Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Computers in Industry 57 (2006) 331–341

COMPUTERS IN INDUSTRY

www.elsevier.com/locate/compind

Towards unified modelling of product life-cycles

G. Thimm [*], S.G. Lee, Y.-S. Ma

School of Mechanical & Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore

Received 11 November 2004; accepted 12 September 2005
Available online 28 November 2005

Abstract

This paper presents the potential of modelling a product's life-cycle using the Unified Modelling Language (UML). The potential benefits and limitations are discussed. An example of a vacuum cleaner is cited in support of this approach. Model consistency across the various life cycle stages of the product is of major concern and an algorithm for constraint management is proposed and prospective research directions highlighted.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Unified Modelling Language; Product life-cycle; Business process; Product model

1. Introduction

Product Life-Cycle Management (PLM) is a strategic business approach that consistently manages all life-cycle stages of a product, commencing with market requirements through to disposal and recycling (see Fig. 1). PLM involves a multitude of stake holders (e.g., customers, suppliers, and regulators), who require various levels of detail and representations of information. For example, the cost accountant may wish to track the costs incurred at certain life cycle stages; regulatory bodies are concerned with data on quality levels and end-of-life disposal options. The type and quantum of data modelled and how much of that data should be visible depends on the desired granularity.

At the February 2004 Georgia Tech-Industry Symposium in Atlanta, USA, a majority of the participants expressed their desire for an ontology for PLM under-pinned by a formal modelling process [private communication]. Twenty-eight percent of the 75 participants (of which 35 were from industry) identified "*single semantic PLM language*; *ontologies*; *data dictionaries*" as priority research thrusts. UML, a graphical modelling language used for computer soft- and hard-ware development [1,2] , offers just this, although it was conceived for object-oriented programming, and therefore, has limitations if applied to other disciplines. Not surprisingly, "*explicit PLM use—cases followed by a formal process modelling UML for PLM*" was voted in third place by 23% of the Georgia Tech-Symposium participants. It can be concluded that industry is in need of a formal modelling technique for PLM embedded in a computer-supported framework, towards which the authors consider this publication a first step.

Nevertheless, to the best knowledge of the authors, no modelling framework using a high-level, top-down modelling technique exists that captures all the aspects of a product's life-cycle stages and translates or connects them seamlessly. The modelling process is not a major concern and conventional approaches as described in [3,4] are suitable. Past experience shows that a translation of models (described in terms proper to each stage) is flawed by information losses and one-way data transfer. Consequently, a PLM modelling framework has to use a unique language for all stages, raising the question of the ideal candidate language. The authors are of the opinion that UML is the most promising candidate, for reasons detailed in Section 1.1. It is not intended here to provide a fine-grained, bottom-up modelling technique, as for example, in feature-based modelling [5,6] , but rather a complementary approach. On the other hand, the authors intend to implement a tool for the creation of such models and to allow them to interact with, for example, CAE tools.

1.1. Why UML is a good PLM modelling language

A PLM-oriented derivative of UML (PLMUL) has many advantages over other approaches and existing modelling

* Corresponding author. Tel.: +65 67904415; fax: +65 67911859.
E-mail address: mgeorg@ntu.edu.sg (G. Thimm).
URL: http://www.drc.ntu.edu.sg/users/mgeorg

0166-3615/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2005.09.003

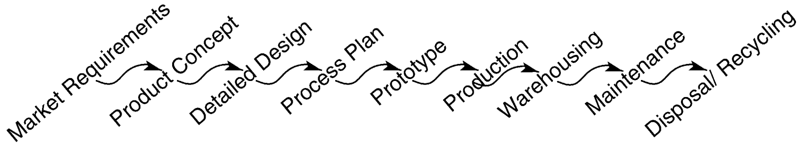


Fig. 1. Stages in product life-cycle management.

techniques (for example, IDEF0 [7] , work-flow modelling [8]). Some of these are:

- Industry has widely accepted UML as a modelling language:
  1. UML can model business processes to some extent [9,10] and underpins various commercial business process planning tools. It is compelling that some business processes are regarded as a part of a product life-cycle and *vice versa* (see Sections 2.4 and 2.5).
  2. UML is presently the most versatile modelling technique in industry.
  3. The Object Management Group [11](an international, not-for-profit consortium including several international companies) endorses UML and uses it in or with other highly industry relevant specifications. Examples of such specifications are the Model Driven Architecture (MDA) or the Common Warehouse Meta-model (CWM).
- The same syntax, that is, the same graphical symbols, can be used across product life-cycle stages. Although a developer only visualises or modifies a very limited number of views at a time, changes are reflected throughout the entire model, which fosters the consistency of PLM models across life-cycle stages. This also assures scalability: rather coarse models can be subsequently refined until a level of detail is reached that allows for example, prototyping or production.
- Modern machines often rely on control by complex software systems modelled using UML. Therefore, if the UML approach is extended to non-software systems, the same supporting tools can be build upon.
- Product models show, depending on the stage of modelling and the role of the persons involved, various levels of detail. UML attends to this and allows purpose-oriented views, favouring communication between designer, project manager, process planner, client, etc.
- UML is consistent with state-of-the-art concepts like functional design [12,13] and end-of-life disposal [14–16].
- UML is an information-rich representation; models can be tested for consistency, analysed, or translated into other representations (Gantt charts, bills-of-material, and so on; see Section 2.5).

1.2. Data modelling for PLM

Although the points stated in Section 1.1 show that UML is a good candidate for a PLM modelling language, its applicability to detailed product design of, for example, mechanical parts is unproven. Although UML is unable to directly describe geometry at this moment, the authors are confident that it is still applicable as it has the same foundation as certain feature-based modelling approaches (all use object-oriented techniques including inheritance, composition, association, and so on). Examples for such feature-based approaches are standard component libraries [17] , assembly feature templates [18,19] , multiple-view feature modelling [20,21] , and unified features [5,22].

These approaches differ mainly from the approach suggested in this paper in that they are very much focused on product details and on developing more universal modelling techniques (that is, a bottom-up approach, [20] has some top-down aspects). For example, in [22] , the STEP standard is extended to include *unified features*in order to rationalise process planning and design for instance, as well as to foster consistent modelling. A similar approach covering four different life-cycles from the conceptual design stage to product assembly is discussed in [20,21].

Opposed to this, the proposed *Product Life-Cycle Unified Modelling Language*(PLUML) is an attempt to model top-down, starting with general, macro models and then working down towards more detailed models. Certainly, either approach has its advantages and draw-backs and it is the hope of the authors that, as in software development, both approaches complement each other.

