RESOLVING ARTIFACT DESCRIPTION AMBIGUITIES DURING SOFTWARE DESIGN USING SEMIOTIC AGENT MODELLING

Daniel Gross and Eric Yu
Faculty of Information, University of Toronto, 140 St. George Street, Toronto, Canada

Keywords: Semiotic Agents, Software Design, Design Modelling, Software Architecture.

Abstract: For software designers to effectively collaborate, they must share an understanding of how software design artifacts contribute to the execution of software processes (the artifact's operational meaning). However, during design, artifact descriptions often lack sufficient detail to unequivocally establish operational meaning. This is because during design, artifact descriptions are initially usually first-cut, and are then successively refined to include additional design details, until the operational meaning of artifacts can be unequivocally demonstrated. Ensuring shared meaning in larger projects is particularly difficult because of the plethora of interrelated artifacts designers deal with, and because the design details in descriptions of different artifacts can vary greatly. In this paper we argue that a semiotic meaning analysis supports clarifying the operational meaning of artifacts during software design, and can help in identifying whether and in what way artifact descriptions must be further elaborated. We further argue that clarifying the operational meaning of artifacts is closely intertwined with design decision-making. Adapting an existing semiotic agent modelling approach, we propose an approach to capturing the evolving operational meaning of artifacts during software design and decision processes, and illustrate the approach with examples taken from a large design project at an insurance company.

1 INTRODUCTION

To collaborate during software system development, designers must discuss software designs. To support effective and efficient design discussions, software designers use specialized terminology and/or conceptual models (e.g., (Gamma et al., 1995, Fowler and Scott, 2000)). The assumption is that these terms and models efficiently and unequivocally communicate design information amongst designers, and help prevent misunderstandings. But does this assumption hold?

Consider the following excerpt from a design discussion reported by an enterprise architect at an insurance company. The enterprise architect is responsible for the overall enterprise architecture of enterprise systems, whereas a number of designers are responsible for the design of individual system components. The enterprise architect asks that a consumer component designer utilizes an *asynchronous messaging* approach to sending insurance policy data from a consumer to a provider, while the consumer component designer argues for a *synchronous messaging* approach. *Consumer*, *provider* and *messaging* are terms taken from the service-oriented architecture (SOA) design style (Erl, 2007, Josuttis, 2007). Broadly speaking, SOA is a distributed system design approach, whereby consumers request, through a messaging infrastructure, computational services from, usually remote, service providers (Erl, 2007, Josuttis, 2007).

Figure 1: High level service-oriented architecture in Business Enterprises.

Figure 1 depicts a simplified schematic illustration of how SOA is usually applied in business enterprises. The figure shows consumer and provider components, as well as a messaging infrastructure component, often called an enterprise service bus (ESB). Components are shown using rectangles. The double sided arrows between components refer to sending and receiving messages between components. We assume the enterprise

Yu E. and Gross D.
RESOLVING ARTIFACT DESCRIPTION AMBIGUITIES DURING SOFTWARE DESIGN USING SEMIOTIC AGENT MODELLING.
DOI: 10.5220/0003270200770086
In *Proceedings of the Twelfth International Conference on Informatics and Semiotics in Organisations (ICISO 2010)*, page
ISBN: 978-989-8425-26-3

architect and system designers have a good understanding of these SOA concepts, including synchronous and asynchronous messaging, before embarking on this project.

The consumer designer prefers to directly request a necessary service from a specific provider, and to deal with an immediate response from the provider. This results in a synchronous style of messaging. From the consumer designer point of view, this is the simplest design which accomplishes the requirements. The enterprise architect explains that requesting a service directly from a provider harms future extensibility of the consumer component. This is because it limits the consumer to that specific provider to process its service requests. If, in the future, other providers need to process the consumer's service request, the need would necessitate changes in the consumer component. The enterprise architect therefore advocates asynchronous messaging, which would result in a loosely coupled design. In contrast the consumer designer's approach would be referred to as point-to-point integration, which results in tight coupling.

Reflecting on this example, we make the following observations:

(1) The terms "synchronous messaging" and "asynchronous messaging" compactly refer to the use of a number of design artifacts, which respectively implement different approaches for integrating enterprise systems.

(2) The collective software behaviour of these artifacts is the intended "operational" meaning that the enterprise architect implies when using these terms during the design discussion.

(3) For a system designer to clearly understand the meaning that the enterprise architect attributes to these terms there must be agreement on the operational meaning.

(4) The design discussion between the enterprise architect and the system designers occurs at a particular level of abstraction. This influences the kind and number of artifacts that are chosen during the discussion that describes operational meaning.

