INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 03 STOCKHOLM, AUGUST 19-21, 2003

TOWARDS AN ACTION LOGIC FOR DESIGN PROCESSES

Filippo A. Salustri

Abstract

While the creative and innovative aspects of engineering designing are arguably its most important aspects, it is equally important that this creativity be tempered and directed through the use of sound and rigorous methods. Formal logic can provide a foundation for the construction of such methods. A formal model of design processes, the Axiomatic Design Process Model (ADPM) is presented. The model, based on the action logic ALX3, is able to model the imperfect reasoning capabilities of human design agents. This paper presents the background and fundamentals of ADPM with particular emphasis on the representation of design principles as goals. Various possible application areas of ADPM are also discussed. It appears, based on the preliminary results, that ADPM can be an appropriate vehicle for modelling engineering design processes.

Keywords: process, systematic product development, descriptive models of designing

1 Introduction

The overall goal of the author's research is to formalise aspects of engineering designing without limiting opportunities for creativity and innovation. An overall framework, Artefact-Centred Modelling [1], distinguishes between product models, modelling languages, and processes that manipulate those models. Previous research has developed the Axiomatic Information Model for Design (AIM-D) [2] uses axiomatic set theory to establish a rigorous product model logic. Similarly, the author is now starting to develop an Axiomatic Design Process Model (ADPM) intended to capture aspects of a generalised design process with formal logic. ADPM is intended to be a formal framework for such tasks as: reasoning about design and design research, formulating new design methods, streamlining existing processes, organisational modelling, teaching design, and developing new computer-based design aids.

This paper introduces ADPM some early results in its development. In particular, an approach for the representation of design principles (e.g. *minimise the number of parts in a product*) is presented to show how empirical and intuitive notions can be formalised.

The rest of this paper is organised as follows. First, some background is presented, to indicate the worth of formal logic in engineering design. The discussion leads to the author's hypothesis that drives this project. The fundamentals of ADPM are then introduced, with emphasis on (a) the boundary conditions that mark the initial and final states of a design process, and (b) the representation of some kinds of design principles. Potential applications of ADPM are then discussed. Finally, some concluding remarks are directions for future work are presented.

2 Background

Design science is the pursuit of formalisms for design using scientific methodologies. Design has important subjective components as well as objective ones. The application of formal systems to the objective aspects of design is well documented; examples include [3-5].

Design is an artificial endeavour, not a natural phenomenon. Thus, strictly scientific approaches to developing a design science are of limited use. Science is, however, based on logic: a fundamental requirement of "scientific" theories is *internal logical consistency*. Others have explored this point in detail, such as [6] and [7]. Because of this distinction, the current author believes that development of a real design science will depend on the use of symbolic logic instead of science.

ADPM is based on two observations made by the author while surveying the literature. First, other attempts to formalise design processes seem to impose either too much or too little structure. Formalisms based on heuristics (e.g. Hubka and Eder [8] and Pugh's Total Design [9]) are flexible but logically unreliable. Other formalisms, such as Axiomatic Design [3] and the work in [10], have relied on highly structured systems that are nonetheless only naïvely connected to formal systems that cannot be verified. Still other formalisms that have relied heavily on mathematical interpretations, such as that of Zeng and Gu [11], allow more reliable reasoning at the expense of flexibility. Most importantly, such theories generally treat design processes as lacking the conscious element of the human designer driving the process forward. While such approaches can highlight certain important features of design processes, they cannot address much of the variability that can occur during design processes.

It would be most advantageous to find a formalism that structures design processes without limiting the flexibility of its application or the creativity of its users.

The second observation comes from the results of the protocol analysis of McNeill et al [12], where it is shown that a designer will interleave periods of analysis, synthesis, and evaluation, cycling between them over the course of the experiment. There also seemed to be opportunistic moments in the reported protocols where concern for function, structure, and behaviour appeared interleaved with one another. It seemed to the current author that designing could consist of multiple concurrent processes that work partly *against* one another with respect to immediate goals. It is this *antagonism* that leads to a deeper understanding of the problem: that there is a context-sensitive *balance* between design goals, and that design processes seek to find that balance. One might then hypothesise that a formalised model of design actions could help designers reach these balanced designs.

Furthermore, these processes work towards a common overall goal but with potentially opposing perspectives *even if there is only one designer*. Of particular note is the interleaved concurrency of synthesis and analysis tasks, which supports the notion that each design decision impacts on the nature of the design problem being solved.

