DOI: 10.2298/CSIS110304048Z

A Reusable Agent Design Pattern with Flexibility and Extensibility

Hao Lan Zhang[1], Wenhua Zeng[2], and Christian Van der Velden [3]

[1] School of Management, NIT, Zhejiang University,
No. 1 Qianhu South Road, Ningbo 315100, China
haolan.zhang@nit.zju.edu.cn
[2] School of Software, Xiamen University,
361005 Xiamen, China.
whzeng@xmu.edu.cn
[3] BAE Systems Australia, 40 River Boulevard
Richmond, VIC, 3121, Australia
christian.vandervelden@baesystems.com

Abstract. Intelligent agent-based systems are regarded as the promising technology in bridging the gap between the physical world and cyber-applications. In spite of the rising demands for reusable information systems; current designs are still insufficient in providing efficient reusable mechanisms for system design. One of the major problems hinders the development of information reuse in most traditional systems is the lack of the autonomous character among system modules or subsystems. The emergence of agent technology is able to solve the problem plaguing many traditional systems. Existing agent design models create an agent as a sole system with built-in domain-specific capabilities. However, this design pattern causes several problems while matching and updating agents' capabilities due to the built-in design pattern in these models decreases agents' extensibility, flexibility and reusability. In this paper we introduce a novel design for agent-based systems, which is able to provide an efficient design pattern for improving the reusability, extensibility and flexibility of agent design. The novel agent capability design offers an open and flexible structure; and implements several practical algorithms that can improve the system performance. An experimental program based on several practical cases has been developed to evaluate the performance of the proposed design. The empirical results reveal the efficiency of the new agent design pattern.

Keywords: Agent Capability Design, Agent Reusability, Domain Specific Components, Agent Design.

1. Introduction

Information reuse has become a key issue for information system designers in order to reduce information redundancy and system development expenses. Various systems have implemented reusable mechanisms; agent-based systems are among these systems with fast-growing demands for information reuse. In this paper we introduce a component- based design for agent capability reuse, i.e. the Domain Specific Component (DSC) design. The mechanisms used in this design can be generally applied to various agent-based systems for capability reuse.

Many existing agent-based systems have been focusing on developing service-oriented agents or component-based agents for complex problem-solving processes [1, 2, 3, 4]. These systems adopt various mechanisms to enhance the system reusability and flexibility. However, agent capabilities developed in these systems are generally integrated to agents and together form a complete component. This design pattern decreases the flexibility and reusability of agents. Under the circumstances, this paper proposes a new mechanism that could efficiently improve the reusability of task-oriented agents.

Unlike traditional agent design models, the DSC structure design deploys the novel slot-item structure, which is supported by several practical algorithms in order to improve the reusability and flexibility. Figure 1 shows the design pattern for a DSC-based agent. The DSC items are the elements of agent functionalities. In other words, an agent's capabilities are based on the DSC items that it carries.

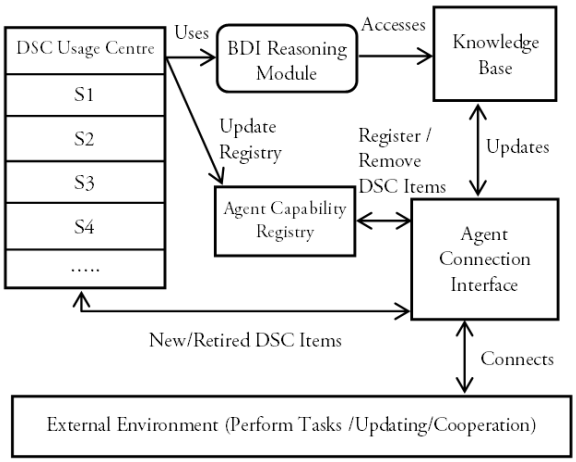

Fig. 1. General agent design pattern of DSC-based agents

A centre is deployed in this design to coordinate various agents. This centre maintains a large DSC warehouse, which enables agents to upgrade their capabilities through receiving the latest DSC items from the DSC warehouse. This design incorporates agent-agent communication (through

the agent-agent connection interface in Figure 1) and centre-agent communication (through the centre-agent connection interface in Figure 1) to form a hybrid structure, which combines decentralised and centralised framework. The performance of the hybrid structure for agent cooperation has been proven as a superior solution [5].

The DSC-based agent design adopts Beliefs-Desire- Intentions (BDI) model for agent reasoning as agent capabilities can be formalised in a framework [6].

A DSC-based agent can efficiently update its capabilities through updating its DSC items. The DSC-based agents are also efficient for agent matching since DSC items provide explicit descriptions about agent capabilities. The retired or dated DSC items are returned to the DSC warehouse; however they can be reused anytime through plugging back to the agent's DSC slots. In general, the DSC-based agent design is able to improve the efficiency of agent capability reuse and provide a flexible and upgradeable structure for integration. This design overcomes the adaptation and integration problems that plague existing software reuse systems [7, 8]; it provides a predefined input and output structure to minimise the costs of components standardisation that plagues many systems [9]. In addition, this design cost-effectively recycles dated DSC items rather than eliminating them from systems.

2. Related Work

Several mechanisms have been suggested to improve the reuse of capabilities and tasks in agent-based systems. For instance, the major mechanisms of describing agent capabilities for information reuse include the Language for Advertisement and Request for Knowledge Sharing (LARKS), Agent Capability Description Language (ACDL) [10], and Interface Communication Language (ICL) [11].

ACDL is introduced to maximise the reuse of agent capabilities over new application domains. It is based on the Knowledge Modelling Framework (KMF). The LARKS allows agents to advertise their capabilities for both syntactic and semantic matching processes. LARKS-based agents are able to use application domain knowledge in any advertisement and request [12].

Modelling of component-based systems is still regarded as a largely unresolved problem in many object-oriented systems according to [13]. The emergence of intelligent agents is helpful to solve the problems in object-oriented systems since agents can be specified on a conceptual level instead of an implementation level. Moreover, agents have strong adaptability and self-learning capability, which make the component integration process in agent-based systems much efficient than in object-oriented systems. The design principles for building component-based agents have been described in [3], which provide several preliminary mechanisms to enhance the reusability of agent design. The 'agent specific task' component described in

their study is similar to the DSC concept. However, it does not provide more concrete mechanisms about the design procedures and the performance of the design.

Previous research on developing an open and comprehensive agent structure has addressed some fundamental issues including the reusability issue for agent design [11, 14, 15]. The reusability issue for agent design is also related to the other issues, such as agent matchmaking, service advertisement, learning and adaptivity, etc. These issues help to draw the basic guidelines for designing DSC-based agents.