1.3. Possible issues in the UML modelling of PLM

As the examples in Section 2 show, the symbols and notions of UML are rather easily interpreted in a more general engineering setting. However, a closer examination reveals some potential problems. The most prominent one is probably the consistency of models: the propagation of changes in UML software-models as well as consistency checks among models. The need for consistency checks within and across engineering models is well recognised and researched on (see for example, [20,22–24]), but no product life-cycle encompassing approach exists. These approaches to maintain consistency are very much limited to part geometry and (low-level) feature compatibility.

Furthermore, even if the semantics of UML is expected to be powerful enough to fulfill PLM-needs, the UML symbols are visually not explicit and the existing associations not quite appropriate. In practice, an engineer probably would like to easily distinguish between electrical, mechanical, and entities of other categories. The consistency of PLUML models is further addressed in Section 3 , based on the case example of a vacuum cleaner. Other issues are the foci of future research as given in Section 4.

2. A PLUML case study: a vacuum cleaner

The purpose of the case example is to demonstrate the feasibility of using UML for PLM including business processes. The management of some life-cycle stages of the humble vacuum cleaner is used as an example and covers all product life-cycle stages in Fig. 1. PLUML models consist mainly of UML symbols: the designs for the vacuum cleaner and its major sub-assemblies (motor, casing, fan and air duct, and power regulator) are represented as objects. Most symbols are conform to the UML language definition [2]; others are defined by the authors and are consistent with it.

The case example is actually a common model although incomplete, for the sake of the size of this publication. Certain symbols occur in two or more diagrams of the partial models shown in Figs. 3–11(*views* in UML terminology). For example, the objects representing the major parts of a vacuum cleaner in Fig. 2 can be found again in the views of Figs. 3, 4, 9, 10, and 11. Or, a consumer, represented by the actor symbol labelled `Consumer`, participates in the use case diagram in Fig. 2, as well as in the end-of-life stage diagram in Fig. 9. These overlaps include also the view on business processes in Section 2.5. This shows that a product life-cycle and business processes may be represented by means of a common modelling language.

On the other hand, these views can accommodate the needs of a certain life-cycle stage or stake holder. Note that the concept of views here is different from that in [20,21] , which focus on geometry and mechanical features of a product.

2.1. View: conceptualisation; function, maintenance, and design constraints

Fig. 2 shows schematically how the design of the vacuum cleaner supports operational and maintenance functions. The large rectangle in the upper part of Fig. 2 groups the use cases of the vacuum cleaner. The links between the use cases and the actor symbols at the top of Fig. 2 indicate who is concerned with which use case. `Clean` and `Repair` are general use cases; each has two specialisations, as indicated by the arrows with triangular heads. The lower, large box represents the design of a vacuum cleaner while its components are represented in the boxes just above the large box as objects in the class `Design`. The dashed arrows from the use cases to the design objects indicate that the design objects are constrained by the use cases. The arrows with the diamond shaped heads from the design objects of the sub-assemblies to the design object of the vacuum cleaner indicate that the latter is *composed of* them (and possibly other).

2.2. View: detailed design; assembly and model consistency

The assembly view of a vacuum cleaner including its main modules (represented as UML objects) is outlined in Fig. 3. The emphasis of this view is on the interdependencies of the modules as well as the associations with external on objects.

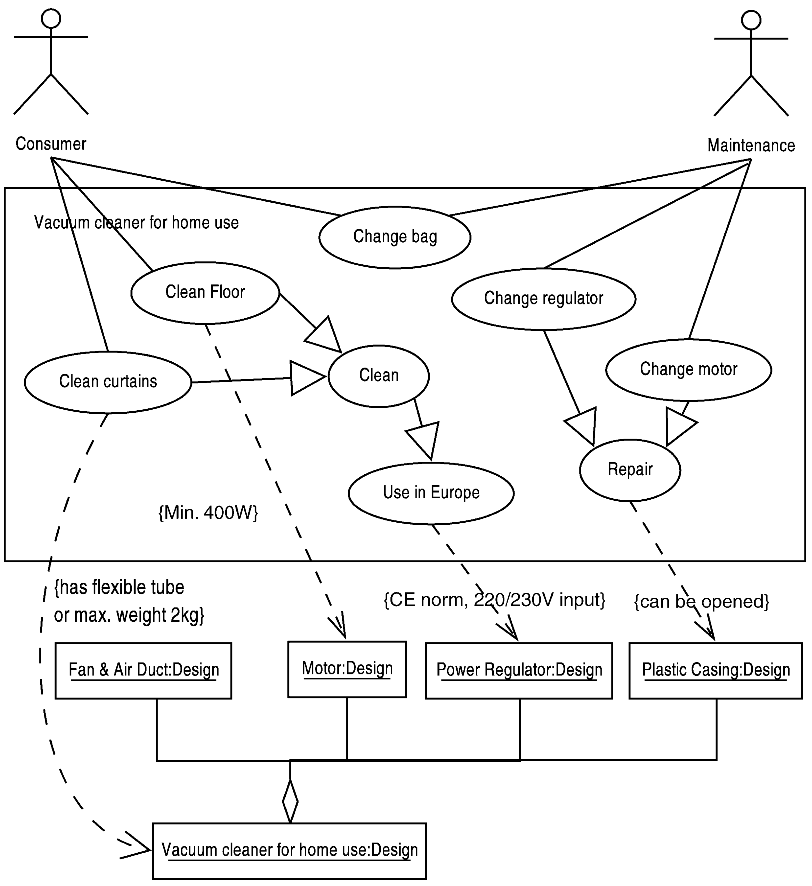


Fig. 2. Use and maintenance.

