A toolkit and methodology to support the collaborative development and reuse of engineering models

Zdenek Zdrahal, Paul Mulholland, Michael Valasek,
Phil Sainter, Matt Koss, Lukas Trejtnar

Knowledge Media Institute,
The Open University,
Walton Hall, Milton Keynes, UK
{Z.Zdrahal, P.Mulholland, P.Sainter, M.Koss, L.Trejtnar}@open.ac.uk

Faculty of Mechanical Engineering
Czech Technical University,
Karlovo nam. 13, 12135 Praha 2
Czech Republic
valasek@felbr.fsik.cvut.cz

Abstract. Engineering design is a knowledge intensive activity. Design is characterized as comprising a number of phases from requirements to detailed specification. Transitions between the design phases stem from decision-making processes supported both by generally available domain and design knowledge and from unique and original knowledge worth of recording for future reuse. The Clockwork project addresses the problems of representing, sharing and reusing knowledge in a collaborative design engineering environment. The Clockwork knowledge management methodology is supported by a web-based toolkit that allows designers to share, formally and informally annotate and reuse engineering models. Reuse is supported through the semantic search of engineering models and of the design rationale by which the engineering models were developed. The approach has been successfully deployed in two industrial organisations.

1 Introduction

Clockwork is a European Union funded R&D project whose aim is to develop methodology and software tools for capturing and sharing valuable engineering knowledge generated and used in design processes. There is increasing pressure to reuse design knowledge in order to expedite the design process and reduce time to market. Generally, design models have to be described and annotated in order that they can be effectively understood and reused. Annotating design models and capturing design rationale imposes additional overheads. It can disrupt the design process, and can require the designer to spend a significant amount of time off task. Clockwork provides methodology and supporting tools for efficient, semantic based annotation of design knowledge focused around crucial decisions in the design process. The Clockwork

toolkit includes a powerful knowledge retrieval mechanism which combines semantic web technologies with informal description of engineering models.

The Clockwork approach has been tested on two pilot applications: modeling and simulation of dynamic systems in mechanical engineering and the development and support of specialized thermal technologies. Both applications have been developed in collaboration with Clockwork industrial partners, INTEC GmbH and ELOTHERM, GmbH.

2 Modelling Design Processes

Design is considered to be an ill-structured problem [5], [9]. This means that prior to the start of the design process the goal specification is not completely known and requirements are often inconsistent. The task specification and its solution are therefore worked out simultaneously.

For ill-structured problems only "weak" problem solving methods (such as generate-and-test) exist and weak methods provide only weak results [5]. By applying domain knowledge, ill-structured problems can be converted into a number of well-structured problems - "the problem is well-structured in the small, but ill-structured in the large" [9, p.190]. For well-structured problems "strong" problem solving methods i.e. specialised algorithms exist which offer much better solution. For this reason knowledge plays a crucial role in design.

In engineering, design processes have been studied [1], [2], [8] with the aim to better understand the underlying activities, organize them and, whenever possible, provide methodological support. Two types of design processes are distinguished in [1]:

- *descriptive* design processes are focused on the analysis of the tasks and their sequences as they typically occur in design processes, and
- *prescriptive* design processes, which, in addition, attempt to provide a sound methodological guidance for these tasks.

Knowledge plays an important but different role in descriptive and prescriptive types of design process. The former rely mainly on heuristic knowledge while the latter ones emphasize the role of logical reasoning as the major inference vehicle for design reasoning. Though individual types differ, they subscribe to a similar set of constituent tasks which divide the design process into four *phases*. They are: *requirement specification*, *conceptual design*, *embodiment design* (preliminary design) and *detailed design*. Each design phase constitutes a design task, each task results in an *engineering model*. The tasks are executed sequentially each task refining the model produced by the previous task. The design process is nonlinear: the designer backtracks or iterates in a loop if he or she finds out that the current solution has reached a dead end. This is a consequence of design being an ill-structured problem and therefore search being an appropriate problem solving method. However, regardless of the sequence of steps taken in the design process the final design must include all phases.

