Modelling Requirements for Self-integrating Manufacturing Systems

Peter Denno
National Institute of Standards and Technology, U.S.A.
Email: peter.denno@nist.gov

Keywords Self-integrating systems, Information Infrastructure, Self-describing systems

Abstract This paper discusses modelling requirements to support self-integrating manufacturing systems. Integration requires reconciliation of mismatch in various forms of context. Four forms of mismatch involving software components and aspect of the enterprise are identified. Although the paper stops short of describing a complete solution, aspects of the solution, as they relate to the four forms of mismatch, are discussed.

INTRODUCTION

Self-integrating manufacturing systems (SIMS) is a new area of research, so it is likely that there are various visions for what it should be. Some goals that might be attributed to SIMS are: implementation of virtual enterprises, supply chain integration, and plug-and-play factories [8]. At an extreme end of the spectrum, dynamic relationships between functional entities may be formed through resource brokering. From the standpoint of this paper, however, relationships are relatively static and the goal is simply to reduce the cost of business process re-engineering through automation of much of the re-engineering activity. This goal is, of course, not unique and even some aspects of the solution are not new [10], [2], [8]. For example, an essential aspect the SMART [2] architecture is self-descriptive software components, that is, components that provide a profile of their functional and interface characteristics. These descriptions may be used by higher-order systems to generate a middleware solution to inter-component communication.

This paper assumes that the integration process of self-integrating systems will rely on a comprehensive enterprise model. An instantiation of the model provides a context under which a self-describing component may be integrated into the enterprise's manufacturing operations. The processes and components involved with integration, as envisioned, are illustrated in *Figure 1*. As this figure suggests, the principle processes involved are (1) *coherence matching*, that is, identifying the relationship between the activity performed by the candidate software and the goals of the enterprise and (2) *technical reconciliation*, that is, overcoming differences in protocol, semantics and information structure. Both of these processes rely on the enterprise model, which describes the goals, processes, organizational structure, resources, interfaces and information entities that are subject to modification in a business process re-engineering project.

Enterprise models and modelling disciplines such as identified by CIMOSA [1] and ARIS [11] identify much of necessary content and methodology. However, in representing the relationship among goals, agents and interfaces, self-integrating systems have an additional requirement: it is necessary to make apparent the mismatch of technology, semantics and function that may occur between the existing and future work process. Whether integration of process is feasible, and whether the technical and semantic differences can be bridged, depends on the characterization of mismatch that can be made between the existing and future enterprise model instances. The remainder of the paper provides a characterization of the forms of mismatch after considering the aspects of communication and context that gives rise to the mismatch.

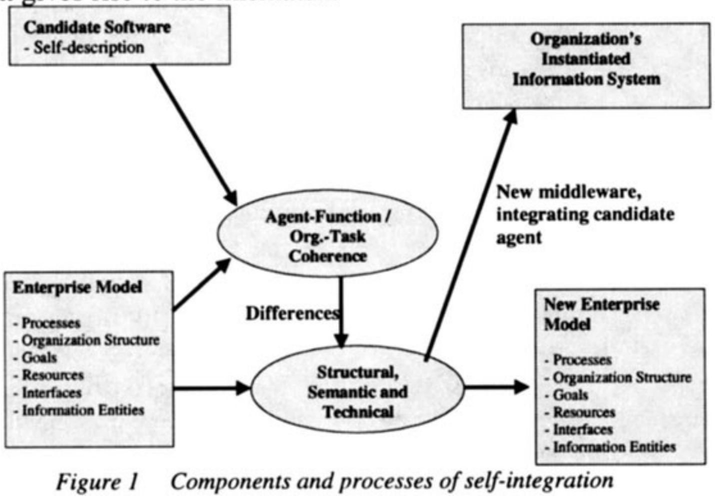

Figure 1 Components and processes of self-integration

INTEGRATION, COMMUNICATION AND CONTEXT

In this paper, *integration* refers to enabling communication among humans and software entities for the purpose of achieving a goal. That is, integration is the enabling of useful communication. *Self-integration* is integration where human intervention is not required.

The principle challenge of self-integrating systems is that of getting components to communicate with each other. Broadly speaking, an agent communicates with another for the purpose of influencing the behavior of the recipient. There is no other purpose. With respect to communication among humans in particular, the speaker, in choosing how to word his utterance, must have in mind the recipient's background knowledge and goals as well as the environment (time and place) in which the utterance is to occur. *Context* is a word for those things (background knowledge, recipient's goals and environment) the speaker must consider in designing an utterance.