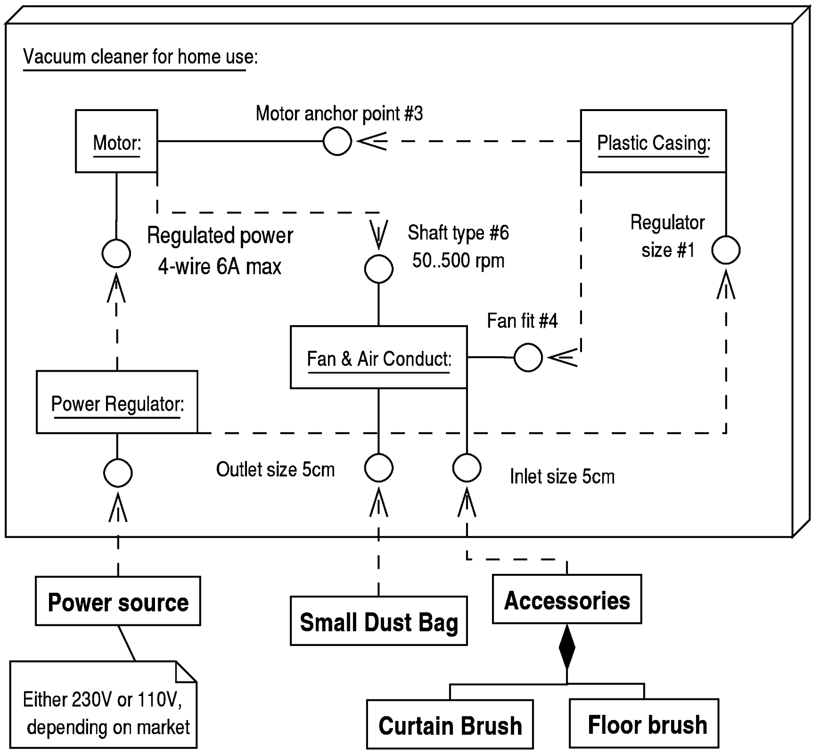


Fig. 3. Assembly of a vacuum cleaner.

For example, the specification of the power regulator depends on the voltage of the power supply in the targeted market. Modules in the diagram possess functional interfaces (circles linked by a solid line to the module), through which other objects are connected. In Fig. 3 , the motor has two interfaces: the regulated power input and the anchor points used to attach it to the casing. Furthermore, it uses an interface of the fan and air duct unit. In other words, modules are interchangeable if they have and depend on the same or compatible interfaces. Similarly for the vacuum cleaner, each module can be further decomposed into sub-modules or components.

In the case that this interchangeability does not exist, this view allows the dependencies to be tracked and thus ensures design consistency. For example, if a replacement motor has a different shaft, this change can be propagated to the fan on which a corresponding modification can be triggered automatically.

2.3. View: production, warehousing, and process planning

Fig. 4 exemplifies constraints related to production and warehousing. For example:

- The targeted production requires an automatic assembly line, and injection molding machines with an appropriate molding cycle time to meet the production target. In the example, it is assumed that only the top and bottom casings are produced

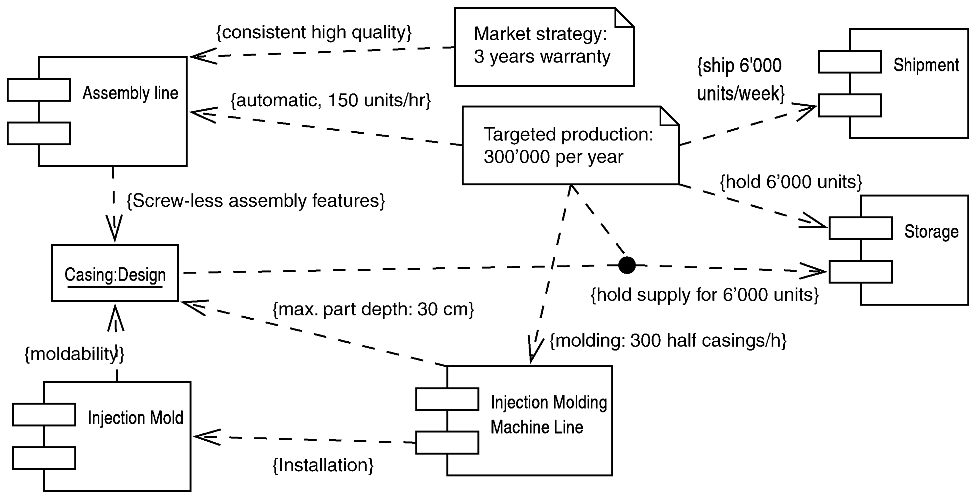


Fig. 4. Production constraints.

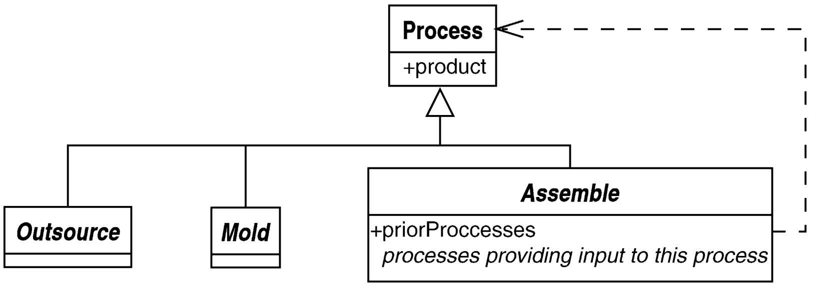


Fig. 5. Classes of processes.

in-house. Allocating 250 8-h working days for production results in a minimum output of 150 units per hour. This implies a minimum throughput of the injection molding machines of 300 half-casings per hour.

- The existing automated assembly line allows only for a screw-less design.
- The casing must be moldable and its parts not exceed a maximum depth of 30 cm in the direction of mold opening.
- For some reason, it is decided that production must run uninterrupted for up to 1 week, implying that (at least) the supply for 6000 units must be available at all times.

Fig. 5 concerns processes and their properties. A general process embodies a product that may be molded or outsourced, a sub-assembly, or the vacuum cleaner itself. Processes in the classes `Outsource`, `Mold`, and `Assemble` inherit this property. The `Assemble` class in turn has another field: a list of processes that provide input to this process. The dashed arrow indicates that an assembly process needs other processes.

Based on the definition of processes in Fig. 5, a process plan is defined in Fig. 6 as an object named `processPlan` (the name is underlined; the colon separates the name of the object from its class) of a specified class that is not here specified (no class name is given after the colon). Fig. 6 shows that a process plan is an aggregation of a number of objects in the class `Process` (that is, a set of out-sourcing, molding or assembly processes; the double box indicates that multiple objects of this kind are present) and is an instance of a process sequence (represented by the `Sequence` box). An almost trivial constraint usually links a product to a process plan: all parts must be the result of some process and the output of all processes (except the final assembly process) must be the input of another process.

Process interdependencies are often implicit in the structure of a product and the nature of the processes. These interdependencies in turn constrain the sequence of processes in a process plan. Fig. 7 exemplifies internal dependencies (though these can be automatically deduced from the process plan and design; many parts of the vacuum cleaner are left out) in the form of an assembly process hierarchy. Processes that have as output outsourced and molded parts do not depend on other processes, whereas assembly processes do. These dependencies are the cause of certain process precedences represented by the constraints on the processes sequence in Fig. 6.

