Goal-Capability-Commitment Based Context-Aware Collaborative Adaptive Diagnosis and Compensation

Wei Liu[1,2(✉)], Shuang Li[2], and Jing Wang[2]

[1] School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan, China
liuwei@wit.edu.cn
[2] Hubei Province Key Laboratory of Intelligent Robot, Wuhan, China
146108697@qq.com, 8783763390@qq.com

Abstract. Agent commitments as a protocol among the participants to melding collaborative patterns typically defined at design-time may not fit the need of collective adaptive systems (CASs). In this paper we propose a Goal-Capability-Commitment (GCC) based context aware collaborative adaptive diagnosis and compensation approach. First, we give a formal and semantic representation of GCC model and define semantic match relations between goals and capabilities/commitments. Second, diagnosing rules are proposed to determine the failures of capabilities and commitments. Third, a generation algorithm apply the capabilities in commitment generation and optimization to verify whether the agent and its partner can collaborate in order to achieve the desired goal under current environment. At last, two experiments over the simulated scenario of automated hospital cart transportation system provide an empirical evaluation of GCC approach.

Keywords: GCC · Agent capability · Semantical match · Context state AGV

1 Introduction

The indispensable aspects of collective adaptive systems (CASs) are adaptation and collaboration [1]. The autonomous agents as executing units in CASs must be capable of adapting their individual behaviors or coordinating their social behaviors over time to achieve a set of goals under dynamically changing and uncertain environment.

Goal models for self-adaptive system to provide an effective mechanism to represent possible changes and making adaptation decision. Baresi et al. [2] propose FLAGS requirements models which are based on the KAOS framework and are targeted at adaptive systems. Ali et al. [3] proposes an automated analysis to reason about contextual goal models that extends Tropos goal model introducing variation points where the context may influent the choice among the available variants of goals satisfaction. Lamsweerde et al. [4] proposes a framework to capture the effects of domain variability on a system using a single context-parameterized i* model and to automatically produce variations of that model based on the currently active context. Such

P. Cong Vinh et al. (Eds.): ICCASA 2017/ICTCC 2017, LNICST 217, pp. 79–89, 2018.
https://doi.org/10.1007/978-3-319-77818-1_8

approaches usually assume that the goals are decomposed until they reach a level of granularity which can be represented as task. In other words, the above assumption is often not true in open CASs for two major reasons: (1) *Unpredictable agents*. The appropriate adaptation decision depends on not only requirements and contexts but also depends on the current status of the autonomous agent configured in CASs. In that sense, there can be impossible to decide which autonomous agent can be thought of as achieving one or more objectives until the system run. (2) *Unpredictable collaboration*. Owing to the autonomous agent's unrestrained joining in or quitting from CASs, the system works as dynamical network of interactions and collaborations.

There are two lines of work that are closed related to the new problems in agent-oriented CASs. One line of work studies agent capability modeling. The capability concept was introduced into the adaptation methodology for it could represent the dynamic and collaborative characteristics [5, 6]. Most previous works focused on action representation formalism of agent capability. Wickler [7] used expressive and flexible action representations to reason about the capabilities for agent cooperation. Cassandra et al. [8] presented an approach to agent capability description and matching that is expressive enough to capture complicated agent functionality. Sycara et al. [9] defined and implemented a language Larks for agent advertisement and request and a matchmaking process. These approaches assume that the decisions about which agent could achieve the goal by their own capabilities are determined at design time. The second line of work studies agent commitment modeling. The commitment as a protocol to be created among the participants to melding selected collaborative patterns. Chopra et al. [10] develop a semantic relationship between goals and commitments. Akin et al. [11] study in developing dynamic protocols, that can be modified by agents while interacting according to the situation they are in. The previous works assume that a participant agent has a commitment or creates a commitment to another participant to help achieve its goal. However, these approaches generates commitments uses only the locally available knowledge revised by their beliefs. In our case, every agents could able to understands and interact with each other. Hence, we need to propose a flexible diagnosis and compensation approach at runtime that not only decide which agent can be thought to achieve the goal but also allow an agent could make dynamical determination.

