to $\mathcal{CC}$. Recall from section 2 that the corresponding actor type can be found by using the *actor type* function: $\mathsf{AType}(a) = j$. With this in mind, a *supply* function can be modeled that returns a value expressing to what extent an actor type offers a certain cognitive characteristic:

$$\mathsf{Supply} : \mathcal{AT} \rightarrow (\mathcal{CC} \rightarrow \mathcal{CRN}) \tag{16}$$

The expression $\mathsf{Supply}_{\mathtt{transactor}}(\mathtt{s}) = 10$ shows that an actor characterized by the transactor type offers the sentience characteristic and is at least capable to perform this characteristic at level 10. Note that the word 'sentience' has been abbreviated to the letter 's'. For readability reasons we will continue to use this abbreviation for the remaining example expressions. The resulting value '10' is part of a characteristic rank domain $\mathcal{CRN}$ which contains integer values within the range $[0, 10]$. The hard values as part of a domain of values can be found using the following function:

$$\mathsf{Numerical} : \wp(\mathcal{RN}) \rightarrow \mathbb{R} \tag{17}$$

Here, the set $\mathcal{RN}$ contains rank values and $\mathcal{CRN} \subseteq \mathcal{RN}$. Formally, the characteristic rank domain includes the following hard values: $\mathsf{Numerical}(\mathit{CRN}) = [0, 10]$. A value of 0 means that an actor is not able to offer a certain characteristic, a value of 5 means that an actor is able to offer a characteristic at an average level and a value of 10 means that an actor is able to offer a characteristic at the highest level. So, in the case of the example, the transactor is able to offer the sentience characteristic at the highest level.

It is possible here to introduce a characteristic rank set containing linguistic (soft) values instead of a characteristic rank set that contains numerical (hard) values. A linguistic value differs from a numerical value in that its values are not numbers but words or sentences in some language. The resulting values of the examples reported, however, are mapped on a domain containing hard values only. As discussed in section 5, the next step in this research will be concerned with *fuzzy assessments* to indicate a certain capability level. In the case of the example above, this would mean that we are able to reason that the transactor is able to offer the sentience characteristic at e.g. a *very high* level. Besides modeling a supply function, a demand function is needed that returns a value expressing to what extent a cognitive characteristic is *required* for a certain task type:

$$\mathsf{Demand} : \mathcal{TT} \rightarrow (\mathcal{CC} \rightarrow \mathcal{CRN}) \tag{18}$$

The expression $\mathsf{Demand}_{\mathtt{synthesis}}(\mathtt{s}) = 10$ indicates that a sentience characteristic is required at the highest level in order to fulfill a task of the synthesis type. The supply and demand functions can now be used together to compute the characteristic match.

3.1 Characteristic Match

In this section, a characteristic match function is defined to compare the resulting values from the supply and demand functions. This comparison should provide insight in the way supply and demand of cognitive characteristics are matched with each other. In order to model a *characteristic match* function, an actor type as well as a task type are required as input, together with a cognitive characteristic from the set $\mathcal{CC}$ of cognitive characteristics:

$$\mathsf{CharMatch} : \mathcal{AT} \times \mathcal{TT} \rightarrow (\mathcal{CC} \rightarrow \mathcal{MRN}) \tag{19}$$

As can be seen in figure 3, the characteristic match function returns a value from the match rank domain, where $\mathcal{MRN} \subseteq \mathcal{RN}$. The match rank domain includes the following values: $\mathsf{Numerical}(MRN) = [0, 10]$.