2.4. View: disposal and recycling

This view demonstrates how associations can be redefined to suit particular modelling needs and at the same time to introduce visually easily identifiable entities. In Fig. 8, the dash-dot-dotted arrows are specialisations of an association between a product (or a part of it) and the company by which it is disassembled, recycled, or disposed of. The way a product is reprocessed is indicated by the role of the actor at the right side; the arrow heads allow for an effortless distinction. Throughout this publication, only these links are not part of the original UML definition (though they nonetheless conform to it as they are derived from an association). The general `re-processing` association on top defines the general properties, whereas the lower, horizontal lines are the specialised associations. The vertical arrows indicate from which association they are derived. These re-processing associations are used in Fig. 9.

Fig. 9 shows simple assembly models of domestic and industrial-grade vacuum cleaners. Both types (classes) are specialisations of the general (class) vacuum cleaner, distinguished by whether they have a metal or plastic casing. The component motor is further decomposed into three main constituents: rotor, stator and coils (in UML: is composed of a rotor, stator, and coils). Fig. 9 shows which vendor or contractor disposes of or recycles vacuum cleaners and their constituents.

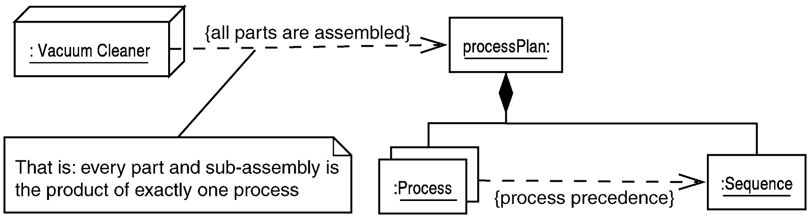

: Vacuum Cleaner
{all parts are assembled}
processPlan:
That is: every part and sub-assembly is the product of exactly one process
:Process
{process precedence}
:Sequence

Fig. 6. Process plans are composed of processes and a sequence.

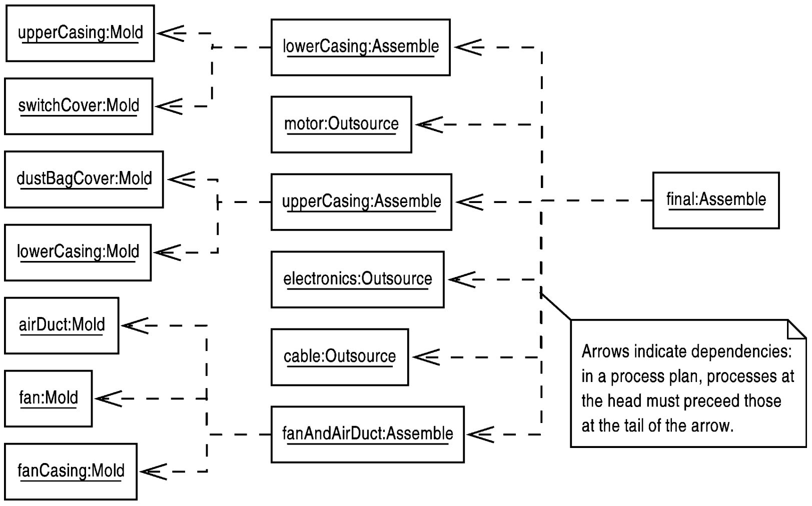


Fig. 7. Constraints on process sequences in a process plan.

For instance, the figure shows that the company `TakeApart` dismantles the vacuum cleaners with the exception of the motor, which is taken care of by the company `Scrap&Metal`.

The flow of material associated with the recycling of vacuum cleaners is shown in Fig. 10 in the form of an UML-sequence diagram. The arrows indicate who is passing which parts to whom (represented as UML messages). The texts above the arrows indicate the designs the object belong to. The costs for disposal and transport, as well as revenue from raw material, can be included into this diagram by the means of "money messages" and additional actors representing the transport companies. Diagrams similar to those of Figs. 8–10 can be drawn for procurement and assembly—additional associations would have to be defined and used to link the actors to the respective parts.

2.5. View: market requirements; PLUML and business processes

The main intent of this view is to demonstrate that a PLUML model can comprise entities used to model business processes and engineering entities. To this end, *purposes*, *objectives*, *results*, and *deadlines* as discussed in [9] are used in Fig. 11 to plan the development of an industrial vacuum cleaner. In order to simplify the figure, no *results* are shown. Furthermore, deadlines are adorned with the clock symbol while business objects are given a gray background. Both variations of the symbols comply with the UML standard.

Additionally, the view Fig. 11 shows how a time line can be represented in a PLUML model. For example, the design and fabrication of the tooling for the casing is implicitly constrained to not more than 5 months: the inner shape of the casing depends on the choice of the motor, the design of which can only be finalised once the objective `finish prototyping` is achieved in September 2006. Prototyping, design, material, and other changes can be reflected upstream. In the given example, tooling for the casing can only start after this date, leaving only 5 months to February 2007. The automatic creation of a simple Gantt-chart from the diagram in Fig. 11 is rather straight forward: the objectives and the corresponding deadlines give the time line for benchmarks; the constraints linking deadlines and dependencies among the

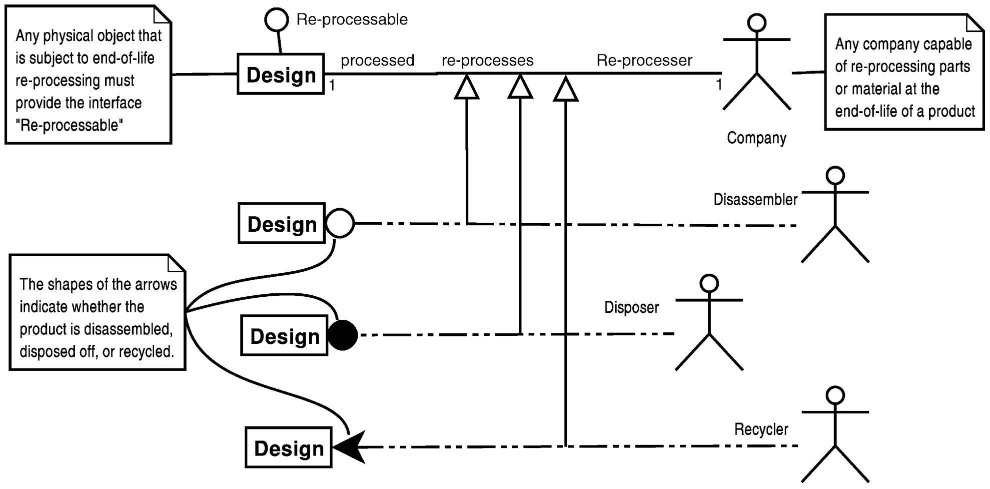

Any physical object that is subject to end-of-life re-processing must provide the interface "Re-processable"
Re-processable
Design
processed
1
re-processes
Re-processer
1
Company
Any company capable of re-processing parts or material at the end-of-life of a product
Design
Disassembler
The shapes of the arrows indicate whether the product is disassembled, disposed off, or recycled.
Design
Disposer
Design
Recycler

Fig. 8. The definition of end-of-life-cycle associations.