The problem of communication is the problem of attending to context well enough to design an utterance that achieves, through the recipient, a desired behavior.

This definition of integration, as well as the use of "communication" applies as well to human agents as it does to software components. This is not an accidental coincidence but rather a design requirement for self-integrating systems. Because the mix of automation versus human endeavor varies from organization to organization (and component integration often changes the mix) it is valuable to have a language to model these notions that is neutral to whether a human or an automaton is committed to the task.

The problem and challenge of self-integrating manufacturing systems centers on the notion of context and the fact that the various, disparate components that might play a role in the manufacturing enterprise were designed in relative isolation; each built with a certain, idiosyncratic context in mind. Whereas the notions of communication and integration are neutral to whether human or automatons are involved, the notion of context is decidedly not. The context in which software agents communicate is narrow and artificial, not the 'lived experience' that we must grapple with as communicating humans. Context is, for the creator of artificial agents, a design problem; in communicating software agents, the crucial design question is that of how agents establish a context for communication. Assuming that the designer of the agent did not expect to exploit some unusual and extraordinary ability the agent might possess to behave usefully in ambiguous circumstances, the narrowness and artificiality of context among communicating agents is a fortunate accident. This narrowness facilitates integration.

CONTEXT AND FORMS OF MISMATCH

Persons managing a manufacturing enterprise have the responsibility of specifying the basic architecture of the enterprise. Broadly speaking, this entails identifying goals, defining processes, committing resources, and selecting systems for automating tasks. Through time, commitments of this sort are revised to better address the changing business environment. With respect to production equipment and software, management is faced with the choice of (1) building its own solution to address the unique requirements that management has imposed by other commitments, or (2) applying an existing solution such as a commercial off-the-shelf (COTS) software package. The latter is generally the less expensive choice, at least in terms of initial cost. It is, increasingly, the more common choice.

When a choice is made to accomplish a task with commercial off-the-shelf software, the enterprise is immediately faced with the difficulty that the selected software was quite likely designed in relative isolation from this enterprise's commitments; that is, it was designed with a different context in mind. The nature of the mismatch of context and how it may be expressed is the subject of the remainder of this paper.

FORMS OF MISMATCH

Generally speaking, *mismatch* (to resemble or harmonize unsuitably or inaccurately) applies to many forms of mismanagement: hiring the wrong persons, providing him with conflicting or ambiguous goals and so on. However, here the concern is specifically with the mismatch between components of the enterprise's architecture (component to component) and mismatch between component function and agency goal (function to goal). Four forms of contextual mismatch will be discussed: semantic, structural, functional and technological mismatch.

Semantic mismatch

A principle challenge of system integration is that of identifying sense differences that occur between information elements at points of communication. We envision that the self-description of an agent to be integrated would include assertions regarding how its information elements relate to terms defined by the enterprise model. In defining these terms the enterprise model is arbitrarily authoritative (it must make *some* ontological commitment) as there is no ultimate ground on which industrial terms might be founded.

An important design concern with regards to the definition of industrial terms in the enterprise model (and even speaking more generally about language [9]) is that meaning is a less useful notion than equivalence of meaning. That is, it is more useful to know whether my notion of "part" is equivalent to your notion of "detail", your notion of "assembly" etc., than it is to know what "part" is, in some fundamental sense.

The self-describing component may define relationship among the information elements of its interface and the terms defined by the enterprise model using sense relationships [3] and other characterizations of semantic proximity [12]. These are discussed below.

- *Synonymy* – having the same or nearly same meaning, substitutable
- *Hyponymy* – taxonomy, 'is kind of' relationship, relating narrower terms to broader terms
- *Antonymy* – having opposite meaning. Crystal [3] identifies three forms of antonymy that may be useful here: (1) *gradable*, permitting the expression of degree, such as good/bad/very good; (2) *non-gradable*, not permitting degree of contrast, such as single/married; and (3) *converse*, two-way contrasts that are inter-dependent, such as buy/sell
- *Incompatibility* – terms that are "mutually exclusive members of the same superordinate category" [3] such as red/green

Wiederhold [12] cites distinctions that follow from the encoding of information:

- *Value semantics* – the choice of threshold values where a quantity takes on another meaning
- *Temporal grain* – the quantum of time on which the value is based (*e.g.* versus monthly salary)
- *Abstraction grain* – a quantum (by some less fundamental metric than time) on which the value is based (*e.g.* production by lot versus monthly production)

The sort of distinction listed here are commonly found in an ontology (a set of terms and definitions in a formal logical language which connect the terms). We envision that the enterprise model would embody an ontology for the purpose of relating the information elements of the self-describing component to the terms of the enterprise model. The ontology can serve to define the nature of the narrowing between hyponymous terms (*e.g.* through membership predicates or other forms of constraint) for example.

