Defining Agents in a COTS-Aware Requirements Engineering Approach

Lawrence Chung

Dept. of Computer Science
The University of Texas at Dallas
MS 31 P.O. Box 830688
Richardson, TX, USA
chung@utdallas.edu

Kendra Cooper

Dept. of Computer Science
The University of Texas at Dallas
MS 31 P.O. Box 830688
Richardson, TX, USA
kcooper@utdallas.edu

Abstract

The goals of developing systems better, faster, and cheaper continue to drive software engineering practitioners and researchers to investigate software engineering methodologies that are novel, yet practical. As the size and complexity of systems continues to grow, there has been a growing interest in the investigation of social paradigms (e.g., agent- and goal-oriented approaches) and the use of COTS components. These are being viewed more and more as promising, and also perhaps inevitable, solutions to this problem. The effective application of social paradigms and use of COTS components, however, requires a systematic approach that provides a set of concepts for modeling the subject matter, a set of guidelines for using such concepts, and tool support to assist the developer.

In this paper, we present an overview of a COTS-Aware Requirements Engineering (CARE) approach that explicitly supports the use of COTS components. Our CARE approach is knowledge based, has a defined process, and is agent- and goal-oriented. In more detail, we present the models and ontology that support the definition of agents. Our CARE approach is validated using a part of a Digital Library System; a prototype of the CARE Assistant tool is used to present some preliminary results.

Keywords: requirements engineering, agent-oriented, goal-oriented, COTS components

Introduction

The goal of developing systems better, faster, and cheaper continues to drive software engineering practitioners and researchers to investigate software engineering methodologies that are novel, yet practical. As the size and complexity of systems continues to grow, there has been a growing interest in the investigation of social paradigms (e.g., agent- and goal-oriented approaches) and the use of COTS components. These are being viewed more and more as promising, and also perhaps inevitable, solutions to this problem.

Our work is focused on creating, modeling and analyzing a goal- and agent-oriented requirements engineering approach that explicitly supports the use of commercial off the shelf (COTS) components. Our approach is called the COTS-Aware Requirements Engineering (CARE) approach. It is agent-oriented (as in [23][25][26]), goal-oriented (as in [1][9][24]), has a defined process (as in [17]), and is knowledge-based (as in [16][22]). The CARE approach is intended to provide a framework for integrating and extending existing work in the areas of agent-oriented requirements engineering (RE), goal-oriented RE, nonfunctional RE, conceptual modeling, and COTS research. The CARE approach is intended to assist the requirements engineer in developing high quality specifications, not to

replace their intelligence and experience. We believe this work may be of interest to researchers and practitioners involved in the development of complex systems that are distributed, open, and large-scale.

The CARE approach is being defined with a set of models and is supported with a prototype of the CARE Assistant Tool. Models are used to embody abstract ideas in a concrete representation. In a model, the ideas are more easily communicated, reviewed, and revised. The properties we seek in a model include completeness, consistency, correctness, etc. We also recognize that a model needs both an ontology and a methodology. An ontology is a formal description of entities and their properties; it forms a shared terminology for the objects of interest in the domain, along with definitions for the terms. A methodology describes how the entities are used or created. In CARE, the models include a process, product, and a meta-model. The process model describes the activities performed to define the system agents, goals, system requirements, software requirements, and software architecture (with respect to using COTS components). Each activity defines who performs it, the steps, constraints, inputs, and outputs. The product model describes the format of the output, or products, created using the process. The CARE meta-model defines the ontology for the approach. Entities include agents, goals, system requirements, software requirements, COTS components, assertions, and activities.

We recognize that developing the CARE approach is an ambitious project. To accomplish the work, we are using an iterative approach to define, refine, and validate the CARE models and tool support [5][6]. This iteration is focused on integrating existing agent-oriented work that supports the definition of agents. We define a product model, process model, and ontology; the prototype of the CARE Assistant Tool is also extended to support the agent definition. The approach is validated by applying it to (part of) a Digital Library System (DLS) example. A DLS is selected as the example because it is a complex, large-scale, non-proprietary system with a rich set of functional and non-functional requirements. A detailed description of the CARE approach (including the process model, product model, meta-model, and ontology) is available in the complimentary technical report [5].

This paper is organized as follows. Following the introduction, we describe related work and then provide an overview of the CARE approach. Subsequently, parts of the process, product, and ontology that support the definition of agents are presented. Some of our lessons learned are described; this is followed by our conclusions and future work.

Related Work

The CARE approach draws upon research in a number of areas including agent- and goal-oriented paradigms, knowledge engineering, process engineering, and COTS based software engineering work. As the focus of this work is integrating agent-oriented research in our CARE approach, we focus the related work section on research in the use of COTS components and agent-oriented paradigms in software engineering.