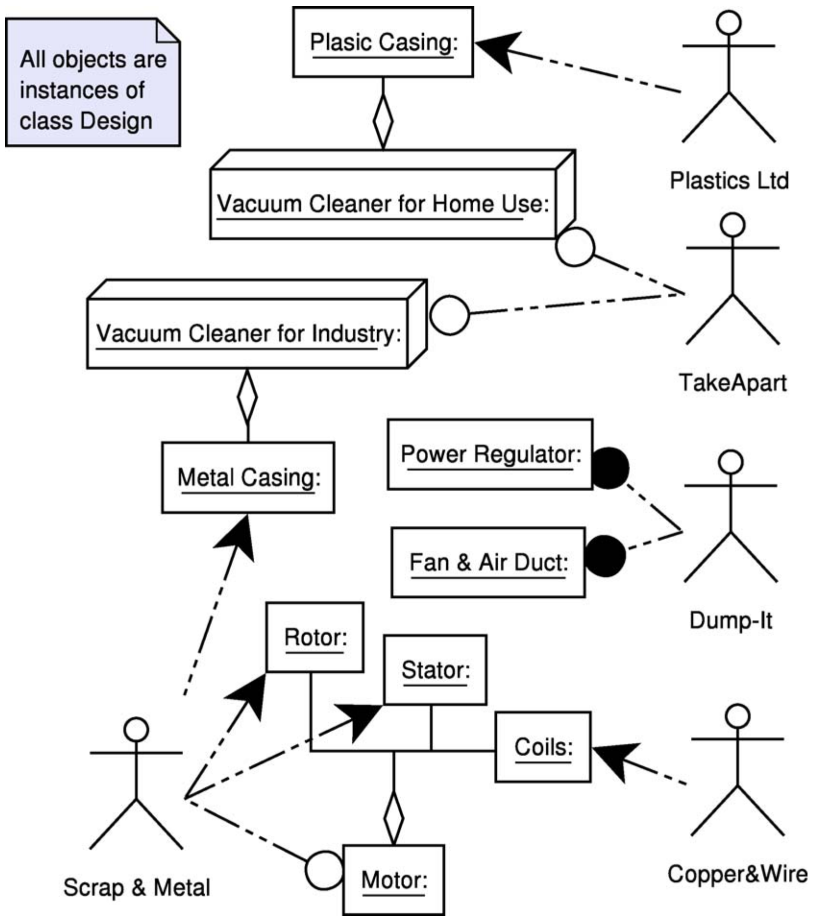


Fig. 9. The end-of-life responsibilities of vendors and contractors (compare Fig. 8).

various objects result in the times available to create the objects.

The link between business processes and the time line given by the deadlines in Fig. 11 can help to propagate constraints across the two models. Say, a competitor announces the launch of a similar product. This would trigger the addition of a corresponding objective, a change to the deadline of the `on market` objective, and a consequent revision of the other deadlines.

3. Model consistency and constraint management

Product changes take place over time and can affect one or more life-cycle stages. For instance, a design change may be necessitated by a manufacturing process; product servicing may necessitate design changes to facilitate servicing and repair. Unless the effects of such changes are reflected throughout the model, inconsistencies will settle in. Consequently, a mechanism is needed that maintains the consistency

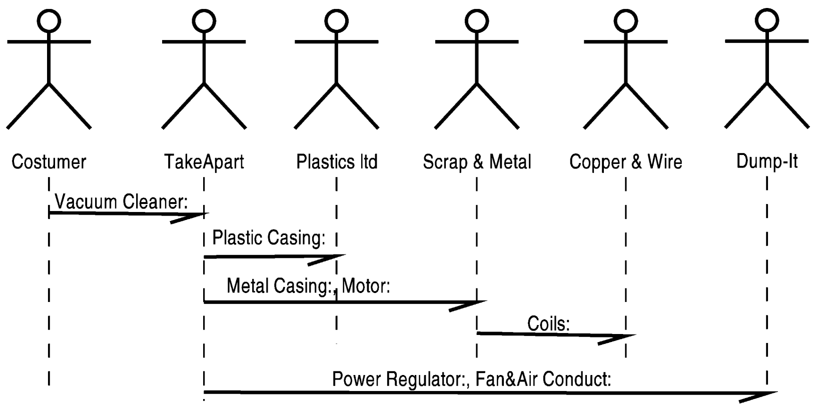


Fig. 10. The end-of-life stage: components of vacuum cleaners are recycled in different flows.

