TOWARDS A LOGICAL FRAMEWORK FOR ENGINEERING DESIGN PROCESSES

Filippo A. Salustri, Ph.D.,P.Eng.
Ryerson University
Toronto, Canada
salustri@ryerson.ca

Abstract

This paper reports on a new aspect of the author's ongoing research to develop a logical framework for engineering design processes. Since this is a new project, the goal of this paper is to describe the nature of the problem being studied, to sketch the architecture of the system being developed to address it, and to identify various areas of application. Reasoning about design processes is found to be a task essentially about reasoning about product knowledge. A recently developed logic, called ALX3, has been found to provide all the basic mechanisms for representing design processes. ADPM will extend ALX3 to support directly the kinds of knowledge needed for design processes to occur. Some examples from the area of automotive engineering are used to indicate some of ADPM's benefits in practical environments.

Keywords: action logic, design process, design science, knowledge-based systems

1. Introduction

The goal of all the author's research is to formalize the design endeavor without limiting creativity and innovation. The author's previous research has developed a model of designed products; the Axiomatic Information Model for Design (AIM-D) [Salustri, 1996] uses axiomatic set theory to establish a logically rigorous product model. Now, the author is pursuing an Axiomatic Design Process Model (ADPM) intended to formalise portions of a generalized design process. Since ADPM is a new project, the goals of this paper are to describe the nature of the problem being studied, to sketch the architecture of the proposed solution, and to identify various areas of application.

Design science pursues formalisms for design using scientific methods. Design has important *subjective* and *objective* components, the relative importance of which depend on the particular discipline. *Engineering design* is constrained

by the laws of nature and the need for robustness and efficiency. It seems likely, then, that engineering design (just "design" hereunder) would benefit most from scientific formalism. However, the kinds of experiments needed to scientifically study real-world design processes is rarely feasible. Thus, until the field matures further, researchers must rely on theoretical techniques, including *formal methods.*

A fundamental requirement of scientific theories is *internal logical consistency.* This is typically afforded through the use of mathematics. One might therefore expect a reliance on mathematics in design science. However, design is an endeavor, not a phenomenon, and there are many substantive distinctions between the two. These issues have been examined by many researchers (e.g. [Simon, 1981; Alexander et al., 1977]. The author believes that a meaningful design science can be based on various forms of *symbolic*, rather than mathematical, logic, and that theorizing about design processes is a reasoning task about product *knowledge.* As such, logic, especially *model theory* will be shown to be a much more natural *fit* than mathematics.

This paper will motivate the use of *model theory* as a proper foundation of a logic of design processes and examine how three fundamental kinds of design tasks - synthesis, analysis, and evaluation - may be addressed by a particular logic, called ALX3, which is the foundation of ADPM. Various possible application domains are then examined.

2. Fundamentals

The author's approach for ADPM is based on two observations. Firstly, other efforts to formalize design processes have imposed either too much or too little formal structure. Formalisms based on heuristics, such as [Hubka and Eder, 1992; Pugh, 1991], are very flexible but not logically rigorous. Others, such as [Suh, 1990; Park and Cutkosky, 1999], have relied on highly structured systems that are nonetheless only naively connected to formal systems. Still others that have relied heavily on mathematical interpretations, such as [Zeng and Gu, 1999], allow reliable reasoning at the expense of flexibility: the logic becomes prescriptive, rather than descriptive. It is would be best to find a design process structure that does not limit the flexibility of its application.

Secondly, recent results of protocol analyses [McNeill et al., 1998] suggest to the current author that a design process consists of concurrent subprocesses that work towards a common overall goal but with potentially opposing perspectives *even if there is only one designer.*

These observations have led the author to propose the following hypothesis: *A design process is a discrete-event system occurring as the result of multiple "agents" acting towards a common general goal, each agent having its own priorities, context, and domain knowledge.* This hypothesis applies to single

designers as well as groups. The author believes that a single designer's cognitive processes may be modeled (in a limited way) as a group of interacting agents.

ADPM is not intended to formalise *entire* design processes. Rather, it is a formal framework for fragments of tasks such as: reasoning about design and design research, formulating new design methods, teaching design, and development of computer-based design aids. The goal of ADPM is to aid in thinking about design at both abstract and practical levels so as to improve the robustness of design processes and the efficacy of human designers.