These observations have let me to adopt the following hypothesis: *a design process is a discrete-event system occurring as a result of multiple "agents" acting towards a common general goal, each agent having its own priorities, context, and domain knowledge*. No commitment is made here to the number of designers involved. Indeed, one of the most interesting possibilities here is that the cognitive process of a *single* designer may be successfully modelled with multiple agents.

The author believes that formal logic should provide a better mechanism than mathematics for this. Logic is the foundation of mathematics. As mathematics is the logic of numbers, the author believes that we need a new logic of design. There are many different logics. Some

have been applied successfully to explain and reason about belief systems, economic systems, organisations, and other essentially non-deterministic and social entities. Logic, in this sense, is a framework that can expose, represent, and support the manipulation of, the structure that underlies a process, even if that process has elements that are not "logical".

There are various ways that formal logic can be used to formalise design processes. The author has investigated a number of these approaches, which have been reported elsewhere [13]; space restrictions do not permit a review of that work here. The result of that investigation is that a particular family of logics, called *action logics*, are best suited to design process formalisation, and that one action logic in particular, called ALX3 [14] is the best of those. A brief summary of the author's investigations is as follows. Action logics fall under the branch of logic called *model theory*. In this approach, two 1st order theories are layered one atop the other. This avoids problems of self-reference that can lead to serious logical contradictions. The first theory, called the *object theory*, describes the domain of interest (i.e. product models). The author uses AIM-D, a formal product modelling system based on set theory (other systems could be used). The second theory, called the *meta-theory*, formalises reasoning about the object theory. In this case, it formalises the processes that generate product models: design processes. This layering of 1st order theories helps ensure a robust system. The soundness of ALX3 has been demonstrated in [14].

Finally, ADPM is intended to be compatible with other theories of designing (e.g. Axiomatic Design). Clearly, further work is needed to determine how, if at all, specific other theories can be represented in ADPM. The point is, however, to provide a formal framework for treating theories of designing and *not* necessarily to prove or disprove any one other theory.

3 The axiomatic design process model

In this section, some fundamental statements about design processes will be presented using ALX3. These statements form a set of design axioms that constitute the basis of ADPM.

We assume a separate language exists for the product models (in this case, AIM-D). This means that whole product models are treated as single variables in ADPM. Similarly, we assume the design requirements are separately treated and are thus variables in ADPM. The author notes that the only design requirements included here are those pertaining to the product; requirements about the manufacturing system for the product, engineering management, scheduling, workflow, and other aspects of product development are *not* included. This is a shortcoming of AIM-D, not the current ADPM work. While these aspects are essential, they are beyond the current work and will be treated properly in the future. This also maintains the 1st order nature of ADPM.

The domain of ADPM consists of states. Each state contains a different product specification and requirements specification, so the domain includes every possible model that can be represented with AIM-D. Only one state is the actual state; other states are only potential ones and may become the actual state through the execution of some action or sequence of actions. An *action* in this sense is a behaviour; i.e. ADPM does not attempt to formalise the cognitive, creative activities of a design, but rather seeks to provide a formal reasoning framework for the behaviours exhibited by designers *as a result* of those cognitive, creative activities. It is in essence a language for formally expressing design agent behaviours, and for reasoning about the processes that consist of sequences of those behaviours.

We assume a constant function (i.e. the same in every state) that evaluates a product model with respect to a set of requirements. This function is able to (a) determine if the product

model in a particular state satisfies the requirements in that state, and (b) produce a *rank* of the product model with respect to the requirements. This rank can be used to order different alternative designs. The exact nature of the evaluation function is a subject of ongoing study.

We can begin now to identify the boundary conditions of a design process. For example, consider a design enterprise whose bid for a design project was accepted. It is reasonable for the designers (*agents* in ADPM) to believe the design requirements specified in the bid are now fixed and constant. Admittedly, this is not necessarily realistic, but we assume this as a simplifying assumption to establish the foundation of ADPM. This can be expressed as:

$$\text{bid (r)} \Rightarrow \mathbf{B}_{\text{d}}(\text{r}\mathbf{P}_{\text{d}}\text{r'}), \tag{1}$$

which is read "*if a bid's requirements have been accepted, then a designer believes that those requirements are preferred to any other set of requirements.*" $\boldsymbol{B}_d$ is the ALX3 believe operator such that $\boldsymbol{B}_d\ (x)$ means "*agent* d *believes statement* x *is true*." $\boldsymbol{P}_d$ is the ALX3 preference operator such that $x\boldsymbol{P}_d\ y$ means "*agent* d *prefers state* x *to state* y."