(5) To achieve effective communication, the enterprise architect and system designer may need to understand each other's design situation, particularly, the different demands that each of them faces and these demands may well be in conflict. A design situation includes objectives, constraints, solution alternative, artifacts, evaluations, and the like.

We now review these observations in more detail, and explain why the use of specialized terminology and/or notation is usually not sufficient to support unequivocal understanding among designers during design discussions.

Consider the first three observations. A specialized term, such as "*synchronous messaging*", translates in the enterprise architect's mind into a number of artifacts that collectively exhibit some software system behaviour. Usually, such a translation is not mechanically derivable, but involves interpretation and decision making. For example, the enterprise architect explains that during synchronous messaging the consumer knows and makes use of the physical address of the provider to send a service request to the provider. However, as we will demonstrate, the term synchronous messaging allows for an alternative operational meaning, and does not necessarily involve such a physical point-to-point integration. During the design discussion the consumer designer may have this other alternative operational meaning in mind. Shared operational meaning of the terms synchronous and asynchronous messaging is thus problematic.

Consider now the fourth observation – levels of abstraction. The meaning attributed by the enterprise architect to the point-to-point integration style involves artifacts at a particular level of abstraction. For example, the enterprise architect may have thought of the address artifact of a second system as a physical address, such as a static IP address, that unequivocally identifies a first and a second system in a network. Alternatively, the enterprise architect may indeed have considered a logical address artifact that allows for some routing decisions within the ESB, thereby "loosening" the operational meaning of point-to-point integration style, but still objecting to this approach. Clarifying the level of abstraction at which a design discussion takes place is thus crucial for understanding operational meaning and for clarifying design intents.

Finally, consider the fifth observation. During a design discussion both the enterprise architect and the system designers are actively involved in the design of enterprise systems, but from different vantage points. The system designer is responsible for the design and goal achievement of a single system, while the enterprise architect is responsible for enterprise-wide goals. This includes also dealing with single systems, however from a systemic point of view, such as in relation to other systems. Each designer therefore comes to the design situation with different objectives in mind, which can conflict, and different design intents may lead designers to interpret design approaches quite differently.

In this example, the design intent of the enterprise architect to avoid a point-to-point integration style relates to the additional development effort needed to support a service request from the consumer to be processed in the future by additional providers. This intent leads the enterprise architect to focus on ensuring that the provider address is not directly known to the consumer component when sending the service request message. However, suppose the consumer designers thinks that a main concern of the enterprise architect is, instead, to avoid the (symbolic) inclusion of the provider service interfaces within the consumer code, since this can create a costly syntactic and semantic dependence between a provider and its consumer components, when the provider's interface is changed in the future (interface has a broad meaning, including knowledge of a messaging protocol between a consumer and provider, and which can greatly vary for synchronous messaging). The operational (and structural) meaning that the consumer designer then attributes to the point-to-point integration term, would then focus on artifacts embodying interface knowledge within the consumer, rather than on a physical or logical provider address. Knowing each other's design demands may thus help clarify what artifacts and behaviours are involved in the operational meaning of terms.

The core problem we identify is the need to clarify what designers mean when they use specialized terminology during discussions or in conceptual models, while acknowledging that meaning is constructed from interpreting the operational meaning of existing artifacts in the design space (e.g., consumer, provider, etc.), and from interpreting operational meaning of new artifacts the designers envision included in the design space (e.g., consumers' and providers' messaging routines). Searching for an appropriate theoretical approach to capturing and analyzing meaning of design artifacts during design leads us to consider semiotics in general, and the semiotic agent modelling approach developed by Stamper, Liu and colleagues for dealing with meaning during information system requirements and design (Liu, 2000, Stamper, 1973, Stamper, 2006, Stamper et al., 2003, Liu et al., 2001) in particular. More specifically, we adopt Stamper's actualism ontology that accepts as assumption that knowing depends on a knowing (semiotic) agent, and that knowledge depends on the actions afforded by the agents perception (Stamper, 2006, Michaels and Carello, 1981, Gibson, 1977). We further adopt Stamper's ontological dependence schemas which applies these philosophical notions in the form of a semantic analysis approach using semiotic agents (Liu, 2000, Stamper, 2006).

We extend ontological dependence schemas to software design. We define a new logical relationship between semiotic agents (a "modifies" link), that supports defining new agents by extending existing agents and their ontological dependence schema; we distinguish between regular semiotic agents, and symbolic semiotic agents which refer to "symbol processing" (computational) artifacts designed by semiotic agents; we specialize affordances (the "things" and "actions" agents perceive and do) to distinguish between substantive (design application domain specific) and symbolic affordances (such as software operations). Finally, we support linking ontological dependence schemas to agent and goal-oriented analysis models that support capturing and reasoning about intents and decision making in development organizations (Gross and Yu, 2001a, Gross and Yu, 2001b, Gross and Yu, 2010, Yu, 1994a). This last contribution is however only briefly discussed and not illustrated in this paper. Note that the term "agent" appears in semiotic work (semiotic agent) as well in our work on intentional modelling and analysis in organizations (intentional agent). These agents have different meanings and application, and we briefly discuss these in the discussion section.

