Agent-Oriented Methodologies – Towards A Challenge Exemplar

Eric Yu[1] and Luiz Marcio Cysneiros[2]
[1]Faculty of Information Studies
yu@fis.utoronto.ca
[2] Department of Computer Science
University of Toronto
cysneiro@cs.toronto.edu

Abstract. The agent-oriented approach to software development is poised to transition from the prototyping done by researchers to the development of large-scale industrial-strength applications by software professionals. For this to succeed, methodologies are needed to systematically guide and support developers through the various stages of system development. A number of agent-oriented methodologies have been proposed recently, offering a variety of conceptual frameworks, notations, techniques, and methodological steps. The diversity of approaches offers rich resources for developers to draw on, but can also be a hindrance to progress if their commonalities and divergences are not readily understood. One way to establish a common context for probing and relating various methodologies is to define and adopt a standardized example setting (or "exemplar") to focus discussion and debate. This paper proposes an exemplar from the health care domain. It is structured into a set of scenarios, supplemented by a series of questions to be posed to each methodology. We consider how an exemplar might serve the needs of the agent-oriented methodology community, and discuss the criteria for selecting an exemplar.

1. Introduction

Despite rapid advances in agent technologies, their adoption in mainstream application areas such as large-scale information systems is still limited. It is generally recognized that a major reason is the lack of systematic methods to guide the development of agent-oriented systems. Agent-oriented methodologies have thus become an important and urgent area of research. In the recent past, more than a dozen methodologies have been proposed. They offer a range of modelling concepts, elaboration and analysis techniques, and opportunities for tool support. They vary in maturity and scope of coverage The diversity of approaches offers rich resources for developers to draw on, but can also be a hindrance to progress if their commonalities and divergences are not readily understood.

One way to advance the state of research in agent-oriented methodologies is to define a suitable example problem that can serve as a focal point for discussion and exchange of research ideas and results. Examples are indispensable for illustrating and explaining methodologies. They provide concrete instances that allow abstract concepts and descriptions to come alive through domain settings and scenarios. Understandably, each methodology would typically be illustrated with examples that are best suited (possibly tailored) for demonstrating its specific features. They are likely to emphasize certain aspects of the problem domain more than others.

A common example that has become familiar to a research community can be a valuable resource in a number of ways.

- It can be used to capture and embody the set of problems faced by the community. It can present the problems with a clear separation from any solution approaches. Specific research projects can use it to delineate research objectives without necessarily addressing the entire set of problems.
- It can provide a familiar application context as a basis for defining and discussing the methodological issues. Knowledge and experience gained about the context can be "reused," allowing attention to focus on the solutions to problems.
- For someone learning about a new methodology, the availability of a familiar example can help quickly place the methodology in relation to other methodologies, to understand what problems it addresses, and how it advances beyond existing ones.
- For someone developing a methodology, it provides a more objective vehicle for assessing capabilities and limitations. It can also be invaluable for third-parties to compare and evaluate different methodologies according to some criteria.
- It can also be a great help to someone from outside the research community, to quickly grasp the issues faced, to see how well they are addressed by the state of the research. Potential users

(methodology adopters) can use it to relate to their own domain problems, thus recognizing the strengths and limitations of agent-oriented approaches versus other approaches.

A common example can therefore be a useful instrument for a research community to help clarify research objectives, consolidate results, and focus future directions.

Common examples have been used to advantage in a number of areas in information systems and software engineering – in IS development methodologies, e.g., [Olle 82], in software specification, e.g., [Wing 88], in software architecture e.g., [Parnas 72], and in software process modelling [ISPW 93], in requirements engineering [Feather97]. [Feather97] has examined the use of such examples in detail, and refers to these as "exemplars". We will follow this usage.

In this paper, we propose the use of a health care information systems example as an exemplar for the agent-oriented methodology community. The exemplar aims to be problem-centered, neutral with respect to particular methodologies or approaches, and rich in embodying real-world issues and challenges. It is intended to test the limits of agent-oriented methodologies, allowing their strengths as well as weaknesses to come through. It is a challenge to individual methodologies to further their strengths and remedy their weaknesses. It is a challenge to the community to combine the best ideas from various research results to synthesize practical methodologies for adoption by practitioners.

We start by discussing the criteria for selecting an exemplar in Section 2. Section 3 introduces the proposed exemplar, and considers how it meets the criteria. Section 4 presents details of the exemplar in the form of itemized scenarios. Section 5 provides a list of questions that probe the capabilities and limitations of various methodologies. Section 6 discusses related work.

2. Criteria for Selecting a Challenge Exemplar

Broad Coverage. Despite the flurry of research activities and advances, most agent-oriented methodologies are still in their early stages of development. There is considerable diversity in concepts (e.g., no consensus on the notion of agent), approaches (formal, informal, hybrids), origins (extensions to object-orientation, knowledge acquisition methodologies, etc.), lifecycle coverage (requirements to implementation), or in even overall objectives. While a standardized exemplar can be a stimulus for movement towards convergence, the exemplar itself should not lead the convergence in any particular direction. As much as possible, it should allow any approach to be presented and explored to the fullest extent. An exemplar that is problem-focused would avoid any bias. The problem should be described in terms of the application domain as much as possible. Where references to technology systems are made, they should be construed as suggestive rather than definitive. Mention of methodological concepts and steps should be avoided as much as possible.

Showcasing Agent Orientation. The exemplar should allow the distinctiveness of agent-oriented approaches to shine through. Agent-oriented systems can offer greater flexibility, robustness, adaptiveness, and many new functional capabilities. Agent-oriented concepts during the development process can offer social organizational views and analyses of relationships among software agents as well as among human agents. Agent abstractions allow higher level analysis and encapsulation of issues and greater independence from implementational issues.