To compute the actual characteristic match value, a *proximity* function is necessary to be able to define the characteristic match function. This proximity function should compute the proximity of the level an actor offers a certain cognitive characteristic related to the level that is required in order to fulfill a task of a certain type. The values that should be used as input for the proximity function are part of the characteristic rank domain. The resulting proximity value is then a value that is part of the match rank domain:

$$\mathsf{Proximity} : \mathcal{CRN} \times \mathcal{CRN} \rightarrow \mathcal{MRN} \tag{20}$$

A normalization function can be introduced that calculates the numerical proximity of demand and supply when a cognitive characteristic is concerned:

$$\mathsf{Normalize} : \mathbb{R} \mapsto [0, 1] \tag{21}$$

The normalization function can be defined by using the supply and demand functions and two additional constants `min` and `max`:

$$\mathsf{Normalize}(\mathsf{Supply}_i(c) - \mathsf{Demand}_j(c)) \triangleq \frac{\mathsf{Supply}_i(c) - \mathsf{Demand}_j(c) + \mathtt{max} - \mathtt{min}}{2 \cdot (\mathtt{max} - \mathtt{min})} \tag{22}$$

Here, i is an actor type of the set $\mathcal{AT}$, j is a task type of the set $\mathcal{TT}$ and c is a cognitive characteristic of the set $\mathcal{C}$. The values of the constants $\texttt{min}$ and $\texttt{max}$ can be determined by interpreting the minimum and the maximum value of the characteristic rank domain. So, in the case of the running example $\texttt{min} = 0$ and $\texttt{max} = 10$. The minimum value that can be returned by the normalization function is 0. This occurs if there is absolutely no supply (i.e. an incapable actor is concerned) but there is a maximum demand of a certain cognitive characteristic in order to fulfill a task of a certain type. This situation is depicted below:

$$\mathsf{Normalize}(0 - 10) = \frac{0 - 10 + \texttt{max} - \texttt{min}}{2 \cdot (\texttt{max} - \texttt{min})} = 0$$

In the case of an overqualified actor that is more capable to perform a cognitive characteristic than is required, the normalization function returns 1:

$$\mathsf{Normalize}(10 - 0) = \frac{10 - 0 + \texttt{max} - \texttt{min}}{2 \cdot (\texttt{max} - \texttt{min})} = 1$$

This means that the normalization function normalizes the proximity of supply and demand between 0 and 1. Using the normalization function, the proximity function can now be defined as follows:

$$\mathsf{Proximity}(\mathsf{Supply}_i(c), \mathsf{Demand}_j(c)) \triangleq \mathsf{Normalize}(\mathsf{Supply}_i(c) - \mathsf{Demand}_j(c)) \tag{23}$$

Regarding the running example the proximity function as defined above results in:

$$\mathsf{Proximity}(10, 10) = \mathsf{Normalize}(10 - 10) = 0.5$$

Now with the introduction of a proximity function the characteristic match can be defined by computing the proximity of demand and supply in the context of a given characteristic:

$$\mathsf{CharMatch}(i, j) \triangleq \lambda_{c \in \mathcal{C}} \cdot \mathsf{Proximity}(\mathsf{Supply}_i(c), \mathsf{Demand}_j(c)) \tag{24}$$

Recall from section 3 that an actor of the transactor type is able to perform the sentience characteristic at level 10, which equals the level to what extent a sentience characteristic should be mastered for a synthesis task type. In the case of our example the characteristic match results in:

$$\begin{aligned}&\mathsf{CharMatch}(\texttt{transactor}, \texttt{synthesis}) = \\ &\lambda_{\texttt{s} \in \mathcal{C}} \cdot \mathsf{Proximity}(\mathsf{Supply}_{\texttt{transactor}}(\texttt{s}), \mathsf{Demand}_{\texttt{synthesis}}(\texttt{s})) = \\ &\mathsf{Proximity}(10, 10) = 0.5\end{aligned}$$

This example shows that if an actor characterized as a transactor masters a sentience characteristic at level 10 and if it is also *needed* to master the sentience characteristic at level 10 to fulfill a task instance of the synthesis type, the eventual *proximity value* is 0.5. However, this proximity value is only related to the demand and supply of one specific cognitive characteristic. To compute a total match of the required cognitive characteristics in a task type and the characteristics offered, a *weighed suitability match* is introduced in the following section.