There are various ways to represent a general design process with the language of mathematics, such as in [Zeng and Gu, 1999]. The author's own mathematical representation may be posed as $s_{i+1} = D\left(\sum_{j=0}^{i} s_j\right)$.

The state of a design process, s, specifies the design problem and its solution. D, the design process, operates on the *sum*[1] of all previous states to some point i - thereby providing access to the design history - and transforms it into the next reasonable state. The process itself is responsible for determining when a design is complete (i.e. when the final state has been reached).

Now, however, it is very difficult to specify the kinds of tasks (e.g. synthesis, analysis, and evaluation tasks) that should occur to transform one state into another. This is because of the limitations of mathematical process representations that do not treat issues like non-determinism and concurrency of executing actions. Instead, we would like to be able to write a set of rules that govern the *possible* actions that can be taken at some point in a design process such that the results of taking one action can be evaluated with respect to criteria governing the successful completion of the overall process.

3. ADPM, Version 1.0

Although mathematical approaches seem restricted, the state-transition representation on which they are based is both sensible and commonplace in the literature. By adopting this paradigm for design processes, we can begin to formalize individual actions without placing undesireable limitations on the overall processes. To maintain flexibility, we need to describe possible states without enforcing a prescriptive overall process. Furthermore, the author believes that a discrete-event formulation can provide a good foundation upon which to build a formalism for design processes. That is, that there are some kinds of crisply distinguishable events that occur during a design process that mark substantive changes to a design. More specifically, ADPM assumes the following representation of the *universe*[2] of design processes.

The *state* of a design is a set of first-order logic statement that describe a problem's requirements *and* the parameters that describe a product able to meet them. Generally, the initial state has no satisfied requirements and no defined

parameters, and the final state has a fully specified set of parameters satisfy the known requirements. It is then possible to select and order sequences of states that identify the shortest path from an initial to a final state. Actual paths in "real-life" design processes will likely be different. Development of strategies for keeping actual paths as close as possible to the ideal should improve design processes by shortening lead-times and increasing resource utilization.

A key feature of a design process is that it creates information; i.e. during the process, some information is not known. This suggests that a *three-valued* first-order logic is needed with constants $\top$ (true), $\bot$ (false), and $\dashv$ (unknown). Three-valued logics are well understood constructs.

It then becomes a simple matter to define axioms that identify various kinds of states (e.g. initial and final states). For example, we can specify that one a requirement is satisfied in some state, then it is satisfied in every subsequent state (assuming all subsequent actions are both correct and correctly executed)[3]

With such an axiom, we eliminate any action that would cause a satisfied requirement to become violated. A computerized designer's aid embedding this rule could warn a designer that an action will yield undesirable results. Clearly such a capability would be advantageous.

However, there is a problem. The state in which r is evaluated has not been specified, which means that r is in fact true in *every* state. This is not our intention. This problem can be avoided by using a *second-order logic* able to represent statements about statements. However, second-order logic both less well-developed and more complex than first-order logic. As a result, the author prefers not to have to depend on it at this time.

There is another kind of solution to this problem, fortunately, that remains within the bounds of first-order logic and maintains our intuitive notion of design as state transformation: model theory.

4. A Model Theoretic Solution: ADPM, Version 2.0

Model theory treats the relation between formal languages and its interpretations (or models) [Chang and Keisler, 1977]. It is typical to consider each model in a particular theory as a description of a "possible world," i.e. a description of the way things could be, the extent of which is restricted to only the domain of the formal language. A model theory allows the description of states and the relationships that allow one state to be transformed into another. Model theory is, then, a way of describing one logic with another logic, thereby circumventing the need for higher-order logics. In the case of ADPM, the formal language is AIM-D, the author's product formalism.

A possible world can represent the state of a design agent. An agent is said to *know a fact* if that fact is *necessarily* true in all worlds the agent considers possible, and *possibly* true if it is true in only *some* of those worlds. Model

theory also eliminates the need for three-valued logic: facts are only either true or false, and an agent either knows or does not know it. These capabilities are exactly what is needed to construct a formalism of design processes as described above; thus, model theory is used to develop ADPM.

To see the potential of a model theory for design, consider a recent experience of the author. A team of designers at an automotive manufacturer was charged with designing a new engine. The team included one designer for the engine block, and another for the cylinder head. Typically, the block designer begins by establishing key parameters, including the *block deck height*[4], which is passed to the cylinder head designer. Based on the total engine package height, the head designer can set a maximum height for the cylinder head. Once the design team had finished their preliminary work, a mockup of the engine was built. It was too tall. *Multiple* iterations over most of the design were required before they found that the block designer and head designer had each assumed the engine *gasket* was their own responsibility. This resulted in an engine that had *two* gaskets. The extra gasket made the mockup too tall.