The next section illustrates our proposed approach through a semiotic analysis of the design discussion between the enterprise architect and the designer of the software component. Section 3 discusses our approach and related work, while section 4 concludes and points to future work.

2 SEMIOTIC AGENT MODELING OF AN ARCHITECTURAL DECISION

2.1 Semiotic Agent Modelling of Terms

Affordance is a central concept during semiotic agent modelling. Generally speaking, one "thing" or concept affords another, if the first helps the second in some way. For example, an *ink pen* affords *writing*. Gibson elaborates that affordance must be understood in relation to an agent, or more precisely, the perceptive capability of the agent (Gibson, 1977, Michaels and Carello, 1981). It is the writer who can

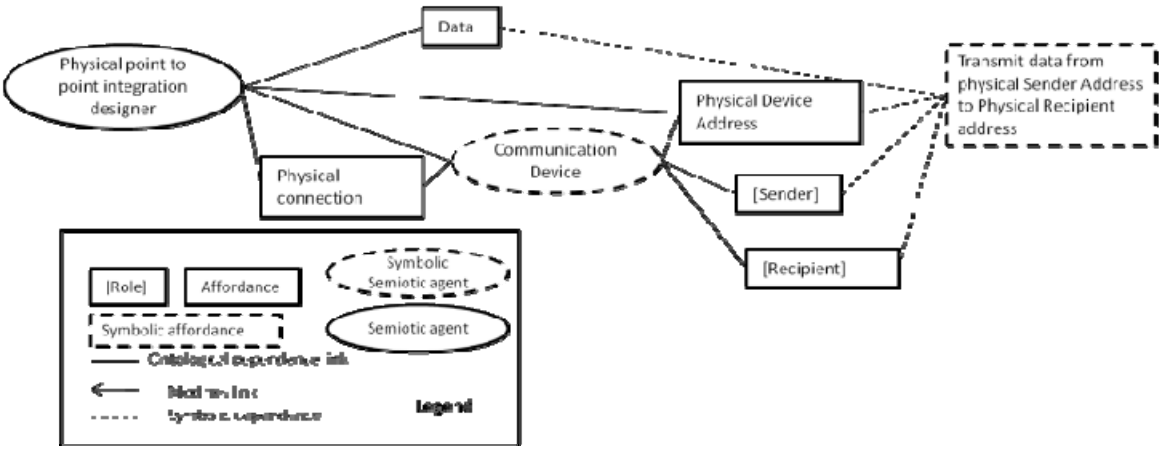

Figure 2: Semiotic analysis of point-to-point integration style.

perceive the affordance, and thus can understand the meaning of an ink pen for writing. Stated differently, the meaning of the concept "ink pen" is operational ("for writing"), and is a perceived affordance, and its perceptibility is dependent on the writer's writing capability. In short, perceived affordance is perceived operational meaning. A person who cannot write, nor has seen writing before, would not perceive the pen's affordance for writing, and thus fail to understand the ink pen's meaning (that it is for writing). An agent's ability to perceive affordances are "preconditioned" by what the agent is in principle capable of sensing and doing, and by learned experiences of past encountered affordances in the agents living environment. In Gibson's theory of affordance, the relationship between environment and perceptive capacity of an agent is interrelated, the living environment of an agent affords the evolution of capabilities in the agent, and the agent's evolved capabilities afford perceived affordances in the environment. According to Gibson, it is possible that some objects that have no perceivable affordances to an agent, would not be perceived, and thus do not exist for that agent.

Stamper's ontological dependence schema captures such ontological necessity between affordances. For instance, that "person stumbling" can only exist if "person running" or "person walking" first exists; without these affordances "stumbling" is not perceivable by person, and hence does not exist. A semiotic agent analysis of the term "physical point-to-point integration" attempts to identify, and capture in the form of an ontological dependence schema, affordances that are necessary for a software system designer to perceive actions or operations relevant to "physical point-to-point integration", such as the physical point-to-point transmission of data, and the designing of a physically point-to-point integrated consumer component. Note, that we distinguish between "action" directly performed by the agent and "operation" performed by a software system.