3.2 Weighed Suitability Match

The cognitive matchmaker system is completed by introducing a weighed suitability match, as is shown in the rightmost part of figure 3:

$$\mathsf{Match} : \mathcal{AT} \times \mathcal{TT} \rightarrow \mathcal{SRN} \tag{25}$$

This function returns a value from the suitability rank domain, where $\mathcal{SRN} \subseteq \mathcal{RN}$. The suitability rank domain includes the following values: $\mathsf{Numerical}(\mathcal{SRN}) = [0, 10]$. This means that an actor of a certain type can have suitability levels ranging from 0 to 10. To determine the suitability of the transactor fulfilling the synthesis task, the calculated proximity of demand and supply of a cognitive characteristic $c \in \mathcal{CC}$ can be weighed:

$$\mathsf{Weigh} : (\mathcal{CC} \rightarrow \mathcal{MRN}) \rightarrow (\mathcal{CC} \rightarrow \mathcal{SRN}) \tag{26}$$

To define the weigh function several other functions are necessary, though. As can be seen in figure 3 the weigh function uses the input from the characteristic match function and returns a value from the suitability rank domain as output. To construct the weigh function, a function is needed that has a match rank metric (i.e. the proximity value) as its input and a suitability rank metric as its output:

$$\mathsf{Metric} : \mathcal{MRN} \rightarrow \mathcal{SRN} \tag{27}$$

For instance, $\mathsf{Metric}(0.5) = 0.5$ shows that the value 0.5, which is the proximity value, equals the value 0.5 which is a suitability rank metric. A characteristic weigh function is needed to actually weigh the importance of a certain cognitive characteristic to fulfill a task of a certain type:

$$\mathsf{CharWeigh} : \mathcal{CC} \rightarrow \mathcal{SRN} \tag{28}$$

So, $\mathsf{CharWeigh}(\mathtt{s}) = 1.5$ means that a weigh factor of 1.5 is given to indicate the importance of mastering the sentience cognitive characteristic (for a certain task). Finally, the $\otimes$ operator is also needed to define a definite weigh function:

$$\otimes : \mathcal{SRN} \times \mathcal{SRN} \rightarrow \mathcal{SRN} \tag{29}$$

The $\otimes$ operator is necessary to multiply the metric value with the characteristic weigh value. If the values mentioned above are multiplied this results in $0.5 \otimes 1.5 = 0.75$. The weigh function can now be defined as:

$$\mathsf{Weigh}(c, \mathsf{CharMatch}(i, j)) \triangleq \lambda_{c \in \mathcal{CC}} \cdot \mathsf{Metric}(\mathsf{CharMatch}(i, j)) \otimes \mathsf{CharWeigh}(c) \tag{30}$$

Here, $c \in \mathcal{CC}, i \in \mathcal{AT}$ and $j \in \mathcal{TT}$. Continuing the running example, we would like to calculate the suitability of the transactor that is fulfilling a task instance of the synthesis type. Considering the sentience characteristic only, this can be computed as follows:

$$\begin{aligned} &\mathsf{Weigh}(\mathtt{s}, \mathsf{CharMatch}(\mathtt{transactor}, \mathtt{synthesis})) = \\ &\lambda_{\mathtt{s} \in \mathcal{CC}} \cdot \mathsf{Metric}(0.5) \otimes \mathsf{CharWeigh}(\mathtt{s}) = \\ &0.5 \otimes 1.5 = 0.75 \end{aligned}$$

In order to calculate the suitability match of the transactor actor type related to the synthesis task type of our example, it is mandatory to determine the

cognitive characteristics supplied by the actor and demanded by the task. The transactor actor type supplies the *volition*, *sentience* and *independency* characteristics as is shown in table 1. The synthesis task type can be characterized by the *applicability* and *correctness* characteristics [7]. These characteristics are explained as follows. An actor should provide the applicability characteristic to be able to apply knowledge during task fulfillment and to make sure that the applied knowledge has a useful effect on successfully completing the task. An actor should provide the correctness characteristic to be able to judge the usefulness of applied knowledge in a task and to be sure that applied knowledge meets its requirements.