3. DSC Usage Centre Overview

3.1. DSC Usage Centre – A Slot Container

Each agent has a DSC usage centre, which provides information resources for agents. This component distinguishes the DSC-based agents from existing middle agents or agent facilitators that mainly play as the role of an agent coordinator. The DSC usage centre upgrades the agents' functionalities through acquiring information from the external environment and sending the acquired information in DSC formats back to the centre. The DSC usage centre manages the unused and active DSC items for agents to update their capabilities. Each DSC item is executable and has standardised and predefined inputs and outputs.

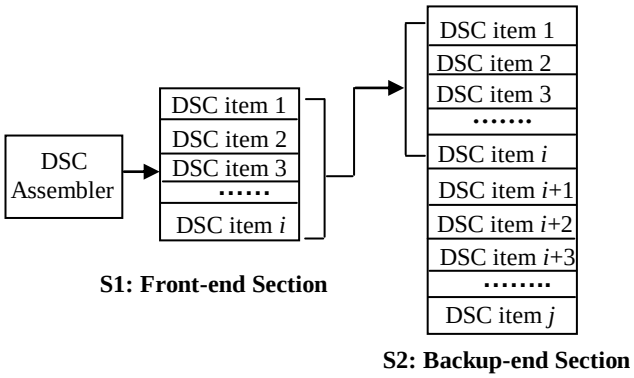

Fig. 2. Slot design of a DSC usage centre

Each DSC usage centre in an agent is a slot container, which offers numerous slots for containing the specialising domain components as shown in Fig 2. Each specialising domain component possesses some special capabilities. Each domain component can plug into a DSC slot to perform specific tasks as an item, for example a domain-component can connect the

agent knowledge base to a stock market database to acquire useful knowledge for the agent.

3.2. Inputs/Outputs of DSC Items

The inputs and outputs of a DSC item contain information as shown in Fig 3.

Input Section: [Input 1: String (Compulsory); Input 2: Integer; Input 3: Float; Input 4: Float (Compulsory).]
Output Section: [Output 1: String; Output 2: Float.]

Fig. 3. Inputs and outputs in a DSC item

In the input section, the information includes the input content, data type, and constraints. The input content indicates the value of the inputs such as a string or a number. The data type indicates the input content's data type, which include string, integer, float, double, etc. The constraint indicates whether the input content can be null. If the constraint value is 'compulsory' then the input content must not be null. Otherwise the input content can be null. The output section consists of the output content and data type; they basically have the same meaning as the input section.

A DSC assembler is employed in the DSC usage centre to evaluate the suitability of a DSC item for an agent through comparing the inputs and outputs from both sides. It also plays the role of extracting top-value of DSC items from the backend section of the DSC usage centre. The comparison between a DSC item and an agent's request is mainly based on the data types and the number of inputs and outputs. For instance, an agent requests a DSC item that can provide two main outputs including the total sales amount and the average salary. Then a selected DSC item must provide these two outputs. The following procedures show the comparison process:

Step 1: If the agent's request does not specify inputs then skip to the next step, otherwise we compare the length of the inputs of the agent's request and the DSC item. We eliminate all the non-compulsory inputs in both DSC items and agent's requests. If the length of compulsory inputs (*LC*) of both sides is equal then we continue the comparison process, otherwise the DSC item is not appropriate for fulfilling the agent's request. The following example illustrates the verification process.

Agent's request inputs: 1 0 0 0 1
DSC item inputs: 1 1 0 0 1
⇩
Agent's compulsory inputs: 1 1
DSC compulsory inputs: 1 1 1
'0' denotes that the input is non-compulsory;
'1' denotes that the input is compulsory.

According to the above example, we have: $LC_a = 2$ and $LC_d = 3$, where LC_a denotes the length of the compulsory inputs of the agent's request; LC_d denotes length of the compulsory inputs of the DSC items. Here, $LC_d > LC_a$. Hence, the selected DSC item is not appropriate for the agent's request.

If $LC_d = LC_a$, then we compare the data types of the compulsory inputs from both sides. If $DIA_i = DID_i$ then go to the next step, otherwise the selected DSC item is not appropriate for the agent's request. DIA_i denotes the input data types of the agent's request; DID_i denotes the input data types of the DSC items; i is from 1 to LC_d, with the incremental change = 1.

Step 2: If the agent's request does not specify outputs then the verification process is accomplished. Whether the DSC item is appropriate for the agent's request is based on the semantic matching of the capabilities descriptions between the two sides, and the comparison process performed in the previous step. If the agent's request specified the output, then we compare the length of the outputs of both sides. The outputs do not distinguish the compulsory and non-compulsory values; therefore, it is not necessary to perform the elimination procedure of non-compulsory values.

If $LP_a \neq LP_d$, then the DSC item is not appropriate for the agent's request. If $LP_a = LP_d$, then we compare the data types of the outputs from both sides. If $DOA_i = DOD_i$ then the DSC item can be selected as an item for the agent's request, otherwise the selected DSC item is not appropriate for the agent's request. LP_a denotes the length of the outputs of the agent's request; LP_d denote the length of the outputs of the DSC item; DOA_i denotes the output data types of the agent's request; DOD_i denotes the output data types of the DSC items; i starts from 1 to LC_d, with the incremental change = 1.

3.3. Functionality Redundancy Calculation

The DSC normally communicates with the environment through the agent-to-agent (or central component-to-agent) interfaces; it also can establish communication with the environment directly. A problem arises when some plugged components are rarely used or never used. To solve this problem, a component usage evaluation mechanism is used to examine the usage efficiency of a plugged component. There are two major factors affecting a DSC's usage efficiency, which include the usage frequency factor and functionality similarity factor. The usage frequency indicates the total number of visits to/from DSC items. The functionality similarity indicates the possible redundancy of a plugged item's functionality with the other plugged items' functionalities. In this section, we introduce the mechanism for calculating functionality redundancy for DSC items. The usage frequency calculation process will be introduced in the next section.

We deploy a three-type semantic relationship model, which is used in WordNet [16], to describe the similarity relationships between two words. The three types of relationships [17] are:

(*i*) Synonym: two words are synonymous.

(*ii*) IS-a: two words are in a superset and subset relationship.

(*iii*) Has-a: One word has ownership of another word. Also known as part-whole relationship between words.