In our proposed approach a semiotic agent indicates a design capability and perceptive vantage point of a designer. For example, by naming a semiotic agent "Physical point-to-point integration designer" we indicate a designer's capability of perceiving all affordances necessary for defining or performing any or all actions or operations afforded by a physical point-to-point integration. The name of a semiotic agent captures a modelling intent to explore and delineate the name's meaning, by identifying the indicated affordances. For instance, the adjective "physical" in the above semiotic agent's name focuses the meaning analysis to physical affordances, while limiting the scope of the analysis to exclude the capability of perceiving a "*logical point address"*, and anything this concept affords, which is left for a different semiotic agent to perceive.

In our proposed approach semiotic agents are therefore a structuring mechanism for software system design terminology in terms of afforded design capabilities and actions, which parallels how design responsibilities are identified and allocated amongst designers in development organizations.

2.2 Semiotic Agent Analysis of Point-to-point Integration

Figure 2 illustrates the result of a semiotic agent analysis of *physical point-to-point integration*. The semiotic agent model is read from left to right. Elements more to the left afford elements more to

the right. Elements are linked via ontological dependence links. Captured on the far left using a solid ellipse is the semiotic agent "Physical point-to-point integration designer". The semiotic agent represents a human designer who is capable of designing a physical point-to-point integration. The agent is therefore capable of perceiving the affordances for any or all automated operations as well as relevant designer actions applicable to a physical point-to-point integration.

Figure 2 illustrates one key operation: "Transmit data from a physical Sender address to a Physical Recipient address". This automated operation is captured on the far right using a dashed rectangle indicating a symbolic affordance -- a software operation. Since it's a symbolic operation, it refers to its dependee affordances using symbols; therefore the affordances it directly depends on are linked via symbolic ontological dependence links (dotted lines). Note that as a result of semiotic analysis the label of this operation is more precise than the one we used before ("physical point-to-point transmission of data"), including additional terms derived from its preceding affordances.

"Data" and "Physical connection" are affordances (captured as sold rectangles) that help define the automated operation. Figure 2 further shows that a "Physical connection" affords the "Communication device". The "Communication device" is from the perspective of the semiotic agent perceived as a *symbolic semiotic agent* – an agent that performs automated symbolic operations. In other words it's a computational device that processes software code. The symbolic nature of this agent is indicated by an ellipse with dashed lines. Having a "communication device" affords a "Sender" and a "Receiver" role for the Communication Device, as well as a "Physical Device Address."

The ontological dependence schema in figure 2 indicates that all these affordances are necessary for defining the "Transmit data from a physical sender address to a physical recipient address" operation. Removing any affordance included in the schema should make defining the operations impossible. If this is not the case, then the semantic agent schema is incorrect and needs to be revised.

2.3 A Fuller Semiotic Agent Analysis of Discussed Terms

In the introduction we mentioned the Enterprise Architect's preference for asynchronous messaging, because it involves loose coupling and avoids point-to-point integration, while the consumer component designer prefers synchronous messaging, since point-to-point integration has some properties of advantage to him (e.g. simplicity). In this section we present a fuller semiotic agent analysis of these terms, and illustrate that the exact operational meaning of *synchronous* and *asynchronous messaging* is in fact independent of the operation meaning of *point-to-point* or *loosely coupled integration*.

This analysis reveals that the architect and the consumer designer seem to conflate "orthogonal" operational meanings when discussing synchronous and asynchronous messaging in terms of point-to-point and loosely coupled integration. It is, for instance, possible to transmit messages asynchronously and point-to-point, and it is possible to offer loose coupling and synchronous messaging. Conflating operational meaning may involve implicit, and unintended, design decision making. Semiotic agent modelling helps ensure that such conflations and related decisions are made visible and amendable to analysis, and not made unintentionally, which in turn helps avoid misunderstandings.

Figure 3 shows how ontological dependencies of "higher level" terms, are selectively composed from ontological dependence schemas of "lower level" terms. For example, to construct the operational meaning for the higher level term "Logical loosely coupled integration", we define and link the semiotic agent "Logical loosely coupled integration designer", via a "modifies link" to the lower level "Physical point-to-point integration designer" agent, we defined earlier. This indicates that everything the Physical point-to-point integration designer perceives is also perceived by a Logical loosely coupled integration designer. The Logical loosely coupled integration designer however perceives more, and can perceive affordances relevant to a logical integration, such as, to perceive the notion of a "Logical device address". The "Logical device address", together with the "Sender" and "Receiver" roles (of the Communication device), affords defining the operation "Transmit data from logical sender address to logical recipient address" and "Translate logical device address to one or more physical device addresses" (the latter is afforded by including the "Physical device address" affordance also). Without specifically restricting the meaning of the logical device address (by placing it in context of additional gents and affordances), it can be interpreted quite generally, allowing for different kinds of loose coupling. Figure 3 further illustrates