Overcoming legacy methodology concepts. Conventional development methodologies are typically organized into stages such as requirements, design, implementation, testing, and so on. While the broad categories may still apply, their meanings and specifics will likely undergo substantial revisions under agent orientation. What is a requirements specification if agents have autonomy? What is the proper role of architectural design if a multi-agent system is dynamically configured and embedded in equally dynamic human social organizations? Where is the boundary between requirements and design if agent behaviours are expressed in terms of goals and beliefs at both levels? What decisions are made at run-time by intelligent agents as opposed to at design-time by human designers? The exemplar should provide opportunities to explore these questions without being confined by traditional methodological concepts and terminology.

Real-world Richness, Extensible, Subset-able. The exemplar should reflect real-world concerns. For pedagogical purposes, examples are often simplified and stripped-down versions of real-world settings. As agent-oriented approaches could depart considerably from traditional approaches, the use of simplified examples could potentially introduce assumptions and biases specific to some particular agent-oriented approach (or class of approaches). Since agent-oriented approaches are supposed to be able to handle

richer and more demanding types of environments, the exemplar should offer sufficient richness for these capabilities to be demonstrated. For this reason, the exemplar also needs to be open-ended to allow further richness to be added. An open-ended real-world example can stimulate and inspire additional capabilities for emerging methodologies. On the other hand, an open-ended exemplar can grow unnecessarily and become unwieldy. Efforts will be needed by the community to refrain from adding extensions whose methodological issues are already well covered. For some methodological issues or activities, the exemplar should allow subsets to be defined for more focused examination and discussion.

A Representative Application Domain. To be concrete, the exemplar will necessarily be grounded in some application domain. It will not be easy to find a domain and application setting that can encompass all the relevant issues in a comprehensive way. Further, it should be easily understandable to non-domain-specialists. The general reader should be able to relate to the domain issues and ideally be able to imagine oneself as a stakeholder in the domain. Selecting a generally accessible domain could also leave room for research teams to do further elicitation with actual subjects in the domain. While grounded in a particular domain, the issues and scenarios should be easily imagined in other domains by analogy or extrapolation.

Allowing comparison with Non Agent-oriented approaches. For agent orientation to be widely adopted, its strengths in relation to more conventional approaches must be made easy to comprehend. The exemplar can play an important part here by facilitating and encouraging comparisons with non agent-oriented approaches. If the exemplar is presented only in terms of application domain issues, it would be open to any methodological approach. If one can work out the exemplar using different approaches side-by-side, one will be able to appreciate exactly where the benefits arise, what issues are addressable, supported by techniques and tools, etc. The comparison will therefore be at a fine-grained level. The decision to use software agent technologies, for example, could come about as a result of applying a methodology, rather than as an a priori decision before a project starts. A methodology could also lead to the choice of a mix of agent and non-agent software technologies, or a staged transition from one set of technologies to another. Even among agent-based technologies, there should be guidance from the methodologies on the choices and mixes of technology types, representing different kinds and degrees of mobility, intelligence, autonomy, coordination and communication capabilities, etc.

Neutrality With Respect To Agent Orientation. To succeed in the above objective, it should be clear that the exemplar is not biased in favor of or against any approach. This can be achieved by staying close to the application problem domain. While the exemplar should provide plenty of opportunities to showcase the benefits of agent orientation, those benefits (and limitations) should arise from the application of a methodology, and not inherent in the exemplar itself.

Prominence of Generic Software Engineering Issues and Criteria. While highlighting the variations among agent-oriented approaches, and their contrasts with non-agent-oriented approaches, one must keep in mind that all of these approaches must face the same generic issues that are faced by information systems development and software engineering in general. These should be prominent in the exemplar. For example, performance, costs, scalability, reliability, time-to-market, manageability of development project and team (including management of schedules, risks and priorities, manpower, knowledge and skills, training, learning curves, etc.), quality, maintainability, traceability, evolvability, and reusability of code and other project artefacts – all of these will continue to be central issues. Methodologies – agent-oriented or otherwise – can be differentiated on how (and how well) they address these issues.

Including Non-Technical Issues. Many of the issues crucial to the success of a system or project are non-technical, and they are intertwined with technical issues. The exemplar should provide sufficient complexity for these to be identified and explored. Real-life systems typically have many stakeholders with conflicting interests. There are political, economic, socio-cultural, legal, and regulatory issues. Commercial systems have competitors, as well as other players upstream and downstream along value chains. They operate under different business models, and go through occasional (and increasingly frequent) changes and upheavals. Consortiums, coalitions, and associations (within, among, or across industries, professions, consumer groups, government agencies, and international bodies etc.) might establish guidelines, set standards, and promote or resist change. Dominant players could set de facto standards. Once dominant players can go out of business, start-ups can become dominant players. Non-technical issues are also inherent within project organizations – differences and changes in organizational forms and styles, individual capabilities and limitations, local versus distributed teams, budget and schedule changes, team turnover and reorganizations, mergers and acquisitions, etc. – these are part and parcel of any real project. An exemplar can help illuminate how well the different methodologies perform under these complex, but real-life conditions.