The real problem faced by these two designers was a lack of *common knowledge* (CK) about the gasket, and it can be proved that the problem would have been *trivially* solved if they had had CK. Model theory provides the apparatus to define and manipulate CK [Fagin et al., 1995]. Some of the ways in which model theoretic formalisms can be applied to engineering design situations will be discussed in Section 6.

Model theory is typically used for two tasks. Firstly, it can *validate a formal language* (AIM-D, in this case) by showing the correctness of all its models. Thus, model theory can be used to logically verify product models.

Secondly, model theory can also search for sequences of transitions (*paths*) from an initial to a final state (called *model checking*). Such systems are called *process logics* and *action logics*, and have found application in artificial intelligence (AI) [Chen and De Giacomo, 1999] and distributed agent technologies [Singh et al., 1999]. In design engineering, such systems can help create, manage, and modify design processes.

To be appropriately expressive, a logic for design processes must include aspects of *modal logic* to distinguish between necessity and possibility, *temporal logic* to account for the passage of time, *dynamic logic* to represent actions, and *deontic logic* to include notions of obligation (i.e. a designer is obliged to act in certain ways under certain circumstances). Many of these kinds of logic are typically combined in *action logics*. The author has surveyed a number of recently developed action logic. Almost all have been found substantially lacking in one regard or another. For example:

- [Henriksen and Thiagarajan, 1999] have developed an action logic without branching time (needed to represent process *alternatives*);

- [Harel and Singerman, 1999] report on a logic with support for branches and concurrency of multiple agents, but it has only a weak form of logical negation and lacks modal operators;

- the process logic in [Chen and De Giacomo, 1999] is specificallyonly for model checking and does not support contradictions (which often occur *during* design processes); and

- [Huang et al., 1996] describe an sound, complete, and decidable action logic that supports bounded rationality (i.e. designers can be fallible), but without support for belief operators and time.

Furthermore, all these logics are based on propositional (not first-order) logic, and none of them support deontic operators. First-order logic is required for two reasons. Firstly, first-order logic allows quantification over the domain (e.g. facts like *Agent A knows that there exists a component that provides support for bending moments*), which is not possible with propositional logic. Secondly, first-order logics can be used to model processes built up from primitive or *atomic* actions. The actions taken as atomic need not be computable *per se*; so it is possible to build a formalism that includes the "atomic" action: *establish the subfunctions of the product's main function description* as a task to be performed by a human. Of course, this task may be redefined in time as a composite task made up of other, more primitive ones. This approach allows top-down - rather than bottom-up - development of formalisms for design.

Since no logic was found to be sufficient for modeling design processes, it seemed the author would have to develop a new one "from scratch." Fortunately, however, the author has found a new action logic, called ALX3 [Huang, 1994], that is ideal for modeling design processes. ALX3 is a sound and complete first-order action logic for agent groups that can represent many kinds of design process knowledge, and that is flexible enough to support knowledge operators, accessibility relations, goals, etc. Perhaps most importantly, it requires only *bounded rationality* of its agents: unlike other logics, an agent in ALX3 need not have perfect or complete knowledge of its environment, nor need it be able to reason perfectly. As such, ALX3 is consistent with human designers. ALX3, then, will be taken as the underlying logic for ADPM.

5. Kinds of Design Tasks

ALX3 is a general logic. To apply it successfully, it will have to be made specific to design's needs. ADPM will be a specialization of ALX3 for design process modelling. As such, ADPM will include: (a) a **representation of state** based on AIM-D, and extended with ontologies for product function [Yang and Salustri, 1999] and for design parameters; and (b) an **ontology of design**

actions based on primitive actions that single agents might carry out to reach their assigned goals.

The design action ontology is particularly germane to this paper. In Section 2, the author hypothesized that a design process occurs via multiple agents that can be broadly classified by whether their tasks are of synthesis, analysis, or evaluation, and identified by the changes they incur on design states. States change through the addition or removal of parameters and requirements. Adding parameters and requirements indicates design's evolution; removing them eliminates recognized falsehoods from the design state space.