The medical waste AGV (Automated Guided Vehicle) transportation in San Antonio Hospital [12] is used as our example scenario. In the system there is three agents: Pickup sensor (denoted as A_1), AGV (denoted as A_2) and Cart elevator (denoted as A_3). The hospital building has two floors and the waste dump is on the ground floor. Every floor has some cart pickups. When a cart loaded medical waste is pushed in the cart pickup, a cart sensor in the ceiling of cart pickups detects the cart reads the RFID tag and delivers the calls for AGV. If an AGV takes the cart to the waste dump in different floor, then a cart elevator with sensors transits the AGV automatically to the waste dump floor.

In this paper we define a Goal-Capability-Commitment (GCC) ontology modeling framework for CASs development. To achieve the targets mentioned above, we develop a GCC model based diagnosis and reconcile approach for CAS. First, we give a formal and semantical representation of GCC model and define semantic matching relations between goals and capabilities/commitments. Second, the diagnose algorithm is proposed to determine the failures of capability and commitment at runtime according to three rules. Third, the compensation algorithm applies the capabilities in commitment generation and optimization to verify whether the agent and its partner can collaborate in order to achieve the desired goal under current environment. We experimented our approach over the simulated scenario of automated hospital cart transportation system in our GCC-SAR prototype tool. The experiments over the simulated scenario of automated hospital cart transportation system provide an empirical evaluation of GCC model and the algorithms.

2 Technology Base

2.1 Capability-Commitment-Goal Representation

2.1.1 Capability

Capability was proposed as an abstraction for specific plans that an agent may execute to achieve the goal for agent early matchmaking approach. The capability representation could not only modify agents' execution for achieving new goals, but also derive possible candidate commitments for agents' collaboration.

Capability describes how a process is to be executed within what current states are and what the states will be. A capability can be represented as follows:

$$Capability = <Agent, In\text{-}constraints, Plan, Out\text{-}constraints>$$

Agent is specified as 'who' has the capability. *In-constraints* denotes the preconditions of the capability could execute. *Plan* describes the sequence of actions for realizing the capability. *Out-constraints* describes the effects after the performance of capability. *In-constraints* and *Out-constraints* consists of the context states which are defined for representing the value of context factors in our previous work [13]. Context state is denoted as $cs = <f, p, v>$. f means a factor, p is a predicate and v represents a value of factor. For example, "detect_cart" (c_1) is a capability of the pickup sensor (A_1). If the cart is in the pickup, then the capability c_1 could be triggered. After c_1 is executed, the pickup position information is obtained. c_1 = <A_1, {cs_1}, p_1, {cs_2}>. cs_1 = <cart, in, cart_pickup> and cs_2 = <pickup_position, is, available> are two context states.

2.1.2 Commitment

Commitment is a promise made by the debtor commits to the creditor to bring about the consequent provided the antecedent holds. The form of commitment has defined as follows:

$$Commitment = <Debtor, Creditor, Antecedent, Consequent> .$$

Commitment means that a debtor is committed to a creditor for the consequence if the antecedent holds [11]. *Debtor* and *creditor* are cooperative partners who have their own capability and they could execute in order as an alliance. In our approach, the contractual relationship between *debtor* and *creditor* is in accordance with the collaboration of their capabilities. Comparing with previous representation of agent commitment, in commitment antecedent and consequent are not propositional variables but the set of context states. For example, a commitment between AGV (A_2) and Cart elevator (A_3) is denoted as co_1 = <c_{11}, c_{10}, {cs_{10}, cs_{16}}, {cs_{17}}>. AGV's capability "move_to_elevator" (c_{10}) represents that if the floor destination is different from the floor of AGV then the AGV will move to the cart elevator. Cart elevator's capability "transport_AGV" (c_{11}) represents that if the elevator call is available and the AGV is in the area of elevator sensor then the cart elevator will transport the AGV to the destination floor. In co_1, cs_{10} = <AGV, hasState, busy), cs_{16} = <AGV_floor, notSame, destination_floor and cs_{17} = <same (AGV_floor, destination_floor) are three context states.