3. A Health Care Domain Setting

We have chosen to use a domain setting from health care. We draw primarily from the proposal document [Szolovits 94] of a project called "Guardian Angel" (GA). The document, also referred to as the "manifesto", envisions a multi-agent information system infrastructure for health care which enables patients to have much greater control over their health information and care processes. [ga.org]

The Guardian Angel project aims to "construct information systems centered on the individual patient instead of the provider, in which a set of "guardian angel" software agents integrates all health-related concerns, including medically-relevant legal and financial information, about an individual. This personal system will help track, manage, and interpret the subject's health history, and offer advice to both patient and provider. Minimally, the system will maintain comprehensive, cumulative, correct, and coherent medical records, accessible in a timely manner as the subject moves through life, work assignments, and health care providers. Each GA is an active process that performs several important functions: it collects patient data; it checks, interprets, and explains to the subject medically-relevant facts and plans; it adapts its advice based on the subject's prior experiences and stated preferences; it performs "sanity checks" on both medical efficacy and cost-effectiveness of diagnostic conclusions and therapeutic plans; it monitors progress; it interfaces to software agents of providers, insurers, etc.; and it helps educate, encourage, and inform the patient. All this serves to improve the quality of medical decision-making, increase patient compliance, and minimize iatrogenic disease and medical errors". [Szolovits 94]

We believe that the GA project vision description can be fashioned into an exemplar that meets most of the requirements set out in Section 2. The domain setting offers plenty of opportunities for software agents to be introduced. The GA vision paints a detailed picture of how one might use software agents in this setting. The technology vision is described at a high level, so there is considerable room for alternative ways of defining requirements. In particular, the notion of software agent is quite open. As an exemplar it highlights the need for the methodologies to specify the appropriate agent notions at appropriate development stages. The primary focus is on the care relationship between the patient and the physician, which is to be enhanced by the GA software agent. There can be many other agents providing additional supporting, facilitating, mediating, monitoring or administrative roles. These other agents could be associated with the physician, the parent (of children patients), other care providers (nurses, administrators), or organizations (hospitals, agencies, etc.). Each of these could itself be made up of multiple agents. There can be interactions with many remote databases or knowledge bases. Some of the agents could be mobile. Numerous design alternatives exist. – how functions and responsibilities are distributed among agents, how they communicate and coordinate, etc. These are heavily constrained by issues of privacy, security, ownership and control, and desires for autonomy. Some portions of the project vision have been carried through to design and implementation.

For our purposes, we will focus on issues arising from the vision description. We are not interested in the particular systems and designs that might be best for GA. Instead, we ask what kinds of systematic and principled approach can guide us towards those designs and solutions. The rich descriptions of the domain setting therefore provide us with the material for our methodological studies.

The vision provides us with vivid depictions of likely scenarios. We have selected excerpts and itemized them for reference (in Section 4). The domain is readily comprehensible to the layperson. Readers may be able to relate some of the scenarios to personal experiences. Non-technical issues abound, often driving the technical ones. The core issue is a human social one – the desire for more patient control and autonomy. The technology could potentially improve health outcomes and lower overall costs. Legal, socio-cultural, political forces might affect how the technology develops and evolves.

Among software engineering issues, security and reliability will be prominent. The setting will bring out the capabilities of software agents in addressing them, as well as for many others such as scalability, performance, maintainability, etc. Although the vision is expressed in terms of software agents, much of the vision can potentially be achieved using more conventional software technology and methods. The extent to which different issues are addressed could be quite different. The contrast between agent-oriented approaches and non-agent-oriented approaches can then be examined in considerable detail, both in terms of the technologies and the development methodologies.

To take a neutral stance, any mention of software agent should be interpreted more generically as "software unit", with the choice of agent notions and capabilities to be determined by the methodology in

question. The application setting is illustrative and representative of large-scale information systems, with many stakeholders and many interacting systems and parts. It can be used to cover methodological issues quite extensively. The premise of patient-centered care is provocative and goes beyond the mere automation of existing practices. This helps to raise many questions and can lead to innovative solutions that may involve the technological as well as non-technological. We therefore believe it is a good vehicle for examining how well the different methodologies can guide us along these explorations toward realized systems.

4. Challenge Scenarios

This section presents the detailed scenarios for exploring methodologies. Each scenario presented in this section should be confronted with the different methodologies to see how well they can handle that scenario. The questions were built having in mind different phases that concerns software engineering. Thus, the questions aim to address aspects like requirements elicitation, design decisions, software architecture evaluation, coding, test generation and software evolution, though new methodologies could bring new interpretations to these software engineering concepts.

The scenarios (and later the questions, in Sec 5) are grouped into two different sets. The first one, represented as EAx.y – where x and y are sequenced numbers, would represent scenarios that pose questions pertinent to the software development process, while the second one, represented as EBx.y, would represent scenarios that pose questions pertinent to the evolution of the software, i.e. after the software is deployed. Scenario EA0.n depicts the general idea of the domain used to extract the scenarios. Most of the scenarios can be found there were directly extracted from the project report GA proposal document [Szolovits94] and are italicized. Others were added to tackle specific questions that we wanted to raise.

The General Scenario

EA0.0- Guardian Angel proposes, among other things, to shift the focus to the consumers, i.e. the patients, rather than its providers, i.e. hospitals, clinics etc. *One of the main reasons for changing the focus relies on the fact that nowadays when a patient moves or changes doctors or hospitals, it is most likely that many if not all of his or her medical records become effectively lost. The GA would store and integrate over a lifetime all health-related information about a patient. The software is not intended to be a simple repository of data; rather, it is intended to be proactive giving information and alternative solutions to patient's problems. Among other features, GA is supposed to engage in data collection interacting with the patient as well as with other devices as blood pressure meters or glucometers, to monitor progress of medical conditions, to help explain new facts to the patient, to allow patient to customize therapy plans, to understand patient's preferences and negotiate them with other systems e.g. scheduling appointments. It is supposed to be run in a small device such as a PDA that will in turn communicate with a home computer and other facilities like hospitals, clinics and laboratories to send appropriate data when necessary. The GA allows the patient to customize therapy plans with bounds established by care providers, giving the patient "ownership" of his or her therapy.*

EA0.1- Although envisioning in the long run that GA will be a routine health assistant to cover a wide range of diseases and all classes of patients, the GA will initially focus on patients with chronic conditions such as diabetes and high blood pressure problems. Therefore, the software architecture has to be flexible enough to allow new diseases and services to be added over the years.