Synthesis tasks refine a design space by adding/removing design parameters to/from a design state, and by refining the ranges of parameter values. These tasks are typically *inductive* or *abductive*. Induction is reasoning about actions, whereas abduction is reasoning about state. In both cases, the reasoning agents are neither omniscient nor perfect reasoners. Courses of action are determined by an agent's *preferences* and *beliefs*.ALX3, which supports preferences and beliefs, is again well-suited for these kinds of tasks.

Analysis tasks refine a design problem by adding/removing requirements and constraints to/from a state. Analysis is deductive in nature; deduction is immediately supported by ALX3. Deductive processes do not change the state of a design, but they change what an agent knows about a particular state. For example, a stress analysis of a part does not change the behaviour of the part (i.e. the state), but it does inform the agent regarding that behaviour.

Finally, **evaluation tasks** determine the veracity of the agents' knowledge of a design with respect to its requirements, performance, and other operational criteria. In ADPM, a design is complete when a state is reached such that all the values of (known) parameters satisfy the (known) requirements. The requirements would be specified as logical formulæ. To know if all the requirements are satisfied, it is sufficient to evaluate the logical conjunction of those requirements in a context containing all the parameters.

There is no necessary relation between the requirements that are known in a particular ADPM model, and the requirements that need to be met to ensure a well-designed product. The strategies needed to ensure this are the subject of ongoing research by the author.

6. Examples and Applications

ADPM will find application in a wide range of design-oriented reasoning tasks. This section outlines some of them.

6.1 The Role of Common Knowledge

A fact is common knowledge (CK) when (a) it is known by all agents, human or otherwise, and (b) every agent knows that they know the fact, every

agent knows that they know that they know the fact, etc. The resulting *infinite deduction* indicates that an agent can never attain CK [Fagin et al., 1995], yet it is also evident that both human and software agents can act as if they have large amounts of CK. This paradox can be resolved if either the knowledge that agents have, and believe to be CK, is not really CK as defined, or the formalisms currently available are too simplistic to capture "real" CK. It happens that both cases are true, depending on the situation.

Existing theories of CK have found application in the fields of artificial intelligence and economics, distributed agent systems, and natural language processing and cognitive science. There are features of these theories that may also be applied to design. CK arises through shared experiences, i.e. simultaneous changes in the states of a group of agents [Fagin et al., 1999]. However, there is no experience that is strictly speaking *simultaneous* to its participants, which means CK should be unattainable.

Sometimes, only *shared* knowledge is needed to solve a problem, which requires only a finite deduction. The kind of coordination problem of this sort is exemplified in Section 4. The number of deductions is directly proportional to the number of agents sharing a condition. This raises the interesting possibility *tuning* the roles of designers in teams to minimize the number of agents that can share a condition, thus lowering the number of iterations required to remedy it. This could shorten product development times in the long run.

Secondly, exact simultaneity of action, which assumes infinitely precise knowledge of the times when events occur, is not necessary to achieve CK in practice. The human mind naturally operates in a relatively vague mode, accepting as simultaneous any events that appear so. This temporal *resolution* allows us to approximate events as simultaneous. Determining appropriate temporal resolutions for design tasks and processes could help to simplify scheduling, workflow, and organizational structures.

Thirdly, the author has found that temporal resolution can be extended to treat other aspects of design process knowledge. This explains why agents can agree on a common process model in general, yet disagree about the model's details: at a coarse level, any number of different models can appear the same. The same logical apparatus that treats temporal resolution may then be used to quantify the compatibility of other kinds of models of varying abstraction, which may help tune the information flow in design processes, especially in multi-model design environments.

Fourthly, CK can arise when agents are in a common situation. This *copresence* explains why "face to face" meetings are preferred for coordinated tasks, negotiation, and conflict resolution. Through the formalism of CK, tools may be developed to improve the efficiency and effectiveness of meetings.

Finally, CK may be implicit by being *distributed* over multiple agents. By identifying *islands* of expertise, tools may be developed to help structure design teams, manage workflow, and other organizational aspects of design enterprises.

Although ALX3 does not deal specifically with CK and its variations yet, it does support all the necessary mechanisms. The author will undertake to translate appropriate theoretical results from the CK literature into the ALX3 framework; this will become an essential part of ADPM.

6.2 Example: Engine Design

Consider again the engine design case discussed in the example in Section 4.