It is also reasonable that all the designers believe that there must exist a product that will satisfy all the design requirements stipulated in the bid. This can be written as:

$$\text{bid (r)} \Rightarrow \mathbf{B}_{\text{d}}\ (\mathbf{A}_{\text{d}}\ (\text{E (r, p)})), \tag{2}$$

which is read "*if a bid is successful, then a designer believes that there is a process by which a product can be designed that satisfies the bid's design requirements.*" $\boldsymbol{A}_d$ is the ALX3 accessibility relation, such that $\boldsymbol{A}_d\ (x)$ means "*there is a sequence of actions executable by agent* d *that results in a state* x."

Finally, at the end of a design process, the designer would know that the design meets the requirements. This is written:

$$\mathbf{K}_{\text{d}}\ (\text{E (r, p)}). \tag{3}$$

The task of the designer is to reach the final state in (3), from the initial state in (1) and (2). Since knowledge is defined as true belief, then achieving the final state involves executing *actions* that (a) establish a set of beliefs about a product, and (b) determine the truth of those beliefs. These actions can be grouped into three broad categories. *Synthesis* tasks are those that expand the belief system about the product. *Analysis* tasks are those that either expand the set of requirements or remove beliefs that are demonstrated to be false. Finally, *evaluation* tasks are those that compare a product's design to the requirements.

Statements that explicate the transition between the initial and final states constitute a description of the process itself. Since there are an infinite number of possible design processes, the author is focusing initially on general statements and principles.

There is evidence, in the form of the protocol studies of McNeill et al [12] and other works such as [3], that analytic and synthetic tasks occur in alternation. Accepting this as a characteristic of design processes in general, we can formalise this idea. For example, it can the case that a designer will prefer to advance *either* the requirements specification of a design *or* the product specification, but not both at once. This is written in ADPM as:

$$\mathbf{K}_{\text{d}}\ (\neg\text{E (r, p)}) \Rightarrow [(\text{r'} \wedge \text{p}) \vee (\text{r} \wedge \text{p'})]\mathbf{P}_{\text{d}}\psi, \tag{4}$$

and is read: "*if the designer knows that the current design does not satisfy the requirements (i.e. the design is not completed), then the designer prefers states where only either the requirements or the product model have been changed*."

In order to represent the transition between synthetic and analytic tasks, we introduce some abbreviations:

$$^{r}P_d =_{def} r \wedge p \wedge (r' \wedge p)\ \mathbf{P}_d\ (r \wedge p'), \quad (5a)$$
$$^{p}P_d =_{def} r \wedge p \wedge (r \wedge p')\ \mathbf{P}_d\ (r' \wedge p). \quad (5b)$$

$^{r}P_d$ indicates that changes to the requirements are preferred to changes in the product model, and $^{p}P_d$ is the converse.

It is not clear from the available literature what condition holds when a change from synthesis to analysis should or does occur. Nonetheless, let us postulate that a designer would change from synthetic tasks to analytic tasks – or vice versa – when he believes that performing more of the same kind of task will advance the design. The author refers to this as the *conservative* approach to task type alternation. In ADPM we can write this as:

$$^{r}P_d \wedge \mathbf{B}_d\ (\neg\langle a\rangle r' \wedge r'\mathbf{P}_d r) \Rightarrow \langle b\rangle {}^{p}P_d \quad (6a)$$
$$^{p}P_d \wedge \mathbf{B}_d\ (\neg\langle a\rangle p' \wedge p'\mathbf{P}_d p) \Rightarrow \langle b\rangle {}^{r}P_d \quad (6b)$$

In (6), $\langle a\rangle r$ means that there is an action *a* whose execution results in achieving a state *r*. (6a) says that given cases where a designer prefers to advance the requirements versus advancing the product model, the designer will change that preference when the designer believes that there are no actions allowing a more-preferred requirements specification to be reached. (6b) says the converse. These statements capture the notion that a designer will change between analytic (or synthetic) tasks only when there is no alternative.

We note the use of the belief operator in (6). There may in fact be further actions that a designer could take, if he were aware of them, but since ADPM agents are imperfect reasoners, they must act on their beliefs when they lack knowledge.

