Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

A Case-Based Service Request Interpretation Approach for Digital Homes

Chiung-Hon Leon Lee
Department of Computer Science and Information Engineering
Nanhua University
Dalin, Chia-Yi, 620, Taiwan
chlee@mail.nhu.edu.tw

Alan Liu
Department of Electrical Engineering and Center for Telecommunication Research,
National Chung Cheng University
Min-Hsiung, Chia-Yi, 621, Taiwan
aliu@ee.ccu.edu.tw

***Abstract*—We propose a case-based service request interpretation approach for digital homes. This approach starts the system construction from the view of software requirements engineering and a goal-driven approach is used to model service requests. The predefined goal model will be encapsulated in a case of a case base. The cases will be future selected and refined by performing a proposed case-based service request interpretation process. The system will base on the selected case to perform a series of actions to satisfy the service requests. Based on the case adaptation process, the system can adapt different user's service request and give a convenient interface for digital homes. We give some scenarios to demonstrate applications of the proposed approach.**

***Keywords*—goal-driven approach, case-based reasoning, digital homes**

I. INTRODUCTION

How to construct a convenient and smart living environment is a trend in developing home automation [1]. If a smart environment has ability to understand the user request from a simple command or direction, the environment can provide more friendly and efficient services for the user [2].

Recently, combining digital appliance, home network and Service-Oriented Architecture, Home Service Systems (HSS) is expected to provide sophisticated services and more comfortable living space for residents. For example, in [3], the author proposed a model and design methodology of continuous services over the HSS. In [4], the author discussed the problem of how to provide a flexible composition mechanism for smart device services. Following the development of variant appliances and smart devices, services that provided by a digital home are more and more complicated. How to integrate variant services and how to provide a convenient, efficient, and friendly human-computer interaction interface become significant factors for promoting smart home service [5].

In the real world, it is easy for a human being to understand requests of the other people from some simple directions. For example, a boss might say "I am thirsty." A smart assistant can understand that the boss wants him to make a drink for the boss. The assistant might has past experience that the boss gets used to drink green tea. He could interact with the boss to confirm boss's request and then make a cup of tea for the boss.

How to bridge the gap between human users and computer-based systems and provide users a better, efficient, and convenient way to interact with the system are long-term research issues of Human-Computer Interaction (HCI), Human-Robot Interaction (HRI) [6], and Natural Language Processing (NLP) [7]. To understand the implicit meaning of a sentence from the other one is easy for a human being. However, to HSS, because lack of common sense and background knowledge, how to interpret the user's real requests becomes a nontrivial task. Besides, how to infer or 'guess' implicit user requests in the user's simple direction also needs more auxiliary information such as the user's preference, system capabilities, and context information.

In research domains of HRI and HCI, there are several ways to facilitate the context information gathering and user intention identification such as visual recognition of hand and body gesture, conversational interaction, force-feedback tactile glove, or fusing the multimodal input [8,9]. However, in many applications, such as the network search engines, Web service systems, and HSS, a keyboard and a mouse or a touch screen are most general input devices for user-system interaction.

Combining NLP technology and other input method can provide users variant interface to facilitate user-system interaction. In [10], a NLP-based user interface for fingerprint recognition systems is proposed.

In this paper, we are interested in how to provide a phrase-based interface similar to traditional searching engines to enhance home services access. The system could infer the user's intention from the entered service request phrase. We proposed a Case-Based service request interpretation approach for interpreting service requests from users of digital home system. Each service request will be mapped to predefined goal models [11] that are encapsulated in cases.

The advantage of using the case-based approach is that the previous request interpretation experience can be stored in a Case-Base (CB), when the system receives a new service request, the old interpretation experience of service request can be adapted to interpret the new service request. Since the new service request has been satisfied successfully and CB has

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

no similar case, the system can store the new solution to CB for future usage. The concept of the proposed approach is shown in Fig. 1.

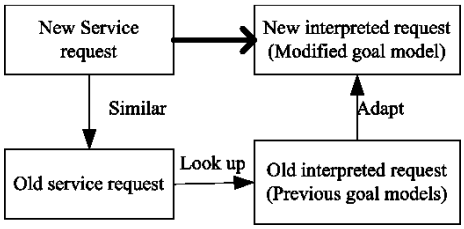

Figure 1. Concept of the Case-based service request interpretation.