If two words are synonym relationship then their similarity value is A_1; if two words are Is-a relationship then their similarity value is A_2; if two words are Has-a relationship then their similarity value is A_3 ($A_1 > A_2 > A_3$). Therefore, we have the following equation for similarity calculation:

$$F = \sum_{i=1}^{n} A_i \tag{1}$$

where *F* denotes the DSC functionality redundancy value of a slot; A_i the similarity value; *n* denotes the total slot number of a DSC; *F* can be further expanded through decomposing the task description of a plugged item into several key terms and comparing these key terms with other plugged items' key terms, see Fig 4.

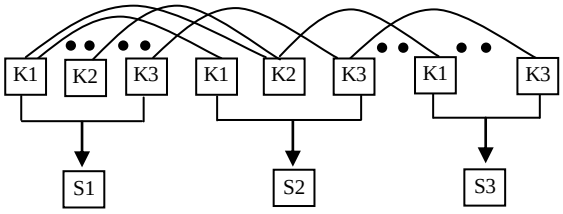

Fig. 4. Functionality similarity calculation based on Cartesian-Product method

In Fig 4, the functionality-similarity value of S_1 and S_2, namely A_i, is based on the *Cartesian-Product* [18] method. In this method, A_i is calculated by exhausting all combinations of choosing one key term from S_1 and one key term from S_2. A_j represents the functionality-similarity value of S_1 and S_3. Therefore, the overall functionality-similarity value of S_1 is the composition of A_i and A_j.

$$F = \sum_{\substack{m,n \in C1 \\ m \neq n}} \sum_{i,j \in C2} S(L_m(K_i), L_n(K_j)), \tag{2}$$

where *m* and *n* denote the DSC slot number (e.g. S_1, S_2, S_3 in Fig 4), *m* must not equal to n because the key terms are only calculated with similarity to other slots but not the same slot; *i* and *j* denote the key term number (e.g. K_1, K_2, K_3 in Fig 4); *S(x, y)* is a function to calculate functionality similarity between *x* and y; $L_x(K_y)$ denotes key term K_y in slot L_x; C_1 denotes the total slot number of the DSC; C_2 denotes the set of decomposed key terms.

If a plugged DSC item has high *F* value compared with other items, and there is no slot for a new item then the high *F* item will be unplugged from the DSC and sent to the central component (the DSC learning centre) for reuse because this item is highly redundant in the DSC compare with other items. A new DSC item will be selected and plugged to the vacant DSC slot. The new DSC item will also be evaluated its similarity to other slots. If the new average similarity value is smaller than the previous average similarity value (before the previous DSC item is removed), then the new item will be plugged in. Otherwise, another DSC item will be selected until it can decrease the average similarity value. This design can enhance the reusability and efficiency of an agent-based system, and it also reduces the redundancy in the DSC.

4. Usage Frequency Calculation for Agent Capability Reuse in DSC Usage Centre

4.1. Front/Back End Structure

The DSC usage centre consists two sections, which include the front-end section and the back-end section. The front-end section extracts the DSC items from the backend section, which are the most frequently used DSC items. This two-section-based structure adopts the methodology of the CPU cache design in operating systems, which stores copies of the data from the most frequently used main memory locations. The reason for dividing the DSC usage centre into two sections is that: the number of the DSC items in the DSC usage centre could be large; however, there are only a number of DSC items are used frequently within a certain period. Therefore, it can improve the system efficiency through deploying a section with a relatively smaller size, which contains the most frequent used items within a certain period.

Fig 5 illustrates the structure the DSC usage centre. S_1 denotes the front-end section; S_2 denotes the item backend section; all the DSC items in S_1 can be found in S_2, which can be expressed as: $S_1 \subseteq S_2$. The DSC items stored in S_1 are obtained from S_2, which are the most frequently used items in S_2. The DSC assembler is responsible for searching a DSC item and importing it to the requesting agent. The following calculation processes illustrate how to extract the DSC items in S_2 and store to S_1.

Step 1: Calculating the recent visit factors in S_2 within time period $[t_a, t_b]$.

$$V(t_i) = \begin{cases} 1, \ the\ DSC\ item\ is\ visited, \\ \\ 0, \ the\ DSC\ item\ is\ not\ visited. \end{cases}$$

where t_i is the time and $V(t_i)$ denotes whether a DSC item is visited at time t_i; i is a variable, which denotes different visiting time within a time period.

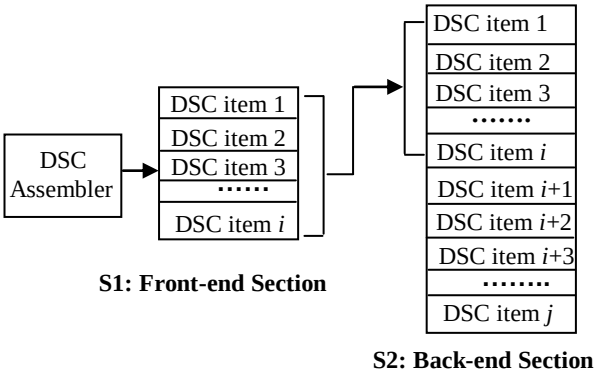

Fig. 5. Two-section-based structure of the DSC usage centre

The total number of visits to a DSC item is calculated as:

$$n = \sum_{i=a}^{b} V(t_i), \text{ and } V(t_i) = 1. \quad (3)$$

where a, b denote the starting point and the ending point of the time period respectively; n is the total number of visits to a DSC item within the time period $[t_a, t_b]$; t_a is the starting point; ti is the time point that the DSC item is visited within the time period. If a DSC item is visited at time ti within $[t_a, t_b]$, then this DSC item is the i^{th} visit to the DSC item and $1 \le i \le n$. For instance: there are a total of 100 visits within $[t_a, t_b]$, i.e. n = 100. The DSC item is visited at time t_{20} ($t_a \le t_{20} \le t_b$) and is the 20th visit to the DSC item, then t_i = t_{20}. All ti and tb are the converted values, which are subtracted by t_a and convert into seconds or a user-defined time scale.