2.1.3 Adaptation Goal

Goals in GoalSPEC are specifies system goals, under the hypothesis the domain is described as a set of states [14]. GoalSPEC is a language suitably created for supporting evolution and self-adaptation. Our approach adds some details about the collaborations of goal satisfaction into the extended system goals. A goal can be represented as follows:

$$Goal = <Actor, TrigConditions, FinalStates>$$

Actor is a set of agents specified as 'who' is/are the main responsible to address the given goal; *TrigConditions* must be held in order to activate the goal; *Final States* must be true in order to declare and the goal is finally satisfied. *TrigConditions* and *FinalStates* consists of the context states. In our approach, the actor of goal will not be determined until the system runs and the actor may be changed under the changing environments. The determined actor may be invalid if the agent could not perform in particular condition. A goal could be achieved by an actor or a list of actors. For example, there are two goals "get cart's position" (g_1) *and* "transport AGV to destination floor" (g_2) in the scenario. The goal g_1 = <{A_1}, {cs_1}, {cs_2}> which is achieved by cart elevator (A_1). The goal g_2 denoted as g_2 = <∅, {cs_{10}, cs_{16}}, {cs_{17}}> at first, which is denoted as g_2 = <{A_2, A_3}, {cs_{10}, cs_{16}}, {cs_{17}}> after achieved by AGV (A_2) and cart elevator (A_3).

2.2 Semantic Matching

Semantic matching provides a machine-readable way to decide whether the goal is achieved by agent's capabilities or commitments between agents. Semantic imply relationship between sets of context states is analyzed according to contains relationship between context states. In our approach, context states with SWRL atoms formats are translated into PROLOG clause. A clause cs_i θ-subsumes another clause cs_j, written $cs_i \preccurlyeq_\theta cs_j$, if there is a substitution θ such that $cs_i\theta \subseteq cs_j$ holds [15].

Definition 1 (Semantic imply). Given two sets of context states S^i_{cs} and S^j_{cs}, if $\forall cs_i \in S^i_{cs}, \exists cs_j \in S^j_{cs}$ and $cs_j \preccurlyeq_\theta cs_i$, then S^i_{cs} semantic implies S^j_{cs}, which is denoted as $S^i_{cs} \Rightarrow S^j_{cs}$.

S^i_{cs} semantic implies S^j_{cs} means that if all the context states in S^i_{cs} are satisfied then all the context states in S^j_{cs} must be satisfied too.

There are two types of constraint-oriented plug-in matching: capability matching and commitment matching.

Definition 2 (Semantic matched by capability). Given a goal g_i = <*actor*$_i$, *trigConditions*$_i$, *finalStates*$_i$> and a capability c_j = <*agent*$_j$, *in-constraints*$_j$, *plan*$_j$, *out-constraints*$_j$>. If *trigConditions*$_i$ ⇒ *in-constraints*$_j$ and *out-constraints*$_j$ ⇒ *finalStates*$_i$, then the goal g_i is semantic matched by the capability c_j, denoted as $c_j \vDash g_i$. If g_i does not be semantic matched by c_j, denoted as $c_j! \vDash g_i$.

A goal can be semantic matched by a capability, which means that the agent should be capable of achieving the goal by itself. If a goal cannot be semantic matched by anyone capability in the system, which means that a agent need to collaborate with other agent in order to achieve the goal. A commitment between the agent and its partner should be generated under current environment. A then, the goal will be semantic matched by the commitment.

Definition 3 (Semantic matched by commitments). Given a goal g_i = <*actor*$_i$, *trigConditions*$_i$, *finalStates*$_i$> and a commitment co_j = <*debtor*$_j$, *creditor*$_j$, *antecedent*$_j$, *consequent*$_j$> generated by *debtor*$_j$ and *creditor*$_j$. If *antecedent*$_j$ ⇒ *trigConditions*$_i$ and *finalStates*$_i$ ⇒ *consequent*$_j$, then the goal g_i is semantic matched by co_j, denoted as $co_j \vDash g_i$. If g_i does not be semantic matched by co_j, denoted as $co_j! \vDash g_i$.

The *debtor*$_j$ and *creditor*$_j$ as different capabilities can not belong to the same agent. If two capabilities belong to an agent could work together to achieve a goal, not a new commitment but a new capability with greater granularity consisting of the two capabilities will be generated.