In software requirements engineering [12], requirements are elicited from system users and the system will be designed to provide a set of software functions to achieve the elicited requirements. In our approach, we start the design of HSS from the view of system requirements engineering. In requirements analysis phase, the requirements specification is generated by goal based requirements analysis [13]. The functional requirements will be extended with goal models [14]. A set of user related goal models that represents the user requirements is selected and refined as the basis of cases for user requests and system functions mapping. A goal model might include or extend by other sub-goal models. The relationships between original goal models and sub-goal models can be derived from software requirements analysis. The relationships among software requirements, service requests, and digital home services are shown in Figure 2.

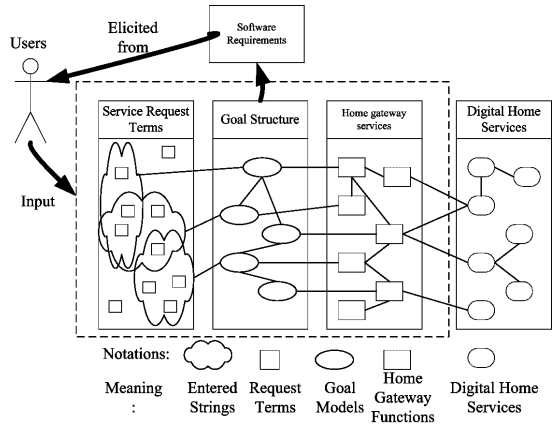

Fig. 2 The relationship between service requests and digital home services.

A service request from the user is composed by a set of service request terms. We assume that the service request implies the user's intention. By using predefined domain ontology and information of goal models, services requests could be mapped to a user-specific goal model in the goal structure. The goal structure consists of goal models and their relationships; it is used to bridges the gap between service requests and system capabilities.

When an original goal model is selected to represent the user's intention, the related sub-goal models of the original goal model is also retrieved to represent the implicit intention of the user. Base on the goal models, the system can generate a plan composed of a set of services of HSS to satisfy the user requests. The whole process of the proposed approach is like a requirements reuse approach. The HSS user enters a phrase to indicate user's requirement, and the system retrieves a previous elicited user-specific requirement and link to HSS services for execution.

This paper is organized as follows. First, we introduce a goal-driven service request modeling approach. Next, a case-based request interpretation approach is proposed. Then, we discuss our primary experimental results. Finally, we give conclusions and future works for this paper.

II. MODELING SERVICE REQUEST WITH GOALS

A service request will be an aim (an object or goal) that guides actions (HSS functions) to achieve it. Goals identification is a crucial factor in the elicitation of software requirements. We analyze software requirements based on a verb and its parameters. The goal model is the main part of a case in CB.

In [15], the authors proposed a goal-driven approach to model implicit and explicit service request. We adapt the approach to represent the service request of HSS. There are three requirements on the goal model. First, the goal model should contain enough information for mapping a service request phrase to a computable goal model. Second, the goal model should have a classification basis for analyzing their relationships. Third, the goal model should contain the information of related sub-goal models and plan to facilitate the retrieval process for the goal models and plans.

Based on these considerations, the basic attributes of a goal model are shown in Fig. 3. A goal is composed by four parts: *contents* to represent service request variables, *properties* to describe attributes of the goal, *relationships* to link original goal, related goals, and sub-goals, and a *plan* linkage will guide the system to retrieve and modify related plan to achieve the goal. Using this approach, a service request can be describe by goal models. A goal structure consists of goal models. The goal structure represents a space in which possible service requests could be identified.

The contents consist of *action*, *object*, *constrain*, and *parameter*. The *verb* in the service request string will be interpreted as an *action* which should be performed to satisfy the goal. *Object* indicates the object which will be affected or generated by the *Action*. The noun in the sentence might be an *Object*. The adjective and adverb play the role to constrain the execution of the *action or object*. The nouns are assigned to parameters which will be the parameters of the action.