Therefore, a DSC item's recent visit factor can be calculated as the following:

$$\begin{aligned} E_i &= \sum_{i=1}^{n} \left(V(t_i) + \frac{t_b - t_i}{t_i} \times V(t_i) \right)^{\log_n (\frac{t_b}{t_i} \times n)} \\ &= \sum_{i=1}^{n} \left(\frac{t_b}{t_i} \times V(t_i) \right)^{\log_n (\frac{t_b}{t_i} \times n)} \\ &= \sum_{i=1}^{n} \left(\frac{t_b}{t_i} \times V(t_i) \right)^{(\log_n t_b - \log_n t_i + 1)} \end{aligned} \quad (4)$$

where E_i denotes the recent-visit-factor value of a DSC item at time t_i, which indicates a DSC item's usage within the period [t_a, t_b]; n is the total number of visits to the DSC and $1 \leq n$, (Eq. 3, 4 only calculate the recent-visit- factor value when there is visit to the DSC item, if n = 0 then E_i = 0.); the exponent in Eq. 4, i.e. $(\log_n t_b - \log_n t_i + 1)$, is to enlarge the recent-visit- factor value. If t_i is more recent then its E_i value is greater, meanwhile log function is used to limit the exponent value. Eq 4 indicates that: when t_i increases (more recent) then E_i increases. In other words, if a visit to the DSC item is more recent then its E_i is greater. The DSC-based system developers or users can define the time period (i.e. [t_a, t_b]), which is configurable, to initialise and update the DSC items in S_1.

The DSC items in S_2 are ranked according to E_i. The DSC items with higher E_i scores are listed on the top of S_2; S_1 extracts a number of the DSC items that are listed on the top of S_2. The next step calculates the number of the DSC items that S_1 extracts from S_2 (i.e. the size of S_1).

Step 2: Calculating the size of S_1.

The size of S_1 is based on the usage frequency of the target system's DSC item; it should also take the miss-rate into consideration. We use a dynamic alteration method to determine S_1's size. An empty S_1 extracts the maximum number of the DSC items with top E_i values from S_2 within S_1's predefined size limit. After a period ([t_0, t_s]) of running (initial running period), some DSC items in S_1 will be replaced by the items in S_2 and some will be removed from S_1 because of low-usage-efficiency (refer to B section). We first calculate the total number of DSC items in S_2 as its size, i.e. A_2. The initial step of this process is set a target miss-rate value and according to the miss-rate value and A_2 to calculate the preliminary size of A_1.

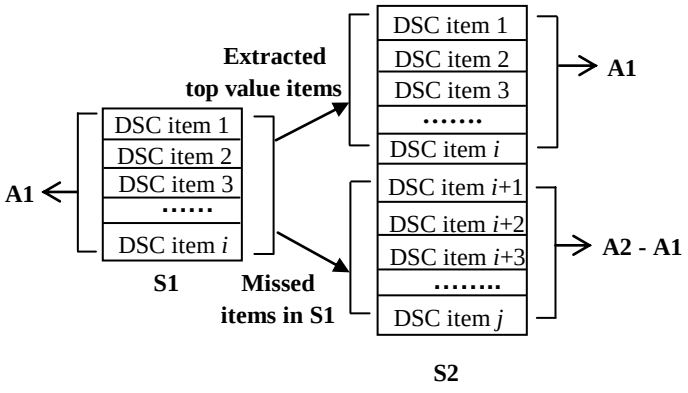

Fig. 6. Dynamic alteration process

Once E_i, A_1, and A_2 are calculated, the DSC usage centre starts to dynamically alter S_1's size according to the miss-rate of searching DSC items in S_1 within a user defined period, i.e. the alteration period. This is because the E_i value of each DSC item is normally changing during the alteration

period and the change affects the size of S_1. Fig 6 illustrates the dynamic alteration process.

In the dynamic alteration process, A_1 is affected by E_i values and the miss-rate of searching DSC items in S_1. The following equitation describes the calculation process.

$$\frac{\sum_{x=1}^{A1} E_x}{\sum_{y=1}^{A2-A1} E_y} \approx \frac{A1}{(A2-A1)} \times \frac{1}{ms}$$

$$\Rightarrow \quad A1 \approx \frac{\sum_{x=1}^{A1} E_x \times (A2-A1) \times ms}{\sum_{y=1}^{A2-A1} E_y}$$

$$\Rightarrow A1 \approx \frac{ms \times \sum_{x=1}^{A1} E_x \times A2}{\sum_{y=1}^{A2-A1} E_y + \sum_{x=1}^{A1} E_x \times ms} \quad . \qquad (5)$$

where ms denotes the miss-rate of searching DSC items in S_1 within the alteration period (this period can be various according to different system applications), and it is always ≤ 1; E_x and E_y denote the E_i values of the DSC items in A_1 and (A_2 - A_1) sections, respectively.

This equitation indicates that: A_1 should be approximately equal to (A_2 - A_1) multiplied by the miss-rate, and the ratio of the total Ei values of all the DSC items in A_1 section to the total E_i values of all the DSC items in (A_2 - A_1) section. In this equitation, A_1 is alterable to satisfy the target *ms* value.

If initial A_1 does not satisfy this equitation, then the DSC usage centre needs to alter A_1 size until it approximately equals the right side of Eq.5. The system users can define the alteration period based on different application requirements. For instance, the alteration period could be longer in some applications because their DSC items require more processing time. Moreover, the process is dynamic, which allows the DSC usage centre to alter the S_1 size regularly according to the computation results based on Eq.5.

The above two steps explain the operating process of the DSC usage centre. The system users can configure the DSC usage centre through adjusting the processing time and the actual database size of the DSC usage centre.

4.2. Reuse / Dismissal of DSC Items

The DSC items in the DSC usage centre that have low usage efficiency will be sent to the DSC warehouse for reuse. A DSC item is regarded as low usage efficiency when this DSC item is constantly on the bottom of S_2 within a period of time [t_s, t_{re}], which is called the critical period for removal and it can be configured by system users.

The usage efficiency is calculated through combining the probability of the visits to a specific item with the recent-visit-factor value E_i. We have assumptions as follows:

There are a total of x DSC items in an agent.

All the DSC items in the agent have been visited the total of m times within period [t_a, t_b].

There is no concurrent visit to the DSC items. In other words, only one DSC item can be visited at the same time.