The author and a colleague were contracted by the auto maker to help improve their design process capabilities. Details of that work are reported in [Lockledge and Salustri, 1999]). We developed a matrix representation that captured bi-directional causal relations between engine systems and components. The relations can be seen as actions that change the state of the design. Since ALX3 can represent actions, ADPM could formalise much of the dynamic structure in [Lockledge and Salustri, 1999] (which remains largely informal). It is expected that ADPM's application to this area will bring to light other patterns of actions, and thus stimulate further improvements to design processes used by the client.

ALX3 also supports *goals*, *preferred* states, and knowledge operators[5]. With these, one can envision systems that can track progress of designers using tools such as that described above, and suggest actions that will lead to goal states most efficiently. Such a system could also warn stakeholders of particular facts when design data changes, and could route appropriate information to them only as and when they need it. One can envision agents that would *know* (to an extent) the nature of design changes and ensure that the appropriate designers were notified and the most preferred actions taken. Such systems could lighten the administrative burden on designers and allow them to focus more on the technical aspects of design problems.

6.3 Simulating Design Processes

Design processes can be *simulated* by devising agents that carry out various goal-oriented strategies for the sake of collaboratively designing a product. The agents would have *very* limited capabilities. Teams of software and human agents could operate together: the software providing the logical aspects, and the humans to provide the creativity and intuition. Such simulated design exercises might yield interesting data regarding the viability of various design strategies (e.g. "opportunistic" design), or suggest new methods and organizational structures that can then be transfered into real design situations.

This kind of research is particularly interesting when applied to single designers. As the author has hypothesized, the cognitive process of design may be modeled as a set of agents working towards a common general goal but possibly with conflicting specific goals. This *could* explain the results of recent protocol analyses. Changes in the designer's cognitive state trigger actions constitute tasks which depend on the current state, includings domain and general knowledge. It would also be reasonable, in this case, to assume that all the agents "running" in a designer's mind have *completely* common knowledge.

A logical formalism of design may also suggest new approaches to protocol analysis: wherein actions, goals, preferences, knowledge, and beliefs can each be identified and modeled, thanks to logics like ALX3.

6.4 Application Domains

There are many possible application domains for a logic like ADPM. ADPM could provide design researchers with a structure within which to **reason about particular methods, representations, and design processes**. The reasoning capabilities that may emerge from ADPM constitute a synthetic technique as well as an analytic technique. If design *patterns* [Alexander et al., 1977] can be established, it may be possible to constitute new generalized design processes that differ substantially from those currently in use, as well as design processes customised on a per-enterprise basis, taking into account particular aspects of specific design environments. Also, in combination with other AI techniques such as case-based reasoning [Maher et al., 1995], abductive reasoning [Roozenburg, 1992], and distributed artificial intelligence [Singh et al., 1999], it may be possible to construct intelligent agents that can act as designers' aids and operate in symbiosis with (groups of) designers. Products are affected by the design processes that create them which in turn are affected by organisational structure. It should also be possible to deduce some of the properties of a design enterprise that could support logically sensible design processes. In this way, design research may find application in areas like enterprise re-engineering, a point emphasised in the literature on enterprise integration (e.g. [Dabke, 1999]), wherein frameworks for product development are used to ensure enterprise models are effective from both the business *and* technical perspectives. Finally, it should be possible to create a "naive" version of ADPM, constituted as rules and guidelines, that can be effectively taught. The prospect of enabling design engineering students with a more structured mental framework should lead to more able designers.

7. Conclusions

An overview of the Axiomatic Design Process Model (ADPM) project has been presented.

ADPM can (a) facilitate reasoning about design in a systematic manner, (b) lead to new computer-based design aids, (c) develop new and custom design and product development processes, and (d) lead to innovative techniques to teach design. Clearly, ADPM is embryonic. This paper has established the problem being studied, laid out and justified the tools to be used, and indicated the possible benefits that may be achieved. A great deal of work remains to be done. It is hoped that as the project matures, it will find use in various areas of design engineering research and practice.

Acknowledgments

The author gratefully acknowledges the National Sciences and Engineering Research Council of Canada for funding this work under grant number OGP0194236.

Notes

1. Here, *sum* intends some kind of merging interrelated knowledge bases of all available information.

2. The *universe* of a system of logic, such as that being discussed here, is the collection of all entities about which one makes statements within the system.