Approaches that offer specific technical solutions that are applicable to the requirements engineering phase include the Rational Unified Process (RUP), Model-Based Architecting and Software Engineering (MBASE), and the Procurement Oriented Requirements Engineering (PORE). RUP is a comprehensive, object oriented software engineering process model [12], which is based on four phases (transition, construction, elaboration, and inception) and a collection of core process disciplines (or workflows) including business modeling, requirements, analysis and design, implementation, test, deployment, etc. We include the RUP in our survey because it is an established, well accepted process. Although the RUP is not goal- or agent-oriented, it does support the use of a component based architecture. A component is defined in [15] as *a nontrivial piece of software: a module, package, or subsystem that fulfills a clear function, has a clear boundary, and can be integrated into a well-defined architecture*. In UML, a COTS component is represented as a component, a physical and replaceable part of the system that provides a set of interfaces and typically represents the physical packaging of classes, interfaces, and collaborations [14]. With the RUP, the use of component based architecture can proceed in several ways. In

addition to supporting a more traditional approach in which a system is built from scratch, RUP also supports building a system with the intent of developing re-usable, in-house components and an assembly approach of building software from commercial off-the-shelf components. For our requirements engineering work, we integrate the iterative approach in our process.

The MBASE approach focuses on the early phases of software engineering. It considers four types of models: success, process, product and property [3] and is consistent for use with COTS components [4]. MBASE uses four guiding principles to develop value-driven, shared-vision-driven, change-driven, and risk-driven requirements. From the MBASE framework, we draw on the ideas of developing a consistent set of models that are goal (i.e., success) driven and support the use of COTS components.

From the PORE technique, we draw on its support for the evaluation and selection of COTS components [17][20]. The PORE process model identifies four goals in a thorough COTS selection process: a) acquiring information from the stakeholders, b) analyzing the information to determine if it is complete and correct, c) making the decision about product requirement compliance if the acquired information is sufficient, d) selecting one or more candidate COTS components. The PORE approach also defines a meta-model that is composed of the following constructs: agents, actions (these are performed by agents), events (these trigger actions), goals (these are accomplished by actions), objects in the domain, state of an object, state transitions of objects, components, and functions provided by the product.

In agent-oriented research, agents have been characterized by four properties [23][25][26]. First, an agent is intentional and possesses intentional properties including goals, beliefs, abilities, and commitments. Second, an agent is autonomous and makes decisions and choices. Third, an agent depends on other agents to accomplish goals, complete tasks, or furnish resources. This characteristic makes an agent able to accomplish goals they could not achieve on their own. At the same time, however, it makes an agent vulnerable because it depends on other agent. Fourth, an agent is strategic. This allows the agent to analyze its opportunities and risks in the various proposals and configurations of a system.

Overview of the CARE Approach

An overview of the CARE approach is provided in this section (refer to Figure 1) and it is illustrated with examples from the Digital Library System. The reader is referred to the complimentary technical report in [5] for details. The CARE approach is characterized as agent-oriented, goal-oriented, knowledge based, and has a defined process.

The CARE approach supports the definition of agents, goals, system requirements, and software requirements. The first task is to define the agents for the system. As part of the definition, the goals of the agents are captured; these goals are used to drive the development of the system. Goals are very high-level objectives of the system. The may be functional (hardgoals) or non-functional (softgoals). Goals are related to the other goals of the system using the NFR framework [9][18]. This framework has four kinds of relationships: makes (very positive), helps (positive), hurts (negative), and breaks (very negative). As goals and requirements are defined, COTS components are identified as possible matches.

When mapping goals and requirements to possible matches of COTS components, the requirements engineer (RE) searches, matches, and selects from the components in the repository. The RE considers the functional and non-functional capabilities and, with the assistance of the software architect, the impact of using a component on the software architecture. If a match is not found, the RE has a number of options. The RE may request that the component engineer attempt to find another component available in the marketplace and add it to the repository. The RE may try bridge the gap between the customer's needs and the capabilities available in the components by asking the vendor to make a change to a component or asking the customer to change a goal or requirement.

Describing the CARE approach requires an agent-oriented notation. We have selected the i* framework as it is well suited for describing goal and softgoal dependencies among agents