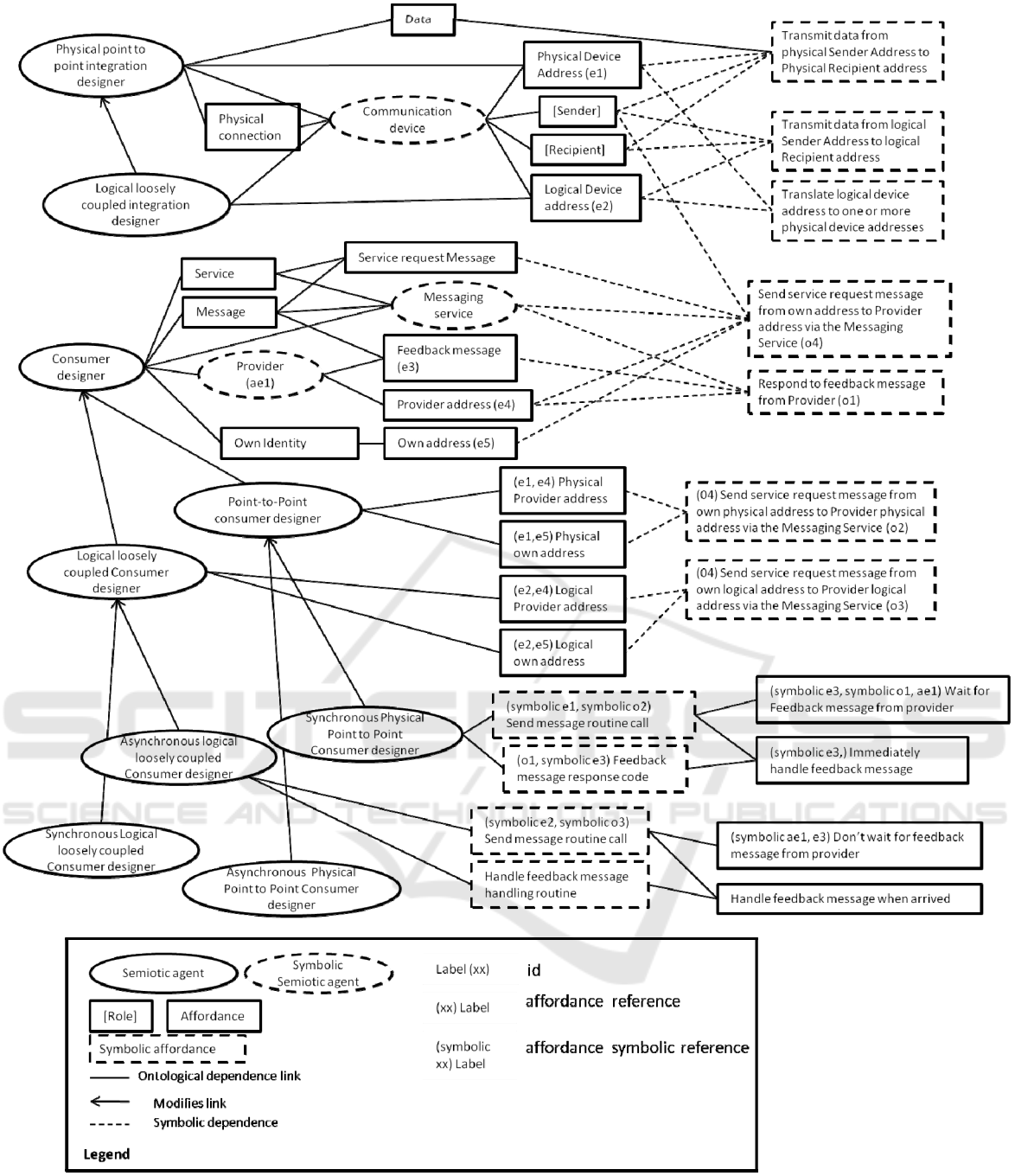

Figure 3: Fuller semiotic agent analysis.

the ontological dependences for the SOA term "Consumer". The "Consumer designer" agent perceives "Service", "Message", "Provider", and "Own identity". Selections of these in turn afford perceiving "Service request message", "Messaging service", "Feedback message", "Provider address" and "own address". Finally, with these affordances defined the operations "Send service request message from own address to provider address via the Messaging Service" and "Respond to feedback message from Provider" are defined. Note that to reduce link clutter in the diagram, we capture some ontological dependence links by a simple identification and referencing scheme. A bracketed identifier, such as (e1), appended to a label uniquely identifies a model elements, while a bracketed

identifier prefixed to a label indicates an ontological dependence link on the element referenced. For example, the affordance "Logical Device address (e2)" is uniquely identified by the identifier "e2". The affordance "(e2, e4) Logical Provider address" indicates that the logical provider address is ontologically dependent on "Logical device address (e2)", and "Logical provider address (e4)". Including "symbol" in a prefixed bracketed identifier indicates a symbolic ontological dependence link.