3 Our Approach

3.1 Failure Diagnosis

We define three rules to diagnose the capability failure and commitment failure, as is described in Table 1. The states of capability and commitment were defined in the work of relating goal and commitment semantics [16].

Table 1. The rules diagnose failures

Rules	Representation (context state is denoted as *cs*)
Rule I	$\dfrac{c.state = Active \wedge (\exists cs \in c.Inc = false \vee \exists cs \in c.Outc = false)}{c.state = Invalid \wedge g.state = Terminated}$
Rule II	$\dfrac{co.state = Active \wedge \exists cs \in co.Ant = false}{co.state = Violated \wedge g.state = Terminated}$
Rule III	$\dfrac{co.state = Active \wedge (ca.state = Invalid \vee cb.state = Invalid)}{co.state = Terminated}$

Capability failure means that the executing capability cannot achieve the goal under current environment. The capability failure is diagnosed by Rule I. As described in

Rule I, when a capability c is *Active,* if not all the context states in the in-constraint*s* of c ($c.Inc$) or the out-constraints of c ($c.Outc$) are satisfied, then the state of c translates from *Active* to *Invalid* and the state of the matched goal g translates from *Active* to *Terminated.* If the current value of context is different from the value described in GCC model, then the context state is not satisfied. Commitment failure means that the executing commitment has no ability to achieve the goal under current environment. The commitment failure is diagnosed by Rule II and Rule III. As described in Rule II, when a commitment *co* is *Active*, if not all the context states in the *antecedent* of *co* (*co. Ant*) are satisfied, then the state of *co* translates from *Active* to *Violated* and the state of the matched goal g translates from *Active* to *Terminated.* Rule III directed against the commitment *co* created by two capabilities c_a and c_b. c_a is owned by a creditor a and c_b is owned by a debtor b. Rule III happens if c_a or c_b is Invalid and *co* is Active, then the state of *co* translates from *Active* to *Terminated.*

As the result of the diagnosis, the goal terminated invalidation will trigger capability and commitment compensation to search for an alternative capability or commitments and make sure that the new adaptive solution can be executed.

3.2 Capability and Commitment Compensation

Capability compensation is to search whether there are some alternative capabilities for achieving the terminated goals. If there is no capability can semantic match the goals indirectly, commitment compensation will try to generate new commitments to achieve the terminated goals. There are two types of commitments: cooperated commitment and assisted commitment.

Definition 4 (Commitment generation). Given a goal g_i = <$actor_i$, $trigConditions_i$, $finalStates_i$>, two capabilities c_i = <$agent_i$, $in\text{-}constraints_i$, $plan_i$, $out\text{-}constraints_i$> and c_j = <$agent_j$, $in\text{-}constraints_j$, $plan_j$, $out\text{-}constraints_j$>, if:

(1) $trigConditions_i \Rightarrow in\text{-}constraints_i \cup in\text{-}constraints_j$ and $out\text{-}constraints_i \cup out\text{-}constraints_j \Rightarrow finalStates_i$, then a **cooperated commitment** $co_p = <debtor_p, creditor_p, antecedent_p, consequent_p>$ is generated, and $antecedent_p = in\text{-}constraints_i \cup in\text{-}constraints_j$ and $consequent_p = out\text{-}constraints_i \cup out\text{-}constraints_j$;

(2) $trigConditions_i \Rightarrow in\text{-}constraints_i$, $trigConditions \cup out\text{-}constraints_i \Rightarrow in\text{-}constraints_j$ and $out\text{-}constraints_i \cup out\text{-}constraints_j \Rightarrow finalStates_i$, then an **assisted commitment** co_q = <$debtor_q$, $creditor_q$, $antecedent_q$, $consequent_q$> is generated, and $antecedent_q = in\text{-}constraints_i$ and $consequent_p = out\text{-}constraints_i \cup out\text{-}constraints_j$.