In the case of the running example (i.e. only when the transactor actor type and the synthesis task type are concerned) the set $\mathcal{C}$ contains the following characteristics: $\{\texttt{volition}, \texttt{sentience}, \texttt{independency}, \texttt{applicability}, \texttt{correctness}\} \subseteq \mathcal{C}$. For all these properties a weigh value needs to be determined using the functions mentioned throughout section 3. This is necessary to compute a final *suitability match* resulting in one suitability rank value. The calculations leading to weighed characteristic matches are elaborated in tables 2 and 3. The actual

Table 2. Example calculations for characteristic matches

Item	Characteristic	Characteristic Match
a.	volition	$\mathsf{CharMatch}(\texttt{transactor}, \texttt{synthesis}) = \mathsf{Proximity}(10, 10) = 0.5$
b.	sentience	$\mathsf{CharMatch}(\texttt{transactor}, \texttt{synthesis}) = \mathsf{Proximity}(10, 10) = 0.5$
c.	independency	$\mathsf{CharMatch}(\texttt{transactor}, \texttt{synthesis}) = \mathsf{Proximity}(10, 0) = 1$
d.	applicability	$\mathsf{CharMatch}(\texttt{transactor}, \texttt{synthesis}) = \mathsf{Proximity}(5, 10) = 0.25$
e.	correctness	$\mathsf{CharMatch}(\texttt{transactor}, \texttt{synthesis}) = \mathsf{Proximity}(5, 10) = 0.25$

Table 3. Example calculations for weighed characteristic matches

Item	Weighed Characteristic Match
a.	$\mathsf{Weigh}(\texttt{volition}, 0.5) = \mathsf{Metric}(0.5) \otimes \mathsf{CharWeigh}(\texttt{volition}) = 0.5 \otimes 2 = 1$
b.	$\mathsf{Weigh}(\texttt{sentience}, 0.5) = \mathsf{Metric}(0.5) \otimes \mathsf{CharWeigh}(\texttt{sentience}) = 0.5 \otimes 1.5 = 0.75$
c.	$\mathsf{Weigh}(\texttt{indepedency}, 1) = \mathsf{Metric}(1) \otimes \mathsf{CharWeigh}(\texttt{independency}) = 1 \otimes 0.5 = 0.5$
d.	$\mathsf{Weigh}(\texttt{applicability}, 0.25) = \mathsf{Metric}(0.25) \otimes \mathsf{CharWeigh}(\texttt{applicability}) = 0.25 \otimes 3 = 0.75$
e.	$\mathsf{Weigh}(\texttt{correctness}, 0.25) = \mathsf{Metric}(0.25) \otimes \mathsf{CharWeigh}(\texttt{correctness}) = 0.25 \otimes 3 = 0.75$

characteristic weigh values (for every cognitive characteristic as part of the set $\mathcal{C}$) denoted in table 3 are: 2, 1.5, 0.5, 3 and 3. Note that these characteristic weigh values always summate to one and the same total value. In the case of our example the characteristic weigh values summate to 10. Thus, no matter how the weigh values are divided across the cognitive characteristics, they should always summate to a total of 10.