Clearly, (6) is not an all-inclusive principle. For example, it does not address *opportunistic design* [15] in which a designer can switch between synthetic and analytic tasks at the first opportunity. We can represent this *progressive* approach to task type alternation by assuming that an agent switches between synthetic and analytic actions as soon as such opportunities are noticed. In ADPM:

$$^{r}P_d \wedge \mathbf{B}_d\ (\langle a\rangle p' \wedge p'\mathbf{P}_d p) \Rightarrow \langle b\rangle {}^{p}P_d \quad (7a)$$
$$^{p}P_d \wedge \mathbf{B}_d\ (\langle a\rangle r' \wedge r'\mathbf{P}_d r) \Rightarrow \langle b\rangle {}^{r}P_d \quad (7b)$$

(7) formalises opportunistic design. The difficulty with this approach is that a designer will be switching *constantly* between analytic tasks and synthetic tasks regardless of whether such a switch is warranted. In real design situations, these criteria are more complex. One possibility for a *balanced* approach to task type alternation is that the evaluation function, *E(r, p)*, is used to rank possible states arising from the selection of either a modified product, *p'*, or modified requirements, *r'*. Such a ranking would have to be based on the designer's beliefs, since it amounts to predicting the future.

$$^{r}P_d \wedge \mathbf{B}_d\ (\langle a\rangle p' \wedge p'\mathbf{P}_d p) \wedge \mathbf{B}_d\ (\langle b\rangle r' \wedge r'\mathbf{P}_d r) \wedge \mathbf{B}_d(E(r, p') > E(r', p)) \Rightarrow {}^{p}P_d \quad (8a)$$
$$^{p}P_d \wedge \mathbf{B}_d\ (\langle a\rangle p' \wedge p'\mathbf{P}_d p) \wedge \mathbf{B}_d\ (\langle b\rangle r' \wedge r'\mathbf{P}_d r) \wedge \mathbf{B}_d(E(r, p') < E(r', p)) \Rightarrow {}^{r}P_d \quad (8b)$$

(8) defines a richer condition for switching between analytic and synthetic tasks. An agent must believe that there is one action that will advance the requirements analysis and another that will advance the product description, and that one advance is better than the other with respect to *E*.

(6), (7), and (8) can all be thought of as design process strategies, but they are not mutually exclusive. If they were all allowed in a design process, one must select among these strategies for a particular process. Indeed, it is also possible that more than one of these strategies could be employed, by different designers, and at different times in the same process. Exactly how this would work is a matter for further investigation.

In summary, notwithstanding a clear need for further development, it seems evident that ADPM can represent some aspects of design processes. Future work will show just how useful such a formalisation is (or is not).

Within the framework of ADPM, some common design principles can be formalised as *goals*. A goal is an implicit description of the kinds of states that represent key achievements in a design process.

To demonstrate the representation of design principles in ADPM, consider the example: *minimise the number of parts.* One may argue that this principle is a design optimisation or a constraint satisfaction task. In practice, however, one can achieve process efficiencies by setting the minimisation as an issue to be considered regularly, and that there reaches a point where further minimisation is not beneficial. This interpretation likens it to goal achievement. ALX3 provides some support for goals that are precisely of this sort of optimisation.

The basis of this goal is an ability to know, calculate, or estimate the number of parts in a product in any state. This function is constant in the domain: the same function is used to calculate the number of parts in any states. Let the predicate *np(x)* be true when *x* is the number of parts of a product.

Next, we establish that a designer prefers states with fewer parts. In ADPM:

$$(x < y) \Rightarrow np(x)\,\mathbf{P}_d\,np(y); \tag{9}$$

that is, "*if x is less than y, then an agent prefers designs with x parts to those with y parts.*"

The preference operator, ***P***, defined axiomatically in ALX3, states facts without explanation; that is, there is no commitment to the rationale for the preference. This is good thing in that it decouples the statement of a preference from its justification, allowing the treatment of these two issues separately. As it happens, however, in this case we can actually interpret (9) as a partial rationale, read as "*a designer prefers designs with x parts to designs with y parts because x is less than y*." This will be discussed further subsequently.

One can imagine (9) now as a criterion for selecting actions. Preferred actions lead to preferred states. However, we must avoid local optima of these preferences. Without formally defining some restrictions on this preference, it is possible that an agent will select an action that leads to a preferred state from which an even more preferred state is not accessible.