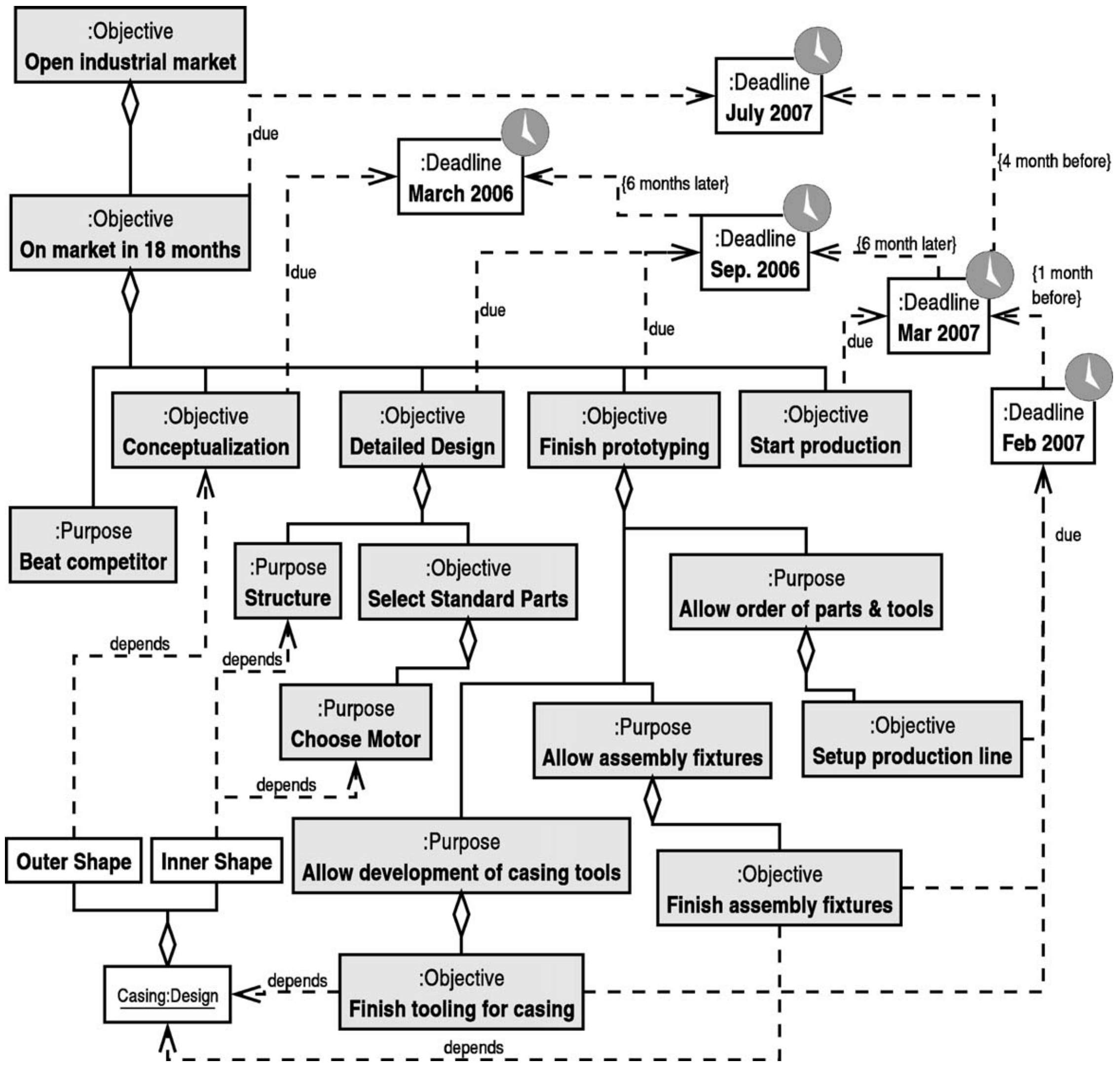


Fig. 11. A business plan interacting with engineering entities.

of the overall model (e.g., the design and process plan must be consistent; the final product must fulfill the objectives defined during product conceptualisation).

Furthermore, external constraints imposed by the production environment, regulation, budget, market and so on, must be linked to the model and, in the case of dynamic constraints (that is, constraints that change, emerge, or vanish while the product life-cycle model is created or used), the model has to be rectified. Within a PLUML model this could be achieved by adding an object to it representing the source of the constraint, and then using UML-constraints to link it to the constrained object in the product-related part of the model. Examples of dynamic constraints are:

- The level of available technology; new technology (newly-acquired machines, for example) changes to constraints on processes, and consequently process plans and designs (and usually lead to more efficient production). The change of constraints likely imply changes to many of the product life-cycle stages.
- Strategic product planning: at almost any time, the launch of competing products may alter the targets set during product conceptualisation and therefore the constraints of potentially all life-cycles stages.
- The market: legislation, safety, and environmental regulations, the number of targeted sold units, and other constraints depend on the country for which the product is destined.

3.1. A consistency maintenance algorithm for PLUML models

An obvious approach to model constraints in PLUML is to use concepts already present in UML: general constraints, interfaces, inheritance, and other types of associations. However, as UML does not provide a means to check or enforce constraints, the graphical modelling tool has to be augmented by an implementable strategy to do this.

This strategy (outlined in Section 3.2) differentiates constraints present in a PLM model according to their nature. The classes of constraints are defined as:

- *Functional*: a mathematical function states whether or not the constraint is fulfilled and allows the calculation of any parameter based on the changes given. For example, the specified power of the motor and the rating of the power supply (110 or 230 V) characterise the power regulator's current rating; if the former parameters are changed, the latter can be automatically deduced.
- *Technical*: though the constraint is defined in mathematical terms, the constraint can only be tested for validity. In the case of a conflict, human intervention is necessary to rectify the model. For example, a part's resistance to force exerted on it can be determined using finite-element methods, but in the case of an unacceptable result, only a designer can take corrective measures.
- *Subjective*: only a human is able to validate the constraint (e.g., judge aesthetic aspects of a product or perform experiments on a prototype). Constraints in this class may even be imprecise (an object is perceived to be more or less appealing by different persons; whether a product is sufficiently reliable is up to human judgement).
- *Literal*: a constraint given in a descriptive form. Even if the constraint has a precise technical significance and could be formulated in this form, this is not done for some practical reason. For example, the casing of the vacuum cleaner must be strong enough to withstand the force of the motor, which could be literally stated in a single phrase and "evaluated" by an experienced engineer. Alternatively, a technical constraint comprising forces and the design could be evaluated by the means of a finite-element model, which may be avoided for economical reasons.
- *Pending*: any of the above constraints can be further classified as "pending" if its validity cannot be evaluated as the product model is not yet sufficiently evolved. Two examples are:

  Production cost is a functional constraint typically asserted during product conceptualisation. On the other hand, it can only be ascertained when the design is near completion and production methods are determined.

  Constraints on in-house designs (the inner shape of the vacuum cleaner) are pending until out-sourced parts (the model of the motor) are selected.

The next concern is the strategy to resolve conflicting constraints. Clearly, only a semi-automatic approach to achieve and maintain consistency is viable. Due to the large number of constraints, holistic approaches, for example, variations of general-purpose optimisation techniques (hill climbing, simulated annealing, and so on) [25] , logic and constraint logic programming [26,27] , truth maintenance systems [28] , genetic algorithms [29] as well as expert or rule-based systems [25] cannot be applied easily and if so, only locally. A further obstacle in using these approaches is the very nature of the constraints. In particular, in the earlier modelling stages, a majority of the constraints are pending, literal, and/or subjective.