Value m is based on Eq. 3. Therefore, $m = \sum_{i=a}^{b} V(t_i)$, and $V(t_i) = 1$. Based on Eq. 4, the average value of all the DSC items' E_i values, i.e. $AVG(E)$, within [t_a, t_b] is:

$$AVG(E) = \frac{\sum_{i=1}^{m}\left(\left[\frac{t_b}{t_i}\times V(t_i)\right]^{(\log_n t_b - \log_n t_i + 1)}\right)}{m} \tag{6}$$

The standard deviation [19] of all the DSC items' E_i value is:

$$SDV(E_i) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\left(\frac{t_b}{t_i}\times V(t_i)\right)^{(\log_n t_b - \log_n t_i + 1)} - AVG(E)\right)^2}$$

$$= \sqrt{\frac{1}{m}\left(\sum_{i=1}^{m}\left(\frac{t_b}{t_i}\times V(t_i)\right)^{2(\log_n t_b - \log_n t_i + 1)}\right) - \left(2AVG(E)\sum_{i=1}^{m}\left(\frac{t_b}{t_i}\times V(t_i)\right)^{(\log_n t_b - \log_n t_i + 1)}\right) + m(AVG(E))^2}$$

$$= \sqrt{\frac{1}{m}\left(\left(\sum_{i=1}^{m}\left[\frac{t_b}{t_i}\times V(t_i)\right]^{2(\log_n t_b - \log_n t_i + 1)}\right) - m\left(AVG(E)\right)^2\right)}$$

$$= \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{t_b}{t_i}\times V(t_i)\right)^{2(\log_n t_b - \log_n t_i + 1)} - (AVG(E))^2} \quad (7)$$

We consider removing the DSC item with an E_i value far below *AVG(E)*. Therefore, the lowest E_i value is:

$$E_k = MIN(E_1, E_2, ...E_m),$$

where dataset (E_1, E_2,…E_m) denotes all the DSC items' E_i values within the period [t_a, t_b]; E_k denotes the lowest E_i value in the dataset. If we have:

$$\frac{|E_k - AVG(E)|}{SDV(E)} \geq 3 \quad (8)$$

We define this DSC item as low usage efficiency. This definition is based on *Chebyshev's* theorem [20]:

"If μ and $\sigma$are, respectively, the mean and the standard deviation of the distribution of the random variable *x*, then for any positive constant *k* the probability that *x* will take on a value which is at most $\mu - k\sigma$ or at least is less than $\mu + k\sigma$ or equal to $1/k^2$."

This theorem is also expressed as: $P(|x-\mu| \geq k\sigma) \leq 1/k^2$. According to *Chebyshev's* theorem, at least 89% of the E_i values disperse within 3 standard deviations from the *AVG(E)* value. Therefore, any value, which is greater than this range, is considered as an outlier that has low-usage-efficiency.

If a DSC item in an agent's DSC usage centre is identified as a low-usage-efficiency item, then it will be sent to the DSC warehouse. These low-usage-efficiency DSC items can be discovered and re-deployed by other agents, or can be updated by the DSC warehouse through information updating.

The DSC-based agents also use this evaluation methodology to identify whether a DSC item is low-usage-efficiency. If a DSC item is identified as low-usage- efficiency in an agent then it will be unplugged from the DSC container and sent to the DSC usage centre. This reuse and dismissal method is also used to remove the low-efficient items in the front-end section of the DSC usage centre; it can reduce the section size and improve the system efficiency.

5. Experimental Results

Many existing agent-based systems have the difficulties in designing efficient processes for agent capability matching and reuse. The DSC-based design aims to provide efficient mechanisms for agent capability matching and reuse. In order to evaluate the performance of the DSC-based mechanisms, we conducted a set of experiments based on a real case scenario for solving

the automated feature recognition problem in aerospace component design process, as shown in Fig 7 [21].

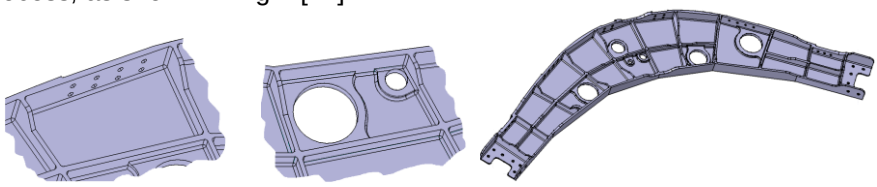

Fig. 7 Typical CAD models of stiffener-panel design.

These experiments demonstrate how the agent-based technology, in particular the DSC mechanism, can be used in the real world engineering design processes.

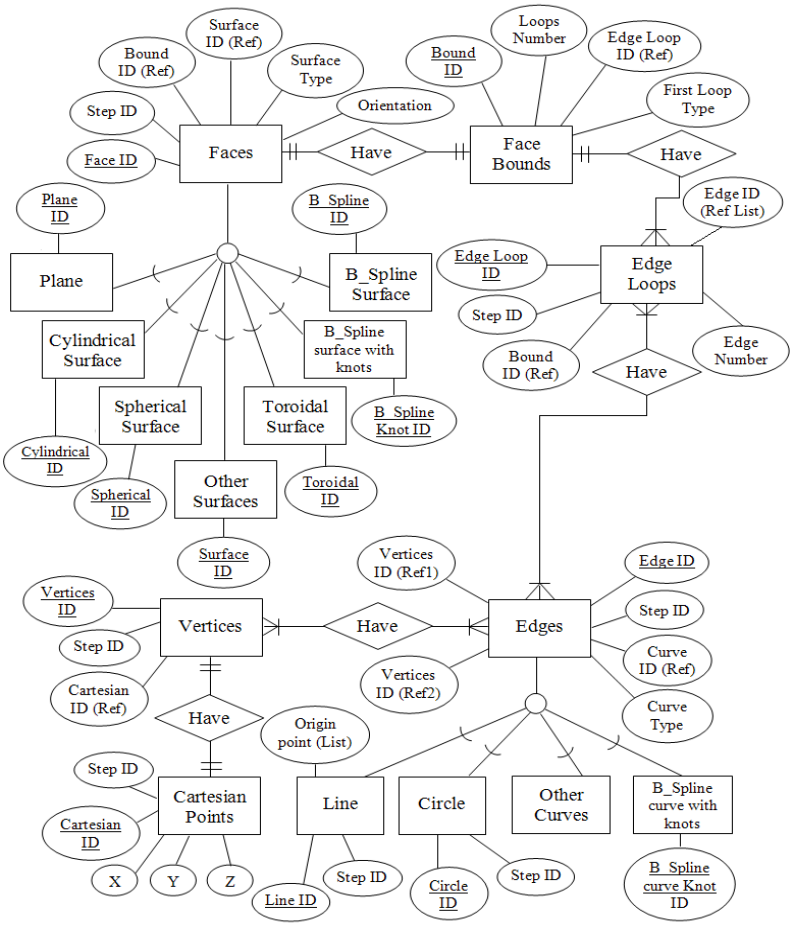

Fig. 8. Entity relationship diagram based on common CAD models [21].

In this case, agents extract key components from CAD models such as the models listed below. There are a number of domain specific agents with different capabilities including the Face agent, Panel agent, Stiffener agent, Hole agent, etc. These agents are responsible for extracting their specified components.

The experimental program matches the user requests against the information from the CAD-based database. The functionalities of the automated feature recognition system are transformed into DSC-based items, i.e. agent capabilities. The experimental program consists two parts: the first part evaluates the performance of the functionality redundancy calculation mechanism; the second part evaluates the performance of the two-section-based structure.