Each task is self-contained in the sense that it has an associated set of objects from which the solution is constructed and a language for expressing the solution. The design process can be viewed as a sequence of transformations between representations of the design phases, i.e. based on the requirements specification the designer con-

structs a conceptual model which is further elaborated into an embodiment model and eventually to the final detailed design. Each of these transformations refines the outcome of the previous phase by imposing additional assumptions and by reasoning supported by design and domain knowledge. The design result is associated with assumptions and justifications for each individual decision made along the design process. A generic model for design reasoning includes a structured account for design hypotheses, goals, actions and justifications [3].

Though it has been accepted that knowledge plays an essential role in design, support for knowledge reuse in practical applications is still insufficient. Standard design packages do not include suitable tools and therefore valuable design knowledge situated in the context of a concrete problem and solution is usually lost. Despite the obvious benefits, design rationale is rarely recorded. Often, even the results of the intermediate design phases, such as the conceptual design, are discarded. Designers tend not to make their decisions explicit because the design tools usually do not provide user-friendly facilities for annotation and knowledge capture and without proper tools and methodology, such activity would distract them from designing and consume their time [7]. However, despite of the obvious lack of support, knowledge reuse is a standard and widely used design technique.

Attempts have been made to tackle the problem by allowing the designer to carry on his/her work and to reconstruct the design reasoning in a non-intrusive way.

The Rationale Construction Framework (RCF) project at SRI International aims at inducing and interpreting design rationale from the trace of designer's activities when interacting with the CAD tool [4]. The project is focused on detailed design, i.e. the last of the design phases. From the design event log, RCF abstracts a symbolic model of the emerging design which is used to reason about designer's intent.

The ASSISTANCE system, under development at MIT [6], applies similar ideas at the other end of the "design phase" sequence. The system interprets designer's sketches and spoken comments to derive a causal model of the conceptual design and allows reasoning about the structure and behaviour of the designed artifact.

Unlike RCF and ASSISTANCE, the Software Integration and Design Rationale Capture Tool SSPARCY [10] supports only semiautomatic annotation of design rationale. The tool has been developed to support simulation and modeling of aerospace design. Similarly as RCF, the system is linked to low level design tools, such as MATLAB and Excel, from which it extracts, parses and organizes design data and records the design history. The designer is encouraged to associate textual annotation with the data as design rationale.

3 Clockwork Knowledge Modeling

Clockwork addresses the problem of sharing and reuse of design knowledge based on the following philosophy: If professional designers document their reasoning to communicate their ideas among themselves or to reuse them in the future, they record only decisions which are valuable. These are for example: new and unique design "tricks" discovered when solving the case; decisions which are meaningful only within the specific context; decisions which relate to an assumption made earlier or

which respond to a concrete design requirement imposed by the client or a collaborating partner. Standard design knowledge which can be found in any textbook is not recorded, because it would only increase the complexity of documentation, decrease the efficiency of information retrieval and take designer's time. Moreover, this knowledge is implicitly represented in the design and each experienced designer can immediately find it. Clockwork therefore focuses only on important knowledge, and does it across all design phases.

Design phases are represented by *engineering models* and transitions between them. Engineering models are the result of work of design engineers, when designing cars, complex technologies, control systems or simulation models. They embody an implicit form of design knowledge. Engineering models often need additional information to define their applicability. For example, control systems or simulation models are designed for certain input and performance criteria. This contextual information propagates in the design process and is refined together with the engineering model. In Clockwork, we introduce the concept of *design world* which includes the engineering model and all associated contextual information. Therefore we say we have a requirement world, conceptual world etc.