After defining the semiotic agents and respective ontological dependence schema for "physical point-to-point integration", "Logical loosely coupled integration designer", and "consumer", we construct higher level semiotic agents. The "Physical point-to-point Consumer designer" agent is a modification of the "Consumer designer" agent, and defines the terms "Physical Provider address", "Physical own address" and the operation "Send service request message from own physical address to Provider physical address via the Messaging service". We similarly define the "Logical loosely coupled consumer designer" agent.

Stepping up another level we modify these agents to define two synchronous and two asynchronous messaging related semiotic agents, the synchronous physical point-to-point consumer designer, the asynchronous physical point-to-point consumer designer, and synchronous logical loosely coupled consumer designer and finally, the asynchronous logical loosely coupled consumer designer. The ontological dependence schema for synchronous messaging indicates that with respect to the consumer designer, synchronous messaging is defined by a designer action (captured by a solid rectangle) embodied in the manner software code is strung together so that when executed the consumer waits for a feedback message from the provider, and then immediate handles the feedback message.

A careful analysis of the ontological dependences of these designer actions indicates that they are afforded by any consumer agent (and selected consumer's affordances), including all agents that modify the consumer agent (the modification link, acts with this respect like an ISA relationship in object-oriented analysis (Fowler and Scott, 2000)). Synchronousity is thus applicable to Physical point-to-point consumer designers and to logical loosely coupled consumer designers. This is also the case for Asynchronousity. To establish this, we trace back the affordances of the designer action and note the semiotic agents whose affordances are required. We also perform a "factoring" analysis, a type of analysis afforded by the "modifies" relationship we introduced, to identify for some affordances lower level affordances that can be "factored out", to leave only higher level affordance in the ontological dependence chain.

For example, tracing back from the "Wait for feedback message from provider" designer action of the "Synchronous Physical Point-to-point Consumer designer" agent, we find that it is afforded by "Feedback Message", "Response to feedback message from Provider", "Provider", "Feedback message response code", "Send message routine call", "Send service request message from own physical address to Provider physical address via the Messaging Service", "Message", "Service", "Physical Provider address", "Physical own address", "Provider address", "Own address ", "Physical Device Address", "Communication Device" and "Physical Connection".

If we trace back from the "Don't wait for feedback message from provider" we find that affordances found differ only with respect to the following alternatives: "Logical Provider Address" vs. "Physical Provider Address" and "Logical own address" vs. "Physical own address". If these different preceding affordances are not relevant with respect to the affordances factoring analyzed (the wait and not wait for feedback message from provider operation), then the differences can be factored out and subsumed into a relevant higher level affordance such as "Provider Address" and "Own address", which then also removes all lower level dependent affordances. This leaves a common higher level "thread" of dependent affordances, which in our case, are link to the consumer designer semiotic agent only. Factoring analysis also helps in identifying restructuring opportunities of agent models to making them more concise by introducing, moving, redefining and and/or relinking affordances.

The semiotic agent model in figure 3 therefore reveals that whether a consumer is involved in a direct point-to-point integration and uses a provider physical address, or whether it is loosely coupled using a logical or symbolic address, has no bearing on the operational meaning of synchronous and asynchronous messaging. Equipped with such a semiotic agent analysis, architects and designers can clarify what exact operational meaning they have in mind during design discussion, and make explicit what level of abstraction they assume unproblematic, which facilitates architectural design and reasoning, and helps avoid misunderstandings and unintended design decisions.

3 DISCUSSION AND RELATED WORK

Misunderstandings and ensuing decision errors during architectural design carry a cost. While it is difficult to provide an empirical answer to what the cost is, mainly due to the difficulty in collecting relevant data in industry to study these costs (Westland, 2002), there is empirical research available that does provide some indications. Westland for instance found that errors that require significant system redesign generate significant costs, and that unresolved errors become exponentially more costly with each phase in which they are left unresolved (Westland, 2002). It is therefore clear that architectural design errors can become very costly. They do require significant system redesign to correct, and do occur at early phases during software development. It thus appears justified to perform a semiotic analysis during architectural design to reduce misunderstandings amongst designers and developers.

A key question, however, during semiotic agent analysis is how much analysis to perform. How many levels of semiotic agents to explore and how fine grained to develop the affordance dependence structures. Ultimately, we believe that the answer to these questions is subjective, and it is up-to the designers involved in semiotic agent analysis to decide when they feel operational meaning has sufficiently been clarified. However, the semiotic agent concept provides a useful focal point to guide designers, while making these subjective decisions. Using semiotic agents the question "how much semiotic analysis" is transformed to the question: to what extent to rely on the experience and know-how of a designer (captured as a semiotic agent) to interpret the meaning of terms, and whether it matters that the agent may invoke different interpretations.