For example, there is a goal "transport AGV to different floor" (g_3) which is denoted as $g_3 = <\varnothing, \{cs_{10}, cs_{16}\}, \{cs_{17}\}>$, and two capabilities "move_to_elevator" $c_{10} = <A_2, \{cs_{10}, cs_{16}\}, \{cs_{14}, cs_{15}\}>$ and "transport_AGV" $c_{11} = <A_3, \{cs_{14}, cs_{15}, cs_{16}\}, \{cs_{17}\}>$. cs_{14} = <elevator_call, is, available> and cs_{15} = <AGV, in, elevatorSensor_area> are two context states. According to the definition of commitment generation, a assist commitment is generated $co_1 = <c_{11}, c_{10}, \{cs_{10}, cs_{16}\}, \{cs_{17}\}>$. The goal g_3 is denoted as $g_3 = <\{A_2, A_3\}, \{cs_{10}, cs_{16}\}, \{cs_{17}\}>$.

We present the commitment generation algorithm in Algorithm 1. The algorithm takes two parameters as inputs: the queue of goals to be supported (I_{tg}), the queue of inactive capabilities (I_c).

Algorithm 1 GENERATION(I_{tg}, I_c)

Require: I_{tg}, the queue of goals to be supported
Require: I_c, the queue of inactive capabilities
1: **for each** g in I_{tg}
2: **do for each** c in I_c
3: **do if** $c \models g$ **then**
4: $c.state \leftarrow active$ and $g.state \leftarrow active$
5: g.ADDMATCHEDCOMPONENT(c)
6: I_{ag}+={g}
7: **if** $g.state$=$terminated$ **then**
8: tc=GETTRIGCONDITION(g) **and** fs=GETFINSTATES(g),
9: **for each** c_i in I_c
10: **do** $in_i \leftarrow$GETINCONSTRAINTS(c_i) and $out_i \leftarrow$GETOUTCONSTRAINTS(c_i)
11: **for each** c_j in I_c
12: **do** $in_j \leftarrow$GETINCONSTRAINTS(c_j) and $out_j \leftarrow$GETOUTCONSTRAINTS(c_j)
13: **if** $tc \Rightarrow in_i \cup in_j$ **and** $out_i \cup out_j \Rightarrow fs$ **then**
14: $co \leftarrow$GENERATECOMMITMENT(c_i, c_j)
15: co.SETANTECEDENT($in_i \cup in_j$) and co.SETCONSEQUENT($out_i \cup out_j$)
16: I_{co}'+={co}
17: $co.state \leftarrow active$ and $g.state \leftarrow active$
18: g.ADDMATCHEDCOMPONENT(co)
19: I_{ag}+={g}
20: **else if** $tc \Rightarrow in_i$ **and** $tc \cup out_j \Rightarrow in_j$ **and** $out_i \cup out_j \Rightarrow fs$ **then**
21: $co \leftarrow$GENERATECOMMITMENT(c_i, c_j)
22: co.SETANTECEDENT(in_i) and co.SETCONSEQUENT($out_i \cup out_j$)
23: I_{co}'+={co}
24: $co.state \leftarrow active$ and $g.state \leftarrow active$
25: g.ADDMATCHEDCOMPONENT(co)
26: I_{ag}+={g}
27: **return** I_{ag}

Line 1 diagnoses each terminated goal g in I_{tg}. Line 2–6 search a new capability to semantic match the goal. If there is no capability in I_c could support the goal, then the algorithm will try to generate a new commitment to semantic math the goal according to Definition 4. Line 6–11 get the variables used in the following analysis. If the conditions of the first situation mentioned above are satisfied, then a cooperated commitment co is generated and which is added into I_{co}' (Line 13–19). If the conditions of the second situation mentioned above are satisfied, then an assisted commitment co' is generated and which is added into I_{co}' (Line 20–26). Line 27 returns the queue of goals which are achieved under the current executable environment I_{ag}.

4 Experiments and Discussion

In order to justify the above discuss and test the execution performance of our algorithms we developed a GCC based self-adaptive reconfiguration (GCC-SAR) prototype tool as a plug-in of protégé 3.4.4 (http://protege.stanford.edu/). We experiment our approach over the simulated scenario of automated hospital cart transportation system in the GCC-SAR prototype tool. We performed our experiences on an Intel Core(TM) i5-62002.4 GHz processor with 8 GB memory running Windows 10 professional edition.