Detailed Scenarios, During Development

EA1.0- *When hypertension is first suspected or noted during a routine examination the physician will load and activate the hypertension GA in the patient's computer.*

EA1.1- *The GA should assist the screening and the diagnosis process for the physician collecting any additional needed demographic information together with latest risk factors for myocardial infarction and stroke. Still, all information gathered in the clinics must be transferred to the patient's computer at home.*

EA2.0- *Once the patient and physician have established a management plan, the GA will help the patient follow the plan and monitor the progress.*

EA2.1- *For most patients, who are able to check their blood pressure with the home pressure cuffs now available, the GA will track the pressure against the expectations for therapy effect and use the data to provide correction and reinforcement for the patient.*

EA2.2- *The GA will also monitor other aspects of the plan including drug compliance, weight, and lifestyle changes.*

EA2.3- *The degree of monitoring of side effects and lifestyle changes such as diet and exercise will depend on the needs and desires of the patient. It could vary from simply making information available to having the patient fill out a progress report each session. Importantly, the GA will be able to ask about most of the potential side effects of the drugs.*

EA2.4- The *GA allows the patient to customize therapy plans with bounds established by care providers, giving the patient "ownership" of his or her therapy*. For example, the GA can use its knowledge of the patient's tastes and budget to suggest variations on the diet or changes to the amounts and types of exercises. On the one hand, the GA software wants to stay close to the original therapy plan to achieve best results and for safety. On the other hand, overly strict enforcement could result in the patient dropping out of the diet or exercise regime.

EA3.0- *Abby Kaye is a 14 year old girl who has had insulin-dependent diabetes mellitus for the last five years. Two years ago she began to measure her blood glucose and administer insulin without direct parental supervision. She has recently been enrolled as a participant in the Guardian Angel program.* Today, *Abby just come back from school upset because one of her friends teased her about becoming chubby. She steps on the bathroom scale and notes her weight; she has gained 10 pounds since her last doctor visit 4 months ago. She has already tried skipping her snacks but that leaves her feeling awful from hypoglycemia (and even hungrier) subsequently.*

EA3.1 *The GA_PDA is aware of several unexplained instances of hypoglycemia in the past month. The pattern of insulin dosing and recorded periods of exercise makes it most likely that these hypoglycemic episodes are due to skipped snacks. The GA_PDA makes a note to itself to use the opportunity of a query about diet to ask her at the end of the interaction whether she has been missing snacks and to warn her about the dangers. The GA_PDA is also aware of the heights and weights obtained at the doctor's office in the last two visits (and downloaded from the IHIS by the GA_hospital_process to the GA_home_computer and then to the GA_PDA) and recognizes that Abby has indeed an increased weight for height over the past visits. GA_PDA asks Abby if she would like to review her meals and snacks for the past days as well as go over her favorite foods.*

EA4.0- *Abby is uncertain what insulin dose to give this morning as she has a double session dance class at 10:00 and she remembers all too well that she has had mild hypoglycemic symptoms towards the end of even single session dance classes. She draws an exercise symbol spanning 10 to 11:30 on her daily schedule on the GA_PDA interface and then selects the Advise Dose icon. The GA_PDA informs her that she can either keep the dose unchanged if she thinks she can manage a double carbohydrate snack before the dance class or she can reduce her morning dose of insulin by two units of short acting (regular) insulin.*

EA4.1- *Two weeks later, Abby's morning blood sugars have continued to climb and they are in the mid 200's. Abby has not modified the GA_PDA default authorization to communicate with her parents' desktop computer. Therefore, when according to schedule, the weekly blood sugars are uploaded by the GA_PDA component to the GA_home_computer component this worrisome trend is displayed to the parents. However, to be compliant with personal feelings of some children the GA must be able to understand that if the parents are getting too many communications the GA must either start sending the communications directly to the physician or start an education program to enforce the need for addressing the problem. The GA_home_computer makes several suggestions as to how to modify the nightly NPH. If the parents do not feel comfortable following these suggestions, they are given several options for communications with the health-care providers including electronic mail with attached measurements, directions for paging or an emergency phone number. If they are comfortable*

with the suggestions, then the recent data along with the alert are uploaded during the routine daily modem connection of GA_home_computer with the GA_hospital_process running on the hospital's information system.

EA4.2- *Two months later: Abby's father realizes that next week Abby will be performing with her entire dance class for a school performance. However, this performance occurs the same day as a scheduled visit with Abby's endocrinologist. Abby's father goes to the weekly schedule view of the GA_home_computer and cancels the endocrinologist visit. The GA_home_computer asks him if he wants to reschedule (he does) and then negotiates with the hospital scheduling system (via the GA_hospital_process) two possible appointments for Abby in the coming month. After one is selected, the GA_hospital_process sends appropriate notifications to the hospital-based care providers.*

EA5.0- *Most common over-the-counter remedies for upper respiratory infections warn the patient to avoid them without consulting their physicians. GA should provide a selection of minimally problematic drugs and then monitor the patient's blood pressure response to those drugs and note the results for future reference.*