The real case scenario is based on several CAD models for aerospace component design. Domain agents using the DSC-based design can improve the efficiency of finding appropriate agent capabilities for performing the feature recognition process. An entity-relationship diagram based on common CAD models is shown in Fig 8.

An example table describes the three-type relationship is deployed in the functionality redundancy calculation process. The example relationships are shown in Table 1.

Table 1. Similarity relationships used in the FRC experiment

Comparison words	**Similarity relationships**
Plane Face, Face	IS-a
Cylindrical Face, Face	IS-a
Face, Face Bound	Has-a
Advance Face, Face	Synonym
Face Bound, Edge Loop	Has-a
Edge Loop, Edge	Has-a
Line, Edge	IS-a
Circle, Edge	IS-a
B Spline curve, Edge	IS-a
……	……
Panel, Panel Face	Has-a

In the first 16 sets of the *Functionality Redundancy Calculation* (FRC) experiments, we evaluated the impact of request number on the success rate. Among the 16 sets, 8 sets based on the FRC mechanism are called the FRC sets; another 8 sets without using the FRC mechanism are called the NFRC sets. In each experiment, a request is generated randomly based on a knowledge base, which holds around 103 capability descriptions. The DSC usage centre is also generated based on the knowledge base. The DSC slot number is alterable; system users can increase and decrease the DSC slot

number according to specific system requirements. In the first part of the FRC experiment, the default DSC slot number in an agent is 3 and there are 5 agents to deal with a request corporately. As shown in Table 1, the relationship table contains 39 three-type relationship descriptions.

The FRC-based sets produce the similarity score of each DSC item compared with other agents' DSC items. If the DSC item has the highest similarity score and is 1.5 times greater than average score of all the participated agents, then this DSC item will be replaced by a new DSC item from the DSC warehouse. The boundary value of trigger a DSC replacement is 1.5 times, which is based on experimental experience and the relevance of the requests.

The NFRC-based sets only match the requests with the DSC item descriptions. The DSC items in NFRC sets will not be replaced or changed in the experimental process. The following figures show the experimental results.

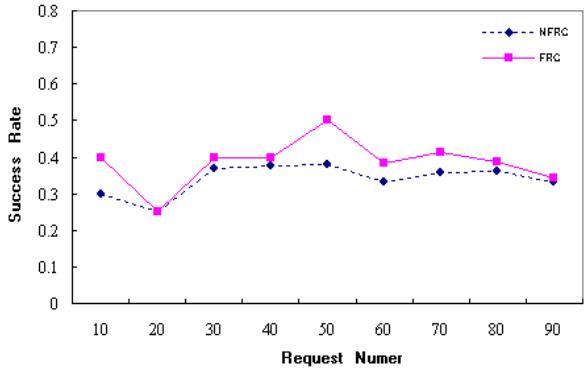

Fig. 9. Success rate comparison based on FRC and NFRC (15 DSC items).

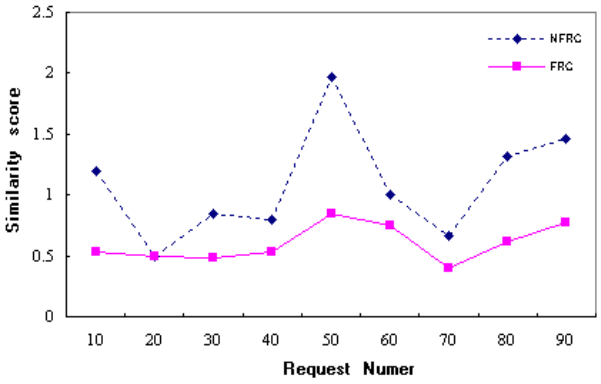

Fig. 10. Similarity score comparison based on FRC and NFRC (15 DSC items).

Fig 9 shows the impact of the FRC method on the success rate of matching requests in comparison with the experimental sets without using

FRC method (i.e. NFRC sets). Fig 10 presents the relationship between the average similarity score and requests number based on FRC and NFRC. The requests matching process in FRC is more efficient than NFRC since the all FRC-based success rates based on different request number are greater than the NFRC-based success rates. The improvement on success rates for a single agent is not enormously large based on FRC. However, this improvement ought to be enormous based on a large number of agents. We also noticed that the success rate is improved significantly when total requests number is 50. The reason for the increase at 50 point is still under investigation. We presume the reason could be: 50 requests for 15 DSC items is a balanced ratio for matching in a DSC usage centre.

Figure 10 shows the average similarity scores based on the NFRC sets is clearly greater than the FRC sets. The average score for NFRC is 1.8 times higher than FRC, which reflects the functionality redundancy in the NFRC sets is much higher than the FRC sets. The functionality redundancy in the DSC usage centre could cause the low efficiency in terms of capability matching and memory consumption for a multi- agent system.

To further evaluate the performance of FRC, we altered the DSC item number (i.e. the DSC slot number) in a DSC usage centre to observe its impact on the success rates of request matching based on 200 requests and remain the other parameters unchanged that are: 39 relationship descriptions and 103 capability descriptions. The results are shown in Fig 11 and 12.

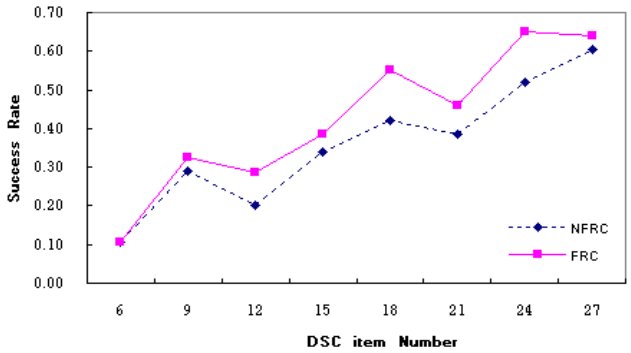

Fig. 11. Success rate comparison based on NFRC and FRC (200 requests)

Fig 11 indicates that the success rates are increasing in both FRC and NFRC sets. Overall, the success rates in the FRC sets are still higher than the NFRC sets. The disadvantage of the FRC method is that: the time consumption of the FRC sets is higher than the NFRC sets as shown in Figure 12. The time consumption in the FRC sets is caused by the processes of calculating the similarity scores, selecting an appropriate DSC item from the DSC warehouse, and replacing the low- efficiency DSC item.