For example, considering a service request likes "Establish a conformable living room at 5 p.m." the service request itself will be the *Contents*, the term "establish" will be an *Action*, "living room" is the *Object* of the *Action*, "5 p.m." are *parameter* of the *Object*, and "conformable" is a *constrain*.

Different from the requirement representation in software engineering, in our approach, we allow a service request string is not a complete sentence and the position of the request variable is not limited. This makes the system implementation more difficult but lets the users have more freedom to use the system.

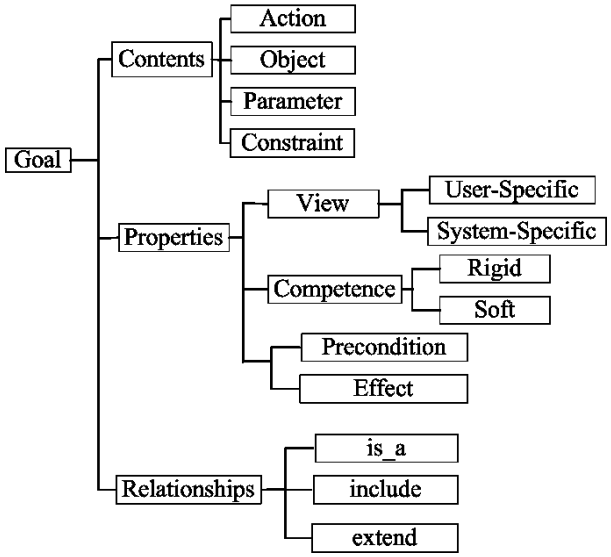

Fig. 3. The goal model.

The properties provide a classification basis of the goals. We use them to analyze the interaction among goal models when constructing the goal models. Four properties are considered as the classification basis of goal models: *View*, *Competence*, *precondition*, and *effects*.

View concerned whether a goal model is *user-specific* or *system-specific*. A *user-specific* goal model is an objective of the human user in using the system. A *system-specific* goal model is requirements on functions that the system provides to support the *user-specific* goal. The original goal derived from the user's request is always a *user-specific* goal because it reflects a service request that the user wants the system to complete. Because the purpose of goal models in our approach is to map a service request to a series of system activities, we take the point of view from whether the goal is used to represent the requirement from human user or a function of the system to aid the achievement of the user-specific goals. For example, "open the air-condition in living room" is a *user-specific* goal and "set proper parameters for the air-condition" is a *system-specific* goal to extend the *user-specific* goal.

The definition of *Competence* property classifies the goal as soft or rigid. A rigid goal must always be satisfied. A soft goal is desired to be satisfied and can be satisfied to a degree. If a goal is soft, there is a set of evaluation criteria to evaluate the satisfaction degree of the goal. For example, "Open a fan" is a rigid goal, because the system has to open a fan for the user. There are only two results of this goal: success or fail. Taking another example, "Tune proper fan speed" is a soft goal, because there is no clear cut about the term "proper". If the system wants to determine whether the fan speed is proper or not, it needs to refer a set of evaluation criteria such as temperature and user preference, etc. Finally, the pre-condition is used to indicate the achievement of a goal to be possible. The effects display whether a goal has been achieved or completed and related information.

There are three relationships defined to support the linkage of goal models and sub-goal models: *is_a, include*, and *extend*. The designer can specify a specialization to generalization relationship between goal models by is_a relationship. The concept of *include* is similar to the composition relationship of a class in object-oriented design. The extend goal model can be used to make the original goal achieved in a 'better' situation. Fig. 4 shows a scheduling flight example to demonstrate these relationships. Establish_LivingRoom is the original goal and inherits Establish_Space which composed of Assert_Event, Adjust_Temperature, Adjust_Humidity, and Adjust_Brightness sub-goals. Establish_Conformable_Space is used to enhance Establish_Space.