The results of the weighed characteristic matches, which are denoted in the rightmost column of table 3, have to be summated to generate a single *suitability*

match value. To summate these values a $\oplus$ operator is required:

$$\oplus : \mathcal{SRN} \times \mathcal{SRN} \rightarrow \mathcal{SRN} \tag{31}$$

Now the final match function can be defined using the aforementioned functions:

$$\mathsf{Match}(i,j) \triangleq \bigoplus_{c \in \mathcal{CC}} \mathsf{Weigh}(c, \mathsf{CharMatch}(i,j)) \tag{32}$$

In the match function $i \in \mathcal{AT}, c \in \mathcal{CC}$ and $j \in \mathcal{TT}$. For the running example this means that the suitability match value of the transactor fulfilling a task instance of the synthesis type is computed as follows:

$$\mathsf{Match}(\texttt{transactor}, \texttt{synthesis}) = 1 \oplus 0.75 \oplus 0.5 \oplus 0.75 \oplus 0.75 = \mathbf{3.75}$$

As a result of the suitability match it can be concluded that the suitability of an actor characterized by the transactor type fulfilling a task instance of the synthesis type is 3.75. Remember that the lowest suitability value is 0 and the highest suitability value that can be reached is 10. The lowest value is reached if the supply of every characteristic is 0 and the demand of every characteristic is 10. The highest value is reached in the case of complete overqualification, i.e. if the supply of every characteristic is 10 and the demand of every characteristic is 0. At this point a decision can be made whether or not the specific actor is suitable enough to fulfill this task or if another actor is present who should be more suitable, i.e. has a better suitability match value. The suitability of an actor to fulfill a certain task is probably best if the resulting suitability value is 5. Underqualification as well as overqualification are both considered undesirable.

A certainty function can now be introduced to make sure how certain it is that an actor is suitable to fulfill a task:

$$\mu : \mathbb{R} \mapsto [0,1] \tag{33}$$

A linear certainty function can be defined as follows:

$$\mu(u) \triangleq \begin{cases} \frac{2}{\texttt{min}+\texttt{max}} \cdot u & \texttt{min} \leq u \leq \frac{\texttt{max}+\texttt{max}}{2} \\ \frac{-2}{\texttt{min}+\texttt{max}} \cdot u + 2 & \frac{\texttt{min}+\texttt{max}}{2} < u \leq \texttt{max} \end{cases} \tag{34}$$

For the running example, where $\texttt{min} = 0$ and $\texttt{max} = 10$, the following expression shows that the certainty that the transactor is suitable to fulfill the synthesis task is 0.75:

$$\mu(3.75) = \frac{2}{0+10} \cdot 3.75 = 0.75$$

This can be interpreted as being 75% sure that the transactor is suitable enough to fulfill the synthesis task. It might be a good choice to let the transactor fulfill the synthesis task, unless an available actor characterized by another type provides a better match.

Throughout section 3 definitions have been discussed along with their corresponding examples. Table 4 provides an overview of the definitions and the examples. In order to also have a graphical representation of the discussed definitions throughout section 3, another ORM model is presented in figure 4. All formalisms mentioned up till now are visualized by means of the ORM models of figures 2 and 4.