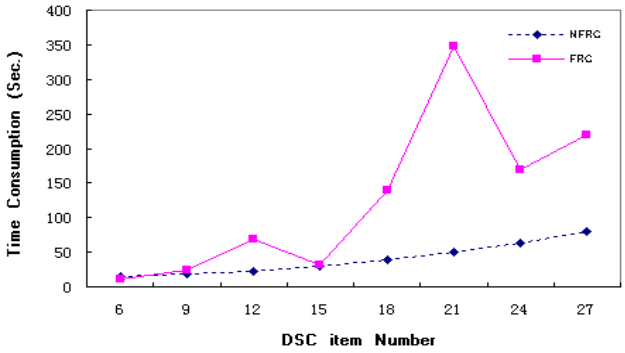

Fig. 12. Time consumption based on NFRC and FRC (200 requests)

In this experimental design, the requests are generated randomly; and this makes the miss-match happens more often than real applications. In other words, the frequency of requesting for a DSC item in a system has a routine pattern in practical. For instance, a 'connecting to the Internet' DSC item might be used most frequently in many organisations, but the random request generation process in this experimental design does not take this factor into account. Therefore, the miss- match will be highly decreased in real cases.

In the second part of the experiments, we evaluate the performance based on the two-section (TS) and Non-two- section (NTS) structures. The database and parameters used in this part are identical to the FRC part.

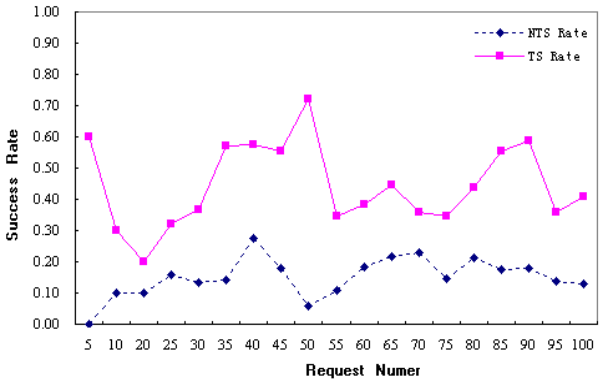

Fig. 13. Success rate comparison based on TS and NTS sets (15 DSC items)

Figure 13 shows that the success rate of matching requests with DSC items in the TS sets are much better than it is in the NTS sets. The average success rate of the TS sets is 2.75 times higher than NTS sets. This result provides a solid proof that TS based mechanism has superior performance for improving matching success rates.

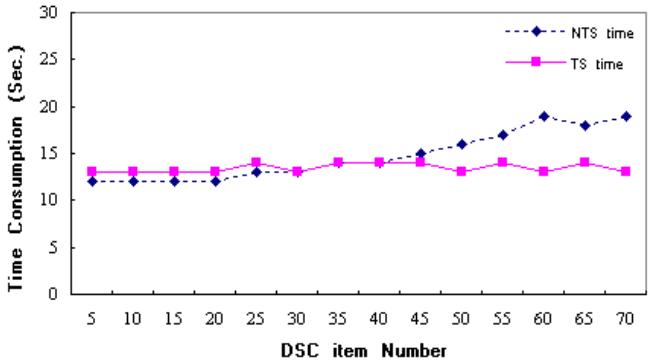

Fig. 14. Time consumption based on TS and NTS sets (100 requests).

To evaluate the time consumption factor of the TS method, the experimental program randomly generated 100 requests and modified the DSC item number to observe the results. Figure 14 indicates that the time consumption of matching requests with DSC items in the TS sets is steady. However, the time consumption increases gradually in the NTS sets when the DSC item number increases.

To further evaluate the results in Figure 14, we increased the requests number to 200. The results shown in Figure 15 are very similar to Figure 14, except that it takes more DSC items in this experiment to increase the time consumption in the NTS sets. Nevertheless, the time consumption of the NTS sets increased from 22 seconds to 34 seconds when DSC items number increased from 5 to 90. However, the TS sets almost remained unchanged.

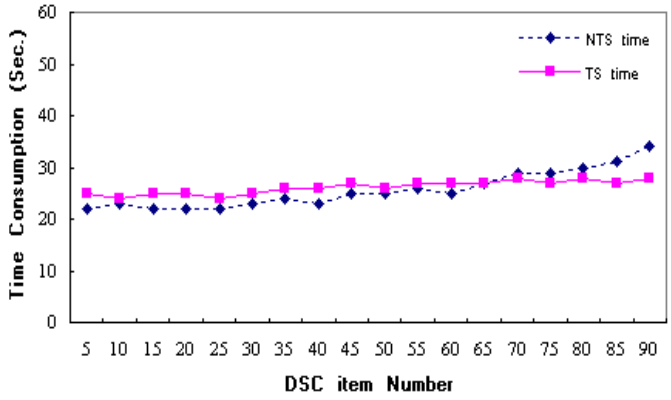

Fig. 15. Time consumption based on TS and NTS sets (200 requests).

6. Conclusion

The DSC-based agent design offers an efficient and cost-effective solution to enhance agent capability reuse and integration. It can be further implemented to various domain agents, such as Web wrapper agents [22], negotiation agents [23], decision support agents [24, 25], etc.

The DSC design mechanism adopts several traditional methods including the cache model, three-type relationship, Cartesian product, and Chebyshev's theorem. These traditional methods are endowed a new meaning when they are applied to the DSC design for agent-based systems. In particular, the novel DSC slot design structure can significantly improve the reusability of agent capabilities.

A DSC-based agent can update its capability through removing the dated DSC items and inserting new DSC items from the DSC warehouse [26]. Instead of destroying retired or dated components in many traditional systems, the DSC-based agent returns the retired or dated DSC items to the DSC warehouse. These returned DSC items can be reused when they can fulfil the users' requests.

The functionality redundancy calculation and two-section- based structure deployed in the DSC-based design are able to improve the success rate of matching DSC items with user requests. The experimental results show that both FRC and TS methods can improve the success rate of request matching. Particularly, the success rate is improved significantly in the TS-based experiments. The similarity scores in the FRC sets are much lower than they are in the NFRC sets. This reflects that the DSC item redundancy is reduced in the FRC-based experiments. The TS sets require less time consumption than NTS sets for matching DSC items, on the contrary, the FRC sets require more time than NFRC sets for matching.

In general, the experimental results based on the CAD models indicate that the TS and FRC methods used in the DSC-based systems improve the system performance in terms of matching requests and reusing DSC components. Hence, the DSC-based agent design offers a reusable and extensible solution, which demonstrates the superior system performance.