3. Assuming linear time and a partial temporal ordering on states, π, such that $s\pi t$ means that state s precedes state t: $\forall(r \in \bar{R})\,[[(r \in s) \bullet r] \rightarrow \exists t\,[(r \in t) \bullet (s\pi t) \bullet r]]$, where r is a requirment and s and t are states.

4. The *block deck height* is the distance from the bottom of the engine to the top of the block.

5. For example, $K_i\, p$ means that agent i knows that p is the case.

References

Alexander, C., Ishikawa, S., and Silverstein, M. (1977). *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, London.

Chang, C. and Keisler, H. (1977). *Model Theory*, volume 73 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, Amsterdam.

Chen, X. and De Giacomo, G. (1999). Reasoning about nondeterministic and concurrent actions: A process algebra approach. *Artificial Intelligence*, 107:63–98.

Dabke, P. (1999). Enterprise integration via corba-based information agents. *IEEE Internet Computing*, 3(5):49–57.

Fagin, R., Halpern, J., Moses, Y., and Vardi, M. (1995). *Reasoning about Knowledge*. MIT Press, Cambridge, MA.

Fagin, R., Halpern, J., Moses, Y., and Vardi, M. (1999). Common knowledge revisited. *Annals of Pure and Applied Logic*, 96:89–105.

Harel, D. and Singerman, E. (1999). Computation paths logic: An expressive, yet elementary, process logic. *Annals of Pure and Applied Logic*, 96:167–186.

Henriksen, J. and Thiagarajan, P. (1999). Dynamic linear time temporal logic. *Annals of Pure and Applied Logic*, 96:187–207.

Huang, Z. (1994). *Logics for Agents with Bounded Rationality*. PhD thesis, University of Amsterdam.

Huang, Z., Masuch, M., and Polos, L. (1996). Alx, an action logic for agents with bounded rationality. *Artificial Intelligence*, 82:75–127.

Hubka, V. and Eder, W. (1992). *Engineering Design: General Procedural Model of Engineering Design*. Edition Heurista, Zurich.

Lockledge, J. and Salustri, F. (1999). Defining the engine design process. *J. Engineering Design*, 10(2):109–124.

Maher, M., Balachandran, M., and Zhang, D. (1995). *Case-Based Reasoning in Design*. Lawrence Erlbaum Assoc.

McNeill, T., Gero, J., and Warren, J. (1998). Understanding conceptual electronic design using protocol analysis. *Research in Engineering Design*, 10(3):129–140.

Park, H. and Cutkosky, M. (1999). Framework for modeling dependencies in collaborative engineering processes. *Research in Engineering Design*, 11(2):84–102.

Pugh, S. (1991). *Total design: integrated methods for successful product engineering*. Addison-Wesley, England.

Roozenburg, N. (1992). *On the Logic of Innovative Design*, pages 127–138. Delft University Press, Netherlands.

Salustri, F. A. (1996). A formal theory for knowledge-based product model representation. In Finger, S., Tomiyama, T., and Mantyla, M., editors, *Knowledge-Intensive CAD II: proceedings of the IFIP WG 5.2 workshop*, London. Chapman & Hall.

Simon, H. A., editor (1981). *The Sciences of the Artificial*. The MIT Press, Cambridge, Massachusetts, 2nd edition.

Singh, M., Rao, A., and Georgeff, M. (1999). *Formal Methods in DAI: Logic-Based Representation and Reasoning*, pages 331–376. MIT Press, Cambridge, Mass.

Suh, N. P. (1990). *The Principles of Design*. Oxford University Press, New York.

Yang, B. and Salustri, F. (1999). Function modeling based on interactions of mass, energy and information. In Kumar, A. and Russell, I., editors, *Proc. 12th Florida Artificial Intelligence Research Symposium, special track on Reasoning about Function*, pages 384–388.

Zeng, Y. and Gu, P. (1999). A science-based approach to product design theory part i: formulation and formalization of design process. *Robotics and Computer Integrated Manufacturing*, 15:331–339.

Filippo A. Salustri received his doctorate in mechanical engineering in 1993 from the University of Toronto. He spent one and one third years at Wayne State University in Detroit as an assistant professor of engineering. Currently, he is an assistant professor in Industrial and Manufacturing Systems Engineering at the University of Windsor, Canada. His industrial collaborations have included Henry Ford Hospital (Detroit), Ford Motors (Dearborn) and Spar Aerospace (Toronto). His primary areas of research are design theory and the development of knowledge-based systems for engineering applications.