Table 4. Definitions of the cognitive matchmaker system with examples

Function	Example
$\mathsf{Supply} : \mathcal{AT} \to (\mathcal{CC} \to \mathcal{CRN})$	$\mathsf{Supply}_{\mathtt{transactor}}(\mathtt{s}) = 10$
$\mathsf{Numerical} : \wp(\mathcal{RN}) \to \mathbb{R}$	$\mathsf{Numerical}(\mathit{CRN}) = [0, 10]$
$\mathsf{Demand} : \mathcal{TT} \to (\mathcal{CC} \to \mathcal{CRN})$	$\mathsf{Demand}_{\mathtt{synthesis}}(\mathtt{s}) = 10$
$\mathsf{CharMatch} : \mathcal{AT} \times \mathcal{TT} \to (\mathcal{CC} \to \mathcal{MRN})$	$\mathsf{CharMatch}(\mathtt{transactor}, \mathtt{synthesis}) =$ $\lambda_{\mathtt{s} \in \mathcal{CC}} \cdot \mathsf{Proximity}(\mathsf{Supply}_{\mathtt{transactor}}(\mathtt{s}), \mathsf{Demand}_{\mathtt{synthesis}}(\mathtt{s})) =$ $\mathsf{Proximity}(10, 10) = 0.5$
$\mathsf{Proximity} : \mathcal{CRN} \times \mathcal{CRN} \to \mathcal{MRN}$	$\mathsf{Proximity}(10, 10) = \mathsf{Normalize}(10 - 10) = 0.5$
$\mathsf{Normalize} : \mathbb{R} \mapsto [0, 1]$	$\mathsf{Normalize}(10 - 10) = \frac{10-10+\mathtt{max}-\mathtt{min}}{2 \cdot (\mathtt{max}-\mathtt{min})} = 0.5$
$\mathsf{Match} : \mathcal{AT} \times \mathcal{TT} \to \mathcal{SRN}$	$\mathsf{Match}(\mathtt{transactor}, \mathtt{synthesis}) =$ $1 \oplus 0.75 \oplus 0.5 \oplus 0.75 \oplus 0.75 = 3.75$
$\mathsf{Weigh} : (\mathcal{CC} \to \mathcal{MRN}) \to (\mathcal{CC} \to \mathcal{SRN})$	$\mathsf{Weigh}(\mathtt{s}, \mathsf{CharMatch}(\mathtt{transactor}, \mathtt{synthesis})) =$ $\lambda_{\mathtt{s} \in \mathcal{CC}} \cdot \mathsf{Metric}(0.5) \otimes \mathsf{CharWeigh}(\mathtt{s}) = 0.5 \otimes 1.5 = 0.75$
$\mathsf{Metric} : \mathcal{MRN} \to \mathcal{SRN}$	$\mathsf{Metric}(0.5) = 0.5$
$\mathsf{CharWeigh} : \mathcal{CC} \to \mathcal{SRN}$	$\mathsf{CharWeigh}(\mathtt{s}) = 1.5$
$\otimes : \mathcal{SRN} \times \mathcal{SRN} \to \mathcal{SRN}$	$0.5 \otimes 1.5 = 0.75$
$\oplus : \mathcal{SRN} \times \mathcal{SRN} \to \mathcal{SRN}$	$1 \oplus 0.75 \oplus 0.5 \oplus 0.75 \oplus 0.75 = 3.75$
$\mu : \mathbb{R} \mapsto [0, 1]$	$\mu(3.75) = 0.75$

4 Discussion

Literature indicates that matchmaking solutions are possible in different ways. The matchmaker system of Shu et al. [13] matches the supply of a (Web) service provider with the demand of a service requester. A division of the concept of matchmaking is made in two categories: syntactic and semantic matchmaking. Relating this division with the matchmaking problem discussed in our study, it can be said that syntactic matchmaking determines a match dependent of the enacted task, which can be an acquisition task, a synthesis task or a testing task in the case of our model. In the case of semantic matchmaking the meaning and informational content of the cognitive characteristics provided by an actor are related with the meaning and informational content of the task requirements. However, it is not self-evident to categorize our matchmaking system in one of these two categories. Our system should be capable of more matchmaking functionality than syntactic matchmaking because of the introduced formalisms that not only take task specifications into account, but also actor specifications and cognitive characteristics. Semantic matchmaking is a matchmaking category that requires an ontology to determine the informational semantics of that what is required and that what is supplied. The disadvantages of this type of matchmaking are related with the necessity of an ontology and the quite complex algorithms needed to compute an actual match result.

A lot of studies in the matchmaking field are especially related to recommender systems. The research of Vivacqua et al. [14] for instance presents how opportunities for collaboration between actors can be determined by matching an actor's current context (as determined by the actor's work environment)