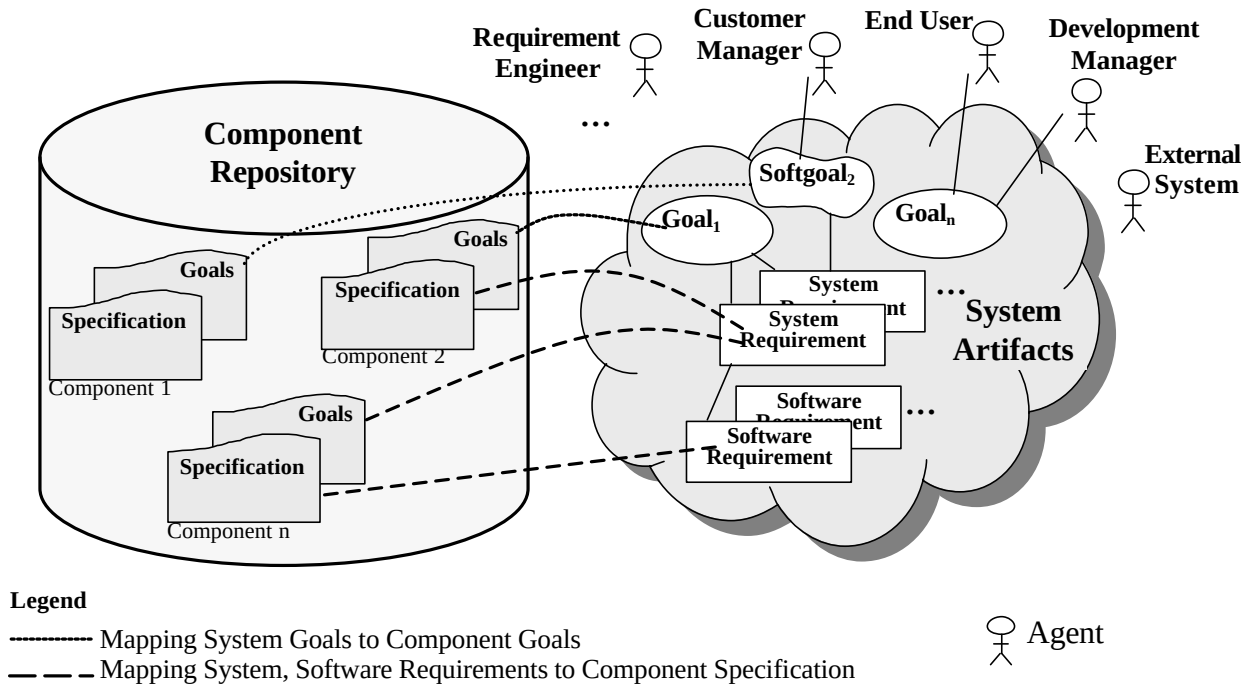

Figure 1. Overview of Agents, Goals, Requirements and COTS Components in CARE

[23]. The i* framework is composed of two models: the strategic dependency and the strategic rational model. The strategic dependency model is used to describe goal and softgoal dependencies among intentional agents (agents that make choices). The concepts of the strategic dependency model are embedded into the conceptual modeling language Telos [19], a descendent of RML [11] - an object-oriented requirements modeling language for functional requirements. As a result, i* provides an extensible, object-oriented representational framework with classification, generalization, aggregation, attribution, and time. The strategic rational model is used to describe how an agent accomplishes a goal in terms of subgoals, softgoals, resources, and tasks. The strategic rationale model is surrounded by an oval with a dashed line. The steps (subgoals, softgoals, tasks) an agent chooses to use are called a routine.

An overview of the CARE process model from an agent-oriented perspective is illustrated in Figure 2. At this level, the agents are three interdependent businesses: Customer, Development House, and the Component Vendor. For example, the Development House depends upon the Customer to validate the system artifacts.

Refining the CARE Models

CARE Process Model: Define Agents

The process model is refined (refer to Figure 3) to illustrate additional details for the *goal* to deliver the system artifacts. In addition, this figure contains the strategic rationale model for the RE. To accomplish the goal of delivering the agents, goals, and requirements for a system, the RE performs the corresponding tasks.

In Figure 4, the model is further refined to illustrate the tasks involved in defining the baselined agents. The RE's tasks include elicit, analyze, correct, validate, and baseline the agent definitions. The RE uses them iteratively to refine the definitions of the agents. Sources of information to accomplish these tasks include the customer (end users, managers, etc.), existing documentation for the system, the expertise of requirements engineers, software architects, and component engineers, etc. Each of these tasks is described briefly below:

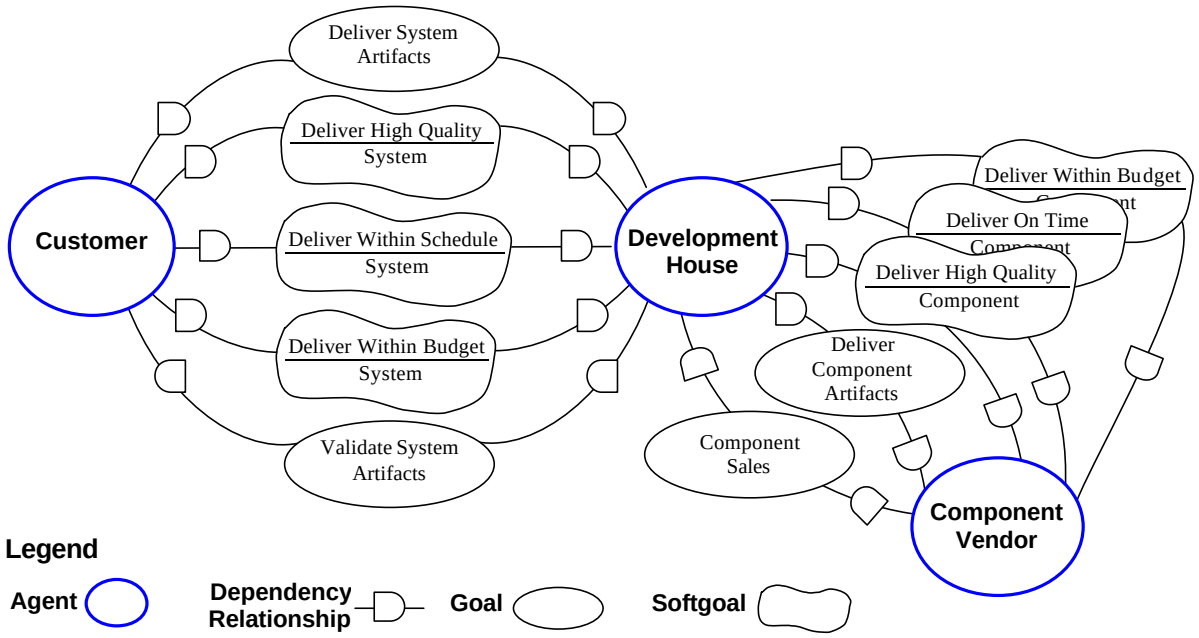

Figure 2. A High-level Agent-oriented View of the CARE Process Model

Elicit Agents. The RE identifies and defines the agents, or stakeholders, for the system under development. The agents may be people, computer systems, or businesses. For example, the people may include the end-users, administrators, managers, people responsible for procuring the system, and people responsible for developing the system. An agent is intentional and has one or more roles to play. Each role has a unique identifier, agent dependencies, and abilities.

To elicit the initial set of agents, the RE begins with the enterprise goals and the initial concept of the system. To identify additional agents, the RE may use interviews, surveys, workflow analysis, and examine the documentation of the current system (if available). A significant amount of work has been done in the area of elicitation (e.g.,[13][21]). In the CARE approach, the RE can choose the specific technique used to elicit the agents. The RE documents the decisions and the rationale for the decisions made performing the task.

Analyze Agents. The RE analyzes the agent descriptions to identify errors of commission (conflicting, incorrect, and redundant agents) and omission (missing agents). The analysis is performed with respect to the initial concept for the system and the enterprise goals. The RE documents the decisions and the rationale for the decisions made performing the task.

Correct Agents. The RE corrects the definitions of the agents based on the analysis results. The RE documents the decisions and the rationale for the decisions made performing the task.

Validate System Agents. The RE uses feedback from the customer to validate the definition of the agents. The RE documents the decisions and the rationale for the decisions made performing the task.

Baseline System Agents. The RE and CM baseline the validated agent definitions and place them under configuration management.