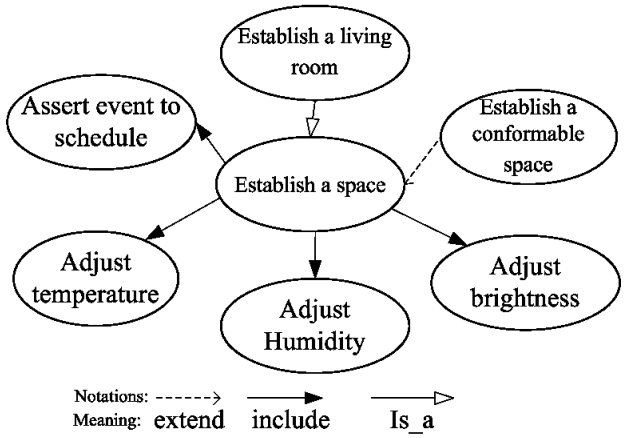

Fig. 4. An example of goal relationships.

III. CASE-BASED SERVICE REQUEST INTERPRETATION

A. Case Representation

The relationships among the case-base, cases, service requests, and goal models are shown in Figure 5. A case of our approach consists of three parts: the description part, solution part, and relation part. The description of the problem will be put in the description of the case here and later we will use this part for similarity measures during case retrieval. The descriptions usually are the feature set of the problem. Various representation methods can be applied here like lists, semantic networks, frames and objects. Sometimes, scenarios are also used here. We use the original service request extracted from requirement specification or entered by the user as the description part.

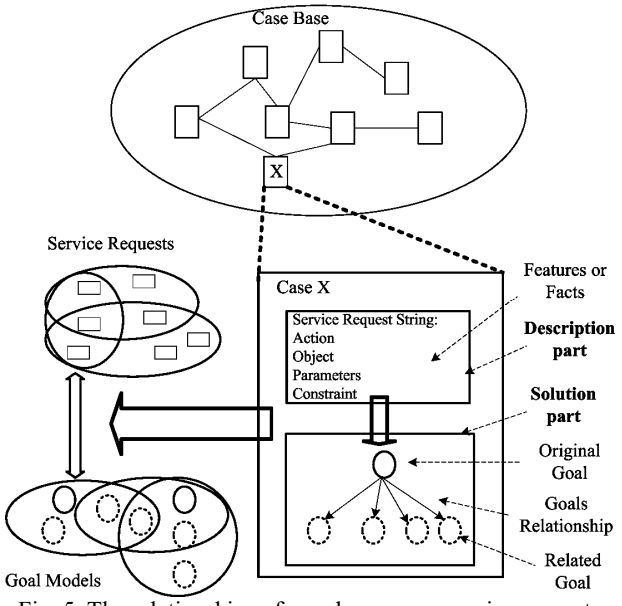

Fig. 5. The relationships of case base, case, service requests, and goal models.

The solution to the problem described in the description part will be recorded in the solution part. Once an identical problem comes, we can apply this solution to solve it. The representation of solution part, like description part, can be implemented by various knowledge representation methods. In our approach, the solution part contains the *Contents* and *Properties* of the goal model. Since the *Contents* and *Properties* are the solution, we can use the case to map the user entered string to a goal model for user intention representation. Finally, the *Relationships* of the goal model will be the relationship part. Using this part, related goal models that recorded in other cases could be retrieved to extend the original goal model.

B. Case Retrieval

To retrieve related cases from the case base, there are three major tasks: case indexing, similarity measures and case selection. Case indexing is helpful to improve the efficiency of case retrieval especially when the Case-Based Reasoning (CBR) system is having a vast case base with large amount of cases in it. It eases the problem of growing size of case base production by case storage. Similarity measures, which lead to the result of case retrieval, calculate the similarity or dissimilarity between new query case and old cases. Usually the case having highest score means it is most similar to the query case, in similarity measures, will be selected to solve the problem in the new query case. Once, if more than one cases are retrieved from case retrieval component, a CBR system have to perform case selection to select a case from these candidate cases.

Because the description part can be seen as a set of keywords and each keyword is a feature of the description part, using the keywords as index seems reasonable. However, because the user might use different service request terms or make minor type error, we prefer to use *n-gram* approach rather than term-based approach to index the cases. *N-gram* is a widely used technique in Information Retrieval (IR) research [16].