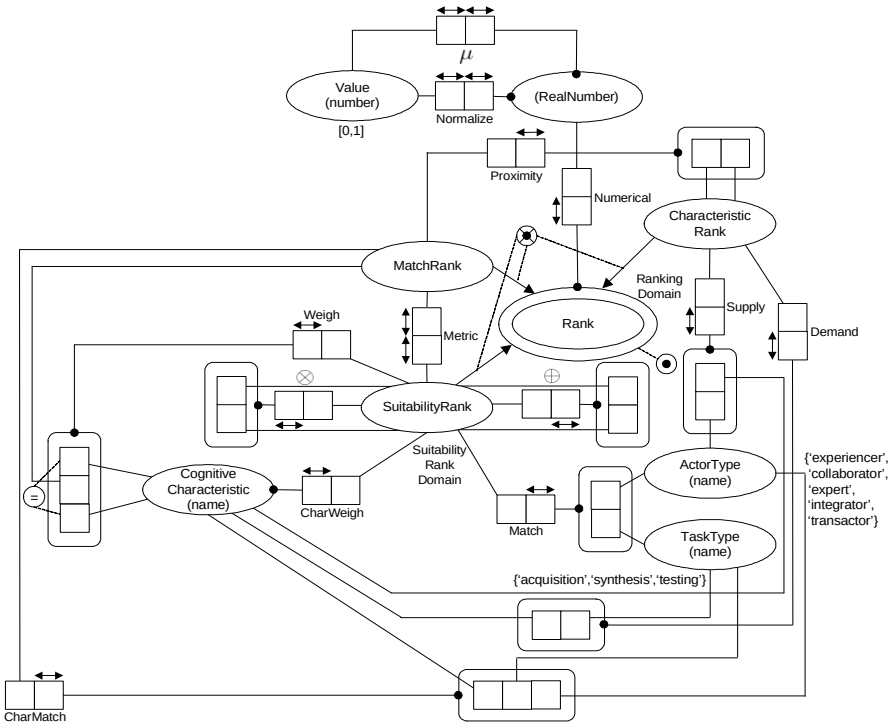

Fig. 4. Object-Role Modeling (ORM) model of the cognitive matchmaker system

with other actors that might have related interests or work. Matches are made through keyword similarity calculation. A drawback of this approach is that a specific algorithm is required to create a list of keywords for *every* document as part of an actor's work environment. Regarding our approach though, no additional algorithms are necessary to compute a match value because of the ranking mechanisms that are present in the theory itself. Also, one may ponder to what extent collaboration opportunities between actors can be determined when only their overlapping documents are taken into consideration. The primary goal of using our system is more straightforward, namely to find actor / task matches based on cognitive characteristics to diminish the cognitive load of an actor as is explained in section 1.

With regard to cognitive psychology, the ideas presented in this paper somehow relate with the notion of *cognitive fit* [15]. Following this theory, if the types of knowledge emphasized in the actor and task elements match, the actor can employ processes (and formulate a mental representation) that also emphasizes the same type of knowledge. Cognitive fit then exists because the cognitive processes used to complete the task match. A difference with our approach is that an actor can be classified based on that actor's current cognitive profile (i.e. the way an actor is able to perform the defined cognitive characteristics), instead

of determining an actor's perception of how to complete a task. 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. However, an actor's cognitive capabilities may change over time (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 which can be disadvantageous.

5 Conclusions & Future Work

This paper describes a categorization and characterization of actors that are able to fulfill knowledge intensive tasks, illustrated by definitions of cognitive characteristics indicating actor abilities for task fulfillment. Proceeding from these definitions a running example, in which a match is determined of an actor characterized by the transactor type wishing to fulfill a synthesis task, shows how the theory can be materialized.

A first aspect of future research is that of expanding the cognitive matchmaker system framework with the capability to compute a suitability match based on fuzzy assessments. Instead of working with numerical (hard) values, it is then possible to work with linguistic (soft) values. The different levels of supply and demand of cognitive characteristics may then be expressed in terms of *low*, *medium* and *high* for instance.