EA6.0- *After GA is in actual use, changes must not ever require a "reset".*

EA6.1- *Integrity of life-long records within a small portable computer that would implement GA should be assured. One can imagine catastrophes that would result in destruction of the device itself, loss or theft, temporary unavailability, accidental or deliberate alteration, etc.*

EA6.2 - *One could choose an infrequent (say monthly) backup tape for people with no serious current conditions. At the other extreme, a direct cellular call to a system providing guarantees of data integrity may be worthwhile when significant events occur in a patient being monitored for possibly life-threatening events. Fruitful synergies may also arise in such architecture. For example, a hospital may choose to go into the commercial business of providing data integrity for its patient's GA systems; in that case, communication costs can be reduced and reliability increased if the same telephone call can be used both to report time-critical data (part of the data repository and the alerting and notification functions) and to assure that it is backed up in case the GA device might malfunction (part of the backup function).*

EA7.0- *The long-term development of GA will require a highly flexible, open architecture that will support expansion of the concept and evolution of its implementation.* Once more, *changes must not ever require a "reset". In addition, it is absolutely critical to our ability to build the system that virtually all technology components of GA be provided by generic, preferably standards-based facilities. These include:*

- *data repositories and retrieval, perhaps based on the object-oriented data model*
- *reliable checkpointing and backup*
- *computer-based patient record*
- *security and authentication*
- *network-based communication*
- *alerting and notification, including pagers, electronic FAX, and email*
- *interfaces to data-capture instruments*
- *easy-to-use user interfaces*
- *formal expressions of guidelines, practice standards, etc.*
- *hypertext-based bodies of medical information*

EA8.0- The GA should interface with many instruments to measure glucose, prothrombim, cholesterol etc. Interface software for these kinds of devices may be commercially available. The GA should incorporate these software rather than develop its own interfaces.

EA9.0– As a result of competitive pressure, halfway through the project, project management decides that a scaled-down version of the GA must be produced for delivery in 3 months. It will support only one type of instrument and offers only a narrower role of advising the patient to change habits

Detailed Scenarios, After Deployment

EB1.0- One year after GA has been put on the market some parents were concerned about the level of freedom that GA gives to a child like Abby (See scenario EA4.0). The GA has now to be changed so it can handle new levels of freedom without prejudicing the existing one.

EB1.1- The GA may now allow parents to know right away when some relevant data are read from instruments. These relevant data must be configurable by each parent based on their experience with their children and should include not only actual readings but also previous readings so parents can establish some sort of profile that could warn them in time to prevent any harm.

EB2.0- Due to the big success on the use of GA, it was asked to develop a new version of GA that includes new features to interoperate with many image instruments like ultrasound, CAT scans, magnetic resonance and others. The GA should handle images to facilitate physicians' diagnosis.

EB2.1- Also, GA must now support the physician with possible alternative treatments for each disease accessing evidence-based medical knowledge bases over the Internet. Each treatment can affect differently each patient. For example, for some patients the use of drugs should be preferable to intense exercise and diet to treat diabetes since they would feel depressed soon and therefore give up the treatment. Hence, GA has to be able to trace a behavioral profile of the patient over the time and adapt its suggestions to this profile.

EB3.0- Despite the success of the GA, many people do not have access to it because it is too expensive to purchase and maintain. A new version of the GA will be produced that does not rely on a home computer. It will have the more limited function of supporting personal assessment. It will be able to read two different instruments (blood pressure meters and glucose meters) and will help patients to control safe levels for both tests.

EB3.1- GA will also keep a historical base of test results so the GA can alert for tendencies to high or low level of glucose and cholesterol.

EB3.2- Abby wants to spend a year abroad. The GA will now have to conform to international standards and national medical practices as well as government regulations.

5. Methodological Questions Suggested by the Scenarios

In this section we present a set of questions probe the capabilities and limitations of various methodologies. This set of questions is intended to be general enough to cover any methodology regardless how they tackle or name each of the traditional software engineering phases. As we do not want to bias towards or against any methodology, these questions are not grouped by any particular aspect in order to give freedom to anyone using the exemplar to answer the questions in ways best suited to each methodology. As this exemplar is intended to be an open-ended real-world example, we intend to keep improving this set of questions over time with input from the community. The full set of questions together with a list of methodologies can be found at: http://www.cs.toronto.edu/km/aometh/

Detailed Questions, During development

QA1 **Pro-activeness-** Reasoning about unexplained instances of hypoglycemia during the past few months and evaluating them towards the pattern of insulin dosing and the recorded periods of exercise leads the GA to conclude that the occurrences of hypoglycemia is most likely to be due to skipped snacks. The GA_PDA makes a note to itself to use the opportunity of a query about diet to ask Abby about possible snacks being missed (EA3.1). This scenario shows the software working in a pro-active manner that is quite important in this scenario, since it serves to alert the patient about a problem that he is not conscious about and therefore would never ask about. How would agent-oriented and non-agent-oriented approaches differ on the modeling and analysis of such cases?

QA2 **Human Autonomy vs software autonomy-** In scenario 4.1 Abby (a human being) has the autonomy to follow or ignore advices from the GA and to modify the GA-PDA authorization to communicate with her parent's desktop computer.. How would the software engineer handle this

autonomy using this methodology? How does one decide which decisions are to be made at design-time and which at run-time ?