A service request is a string which has various features can be used for similarity measurement such as token-based distance and edit-distance metrics [17]. For capturing all of these different similarity types, we adapt the approach proposed in the [18] to build up a set of similarity measurement vectors (SMV) that reflect the each of the different similarity types and use these vectors for similarity measurement. The vectors can be divided into three groups: token vectors, edit vectors, and other vectors. The token vector is the set of token level similarity scores between the service request entered by the user and the description part in the case. The functions for deriving token vectors include Jensen-Shannon distance and Jaccard similarity. The edit vector can be derived from the following edit distance functions: Smith-Waterman distance, Levenstein distance, and Jaro-Winkler similarity. All of these string similarity measuring functions can be found in [17]. Lastly, the other vectors consists of the Soundex score and service request stemmer score between the service request and the description part.

Soundex is a phonetic index method and its key feature is that it codes a term based on the way term sounds rather than on how it is spelled. For example, terms that sound the same but are spelled differently, like “Smith” and “Smyth”, have the same code and are treated similar. Stemming is a technique for reducing words to their grammatical roots. A set of SMV can be derived by applying the string comparison functions to the service request and the description part of candidate cases. For example, a SMV could be <0.40, 0.40, 0.33, -8.0, 0.69, 0.77, 0.77, 0.75>.

Many criteria can be applied for the selection of cases. For example, we can simply select the case with the highest similarity value or maybe the case with the highest adaptability if the CBR system has the ability to evaluate the adaptability of cases. The selection process used in our approach includes two steps: rescoring and testing. The rescoring step rescore each SMV for ranking candidate cases and the testing step test whether the candidate case with highest rank matches the service request.

After all of the candidate cases are scored, we then rescore each SMV for case selection. For each element of SMV, if the candidate cases with the maximum value that the value will be map to 1. The value of the rest candidate cases will be mapped to 0. For example, assume we have 2 SMV for two candidate cases SMV1 and SMV2:

SMV1=<0.42, 0.38, 0.30, -30, 0.77, 0.60, 0.77, 0.8>, and SMV2=<0.31, 0.28, 0.21, -37, 0.46, 0.65, 0.66, 0.94>. After rescoring they become: SMV1=<1, 1, 1, 1, 1, 0, 1, 0>, and SMV2=<0, 0, 0, 0, 0, 1, 0,1>.ψ

The rescoring helps to determine the best possible candidate match for the newly service request. Next, the score in the SMV will be summarized to deriving a ranking score to rank the candidate cases for case selection. For example, the ranking score of SMV1 and SMV2 is 6 and 2. Finally, the

candidate case highest ranking score will be selected to test whether it suits for representing the service request. If the selected case pass the test, it will be chosen as relevant case, if not, this case will be removed from the candidate case set, and the SMV and ranking score will be recalculated again to select new case for testing. If the candidate case set becomes empty and no case pass the testing, it indicates the service request cannot be parsed.

When performing the case testing process for case selection, the system will use the *Action* and *Object* information of the goal model which stored in the candidate case to check whether the synonyms of the *Action*, *Object,* or *Parameters* can be found in the service request. If *Action* or *Object* cannot be found in the service request, the system can use the information of *Parameters* to "guess" possible *Action* or *Object.* If the information of *Parameters* is not enough to "guess" and the *Action* and *Object* cannot be found in the service request, the case will fail the testing.

C. Case Adaptation

Once a similar past case is retrieved in the case retrieval process, this case is sent to the case adaptation process to adapt its solution for generating a goal model to represent the service request intention. Adaptation is important and necessary because we cannot expect the pre-designed goal models stored in the case-base covering all service requests come from the user. In other words, the CBR systems only retrieve a partial matching case to the query case in most of the time. For using this similar past case to solve current problem, we assume that similar problems have similar solutions [19].

There are various techniques proposed for case adaptation. In many commercial CBR systems, null adaptation is used [20]. This approach leaves the problem of adaptation to the users themselves. This kind of system is called case retrieval systems since it is considered to have no reasoning part. Users of this kind of CBR systems must have expertise of the problem domain to adapt the cases by themselves. Automatic adaptation is needed to help those who do not have expertise of the problem domain and thus expand the applicable domains for CBR systems. Also automatic adaptation helps the CBR systems to perform autonomously and thus makes the CBR system learn without the assistance of human users. In addition, case adaptation draws the boundary of the domain within which problems can be solved. A better case adaptation is not only successful to adapt the past solution to current problem but also makes the boundary of the "can-be-solved" problem domain wider.