In this stage of the research, the proposed cognitive matchmaker system computes a suitability match on the type level. In other words, the possible different actor *types* and task *types* are taken into consideration. This is expressed by the *supply* and *demand* functions of section 3 in which the set of actor types $\mathcal{AT}$ and the set of task types $\mathcal{TT}$ are used respectively. A total of 5 actor types and 3 task types that are distinguished up till now would create $5 \times 3 = 15$ matching combinations. A future research goal is to compute suitability matches based on actor and task instances by practically exploiting the formal models mentioned in this paper. Suppose that actor 'John Doe' working at an organization can be classified as an experiencer actor type because he only possesses the sentience cognitive characteristic. John Doe then *instantiates* the experiencer actor type. In the case of John Doe a match can then be determined for the tasks he instantiates during his work.

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.

Finally, possible personal preferences of actors regarding task execution can be taken into account. Such preferences might influence the suitability of an actor fulfilling a task. 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 nega-

tively influence the actor's task performance. Besides the personal preferences, it might be interesting to understand an actor's personal goals to determine a match between an actor and a task. This means that for the near future our matchmaker system can be expanded with the possibility to reckon an actor's personal preferences and goals together with an actor's supply of cognitive characteristics.

References

1. Staab, S., Studer, R., Schnurr, H., Sure, Y.: Knowledge processes and ontologies. IEEE Intelligent Systems **16**(1) (2001) 26–34
2. Kako, E.: Thematic role properties of subjects and objects. Cognition **101**(1) (2006) 1–42
3. Weir, C., Nebeker, J., Bret, L., Campo, R., Drews, F., LeBar, B.: A cognitive task analysis of information management strategies in a computerized provider order entry environment. Journal of the American Medical Informatics Association **14**(1) (2007) 65–75
4. Meiran, N.: Modeling cognitive control in task-switching. Psychological Research **63**(3–4) (2000) 234–249
5. Hertwig, R., Barron, G., Weber, E., Erev, I.: The role of information sampling in risky choice. In Fiedler, K., Juslin, P., eds.: Information Sampling and Adaptive Cognition. Cambridge University Press, New York, NY, USA (2006) 72–91
6. Koehler, D.: Explanation, imagination, and confidence in judgment. Psychological Bulletin **110**(3) (1991) 499–519
7. Overbeek, S., van Bommel, P., Proper, H., Rijsenbrij, D.: Characterizing knowledge intensive tasks indicating cognitive requirements – scenarios in methods for specific tasks. In Ralyté, J., Brinkkempers, S., Henderson-Sellers, B., eds.: Proceedings of the IFIP TC8 / WG8.1 Working Conference on Situational Method Engineering: Fundamentals and Experiences, University of Geneva, Switzerland, EU, Springer, Boston, USA (2007)
8. Dowty, D.: Thematic proto-roles and argument selection. Language **67**(3) (1991) 547–619
9. Davenport, T.: Thinking for a Living – How to get Better Performances and Results from Knowledge Workers. Harvard Business School Press, Boston, MA, USA (2005)
10. Hoppenbrouwers, S., Proper, H.: Knowledge discovery: De zoektocht naar verhulde en onthulde kennis. DB/Magazine **10**(7) (1999) 21–25 In Dutch.
11. Nonaka, I., Takeuchi, H.: The Knowledge Creating Company. Oxford University Press, New York, NY, USA (1995)
12. Halpin, T.: Information Modeling and Relational Databases, from Conceptual Analysis to Logical Design. Morgan Kaufmann, San Mateo, CA, USA (2001)
13. Shu, G., Rana, O., Avis, N., Dingfang, C.: Ontology-based semantic matchmaking approach. Advances in Engineering Software **38**(1) (2007) 59–67
14. Vivacqua, A., Moreno, M., de Souza, J.: Profiling and matchmaking strategies in support of opportunistic collaboration. In Meersman, R., Zahir, T., Schmidt, D., eds.: On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, Berlin, Germany, EU, Springer (November 2003) 162–177
15. Vessey, I.: Cognitive fit: A theory-based analysis of the graphs versus tables literature. Decision Sciences **22**(2) (1991) 219–240