In the first experiment we observed the effect of α parameter on the execution performance of Algorithm 1. This parameter control the maximum depth between a domain class and its ancestors in semantic matching that has a direct impact on the rate of the goals could semantical matched by capabilities or commitment. In order to investigate this issue we conducted an experiment in which we took the initial value of α as 0 and then increased it up to 5 and measure the matching rate of goals with the algorithm. We assumed that all the goals are to be supported in the algorithm initially.

Figure 1 shows the results of the experiment in which we observe the effect of α. In the figure the x-axis is the number of goals that need to be semantical matched and the y-axis is the matching rate of all goals. The six plots represent the six conducted experiments. The chart shows that the matching rate is grows with the growth of α until $\alpha = 3$ and changes little with $3 < \alpha < 6$. Note that in experiment we would expect to see that it is not the bigger the value of α is the higher the matching rate is. If the value of α is less than the maximum depth of domain model, then the matching rate is grows with the growth of α. In addition, if the number of capabilities is limited, then the more the number of goals is the slightly lower the matching rate is.

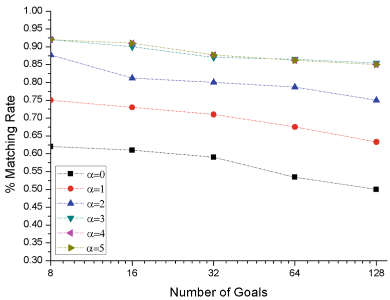

Fig. 1. Comparison of matching rate of the achieved goals based on the maximum depth

In the second experiment we observed the effect of the β parameter on the execution time of the optimization method. This β parameter is the percentage of goals matched by commitments to all matched goals. Since we realize the capability compensate and optimization for achieving each goal, this parameter has a direct impact on the time of the Algorithm 1. We presents the result of the second experiment (we fix the parameter as $\alpha = 3$) in which we observe the effect of β in Fig. 2.

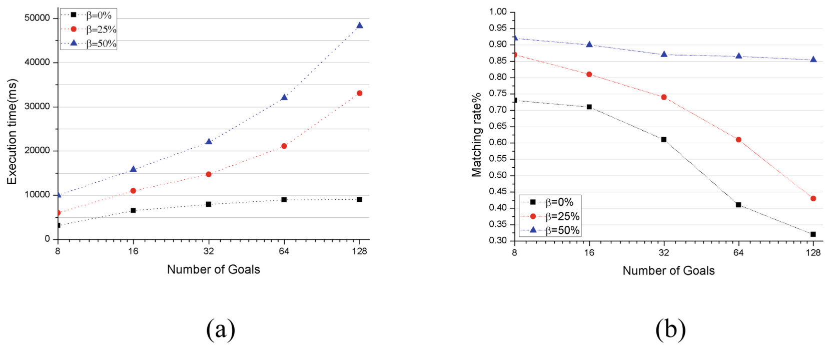

Fig. 2. Comparison the execution time and matching rate based on the rate of achieved goals

Figure 2(a) presents the execution time of Algorithm 1 for different values of β as 0, 25% and 50% by increasing the number of goals (denoted as x) need to be semantical matched.

The value of β is 0 means that all the goals are achieved by the capabilities. With the increasing the number of goals, the very slow growth of the execution time shows that the execution time of semantic match by capabilities does not grow exponentially. When the values of β are set as 25% and 50%, the results illustrate two important phenomenons. (1) The speedy increasing of the execution time illustrates that the semantic match by commitments could spend much more time than by the semantic match by capabilities. (2) When the numbers of goals achieved by commitments are the same in different case, such as ($x = 16$, $\beta = 50\%$) and ($x = 32$, $\beta = 50\%$), ($x = 32$, $\beta = 25\%$) and ($x = 64$, $\beta = 25\%$), ($x = 50\%$, $\beta = 25\%$) and ($x = 128$, $\beta = 25\%$), the similar results show clearly that the number of goals achieved by commitments is the critical factor in the execution times.