In our approach, case adaptation is needed to help the service request interpretation interface to adapt the user's input. However, we limit the service request which comes from the user cannot out of the system capability. For adapting the newly encountered terms in a service request, we index the concepts in the ontology by the *Contents* of goal model: *Action*, *Object*, *Constrain*, and *Parameter*. We add four relationship features: *Related*, *Required*, *Unordered*, and *Mutual-exclusive* to the concepts in the domain ontology. These features are similar to *Relationships* part of the goal model. The identification rules to determine whether the service term is a concept in the ontology is also bind to the concepts. We use these features and identification rules as guideline to explain the new term or to interact with the user for get explanation. For example, the concept "Establish" will be classified as an *Action* and the relationship features store it's related *Objects* such as "living room", "space", and "room". If the term "Establish" is identified in the service request but the system cannot find the predefined related *Object* "living room", the system can "guess" whether the object of the action is "living room", "space", or "room" by using the identification rules stored in the ontology. The features of *Object*, *Constrain*, and *Parameters* are defined in similar way, but the features of *Object* store its related *Parameters*, the features of *Constrain* store its related *Actions,* and the features of *Parameters* store its related *Objects.*

Using this mechanism, the interpretation capability of the system can be extend by learning the newly encountered terms. For example, assume that the service request entered by the user is "Establish a conformable living room at 5 p.m." and the description part of the selected case might be "Make a conformable living room at 5 p.m.". After interact with the user and confirm the new service request can be represented by the retrieved case, a new case "Establish equals to Make" can be added into the case-base.

IV. Discussion

How to separates, parses, and abstracts input keywords to derive a set of annotated terms is an important step for retrieving correct related goal models. In general, accurately assigning correct morphological tags to input text is an important challenge of Information Retrieval and NLP. One problem with assigning parts of speech is that a given word can be used in many ways such as the word Schedule could be a noun or verb. For getting a suitable parsing approach for service request extraction, we implement three parsing approaches for comparison: an augmented transition networks (ATN) parser, a Hidden Markov Model (HMM) parser, and a Case-based parser. The ATN and HMM parser are adapted from [21].

In this experiment, we construct an experimental lexicon which contains some words and related part of speeches of these words. A training service request set is automatically generated by referring the lexicon as the training file for HMM and Case-based Parser. A set of parsing rule for ATN is also designed.

First, we evaluate the recall performance of ATN, HMM, and Case Retrieval (CR) approaches. Some input testing requests are generated by referring the predefined lexicon and without unknown words. The CR retrieves a similar previous service request for tagging the newly entered service request but without learning and reasoning. In our experimental results, we found that performance of HMM is best when using the words in predefined lexicon to generate service request strings.

Next, we evaluate the parsers by inputting requests which contain unknown words. Because different users might use different words for service request, we randomly replace terms

in the automatically generated service request to simulate this situation. Our experiments results show that after a user uses the interface many times, the CBR parser approach has best parsing rate. The advantage of CBR approach is its learning ability but it long and unstable training time is the shortage.

V. CONCLUSION

Following the development of intelligent appliance, home network, and service-oriented architecture, HSS can provide more variant services. If we can integrate variant services and provide users a convenient, efficient, ease to use, and friendly human-computer interaction interface that will promote acceptance of HSS. Because lack of common sense and background knowledge, how to give users an intelligent interface that can understand the user request from a simple command or direction is difficult. In this paper, we proposed a goal model to model software requirement as the background for service request interpretation. We also proposed a CBR approach to map the service request to goal models. Our approach makes possible a natural human-computer interaction.

There are three directions to improve our approach. First, the adaptation knowledge plays an important role to improve the performance of our approach. We can develop a systematic approach for adaptation knowledge construction. Second, we can develop a "guided" interface to guide the service request input of the user. This will give some constrain to the interface but can acquire more precise service request from the user. Third, a realization of the vocal communication processing module can be used for multimodal interaction.