For example, in figure 3 the "Consumer designer" agent perceives a "Service" affordance. However, what is the exact meaning of a "Service"? We have chosen not to analyze that further, and rely on the consumer designer's knowledge and skills to interpret the meaning for us. This choice is further strengthened by our judgment that with respect to analyzing the operational meaning of synchronous and asynchronous messaging, the exact operational meaning of a "Service" is inconsequential. It is of course possible that our judgment is incorrect, however, semiotic analysis affords these kinds of judgment questions and guides when and for what agents and terms to ask them.

Stamper suggests constructing dependence schemas using strict rules, such as that every affordance, apart from the root, depends for its existence on at most two antecedent affordances (Stamper and Ades, 2004). A schema complying with these rules is called in canonical form, which, based on empirical observations, helps expose errors and reduce semantic confusion. The diagrams in this paper are not in canonical form. As we gain more experience with this notation the utility of the canonical form will also become better understood.

The proposed approach should be seen as complementing other design approaches. For example to support terminology use during UML modelling and analysis (Fowler and Scott, 2000). Future work will also look at integrating this approach with an architectural design reasoning and analysis approach (Gross and Yu, 2010).

Besides operational meaning we also distinguish between the structural and intentional meaning of artifacts. Structural meaning relates to a physical or conceptual structure an artifact contributes to. For example, a "Service" offered by a provider affords a "Service Interface", which is a structuring concept in software architecture, and which is, in turn, symbolically referred to by a consumer. The enterprise architect could have the distribution of the providers interface structure to consumer components in mind when discussing point-to-point integration. However, ultimately, every structural meaning leads to operational meaning, such as "what does a service interface, or symbolic reference to an interface afford to do?" The structural meaning of an artifact may be a convenient intermediate concept when capturing artifact meaning.

The intentional meaning of an artifact refers to a higher level purpose of the artifact's operational and structural meaning. For instance, intentionally, asynchronous messaging affords systems that are scalable and easier to extend. Scalability and ease of extension are the intentions that justify the decision to adopt asynchronous messaging. Ultimately, however, also intentional meaning leads to designer actions. The intentional meaning of an artifact offers a bridge between the semiotic agent work and our prior work on intentional agents and decision making (Yu, 1994b, Gross and Yu, 2010).

For example, the semiotic analysis in figure 3 illustrates the operational meaning of synchronous and asynchronous messaging and shows that either one is possible when adopting point-to-point integration or more loosely coupled integrations. However, which of these should, say, a consumer designer choose, and how should a consumer

designer justify his/her choice. These questions lead us to the intentional meaning of artifacts, and intentional modelling and analysis of the artifact in terms of intentional agents in organizations (Yu, 1994b, Gross and Yu, 2010). The exact relationship between intentional agents, who reason about design intents and decision making, and semiotic agents, who perceive operational meaning, needs more research.

As outlined in the preface, the contribution of this work is in applying and extending Stamper and Liu's approach to modelling and specifying meaning during information system development. Little work was done in applying semiotic agent modelling to architectural design. Luo and Liu applied semiotic agent analysis for Information Systems Architecture Design (Luo and Liu, 2009). Their work focuses on using semiotic agents to capturing organizational requirements in preparation for architectural design, rather than offering an approach for a semiotic analyze of architecture artifacts.

Nobel et. al. present a semiotic analysis of object oriented design patterns (Nobel et al., 2006). While their work specifically deals with design patterns, which are artifacts relevant to architectural design, their work uses a more general semiotic framework based on Saussure's binary model of a sign, and Peirce three part relationship. While these allows deriving useful insights into the meaning and relationships amongst design patterns, the analysis is too coarse-grained to provide insights into the operational meaning of design patterns, and artifacts comprising patterns. Our work, which uses an extended version of Stamper's semiotic agent analysis as its underlying analysis framework, offers finer grained analysis of the operational meaning of artifacts.

Berry et. al. have studied ambiguity in natural language requirements (Berry et al., 2003). However, while enumerating many didactic examples of ambiguity, and providing informal writing guidelines how to reduce ambiguity during requirements drafting, no analysis method is offered to help clarify meaning to avoid ambiguity.

4 CONCLUSIONS AND FUTURE WORK

In this paper we offered a semiotic agent modelling approach to dealing with ambiguity during architectural design. The proposed approach extends Stamper's and Liu's work on Semiotic agents and ontological dependence schema's to apply to software design. The proposed approach uses the notion of affordance to helps clarify the operational meaning of technical design vocabulary used during free form discussion, captured in documents or included in conceptual models.