Figure 2(b) presents the he matching rate of all goals with the different values of β as 0, 25% and 50% by increasing the number of goals (denoted as x) need to be semantical matched. When the value of β is set as 50%, the very slow decrease of matching rate with the exponential increasing the number of goals shows that the semantic match by commitments could maintain the matching rate in a high level. When the values of β are set as 25% and 0%, the speedy decrease of matching rate illustrates that the less commitments is generated the lower matching rate is.

5 Conclusion

GCC framework based self-adaptive diagnosis and compensation approach could reduces the gap between the requirements model and the executable model for collaborative adaptive system. The main contribution of the paper include: (1) the semantic representation for goal, capability and commitment could provide foundation of heterogeneous agents recognize the executive behavior information of each other at runtime; (2) capabilities are applied in commitment generation and optimization to verify whether the agent and its partner can collaborate in order to achieve the desired goal under current environment.

We are currently integrating this proposal with other multi-agents interaction modeling techniques based on the agent commitments. As such, more characteristics of social behavior such as competition and disposition need to be considered. We are also beginning to study capability-based components collaborative for CAS.

Acknowledgment. Project supported by the National Natural Science Foundation of China under Grant (No. 61502355), the Doctor foundation for Science Study Program of Wuhan institute of technology (No. K201475).

References

1. Ferscha, A.: Collective adaptive systems. In: Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers, pp. 893–895. ACM (2015)
2. Baresi, L., Pasquale, L., Spoletini, P.: Fuzzy goals for requirements-driven adaptation. In: 18th IEEE International Requirements Engineering Conference (RE), pp. 125–134. IEEE (2010)
3. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. Requir. Eng. **15**, 439–458 (2010)
4. Lamsweerde, A.V.: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley, Hoboken (2009)
5. Cassandra, A., Chandrasekara, D., Nodine, M.: Capability-based agent matchmaking. In: International Conference on Autonomous Agents and Multiagent Systems, pp. 201–202 (2000)
6. Katia, S., Klusch, M., Widoff, S., Lu, J.: Dynamic service matchmaking among agents in open information environments. ACM Sigmod Rec. **28**, 147–153 (1999)
7. Wickler, G.: Using expressive and flexible action representations to reason about capabilities for intelligent agent cooperation (2000). http://www.dai.ed.ac.uk/students/gw/phd/story.html
8. Cassandra, A., Damith, C., Marian, N.: Capability-based agent matchmaking. In: The Fourth International Conference on Autonomous Agents. ACM (2000)
9. Sycara, K., Widoff, S., Klusch, M., Lu, J.: LARKS: dynamic matchmaking among heterogeneous software agents in cyberspace. Auton. Agents Multi-Agent Syst. **5**(2), 173–203 (2002)
10. Chopra, A.K., Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Reasoning about agents and protocols via goals and commitments. In: International Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 457–464 (2010)
11. Günay, A., Winikoff, M., Yolum, P.: Dynamically generated commitment protocols in open systems. Auton. Agents Multi-Agent Syst. **29**(2), 192–229 (2015)
12. Hospital Automatic Guided Vehicle Systems (2015). http://www.agvsystems.com/wp-content/uploads/2015/03/AVANT_AGV_AGC_HOSP.pdf
13. Liu, W., Feng, Z.W.: Context-based requirement modeling for self-adaptive service software. J. Comput. Inf. Syst. **8**(24), 10131–10140 (2012)

14. Sabatucci, L., Ribino, P., Lodato, C., Lopes, S., Cossentino, M.: GoalSPEC: a goal specification language supporting adaptivity and evolution. In: Cossentino, M., El Fallah Seghrouchni, A., Winikoff, M. (eds.) EMAS 2013. LNCS (LNAI), vol. 8245, pp. 235–254. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45343-4_13
15. Sycara, K., Widoff, S., Klusch, M., Lu, J.: LARKS: dynamic matchmaking among heterogeneous software agents in cyberspace. Auton. Agents Multi-Agent Syst. **5**, 173–203 (2002)
16. Telang, P.R., Singh, M.P., Yorke-Smith, N.: Relating goal and commitment semantics. In: Dennis, L., Boissier, O., Bordini, R.H. (eds.) ProMAS 2011. LNCS (LNAI), vol. 7217, pp. 22–37. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31915-0_2