In order to formalise constraints, UML has to be complemented by a constraint-modelling language. The object constraint language (OCL) [30–32] is likely a good approach for doing so. However, as OCL is very much tailored for software engineering purposes and does not offer an approach for constraint maintenance, much work remains to be done. For example, constraints cannot be placed on classes (that is, all instances derived from them); a general constraint like "*All wires* (read: instances of class `Wire`) *must be at a minimal distance from hot parts* (objects that have a flag `hot` set)" cannot be formulated easily.

Furthermore, given that PLUML models are potentially very large with possibly many inconsistent, subjective and pending constraints, the strategy has to be able to handle situations in which an automated updating can result in endless loops.

3.2. The algorithm

In order to resolve these problems, the following strategy is proposed: the resolution of conflicts and the further development of the model are integrated into an approach that allows a stepwise refinement of the model and a possible backtracking to earlier stages in the model. Each step in the refinement (which may include several modifications to the model) is characterised by a model M_t and a set of unfulfilled constraints C_t at time t. The models M_t evolve with time into models with ever greater detail and the set of unfulfilled constraints towards the empty set. The following algorithm realises this strategy (a state is an atomic property of a class, an object, an association, and so on, including the presence or absence of the property):

(1) Replicate M_{t-1} and C_{t-1} in order to obtain M_t and C_t.
(2) Create an empty set of states S. This set is used to track modifications to model M_t.
(3) Prompt the user (that is, one of the stake holders) for
  (a) constraints to be changed and constraints to be added to or removed from C_t,
  (b) changes to states of PLUML symbols (that is, the electronic description of associations, objects, classes, and so on) in M_t, and
  (c) PLUML symbols to be added to or removed from M_t.

  Then modify the model M_t accordingly.
(4) For all constraints $c \in C_t$ do the following:
  (a) If the constraint c is functional or technical (only then it can be tested automatically) and is fulfilled, do nothing.
  (b) If c is functional and unsatisfied, the action taken depends on whether the states c constrains were already modified in the current iteration of this algorithm:
    - If constraint c can be satisfied by modifying states not present in S, the states are changed accordingly and then added to S.
    - If the constraint c cannot be satisfied by such a modification of non-current states, report c to the user.
  (c) In all other cases (that is, the constraint is technical and unsatisfied, subjective, pending, or literal), report c to the user.
(5) If the user desires, revoke the changes by eliminating M_t and C_t. Otherwise, increment t and reiterate the algorithm until C_t is empty and M_t is a completed life-cycle model.

This algorithm still needs some refinements: for instance, it is not necessarily desirable that all constraints are evaluated and presented to the user after each modification (steps 4b and 4c). Thus, constraints in the set C_t must be prioritised ([20] addresses this problem and suggests assigning higher priority to constraints of early life-cycle stages) and, for example, events triggering their reconsideration established.

Note that this algorithm possesses properties usually not present in constraint solving algorithms: backtracking is only initiated if the user desires it (see step 5). The interim solutions are unstable as states may change their values continuously back and forth or even diverge. However, the algorithm has the advantage that a single iteration is guaranteed to terminate (each state variable is modified at most once in model M_t), and that a user has a very good control over the direction the model is developing. A possible way to instill user friendliness and the potential for collaborative design and production into the approach is combine it with an issue (a.k.a. bug or defect) tracking system. In this framework, once the modifying stake holder approves the changes, the unresolved constraints from steps 4b and 4c of the algorithm are routed to the stake holders qualified to address them.

4. Conclusion and future work

Product life-cycle management requires a modelling framework showing the associations among the life-cycle stages, business processes, and stake-holders. The potential of UML as a medium to model a product's life-cycle is presented using the example of a vacuum cleaner. One of the major challenges of UML modelling is to ensure consistent product life-cycle models with respect to constraints within the model as well as with those imposed from outside, such as by safety regulations or by the market. A possible approach to enforce and maintain model consistency is discussed. An algorithm is proposed. The approach recognises that many constraints are subjective or depend on missing information, and that many constraints require human intervention to resolve.

It is the authors' intention to extend the existing UML to allow constraint management and to further integrate it with conventional engineering tools providing functionalities like CAD and CAM. However, before an industrial application can happen, the following issues (research foci of the authors) must first be addressed:

- The multitude of interactions among and the inter-dependencies of parts in a complex machine necessitate the use of a template library of common sub-assemblies and parts.
- Tools to translate PLUML models into representations such as bills-of-material or Gantt charts need to be developed.
- Knowledge must be represented and embedded in the product model and managed to assure consistency with business practices, current technologies, and the production environment.

In conclusion, the authors postulate that a UML model of PLM is feasible and can potentially improve current practices in product life-cycle management.

References

[1] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Object Technology Series, Addison-Wesley, 1999.

[2] J. Rumbaugh, I. Jacobson, G. Booch, The Unified Modeling Language Reference Manual, Object Technology Series, Addison-Wesley, 1999 .

[3] R.G. Askin, C.R. Standridge, Modeling and Analysis of Manufacturing Systems, John Wiley & Sons Inc, New York, 1993.

[4] A.M. Law, W.D. Kelton, Simulation Modeling and Analysis, third ed., McGraw Hill Inc., New York, 2000.

[5] G. Chen, Y.-S. Ma, X. Ming, G. Thimm, S.G. Lee, L.-P. Khoo, S.-H. Tang, W.F. Lu, A unified feature modeling scheme for multi-applications in PLM, CE2005, in: The 12th ISPE International Conference on Concurrent Engineering: Research and Applications, Ft. Worth, Dallas, USA, 2005.

[6] G. Chen, Y.-S. Ma, G. Thimm, S.-H. Tang, Knowledge-based reasoning in a unified feature modeling scheme, Computer-aided Design and Applications 2 (1–4) (2005) 173–182.

[7] Air Force Wright Aeronautical Laboratories Integrated Computer-Aided Manufacturing (ICAM) Architecture, Part II, vol. IV, Function Modeling Manual (IDEF0) AFWAL-TR-81–4023 (June), Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio 45433, 1981.

[8] B. Kiepuszewski, Expressiveness and suitability of languages for control flow modelling in workflows, Ph.D. Thesis, Faculty of Information Technology, Queensland University of Technology (November 13, 2002). http://tmitwww.tm.tue.nl/research/patterns/download/phd_bartek.pdf.

[9] C. Marshall, Enterprise Modeling with UML: Designing Sucessful Software through Business Analysis, Object Technology Series, Addison-Wesley, 1999.