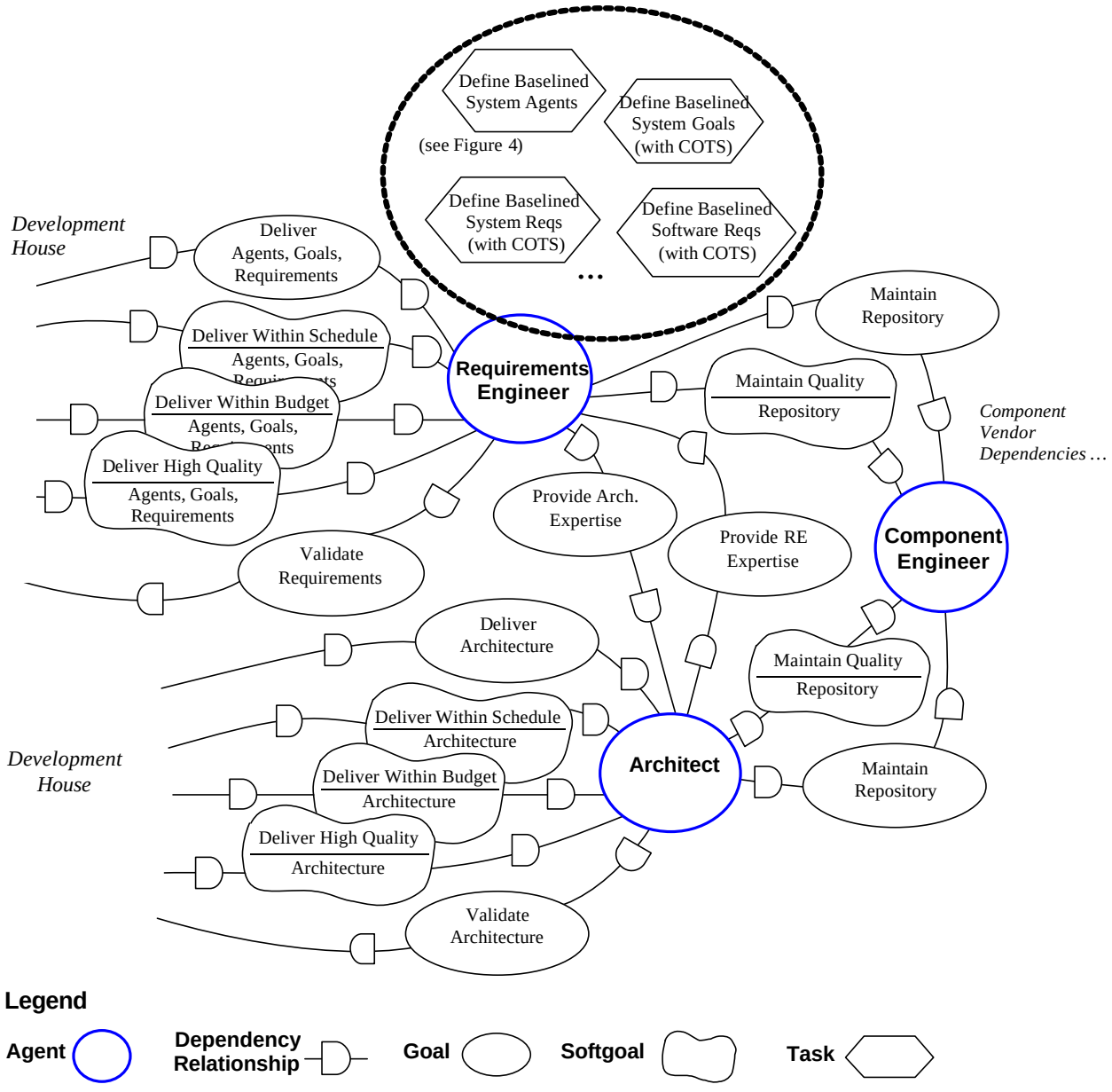

Figure 3. Strategic Dependency and Rationale Models

CARE Product Model: Define Agent

The users of the CARE process may select their own techniques for the product model. In this work, we use a tabular approach to summarize the attributes the RE needs to define. This presentation is similar to examples available in the MBASE approach [3]. The presentation of the attributes defined in this section is not intended to restrict the structure of the knowledge base. The attributes for the Agents are described in this section. Each attribute has a brief description, cardinality, and whether or not the attribute is optional or required. An example of a system agent definition is provided in a tabular format in Appendix A.

Unique Identifier. This attribute is used to support traceability. The Agent Unique Identifier is "A_" followed by a unique three digit integer. (e.g., A_001). Each instantiation of an agent has exactly one Unique Identifier attribute value. The attribute is required.

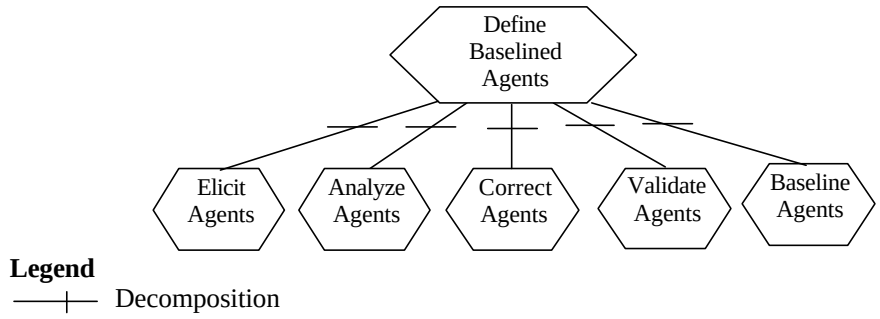

Figure 4. Define Baselined Agents

Type. This attribute defines the specialization of the agent. For example, the Agent may be a User, Administrator, Development Staff, etc. Each instantiation of an agent has exactly one Agent Type attribute value. The attribute is required.

Role. This attribute defines the role an agent is performing. For example, in a DLS, the role may be junior librarian, senior librarian, borrower, system administrator, etc. Each instantiation of an agent has exactly one Role attribute value. The attribute is required.

Association. This attribute defines the association of the agent. For example, the agent may be associated with the company acquiring the system, with the development house, or with a component vendor. The Association attribute is optional.

Perspective. This attribute defines the perspective of the agent. For example, the agent may be viewing the system from a customer's perspective or a developer's perspective. Each instantiation of an agent has exactly one Perspective attribute values. The attribute is optional.

Relevance. This attribute defines why the agent is relevant to the model of the system. For example, how does the agent interact with the system to meet a business need. Each instantiation of an agent has one or more Relevance attribute values. The attribute is required.

Other Agent Dependency. This attribute defines the other agents that are depended upon to accomplish goals. The agents may be needed to accomplish a goal when the product is being built or when the product is being used. Each instantiation of an agent has one or more other agent dependency attribute values. The attribute is optional.

Goals. This attribute defines the goals of the Agent. The goals may be related to building the product or using the product. Each instantiation of an agent has one or more Goal attribute values. The attribute is optional.

Ability. This attribute defines the ability, or skills, the agent has to accomplish the goals. Each instantiation of an agent has one or more Ability attribute values. The attribute is optional.

Assumptions. This attribute defines any assumptions about the Agent. Each instantiation of an agent has one or more Assumptions attribute values. The attribute is optional.

Notes. This attribute defines any additional notes about the Agent. Each instantiation of an agent has exactly one Agent Notes attribute value. The attribute is optional.

Agent Ontology

The most general class in the ontology is the Agent. Agents in this class are specialized as enterprise (i.e., a company), people, and computer systems. Enterprise agents are specialized into businesses involved in developing components, developing a customer's system, and a customer. Associated with the development house are people who work for the company and fill the roles of requirements engineer, software architect, and component engineer. The agents may have one or more assertions (e.g., a system agent must have at least one system goal) and they accomplish goals by performing activities.

Illustration

The support for defining system agents in the CARE approach is validated by applying it to (part of) a DLS example. A DLS is selected as the example because it is a complex, large-scale, non-proprietary system with a rich set of functional and non-functional requirements.

The example is illustrated using the prototype of the CARE Assistant Tool. The prototype supports a subset of the constructs in the meta-model, including the agents. ConceptBase [10] is used as the database for the knowledge base implementation.