QA3 **Autonomy reasoning-** The GA_home_computer and the GA_hospital_process agents may not come to an agreement about rescheduling the consultation (EA4.3). Which methodological constructs can support reasoning about this problem? Once agents can have autonomy (GA agents and human agents) how are their behaviours described and analyzed? Where does requirements end and design begin?*?For example, how would the agent interaction analysis in [Miles 01] compare with the approach in DESIRE [Jonker 00]?*

QA4 **Different levels of Abstraction-** How does the methodology supports navigating from the abstract levels of reasoning to the concrete one and vice-versa, *as in the recursive definitions of agency in Gaia and ADEPT [Jennings 00] or Level 0 and 1 in MESSAGE [Evans 01]?*

QA5 **Identifying participants in the domain-** In scenarios with many participants (e.g., EA1.0, EA2.0 and EA2.1), how can the methodology help identify participants such as the physician and the GA in the patient's computer? *Would the Universe of Discourse model from Cassiopeia [Collinot 96] help?*

QA6 **Capturing, understanding and registering terminology-** Terms like Hypertension in scenario EA1.0 or risk factors in scenario EA1.1 are not common knowledge. How would the methodology help understand the different terms? Does the methodology provide support for using ontologies? *Is there any vocabulary control as in Catalysis [D'Souza 99]?*

QA7 **Domain analysis-** GA involves complex social issues, how does the methodology support the modelling and reasoning about the social relationship involved in the above scenarios? How would they represent for example the fact that patient expects to have a plan to monitor his progress established by the physician as in scenario EA2.0? *Iglesias [Iglesias 99] mention that Mas-CommonKads [Iglesias 98] has an informal phase for collecting the user requirement using use cases and MSC (Message Sequence Charts) [Regnell 96], can the software engineer benefit from this? MESSAGE [Evans 01] offers Organisational and task goal view, how much would it help?*

QA8 **Finding requirements -** How does the methodology help in discovering and refining requirements? *For example, in MESSAGE [Evans 01] how does one arrive at the organization diagram and the task/goal implication diagrams*?

QA9 **Human-machine cooperation.** The diet and exercise scenario (EA2.4) illustrates how the GA might explore alternatives to help the patient achieve therapy goals while respecting personal preferences and life styles. How does the methodology help identify and analyze cooperative problem solving scenarios?

QA10 **Database design** GA implies the use of different and possibly distributed databases for accessing information like drug compliance, diets, exercises programs among others (EA2.2, EA2.3 and EA 2.4). How does the methodology determine the modes of interaction with these databases?

QA11 **Database evolution-** How would the methodology support the fact that these databases like drug compliance and diets, like those mentioned in EA2.2, EA2.3 and EA2.4 will be continuously evolving (EA0.1), eventually offering new features or even changing signatures and capabilities?

QA12 **Database design and legacy-** Risk factors for myocardial stroke (EA1.1) probably need to be retrieved from an existing database. How does the methodology support modeling and reasoning about legacy systems?

QA13 **Reasoning about different non-functional aspects-** Communicating with different databases like diets and exercise program that might be in different servers can be seen in EA2.3 and EA2.4. How does the methodology help determine the appropriate responses from the GA if one or more of the databases cannot be reached? *How would these appear in AUML [Odell 00] interaction diagrams?*

QA14 **Mobility-** Many different agents such as GA_hospital_Process ,GA_PDA, GA_Home_Computer and the physicians are depicted in EA4.0, EA4.1 and EA4.2. Many others could be involved like the physicians assistant software that must be aware of changes in patients' routine, test results and other important information. Moreover, consider that more then one physician may assess one single patient since some cases might demand different expertise. Yet many of these agents may not have an exact location as physicians might also be allowed to use PDAs. Software can also potentially migrate from one host to another. How does the methodology help reason about types and levels of mobility? How much intelligence should each agent contain? At what cost?

QA15 **User interface design-** Abby's double dance class scenarios (EA4.0, EA4.1 and EA4.2) indicate that many different softwares will have to interact with different users. How does the methodology lead to interface designs that respect the diversity of users?

QA16 **Generating test cases-** The "skipped snack" episode (EA3.1) involves many software artifacts interacting with its users and performing different tasks. Can the software engineer benefit from the methodology to generate test cases from models? Which level of tests case (integration, modular, systemic)? What would be tested in the above scenario?

QA17 **User interface Design-** The interaction between the GA_PDA and the patient, depicted in EA3.1 can be a key factor for the success or failure of the project. How would the methodology help software engineers on the user interface design?

QA18 **Architectural design and reasoning-** Scenario EA6.0 portrays an important aspect of the GA, it may never require a "reset" when changes have to be made. How the methodology supports software engineers on modelling aspects like this*? For example, DESIRE [Jonker 00] offers component managing how much it would help on that?*

QA19 **Eliciting and reasoning about Non-Functional aspects-** Possible catastrophes are mentioned in scenario EA6.1, how the methodology facilitates the elicitation of such requirements? *Can softgoals in **i*** [Yu 97] and Tropos [Castro01][Perini01] contribute to modeling them?*

QA20 **Architectural design and reasoning-** The need for flexible architectures that can support evolution is prescribed, among other things, in EA7.0, how the software engineer can model and reason about different architectures to support this scenario?

QA21 **Architectural design and reasoning-** Important aspects like communication costs and reliability are stated in EA6.2. These aspects may have different contributions to different software architecture. How the methodology would support reasoning about different architectures and the several aspects like the ones mentioned above?

QA22 **Validating specification over the life cycle-** Security, authentication alerting and notification are mentioned in EA7.0. How the methodology facilitates to validate if such aspects are being met during design, code and in the final product?

QA23 **Tracing changes in the requirements into design- -** The need for interfacing with a new database containing possible cross-reactions due to the use of a medication is introduced in EA5.0. It also introduces the need for reasoning how it will affect the physician assistant software since drugs prescription assistance would be affected. If scenario EA5.0 is elicited later, during the design phase, does the methodology provides any mechanisms to know where the design will be affected?