Structural mismatch

Structural mismatch refers to differences in encoding or organization between information entities whose semantics are similar. Resolving structural mismatch presupposes that semantic equivalence has been recognized. In fact, structural mismatch often cannot be separated from semantic mismatch. Value semantics, temporal grain and abstraction grain can be viewed as structural problems arising out of the encoding of information, as they are semantic discriminators. The important distinction between semantic and structural mismatch is the manner in which it is addressed in the integration process: structural mismatch may be resolved through the generated middleware integrating the component or through an information mapping engine such as Express-X [4].

Functional mismatch

Functional mismatch refers to the degree to which the behaviour of an agent fails to achieve an expected effect (an enterprise goal, presumably). Functional mismatch is a concern first during the coherence matching process where an assessment of the feasibility of using the agent can be made, and second during technical reconciliation, where useful behaviours and results can be isolated from useless ones. ('Useful' and 'useless' are assessments made relative to the business context).

Concepts of function often rely on more general concepts of business practice and business resources. As described above, we envision that these later concepts and terminology would be addressed in the enterprise model. Therefore the modelling of concepts of function are tightly coupled with modelling of general enterprise terminology used in reconciling semantic mismatch. We envision that much the same relationship would exist between the component to be integrated and the enterprise model as it does in semantic reconciliation, the enterprise model defines terms of function by which the self-describing component describes itself.

Technical mismatch

Technical mismatch refers to differences in the software technology under which components provide interfaces (*e.g.* CORBA, COM, message queues). Components of the enterprise's information infrastructure might communicate through a framework (an architecture where components share a communication technology) or communication channels may be heterogeneous. General consensus in middleware technology is unlikely in the near future. For these reasons, self-integration will often involve bridging technical dissimilarity. The enterprise model, therefore should possess the ability to recognize the nature of the mismatch. Middleware technology can be categorized roughly as based either on message passing or remote procedure calls [5]. Subtle differences between technologies exists and must be made apparent. The development of an ontology of software technology is perhaps the least mature of requirements enabling self-integrating systems.

CONCLUSION

Self-integration, like business process re-engineering through traditional means, requires reconciliation of differences in semantics, function, structure and technology between the component to be integrated and the enterprise's existing information infrastructure. The complete context of the integration, as is understood by business- and technically-minded analysts undertaking a business process re-engineering project, involves aspect which are not traditionally subject to enterprise modelling. These include models of middleware technology, enterprise goals and component (or agency) function. Development of these aspects of the enterprise model defines a complete context for communication between components of the enterprise, enabling self-integration.

Commercial equipment and materials are identified in order to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

REFERENCES

[1] AMICE, (editors) (1991). *CIMOSA: Open System Architecture for CIM* Springer-Verlag, 2nd edition.

[2] Barry, J., et al. (1998) *NIIP-SMART: An Investigation of Distributed Object Approaches to Suport MES Development and Deployment in a Virtual Enterprise.* In The Proceedings of The Second International Enterprise Distributed Computing Workshop (EDOC98), 2-5 Nov.

[3] Crystal, D. (1997) *The Cambridge Encyclopedia of Language* Cambridge University Press.

[4] Express-X (2000) Product data representation and exchange — Description methods: Part 14: The Express-X Language Reference Manual. International Organization for Standards, Committee Draft.

[5] Grasso, M., P. (2000). Getting the Message. *Application Development Trends,* 101communications, LLC, August.

[6] Hunt, V., D. (1996). Process Mapping : How to Reengineer Your Business Processes, John Wiley & Sons.

[7] Liebowitz, J. and Khosrowpour M. (1997). Cases on Information Technology Management in Modern Organizations, Idea Group Publishing.
[8] NEMI, *NEMI Homepage*, http://www.nemi.org/
[9] Quine, Q. V. (1980) From a Logical Point of View : Nine Logico-Philosophical Essays, Harvard University Press.
[10] RosettaNet, *RosettaNet Homepage,* http://www.rosettanet.org/
[11] Scheer, A.. W. (1998) *Business Process Engineering* Reference Models for Industrial Enterprises, Springer-Verlag, Study edition.
[12] Wiederhold, G. (1992) Mediators in the Architecture of Future Information Systems, *IEEE Computer*, March, 38-49.