Masuch and Huang [16] restrict this kind of goal by limiting preferred states to those that allow other even more preferred states to remain accessible. However, the current author does not believe this is appropriate in design. It is easy to imagine cases where an action that lessens the number of parts in a product does so to the detriment of the product. That is, the goal is not defined only by the actions that can be taken to improve one design characteristic.

The author believes that the real limitation is the *trade-off* between *all* the goals to be met (i.e. principles to be applied) by the designer. Improving one characteristic of a product may deteriorate another. This must be avoided. We need a rule, then, that states that improve one product characteristic cannot do so at the expense of any other characteristic.

Since generally we cannot know how some values of characteristics are rated "better" or "worse" than others, we must confine this determination to designers' preferences only. This is not so bad because, as is seen in (9), rationale can be treated separately from preferences. We then come to a version of the principle of minimising parts in ADPM:

$$\mathbf{G}^{t}[\mathrm{np}(x)] =_{\mathrm{def}} \mathrm{np}(x)\,\mathbf{P}_{d}\mathrm{np}(y) \wedge (\mathrm{np}(x) \rightarrowtail \psi(u)) \wedge \mathbf{B}_{d}[\forall z\,[\mathrm{np}(z)\,\mathbf{P}_{d}\mathrm{np}(x) \Rightarrow [\mathrm{np}(z) \rightarrowtail \psi(v)] \wedge \psi(u)\,\mathbf{P}_{d}\psi(v)]. \quad (10)$$

(10) defines a trade-off goal ($\mathbf{G}^{t}$) for a value *x* of the design characteristic *np* (number of parts): an agent prefers products with *x* parts to products with *y* parts, so long as the designer believes that more preferred values, *z*, are such that they cause another characteristic (y) to obtain a less preferred value. Other characteristics might be weight, cost, lifetime, manufacturability, reliability, etc. (The symbol $\rightarrowtail$ denotes the ALX3 causation operator which denotes that its antecedent is the cause of its consequent. Causation is not to be confused with material implication, $\Rightarrow$.) (9) now defines a physical basis of the preference in (10), and (9) and (10) define the goal of minimising the number of parts in a product within a context of other possibly conflicting goals.

We can use this goal statement as a template for a general category of design principles. Let *x*, *y*, *z*, *u*, and *v* be arbitrary values. Let $y(x)$ and $f(x)$ be two functions that associate values with product characteristics (like *np(x)*), and let $xO^{f}y$ be an order for characteristic f (like $x<y$) such that the antecedent (x) precedes the consequent (y) in the order. We can write:

$$\mathbf{G}^{t}[\phi(x)] =_{\mathrm{def}} \phi(x)\,\mathbf{P}_{d}\phi(y) \wedge (\phi(x) \rightarrowtail \psi(u)) \wedge \quad (11a)$$

$$\mathbf{B}_{d}[\forall z\,[\phi(z)\,\mathbf{P}_{d}\phi(x) \Rightarrow [\phi(z) \rightarrowtail \psi(v)] \wedge \psi(u)\,\mathbf{P}_{d}\psi(v)],\quad xO^{\phi}y \Rightarrow \phi(x)\,\mathbf{P}_{d}\phi(y). \quad (11b)$$

(11) is now a general formalised principle of design that embodies two aspects. First, (11a) captures trading off one kind of improvement against other kinds of improvement and provides a criterion for selecting design actions. Second, (11b) gives a partial account of why a particular preference exists for a designer. This allows one to translate between information about a design, and how a designer could seek to improve it. The kind of rationale indicated by (11b) captures the invariant physical rationale over all reasonable cases.

(11) is by no means a complete account of how one might balance a design by trading off conflicting goals. (11a) says nothing about the priority of different characteristics. For example, increasing safety may be more important than decreasing cost in one product whereas the opposite might be true elsewhere. Nothing in (11a) can help a designer account for this.

We can introduce axioms for specific cases that capture preferences explicitly. For example, let *s(x)* be a function that assigns a value *x* to a safety characteristic *s*, and let *c(x)* be a similar function for cost. Let the values of *s* and *c* be ordered by functions O^{s} and O^{c}. We can write:

$$[xO^{s}y \Rightarrow s(y)\,\mathbf{P}_{d}s(x)] \wedge [uO^{c}v \Rightarrow c(u)\,\mathbf{P}_{d}c(v)] \wedge s(y)\,\mathbf{P}_{d}c(u). \quad (12)$$

(12) states that safety is preferred to be high, that cost is preferred to be low, and that high safety is preferred to low cost. This kind of axiom captures the coupling between product characteristics with respect to designer preferences. When trade-off goals as defined in (11) cannot account for such preferences, axioms like this can be used.