[10] H.-E. Eriksson, M. Penker, Business Modeling with UML: Business Patterns at Work, John Wiley & Sons, New York, 2000.

[11] Object Management Group, Inc., Home page: http://www.omg.org, 2004.

[12] Y.-M. Deng, G.A. Britton, S. Tor, Constraint-based functional design verification for conceptual design, Computer-Aided Design 32 (2000) 889–899.

[13] G.A. Britton, S.B. Tor, Y.C. Lam, Y.-M. Deng, Modelling functional design information for injection mould design, International Journal of Production Research 39 (12) (2001) 2501–2515.

[14] S.W. Lye, S.G. Lee, M.K. Khoo, An environmental design evaluation tool, Journal of Engineering with Computers 18 (2002) 14–23.

[15] S.W. Lye, S.G. Lee, M.K. Khoo, A design methodology for the strategic assessment of a product's eco-efficiency, International Journal of Production Research 39 (11) (2001) 2453–2474.

[16] S.G. Lee, X. Xu, A simplified life cycle assessment of re-usable and single-use bulk transit packaging, Packaging Technology and Science 17 (2004) 67–68.

[17] Y.-S. Ma, S.B. Tor, G.A. Britton, The development of a standard component library for plastic injection mould design using an object-oriented approach, International Journal of Advanced Manufacturing Technology 22 (2003) 611–618.

[18] Y.-S. Ma, G.A. Britton, S.B. Tor, L.-Y. Jin, G. Chen, S.-H. Tang, Design of a feature-object-based mechanical assembly library, Computer-Aided Design & Applications 1 (1–4) (2004) 387–404.

[19] W. van Holland, W.F. Bronsvoort, Assembly features in modelling and planning, Robotics and Computer Integrated Manufacturing 16 (4) (2000) 277–294.

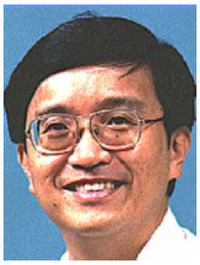
[20] W.F. Bronsvoort, A. Noort, Multiple-view feature modelling for integral product developement, Computer-Aided Design 36 (10) (2004) 929–946.

[21] A. Noort, G.F.M. Hoek, W.F. Bronsvoort, Integrating part and assembly modelling, Computer-Aided Design 34 (12) (2002) 899–912.

[22] G. Chen, Y.-S. Ma, G. Thimm, S.-H. Tang, Unified feature modeling scheme for the integration of CAD and CAx, Computer-Aided Design & Applications 1 (1–4) (2004) 595–602.

[23] S.-H. Tang, Y.-S. Ma, G. Chen, A web-based collaborative feature modeling system framework, in: S. Hinduja (Ed.), Proceedings of the 34th International MATADOR Conference, Springer-Verlag, Manchester, United Kingdom, 2004, pp. 31–36.
[24] R. Bidarra, W.F. Bronsvoort, Semantic feature modelling, Computer-Aided Design 32 (2000) 201–225.
[25] A.A. Hopgood, Intelligent Systems for Engineers and Scientists, second ed., CRC Press, 2001.
[26] L. Sterling, E. Shapiro, The Art of Prolog, MIT Press Series in Logic Programming, The MIT Press, 1986.
[27] J. Jaffar, M.J. Maher, Constraint logic programming: a survey, Journal of Logic Programming 19/20 (1994) 503–581.
[28] K.D. Forbus, J. De Kleer, Building Problem Solvers, MIT Press, 1993.
[29] P.J. Bentley (Ed.), Evolutionary Design by Computers, Morgan Kaufman, Inc., 1999.
[30] J.B. Warmer, A.G. Kleppe, The Object Constraint Language: Precise Modeling with UML, The Addison-Wesley Object Technology Series, second ed., Addison-Wesley, 1999.
[31] Klasse Objecten, Inc., Welcome to the OCL center: http://www.klasse.nl/ocl/, 2004.
[32] Object Management Group, Inc., UML 2.0 OCL Final Adopted Specification, document ptc/03–10-14 (October 2003). http://www.omg.org.

G. Thimm joined the Dalle Molle Institute for Perceptual Artificial Intelligence in Switzerland, after receiving in 1992 a diploma in computer science from the University of Karlsruhe. There, he performed research on higher order neural networks in preparation of a doctor's degree in technical sciences, which he obtained in 1997 from the Swiss Federal Institute of Technology in Lausanne. After some research on image recognition, he joined the Nanyang Technological University (Singapore) in 1999 as research fellow and was converted to assistant professor in 2000. His research interests are in the application of artificial intelligence and graph theory, currently focused on process planning, design for manufacturability, and product life-cycle management. He is member of several editorial boards of internationally recognised journals.



Stephen Siang-Guan Lee is an associate professor and Director of the Design Research Centre in the School of Mechanical & Production Engineering, NTU. Dr. Lee's bachelor degree was in mechanical engineering and earned his MSc in advanced manufacturing technology from the UMIST (UK) and PhD (NTU) in 1985 and 1997, respectively. Prior to joining Nanyang Technological University in 1983, he was engaged in design and consulting assignments in local manufacturing companies. As a faculty member of the School of Mechanical & Aerospace Engineering, Dr. Lee teaches courses related to concurrent engineering, design for the environment, product safety in which areas he has been consulted by local industry. His research interests are in design methodology, product packaging, knowledge-based design and manufacturing, product life cycle management and dynamic enterprise collaboration. Dr. Lee is a registered professional engineer, a Fellow of the Society of Manufacturing Engineers, and a member of the ASME. An active member of the SME since 1987, Dr. Lee held senior appointments in the Singapore Chapter, culminating with the Chapter Chairmanship in 1990. In 1992, the Society of Manufacturing Engineers conferred on him its *Award of Merit* and in 1998, he was elected to its College of Fellows.

Yongsheng Ma is currently an associate professor at school of Mechanical and Aerospace Engineering, Nanyang Technological University (NTU), Singapore. His main research areas include product lifecycle management, feature-based product and process modeling, and engineering IT solutions. Graduated from Tsing Hua University, Beijing with BEng degree in 1986, he further studied at UMIST, UK and achieved Msc and PhD degrees in 1990 and 1994, respectively. He started his career as a Lecturer at Ngee Ann Polytechnic, Singapore from 1993 to 1996. From 1996 to 2000, he worked in Singapore Institute of Manufacturing Technology, as a research fellow, project leader, senior research fellow and group manager. He joined NTU since Sept. 2000.