Future work will focus on further integrating the semiotic modelling approach into our work on representing, capturing and analyzing designers and stakeholder's intents and decision making in development organizations. Such integration promises advantages for semiotic analysis, in that it makes decision making during operational meaning construction visible and amendable to intentional analysis, and it promises advantages to the modelling and analysis of decision making processes in organizations, in that it helps clarify the terminology used and referred to in intentional models. Future work will also focus on further exploring the semantic of the "modifies" link between semiotic agents, and in particular, how to support "polymorphic" affordance definitions that selectively override respective affordances in the "parent" semiotic agent, which supports simplifying agent models.

Finally, since new terminology is introduced as design unfolds, the relationship between semiotic agent creation, to capture meaning of evolving and new terminology, and responsibility creation and assignment during intentional agent modelling needs to be further explored.

Another line of future work is the inclusion of norm analysis and a denotational language to capture architectural rules and guidelines that architects specify in development organizations. Perhaps some activities such as "Wait for feedback from provider" in figure 3 are better captured as norms rather than affordances. Also, a request such as "use asynchronous messaging" may be captured using norms defined over an ontological dependence schema. This future work would also focus on how to incorporate norms into intention agent modelling and analysis.

Finally, future work will also focus on modelling and analysis tools to support designers in capturing, reusing, applying, presenting, analysing and disseminating semiotic agent models during architectural design and change.

REFERENCES

Berry, D., Kamsties, E. & Kriekger, M. (2003). From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity. School of Computer Science, University of Waterloo.

Erl, T. (2007). SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl).

Fowler, M. & Scott, K. (2000). *UML distilled: a brief guide to the standard object modeling language,* Reading, Mass., Addison Wesley.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software,* Reading: Mass., Addison-Wesley.

Gibson, J. J. (1977). The Theory of Affordances. IN SHAW, R. E. & BRANSFORD, J. (Eds.) *Perceiving, Acting, and Knowing.* Hillsdale, N.J., Lawrence Erlbaum Associates.

Gross, D. & Yu, E. (2001a). Evolving System Architecture to Meet Changing Business Goals: An Agent and Goal-Oriented Approach. *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering.* IEEE Computer Society.

Gross, D. & Yu, E. (2001b). From Non-Functional Requirements To Design Through Patterns. *Requirements Engineering,* 6**,** 18-36.

Gross, D. & Yu, E. (2010). Supporting the evolution of software architectures in development organizations using intentional agents. *Fourth International i* Workshop - istar 2010*.

Josuttis, N. M. (2007). *SOA in Practice - The Art of Distributed System Design*, O'Reilly.

Liu, K. (2000). *Semiotics in Information Systems Engineering*, Cambridge University Press.

Liu, K., Sun, L., Dix, A. & Narasipuram, M. (2001). Norm-based agency for designing collaborative information systems. *Information Systems Journal,* 11**,** 229-247.

Luo, A. & Liu, K. (2009). Using Organizational Semiotics Methods for Information Systems Architecture Design. *The 11th International Conference on Informatics and Semiotics in Organizations.* Beijing, China.

Michaels, C. F. & Carello, C. (1981). *Direct Perception,* Englewood Cliffs, New Jersey 07632, Prentice-Hall, Inc.

Nobel, J., Biddle, R. & Tempero, E. (2006). Patterns as Signs: A Semiotics of Object-Oriented Design Patterns. *An International Journal on Communication, Information Technology and Work,* 2**,** 3-40.

Stamper, R. (1973). *Information in Business and Administrative Systems,* Oxford, Blackwell.

Stamper, R. (2006). Exploring the semantics of communication acts. *Proceedings of the tenth international conference on the language action perspective.* Kiruna: Linköping University.

Stamper, R. & Ades, Y. (2004). Semantic Normal Form and System Quality. *Proceedings IEEE Conference on Requirements Engineering, Kyoto*.

Stamper, R., Liu, K., Sun, L., Tan, S., Shah, H., Sharp, B. & Dong, D. (2003). Semiotic Methods for Enterprise Design and IT Applications. Staffordshire University, Reading University.

Westland, J. C. (2002). The cost of errors in software development: evidence from industry. *The Journal of Systems and Software,* 62**,** 1-9.

Yu, E. (1994a). Modeling Strategic Relationships for Process Re-Engineering. *Department of Computer Science.* University of Toronto.

Yu, E. (1994b). Understanding "Why" in Software Process Modelling, Analysis, and Design. *Proceedings of 16th International Conference on Software Engineering.* IEEE Computer Society Press.