The RE has a goal of defining the baselined system agents for the DLS. In this example, the routine to define the set of baselined system agents is composed of the tasks to Elicit, Analyze, Correct, Validate, and Baseline Agents. When the initial set of agents are defined, their analysis identifies errors of omission such as undefined attributes and errors of commission such as incomplete descriptions. The elicitation, analysis and correction of the definitions is an iterative process. When they are considered to be complete, consistent, and correct by the RE, the customer is asked to validate the definitions. In an academic setting, we realize that the tasks are significantly simplified as we do not have real customers.

Using the CARE process and product models for the definition of agents, the classes for the agents and instances for the DLS example are defined in O-Telos. Example is presented below for a junior librarian (refer to Figure 5 for a graphical representation).

```
Individual junior_librarian in UserAgentClass with
UniqueIdentifier
    User_ID: "A_001"
Association
    User_Association:"Library Company Inc."
Perspective
    User_Perspective: "Customer"
Type
    User_Type: "end user"
Role
    User_role: "junior librarian"
Relevance
    User_Relevance:
    "1. Interacts with borrowers to provide reference assistance.
     2. Interacts with DLS to obtain library items.
     3. Interacts with RE to provide domain expertise.
      4. Interacts with RE to validate agent, goals, requirements"
AgentDependency
    Agent_Dependency1: UserAgent_A_004
AgentDependency
    Agent_Dependency2: DeveloperAgent_A_009
Goals
        User_Goals1: Goal_G_002        {* The DLS should be easy to use *}
Goals
        User_Goals2: Goal_G_004        {*The DLS should have high availability*}
Goals
        User_Goals3: Goal_G_005        {*The DLS should have fast performance even when the
                                       system is loaded with the maximum number of users *}
Goals
        User_Goals4: Goal_G_006        {*The users should be able to access a large number of
                                       diverse objects*}
Goals
        User_Goals5: Goal_G_007        {*The users should be able to search, browse, and retrieve
                                       objects quickly and efficiently from remote locations*}
Goals
        User_Goals6: Goal_G_008        {*The librarians should be able to maintain the library
                                       quickly and efficiently*}
Goals
        User_Goals7: Goal_G_009        {*The librarians should be able to provide reference
                                       support quickly and efficiently*}
```

Goals
User_Goals8: Goal_G_011 {*The DLS should be scaleable in order to accommodate new collections or additions to the current collections. *}
Goals
User_Goals9:Goal_G_012 {*The DLS should be secure*}
Ability
User_ability: "Trained in library science, 0-2 years experience"
End

New agents identified in the CARE approach for the development house are the component engineers (CEs). The CEs are responsible for adding, modifying, and deleting COTS components into the repository. They are a source of technical expertise for the REs when matching and selecting possible COTS components. The component vendor, an enterprise agent, is another new addition when considering a COTS-aware approach. People agents working for the component vendor such as the technical sales representative or technical support personnel are also stakeholders in the system development.