QA24 **Tracing changes from design to code-** Once the software engineer ends reasoning about changes in the design to support scenario EA5.0, does the methodology provide support to find the parts of the code that will be affected?

QA25 **Concurrency-** In Scenarios EA3.0 and EA3.1 we can see three different softwares (GA-PDA, GA_Home_Computer and GA_Hospital_Process) that will be necessarily operating concurrently, each one taking its own initiatives. How would the methodology support this requirement?

QA26 **Tracing back to requirements-** Suppose that during a design revision, the software engineer is in doubt if GA should really deal with allergies. Can this be traced back to requirements? How much rationale the methodology provides? Can one find out which stakeholder was responsible for pointing out the need for this requirement?

QA27 **Software Modularity-** The GA needs to be able to handle interfacing with instruments from several companies as depicted in EA8.0. How the methodology supports work in different groups? Can I buy the software for handling instruments interface from other companies as portrait in scenario EA8.0? *For example, can packages from AUML [Odell 00] help on that?*

QA28 **Formal Verification and Validation-** Does the methodology provide any means for formal verification and validation? Can the software engineer validate the different products throughout the software development process? To what extent can inconsistency be tolerated and managed?

QA29 **Project Management-** Suppose that at some point in the development process the project manager decides to give different directions to the project targeting a smaller number of requirements aiming at a faster delivery (EA9.0). How does the methodology help implement this change?

QA30 **Working in distributed teams-** Developing a software usually demands that more then one team be used so different goals can be achieved concurrently. The teams must coordinate activities and

exchange knowledge and ideas. How easy it is to partitioned the activities? How would the methodology facilitate it? What supports are there for forking and joining models for example?

QA31 **Tool support-** How much tool support does the methodology provide?

QA32 **Learning curve-** How easily can the software engineer learn the methodology and its tools?

QA33 **Integration with other methodologies-** How easily can the software engineer use other models together with the methodology?

Detailed Questions, After Deployment

QB1 **Flexibility to evolve / Reuse-** How the methodology would handle the evolution of the system portrait in scenario EB1.0? How easily and in which extent can the software engineer reuse existing design, code and test cases? *For example, although agent interaction analysis [Miles 01] claims that due to agent's characteristics used in the framework the alterations would minimally effect the rest of the system, it is not clear how one can be sure of that. It is not clear also how one agent will be affected and how easily it would be to evolve it.*

QB2 **Project Management-** The evolution requested in scenario EB1.0 will of course demand a certain amount of time and money to be made. How much the methodology helps on determining such issues and therefore to evaluate if the changes are feasible or not? Does the methodology help on establishing precise milestones? Can the software engineer establish a critical path in order to manage the project efficiently?

QB3 **Database Design-** Having to change existing database to comply with scenario EB1.1 might conflict with many of the non-functional aspects present in scenarios EA4.0 and EA4.1, how the software engineer handles that? What are the methodological constructs that can help?

QB4 **Can one address even more complex system-** Is the methodology able to scale up to a software with the complexity added by scenarios EB2.0 and EB2.1?

QB5 **Database design-** How the software engineer would handle the need for different schemas to represent the psychological profiles? What are the methodological constructs that might help on that?

QB6 **Evolvability and Product Lines-** If Abby decides to spend a year abroad as in EB3.2, how would the methodology support changing the GA to comply with new requirements such as variations among national standards?

QB7 **Lightweight versions of methodology for simpler problems-** How the methodology scales to simpler problems like the one stated in scenarios EB3.0, 3.1 and 3.2? Suppose the software engineer feels comfortable enough in the domain, can he use subsets or shortcuts to avoid overwork? *Can one for example choose not to use Organization diagrams from Message [Evans 01]?*

6. Related Work

Examples are indispensable for illustrating and explaining methodologies. A common pedagogical technique is to apply the same example to more than one methodology. For example, [Shehory 01] compares three agent-oriented modelling techniques using a small auction agent example. While small pedagogical examples can be powerful in their own way, they lack the richness of real-life examples that can stimulate researchers to explore issues in depth. A standardized example that a community of researchers can return to again and again for additional insight and inspiration has been called "exemplars". [Feather 97] has studied the challenge of designing suitable exemplars for requirements engineering.

Several works have surveyed and reviewed agent-oriented methodologies (e.g., [Iglesias 99], [Tveit01], [Wooldridge 01]). These are invaluable for orienting the reader to the increasingly numerous and diverse approaches being taken. The exemplar proposed in this paper aims to complement these works by challenging researchers and users of methodologies to do finer grained explorations and broader coverage of issues for each methodology. This will promote more detailed comparisons and more direct contrasts. A similar effort to compare with non-agent-oriented approaches (e.g., object-oriented) can be very illuminating.

7. Conclusion

We have proposed a detailed exemplar that we believe to be suitable for facilitating the exchange of ideas in agent-oriented methodology research. The real-life domain setting is intended to serve as common ground for the abstractions in various methodologies to be elucidated and compared in concrete terms. While the proposal of a challenge exemplar is only the first step, we have formulated a set of questions based on the example scenarios to stimulate the process.

The exemplar will most likely need many revisions and extensions. We hope that it can be useful to the community, and that all interested parties will contribute to its use, refinement, and reuse.

8. References

[Bass 98] Kazman, R., Bass, L., Abowd, G. and Webb, M. "SAAM: A Method for Analyzing the Properties of Software Architectures." In *Proceedings of the 16th International Conference on Software Engineering*. May 1994. Sorrento, Italy. 81-90.

[Castro01] J. Castro, M. Kolp and J. Mylopoulos. "A Requirements-Driven Development Methodology", In *Proc of the 13th International Conference on Advanced Information Systems Engineering CAiSE 0*1, Interlaken, Switzerland, June 4-8, 2001.