REFERENCES

[1] C. Nugent, et al., "Intelligent Person-Centric Services for Smart Environments," Proc. 4th Int'l Conf. on Smart Homes and Health Telematics, IOS Press, 2006, pp. 141-156.

[2] Marsic, A. Medl, and J. Flanagan, "Natural communication with information systems," *Proceedings of the IEEE*, vol. 88 , no. 8 , Aug. 2000, pp. 1354-1366.

[3] M. Aoyama, "A Model and Design Methodology of Continuous Services over the Home Service Systems", Proc. of the IEEE Int. Symposium on Service-Oriented System Engineering, 2008, pp. 50-55.

[4] M. Vallee, F. Ramparany, and L. Vercouter, "Flexible Composition of Smart Device Services," Proc. of the 4th Int'l. Conf. on Pervasive Computing, pp. 91–96, 2006.

[5] J.M. Hsu and I.R. Chang, "Design a Virtual Object Representing Human-Machine Interaction for Music Playback Control in Smart Home," Proc. of the Int. Conf. on Complex, Intelligent and Software Intensive Systems, 2009, pp. 614-619.

[6] C. Breazeal, "Social Interactions in HRI: The Robot View," *IEEE Trans. Systems*, *Man*, and *Cybernetics*, May 2004, pp. 181–186.

[7] J. Allen, *Natural language understanding 2nd ed.*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, 1994.

[8] Marsic, I., Medl, A., and Flanagan, J., "Natural communication with information systems," *Proceedings of the IEEE*, vol. 88 , no. 8 , Aug. 2000, pp. 1354-1366.

[9] Iba, S., Paredis, C.J.J., and Khosla, P.K., "Intention aware interactive multi-modal robot programming," *Proc. of the IEEE RSJ Conf.* 2003, pp. 3479-3484.

[10] V. Conti, et al., "A User-Friendly Interface for Fingerprint Recognition Systems based on Natural Language Processing," Proc. of the Int. Conf. on Complex, Intelligent and Software Intensive Systems, 2009, pp. 736-741.

[11] C.H.L Lee and A. Liu "Model the query intention with goals," Proc. of the USW2005, Mar. 2005, pp. 535-540.

[12] P. Loucopoulos and V. Karakostas, *System requirements engineering,* McGraw-Hill, London, 1995.

[13] J. Lee, N.L. Xue and K.Y. Kuo, "Structuring requirement specifications with goals," *Information and Software Technology*, vol. 43, 2001, pp. 121-135.

[14] J. Lee and N.L. Xue, "Analyzing user requirements by use cases: a goal-driven approach," IEEE Software, vol. 16, no. 4, 1999, pp. 92-101.

[15] C.H.L. Lee and A. Liu, "Modeling Explicit and Implicit Service Request for Intelligent Interface Design," Proc. of the Second International Workshop on Intelligent Interfaces for Human-Computer Interaction (IIHCI-2009), Fukuoka, Japan, March, 2009.

[16] D.A. Grossman and O. Frieder, *Information Retrieval: Algorithms and Heuristics*, Kluwer Academic Publishers, Boston, 1998.

[17] W.W. Cohen, P. Ravikumar, and S.E. Fienberg, "A comparison of string metrics for matching names and records," In Proc. IIWeb 2003 (IJCAI2003 Workshop), 2003, pp. 73-78.

[18] M. Michelson and C.A. Knoblock, "Semantic annotation of unstructured and ungrammatical text," In Proc. of IJCAI2005, 2005, pp. 1091-1098.

[19] W. Wilke and R. Bergmann, "Techniques and Knowledge used for Adaptation during Case-Based Problem Solving", In Proc. IEA/EIA conference, LNCS, vol. 1416, 1998, pp. 497-506.

[20] Watson, *Applying Case-Based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann Publishers, California, 1997.

[21] W. Mark, *Practical Artificial Intelligence Programming in Java*, 2005. Available at: http://www.markwatson.com/opencontent/.