The current agent design pattern is based on the system prototype and agent simulation processes. To further improve the DSC-based design, we plan to develop a complete DSC-based multiagent system to solve some complex problems based on industrial cases, such as the automated feature recognition problem described in Section V. Thus, the DSC-based agent design can be further evaluated and improved systematically.

References

1. Lesser, V., Decker, K., Wagner, T., Carver, N., Garvey, A., Horling, B., Neiman, D., Podorozhny, R., Nagendra Prasad, M., Raja, A., Vincent, R., Xuan, P.,

Zhang, X.Q.: Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. Journal of AAMAS, Vol. 9, 87-143. (2004)
2. Hutzschenreuter, A. K., Bosman, P., Blonk-Altena, I., Aarle, J. V., Poutre, H. L.: Agent-based Patient Admission Scheduling in Hospital. Proc. of AAMAS. 45 – 52. (2008)
3. Wang, M., Wang, H., Xu, D., Wan, K.K., Vogel, D.: A Web-service Agent-based Decision Support System for Securities Exception Management. Expert Systems with Applications. Vol. 27, Elsevier Publication, 439 – 450. (2004)
4. Brazier, F.M.T., Jonker, C.M., Treur, J.: Principles of Component- Based Design of Intelligent Agents. Data & Knowledge Engineering, Vol. 41, Issue 1, Elsevier Press, 1-27, (2002)
5. Zhang, H.L., Leung, C. H. C., Raikundalia, G. K.: Topological analysis of AOCD-based agent networks and experimental results. Journal of Computer and System Sciences, Vol. 74, Elsevier Press, 255–278 (2008)
6. Braubach, L. and Pokahr, A.: Representing Long-Term and Interest BDI Goals. Proc. of ProMAS, AAMAS Foundation Press. (2009)
7. Mili, H., Mili, A., Yacoub, S. and Addy, E.: Reuse-Based Software Engineering: Techniques, Organization, and Controls. John Wiley & Sons Press. (2002)
8. Ghijsen, M., Jansweijer, W., Wielinga, B.:Towards a framework for agent coordination and reorganization, AgentCoRe. Proc. of COIN, IEEE Press. (2007)
9. Mohagheghi, P., Conradi, R.: An Empirical Investigation of Software Reuse Benefits in a Large Telecom Product. ACM Transactions on Software Engineering and Methodology, Vol. 17, 3. (2008)
10. Gomez, M., Plaza, E.: Extending matchmaking to maximize capability reuse. Proc. of AAMAS, New York. (2004)
11. Cheyer, A., Martin, D.: The Open Agent Architecture. Journal of AAMAS, Volume 4 Issue 1-2. (2001)
12. Sycara, K., Widoff, S.: LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. Journal of AAMAS, 5, 173-203. (2002)
13. Vitharana, P., Zahedi, F., Jain, H.: Design, retrieval, and assembly. Communications of the ACM, Vol. 46, No. 11, 97 – 102. (2003)
14. Haigh, K., Phelps, J., Geib, C.: An Open Agent Architecture for Assisting Elder Independence, Proc. of AAMAS, pp. 578 – 586. (2002)
15. Jennings, N. R.: An agent-based approach for building complex software systems. Communications of the ACM, Vol. 44, No. 4. (2001)
16. Li, Y., Bandar, Z. A., McLean, D.: An approach for measuring semantic similarity between words using multiple information sources. IEEE Transaction on Knowledge and Data Engineering, Vol.15, No.4, 871-882. (2003)
17. Gangemi, A., Navigli, R., Velardi. P.: The OntoWordNet Project: Extension and Axiomatization of Conceptual Relations in WordNet. Proc. of ODBASE, pp. 820-838. (2003)
18. Hein, J. L.: Discrete Mathematics, Jones and Bartlett Press, 74-75. (2002)
19. Venables, W. N., Ripley, B. D.: Modern Applied Statistics with S-PLUS, Springer-Verlag Press. (1999)
20. Hogg, R.V. and Craig, A.T.: Introduction to Mathematical Statistics (5th Edition), Prentice Hall Press. (2007)
21. Zhang, H. L., Van der Velden, C., Yu, X., Jones, T., Fieldhouse, I., Bil, C.: Developing A Rule Engine for Automated Feature Recognition from CAD Models. Proc. of IEEE IECON, 3925 – 3930. (2009)
22. Chang, C., Siek, H., Lu, J., Hsu, C., Chiou, J.: Reconfigurable Web Wrapper Agents. IEEE Intelligent Systems, 34 – 40. (2003)

23. Fischer, K., Chaib-draa, B., Muller, J. P., Pischel, M., Gerber, C.: A Simulation Approach Based on Negotiation and Cooperation between Agents - A Case Study. IEEE Transaction on Systems, Man and Cybernetics - Part C, Vol. 29, Issue 4, 531-545. (1999)
24. Vahidov, R., Fazlollahi, B.: Pluralistic multi-agent decision support system: a framework and an empirical set. Information & Management, Vol. 41, Elsevier Press, 883 – 898. (2004)
25. Turban, E., Aronson, J. E., Liang, T.: Decision Support Systems and Intelligent Systems (7th edition), Prentice-Hall Press, 223,. (2005)
26. Zhang, H. L., Leung, C.H.C., Yu, X., He, J.: An Optimised Design for Agent Capability Reuse. Proc. of IEEE/WIC/ACM WI-IAT, Vol.3, 231-234. (2010)

Dr. Hao Lan Zhang is an Associate Professor and Deputy Director of IS discipline at NIT, Zhejiang University. Dr. Zhang received a PhD from Victoria University, Australia. He was with RMIT University as a Research Fellow from 2008–2010. Prior to that, he worked as a research assistant/tutor at Victoria University. His research interests include: multi-agent systems, knowledge-based systems, intelligent information systems, e-health systems, database, etc. He has published over 30 research papers in various journals, conferences and workshops. Dr. Zhang received an outstanding student award from the Chinese Scholarship Council and Chinese Ministry of Education in 2006.

Dr. Wenhua Zeng is a Professor and Vice Dean of Software School at Xiamen University. He obtained a PhD degree from Zhejiang University in 1989. He has been active in areas of Software Engineering, Grid Computing, Cloud Computing.

Dr. Christian Van der Velden is a Control System Engineer at BAE Systems Australia. He received his PhD in 2009 from RMIT University, Australia in the field of Knowledge Based Engineering. He worked as a Research Fellow at RMIT University from 2008 to 2010 working with GKN Aerospace and the Cooperative Research Centre for Advanced Automotive Technology (AutoCRC) in the area of Automated Feature Recognition.

Received: March 4, 2011; Accepted: May 4, 2011.