[Collinot 96] Anne Collinot, Alexis Drogoul, and Philippe Benhamou."Agent oriented design of a soccer robot team" In *Proceedings of the Second International Conference on Multi-Agent Systems(ICMAS-96)*, pages 41–47, Kyoto, Japan, December 1996.

[D'Souza 99] D'Souza, D.F and A.C. Will. "*Objects, Components and Frameworks With UML: The Catalysis Approach"*. Addison-Wesley 1999.

[Evans 01] Evans, R. et all "Message: Methodology for Engineering Systems of Software Agents" Technical Information, Project P907, http://www.eurescom.de/~pub-deliverables/P900-series/P907/TI2/ p907ti2.pdf

[Feather 97] Feather, M., Fickas, S., Finkelstein, A., van Lamsweerde, A. "Requirements and Specification Exemplars". *Automated Software Engineering*. 4(4) 1997.

[Iglesias 99] Iglesias, C.A. and González, J.C. "A Survey of Agent-Oriented Methodologies" In *Proceedings of the 5th International Workshop on Agent Theories, Architectures and Languages* (ATAL'98), LNAI n1555 - Springer Verlag, Paris, France, July 1998, pp:317-330.

[Iglesias 98] Carlos A. Iglesias, Mercedes Garijo, Jos´e C. Gonz´alez, and Juan R. Velasco "Analysis and design of multiagent systems using MAS-CommonKADS" In *AAAI'97Workshop on Agent Theories, Architectures and Languages, Providence, RI, July 1997*. ATAL. (An extended version of this paper has been published in *INTELLIGENT AGENTS IV: Agent Theories Architectures, and Languages*, Springer Verlag, 1998.

[ISPW 93] Proc. Of 8th International Software Process Workshop, 1993.

[Jennings 00] Jennings, N.R., Faratin, P., Normam, T.J. O'Brien, P. and Odgers, B. "Autonomous agents for Business Process Management" Int. *Journal of Applied AI* 14(2), 2000, pp:145-189.

[Jonker 00] Jonker, C.M., Klush,M., Treur, J. "Design of Collaborative Information Agents" In M.Klush and L. Kerschberg (eds.), *Cooperative Information Agents IV, Proc. Of CIA 2000.* Springer, pp:262-283.

[ga.org] The Guardian Angel Web Site - http://www.ga.org/ga

[Miles 01] Miles, S., Joy, M., Luck, M. "Designing Agent-Oriented systems by Analysing Agent Interactions" In *Proceeedings of the First International Workshop on Agent-Oriented Software Engineering (AOSE-2000)* held at the 22nd International Conference on Software Engineering, Limerick, Ireland, 2000.

[Odell 00] Odell, J., Parunak,H.V.D., Bauer, B. "Extending UML for Agents" In G. Wagner, Y. Lesperance, and E. Yu, editors, *Proceedings of the AgentOriented Information Systems Workshop (AOIS) at the 17th National Conference on Artificial Intelligence,* pages 3--17, Austin, Texas, 2000.

[Olle 82] Olle,T. "Comparative Review of Information Systems Design Methodologies - Stage 1: Taking Stock," in T. Olle, H. Sol & A. Verrijn-Stuart (eds); *Information Systems Design Methodologies: a comparative review,* Proc. IFIP WG8.1 CRIS 1, North-Holland, pp. 1-14.

[Parnas 72] Parnas, D.L. "On the Criteria To Be Used in Decomposing Systems into Modules" Reprinted from *Communications of the ACM*, Vol. 15, No. 12, December 1972 pp. 1053 - 1058 Copyright © 1972, Association for Computing Machinery Inc

[Perini 01] A. Perini, P. Bresciani, F. Giunchiglia, P. Giorgini, J. Mylopoulos. "A Knowledge Level Software Engineering Methodology for Agent Oriented Programming". In *Proc. of the Fifth International Conference on Autonomous Agents, Montreal,* Canada, 28 May - 1 June 2001.

[Regnell 96] Bjorn Regnell, Michael Andersson, and Johan Bergstrand. "A hierarchical use case model with graphical representation" In *Proceedings of ECBS'96, IEEE International Symposiumand Workshop on Engineering of Computer-Based Systems*, March 1996.

[Shehory 01] Shehory, O. and Sturm, A. "Evaluation of Modeling Techniques for Agent-Based Systems" in *Proc. Of bar-ilan international symposium on the foundations of artificial intelligence*, Israel Jun 2001.

[Szolovits 94] Szolovits, P., Doyle, J., Long, W.J. "Guardian Angel:Patient-Centered Health Information Systems" Technical Report MIT/LCS/TR-604, http://www.ga.org/ga/manifesto/GAtr.html

[Tveit 01] Tveit, A. "A Survey of Agent-Oriented Software Engineering" NTNU Computer Science Graduate Student Conference, Norwegian University of Science and technology, 2001.

[Wing 88] J.M. Wing, ``A Study of Twelve Specifications of the Library Problem," *IEEE Software*, vol. 5, no. 4, July 1988, pp. 66-76. Runner-up for 1988 *IEEE Software* Best Paper Award. Also CMU-CS-87-142, July 1987.

[Wooldridge 01] Wooldridge, M. and Ciancarini, P. "*Agent-Oriented Software Engineering: The State of the Art*" In Handbook of Software Engineering and Knowledge Engineering. 2001, World Scientific Publishing.

[Yu 97] Yu, E. "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering" *in Proc. Of the 3rd IEEE Int. Symp. on Requirements Engineering*, pp:226-235, 1997.