Furthermore, there are situations where a *significantly* better value of one characteristic is acceptable if there is a corresponding but *very small* worsening of another characteristic. For example, say that some action could reduce the number of parts in a product by 10%, but

only with a 1% increase in cost. (11) does not allow this, even though in practice there could be very good reasons for it. One way to address this is to associate with each value a tolerance range or interval, or even to use fuzzy values. So long as variations arising from instances of (11) remain within the tolerable range, then the associated design actions are acceptable, and the goal is only violated when changes fall outside a value's tolerance interval.

We note that the particulars of the value model do not affect the validity of ADPM although they have obvious implications to the usability of the model. One can use fuzzy, crisp, or interval values without loss of logical rigour. Investigating the implications of different value models within ADPM is a topic for future research.

Also, (11b) is an incomplete rationale for a preference. There might be reasons why a designer might need to violate (11b). For example, a designer may have some past experiences that inform him to prefer designs that violate (11b) because in the long term, there will be a net benefit for the product, but for which there is no known rationale – except his experience. In such cases, an axiom can be development, like (12), to represent this. A detailed study of the impact of design rationale on ADPM is also a matter of future study.

This concludes the introduction of ADPM. Clearly more work is needed, but the author believes that the foregoing has shown ADPM's potential to formalise design processes.

4 Potential applications of ADPM

In this section, some possible areas of application for ADPM are discussed to give a sense of the system's potential breadth and impact.

The author has done some work in the use of variants of the *design structure matrix* in automotive engine design [17]. In that work, we captured bi-directional causal relations between engine systems and components. These relations could be described by actions that change the state of the engine design. Here, ADPM could formalise the actions and to reason about them. Since ADPM supports ALX's inference system, it may be possible to partly automate reasoning about engine design. For example, one might find alternate action sequences to achieve goal states by exploring the ADPM state space with intelligent agents, and selecting those that shorten lead-times by simplifying and re-sequencing design tasks. It may also be possible to develop libraries of action sequences that capture "corporate memory".

Another possible use of ADPM is in the development of new design processes (or, equally, in the redevelopment of existing processes). For example, ISO 16949 defines a process management standard for product development. The tasks set forth in the standard can be thought of as black boxes that together impose system-wide constraints on a design process. If an ADPM model of ISO 16949 were developed, it might be useful as the starting point of process development. Given the formal nature of ADPM, conformance to the ISO standard could be determined unequivocally. Logically manipulating statements within the ADPM framework rearranges design task definitions. The results of such investigations could lead to new process models that remain consistent with the ISO framework.

ADPM can be used directly as a research tool. One may pose hypotheses about how design happens in particular environments and then reason about the possible consequences of actions taken by designers. One could, for example, study the impact of different preferences on the sequencing of design tasks, to identify preferences that lead to improved processes. Such preferences could indicate *best practices*, and their identification without the usual benchmarking exercises, that can be disruptive to daily operations, could facilitate the propagation of those best practices, again leading to improved design capacity. For example, the

axioms of Axiomatic Design [3] can be written in ADPM; the logical implications of the axioms can then be explored formally. The author is currently beginning to study this point.

ADPM can also be used to simulate design processes. Such simulations could be used as test vehicles for new design methods. Clearly, such simulations would be only rough approximations of the real world. Still, the results of process simulations would be at least as useful as other simulations to identify problems and benefits of a system. The simulations can also be used to study the effect of different product modelling languages on design process efficiency. One might reasonably substitute AIM-D with an industry standard such as STEP; in that case, STEP would capture (a portion of) the design state, and ADPM can be used to describe and reason about processes for evolving STEP descriptions of products.

One more area where ADPM could find application is in enterprise modelling. Design processes are tightly coupled to the organisational structures in which they are practised. It makes sense that changes to a design process would likely require supporting organisational changes, such as changes to workflow or the hierarchy of responsibilities. ADPM could model business actions as easily as it can represent the technical perspective. One might expect that such an integrated business/technical model of design enterprises could be useful to initiate and manage overall business processes as they occur in technical environments.