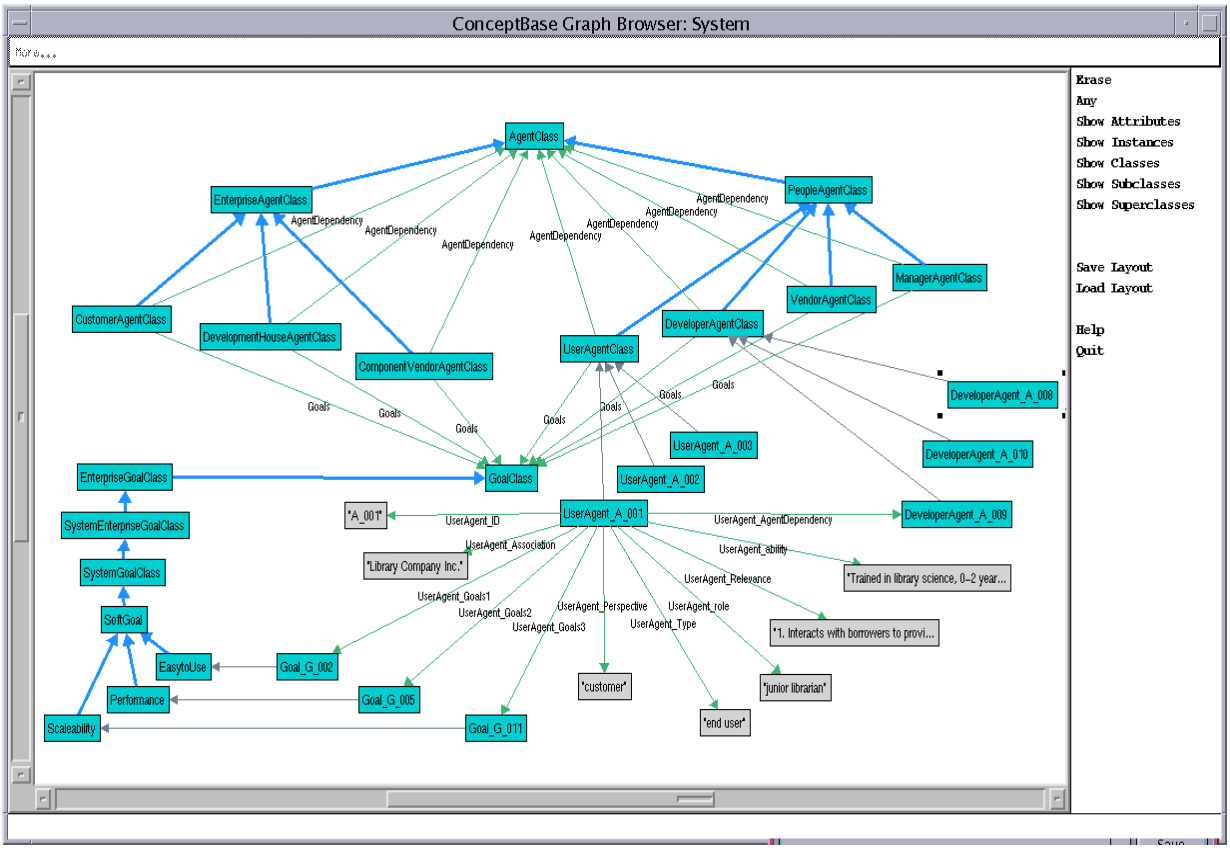

Figure 5. Illustration of Agents and Goals Defined with the CARE Approach

Lessons Learned

We have already learned a number of lessons in developing this phase of the work. The formalization into O-Telos class definitions has encouraged the development of a complete, correct model. As the definitions were being developed, it became an iterative process to define, validate, and correct the definitions.

Attributes that were initially defined, but later appeared to have little value in the validation with the DLS example, were deleted. An example of this case was in the System Agent class. Initially, the name of an agent was included as an attribute. In the validation, it became clear that the role the agent was playing (e.g., junior librarian) was more relevant.

As we modeled the interdependencies among agents, we encountered a problem while trying to strictly adhere to the i* framework. In i*, agents are allowed to be decomposed and refined, however goals are not. Due to the complexity of our model, we also need to decompose goals to ensure the models are straightforward to review and correct. A simple solution has been selected to work around the restriction: we represented the refinement of a goal on a separate diagram. If this extension can be formally added to the i* framework it may be valuable for the RE community.

Conclusions and Future Work

The CARE research project is defining CARE process, product, meta-model, and tool support concurrently. The intent is to have a set of models and tool support that work together and are practical to apply to complex, large-scale systems. An iterative approach is being used to systematically extend and refine the models and tool support. As each part of the work matures, it can be used to detect problems such as incompleteness, inconsistency, or ambiguity in the other parts.

In this paper, our contribution is to present our preliminary work in the CARE approach that supports the definition of agents. We present the process model, product model, and agent ontology; the approach is illustrated with an example from a Digital Library System. A more detailed description of the CARE is available in [5]. The task to define system agents is refined into steps to elicit, analyze, correct, validate, and baseline the definitions. In the product model we capture a set of attributes for system agents including unique identifier, type, role, association, perspective, relevance, other agent dependencies, goals, ability, assumptions, and notes. Classes and instances of agents for part of a Digital Library System are formalized in the O-Telos notation and presented using a prototype of our CARE Assistant Tool. It is important to note that our approach does not capture some characteristics of agents including a level of commitment or its strategic capability. We are investigating solutions to this problem.

As we have only used one example system to validate the work, we recognize that additional validation with different systems is needed. The next example system we intend to use is a telepresence system. We also plan to extend and refine the CARE approach's models (process, product, meta) and tool support. More specifically, we propose to support reasoning about contradictory, non-functional goals using the NFR framework.

REFERENCES

[1] Antón, A., Potts, C., "The Use of Goals to Surface Requirements for Evolving Systems", Int. Conf. on Software Engineering, Kyoto, Japan, 19-25 April 1998, pp. 157-166.

[2] Basili, V.R. and Boehm, B., "COTS-based systems top 10 list", Computer, Vol. 34 Issue: 5, May 2001, pp. 91-95.

[3] Boehm, B., Port, D., Abi-Antoun, M., and Egyed, A., "Guidelines for the Life Cycle Objectives (LCO) and the Life Cycle Architecture (LCA) deliverables for Model-Based Architecting and Software Engineering (MBASE)", TR USC-CSE-98-519, USC-Center for Software Eng.

[4] Boehm, B. "Requirements that handle IKIWISI, COTS, and Rapid Change", IEEE Computer, Vol. 33 Issue: 7, July 2000, pp. 99 –102.

[5] Chung, L. and Cooper, K., "A COTS-Aware Requirements Engineering Approach: Defining System Level Agents, Goals, and Requirements, version 2", TR UTDCS-11-02, The University of Texas at Dallas, 2002.

[6] Chung, L. and Cooper, K., "Defining Goals in a COTS-Aware Requirements Engineering Approach", Proceedings of the International Council on Systems Engineering Symposium 2002 (INCOSE 2002), electronic proceedings.

[7] Chung, L. and Cooper, K., "A Knowledge-Based COTS-Aware Requirements Engineering Approach", Proceedings of the International Conference on Software Engineering and Knowledge Engineering 2002 (SEKE 2002), pp. 175-182.

[8] Chung, L., Katalagarianos, P., Marakakis, M., Mertikas, M., Mylopoulos, J., and Vassiliou, Y., "From Information System Requirements to Designs: a Mapping Framework", Information Systems, Vol. 16, No. 4, 1991, pp. 429-461.

[9] Chung, L., Nixon, B., Yu, E., and Mylopoulos, J., *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishing, 2000.

[10] ConceptBase, A deductive object manager for meta databases, Aachen University of Technology (RWTH), Germany, version 5.2, 2002.

[11] Greenspan, S., Mylopolous, J., and Borgida, A., "On formal requirements modeling languages: RML revisited", 16th Int. Conf. on Software Eng., 1994, pp. 135-147.

[12] Jacobson, I., Booch, G., and Rumbaugh, J., *The Unified Software Development Process*, Addison Wesley Longman, Inc., USA, 1999.s

[13] Kato, J.; Komiya, S.; Saeki, M.; Ohnishi, A.; Nagata, M.; Yamamoto, S.; Horai, H., "A model for navigating interview processes in requirements elicitation", Eighth Asia-Pacific Software Engineering Conference, 2001 (APSEC 2001), pp. 141 -148.

[14] Kruchten, P., "Modeling Component Systems with the Unified Modeling Language", Int. Workshop on Component-Based Software Eng., 1998, http://www.sei.cmu.edu/cbs/icse98/papers/p1.html.

[15] Krutchen, P., "What is the Rational Unified Process?", Rational Edge e-zine, http://www.therationaledge.com/content/jan_01/f_rup_pk.html, January, 2001.

[16] Liu, A. and Tsai, J.J.P., "A knowledge-based approach to requirements analysis", 7th Int. Conf. on Tools with Artificial Intelligence, 1995, pp. 26 - 33.

[17] Maiden, N.A.M., Kuim, H. and Ncube, C. "Rethinking Process Guideance for Software Component Selection", Proc. of the 1st Int. Conf. of Component Based Eng., 2002, pp. 151-164.

[18] Mylopoulos, J., Chung, L., and Nixon, B., "Representing and using nonfunctional requirements: a process-oriented approach" IEEE Trans. on Software Eng., 18 (6), June 1992, pp. 483-497.

[19] Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M., "Telos: Representing Knowledge about Information Systems", ACM Trans. Info. Sys., 8 (4), October 1990, pp. 325-362.

[20] Ncube, C. and Maiden, N., "Guiding parallel requirements acquisition and COTS software selection", Proc. of the IEEE Int. Symp. on Requirements Eng. 1999, pp. 133-140.

[21] Nuseibeh, B. and Easterbrook, S., "Requirements engineering: a roadmap", in A. Finkelstein, editor, "The Future of Software Engineering", Special Volume published in conjunction with ICSE 2000, 2000.

[22] Rolland, C. and Grosz, G., "A general framework for describing the requirements engineering process", IEEE Int. Conf. on Systems, Man, and Cybernetics, 1994, vol. 1, pp. 818 - 823.

[23] Yu, E., "Modelling Strategic Relationships For Process Reengineering", DKBS-TR-94-6, Univ. of Toronto, Canada, Dec. 1994.

[24] van Lamsweerde, A. and Letier, E., "Handling Obstacles in Goal-Oriented Requirements Engineering", IEEE Trans. on Software Eng., Vol. 26, September 2000, pp. 978-1005.

[25] Wooldridge, M. and Ciancarini, P., "Agent-Oriented Software Engineering: The State of the Art," Handbook of Software Engineering and Knowledge Engineering, World Scientific Publishing Co., 2001, pp 1-28.

[26] Wooldridge, M., Jennings, N.R., and Kinny, D., "A Methodology for Agent-Oriented Analysis and Design" Proc. 3rd Int. Conf. on Autonomous Agents, 1999, pp. 69-76.

Appendix A. Semi-formal Agent Definition (Junior Librarian)

Agent Example	
Unique Identifier	A_001
Type	User
Role	Junior librarian
Association	Library Company Inc.
Perspective	Customer
Relevance	1. Interacts with RE to provide domain knowledge 2. Interacts with RE to validate deliverables 3. Interacts with proposed system to provide reference assistance to borrowers.
Other Agent Dependency	1. UserAgent_A_002 (senior librarian) 2. UserAgent_A_004 (borrower) 3. DeveloperAgent_A_009 (requirements engineer)
Goals	1. G_002 The DLS should be easy to use 2. G_004 The DLS should have high availability 3. G_005 The DLS should have fast performance even when the system is loaded with the maximum number of users 4. G_006 The users should be able to access a large number of diverse objects 5. G_007 The users should be able to search, browse, and retrieve objects quickly and efficiently from remote locations 6. G_008 The librarians should be able to maintain the library quickly and efficiently 7. G_009 The librarians should be able to provide reference support quickly and efficiently 8. G_011 The DLS should be scaleable in order to accommodate new collections or additions to the current collections. 9. G_012 The DLS should be secure
Ability	Trained in Library Science, 0-2 years experience