Finally, one can expect new computer-based tools to be developed from ADPM. Logics can usually be implemented, at least in part, as tractable computational systems that can carry out deductive, inductive or abductive inferences. ALX3's inference system is a conventional deductive one, but one might consider other inference systems in the future. One can imagine a transparent AI application implementing ADPM that monitors the actions of a designer, automatically suggesting courses of action in response to the actions taken by the designer.

5 Conclusions

This paper has described a role for formal theories in the study and development of design processes. While many aspects of design cannot (and arguably should not) be formalised, there are other, objective aspects of designing that can benefit. The rigour available to reasoning agents who have formal tools at their disposal is substantial compared to informal and ad-hoc systems. The author believes that logic should form a foundational aspect of engineering designing, and presented one possible formalism for design processes, ADPM, which is based on a sound action logic, ALX3, and which is capable of representing at least some of the fundamental principles and strategies of designing. Of particular note is the system's capacity to represent the actions and beliefs of imperfect reasoning agents such as human designers. As a formal language, it permits the representation of general principles as well as specific facts about particular design processes, and it (currently) permits deductive reasoning about the design process models it represents. Clearly, ADPM is immature, but the author believes it has the potential to extend our understanding of designing and develop new and better systems to aid designers.

Acknowledgements

The author acknowledges the support of the National Sciences and Engineering Research Council of Canada for funding the work reported herein.

References

[1] Salustri F.A., "An artefact-centred framework for modelling engineering design," Proceedings of ICED '95, Vol. 1, The Hague, 1995, pp. 74-79.

[2] Salustri F.A., "A formal theory for knowledge-based product model representation," 2nd IFIP WG5.2 Workshop on Knowledge Intensive CAD, Pittsburgh, 1996, pp. 59-78.

[3] Suh N.P., "The principles of design," Oxford University Press, New York, 1990.

[4] Roozenburg N.F.M. and Eekels J., "Product Design: Fundamentals and Methods," John Wiley & Sons, Chichester, 1995.

[5] Antonsson E.K. and Cagan J., "Formal Engineering Design Synthesis," Cambridge University Press, London, 2001.

[6] Simon H.A., "The sciences of the artificial" 2nd ed., MIT Press, Cambridge, 1981.

[7] Alexander C., Ishikawa S., and Silverstein M., "A pattern language: towns, buildings, construction," Oxford University Press, London, 1977.

[8] Hubka V., and Eder W.E., "Engineering design: general procedural model of engineering design," Edition Heurista, Zurich, 1992.

[9] Pugh S., "Total design: integrated methods for successful product engineering," Addison-Wesley, London, 1991.

[10] Park H., and Cutkosky M., "Framework for modelling dependencies in collaborative engineering processes," Research in Engineering Design, Vol. 11, 1999, pp.84-102.

[11] Zeng Y., and Gu P., "A science-based approach to product design theory part I: formulation and formalisation of design process," Robotics and Computer Integrated Manufacturing, Vol. 15, 1999, pp.331-339.

[12] McNeill T., Gero J., and Warren J., "Understanding conceptual electronic design using protocol analysis," Research in Engineering Design, Vol. 10, 1998, pp.129-140.

[13] Salustri F.A., "Towards a logical framework for engineering design processes," 4th IFIP TC5 WG5.2 Workshop on Knowledge Intensive CAD, Parma, 2000, pp. 211-226.

[14] Huang Z., "Logics for agents with bounded rationality," Ph.D. Thesis, University of Amsterdam, The Netherlands, 1994.

[15] French M.J., "The opportunistic route and the role of design principles," Research in Engineering Design, Vol. 4, 1992, pp.185-190.

[16] Masuch M., and Huang Z., "A case study in logical deconstruction: formalizing J.D. Thompson's *Organisations in Action* in a multi-agent action logic," CCSOM Report 94-120, University of Amsterdam, The Netherlands, 1994.

[17] Lockledge J.C. and Salustri F.A., "Restructuring design communication using a design structure matrix," Proceedings of ICED '01, Vol. 1, Glasgow, 2001, pp. 27-34.

Corresponding author:
Filippo A. Salustri, PhD, P.Eng.
Ryerson University
Department of Mechanical, Aerospace, and Industrial Engineering
350 Victoria Street
Toronto, ON
M5B 2K3 Canada
Tel: 416-979-5000 x7749
Fax:416-979-5265
E-mail: salustri@ryerson.ca
URL: http://deed.ryerson.ca/~fil/