Knowledge description is constructed on the following principles:

- Depending on the problem and the available support, engineering models are developed using different tools, such as CAD packages, MATLAB/Simulink, Solid Edge or they might be just hand-drawn sketches. However, at the knowledge level engineering models are represented in a unified way as *knowledge models* in terms of ontologies and knowledge bases. Ontology is an explicit conceptualisation of the domain. It is a dictionary of engineering concepts i.e. formally defines all available concepts in terms of their properties and relation with other concepts. Concepts are represented as classes of an object oriented language. Knowledge bases contain instances of classes pertinent to a specific case.
- The association of knowledge models with engineering models allows for reasoning about the engineering models. Knowledge models need not describe engineering models completely, only important model components need to be included. The selected components are associated with classes in the corresponding ontology by means of class instances created in the knowledge base. The instances are called *semantic indexes* because by associating objects of the engineering model with concepts in the ontology they assign the meaning to the object.
- Refinement of engineering models can be described as relations between corresponding formal knowledge models.
- Formal knowledge models can be further supported by informal text annotations or sketches.

An example of a knowledge model and its associations with the engineering model is shown in Fig. 1. The class "Lever" has two instances, L1 and L2 and the class "Joint" has an instance J. These are used as semantic indexes describing the components of the lower left arm of the mechanism shown in Fig. 1. Though the drawing is a non-executable conceptual model, the association with a formal knowledge model makes it possible to reason about the three model components.

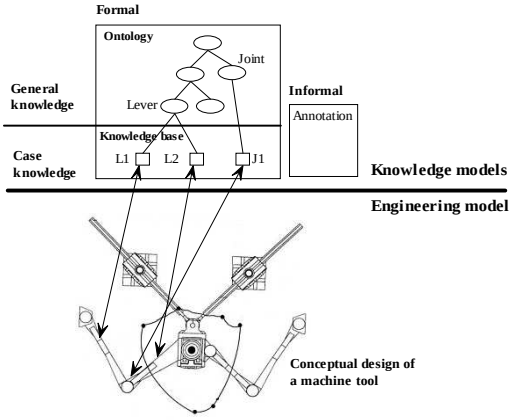

Fig. 1 Clockwork knowledge types**.**

Based on these principles the Clockwork methodology for sharing and reuse of engineering knowledge has been developed and a support toolkit has been implemented. Core to our approach is the web-based Clockwork Knowledge Management Tool (CKMT) that allows designers and other stakeholders such as customers and domain experts to explain, share, discuss and collaboratively develop engineering models. Engineering models stored in the CKMT repository can be annotated formally by association with domain ontologies or informally using notes or sketches.

For developing ontologies, Clockwork provides the Apollo ontology editor, which allows the user to define classes, their structure and hierarchies and create instances. The ontology and knowledge base created by Apollo can be exported into various formats including XML, RDF, Lisp and Operational Conceptual Modelling Language (OCML). The current version of Apollo is available at http://apollo.open.ac.uk.

CKMT has been tested on two case studies.

4 Case Study 1: Simulation and Modelling of Dynamic Systems.

The knowledge sharing and reuse framework introduced in the previous section has been instantiated for the design of simulation models of dynamic systems and implemented using the CKMT. Since this is a typical design problem, the four design phases introduced in Section 2 have a straightforward interpretation in the simulation and modeling context. They are:

Requirement analysis. When constructing a simulation model, the real world system to be modeled specifies the requirement for the model design task.

Conceptualisation. From the real world system, the conceptual model is elaborated. Conceptual model specifies the components which will be included in the design, their properties and interactions accompanied by the description of their function from which causal and functional explanation can be inferred. For example, for the horizontal machine tool DYNA-M (real world system) the conceptual model shown in

Fig. 1 consists of two arms in the lower part, a spindle in the middle and two moving screws with drives in the upper part. Drives and moving screws convert rotation to translation and allow positioning of the spindle.

Modelling. In the next step the conceptual model is converted into the corresponding physical model composed of ideal modelling objects. The physical model is still independent of the simulation environment which will be used for implementation.

Simulation. Finally, the physical model is implemented in a selected simulation package such as MATLAB/Simulink or Simpack. The four design worlds shown in Fig. 2 are called Real World Object (RWO), Conceptual Model Object (CWO), Model World Object (MWO) and Simulation World Object (SWO).

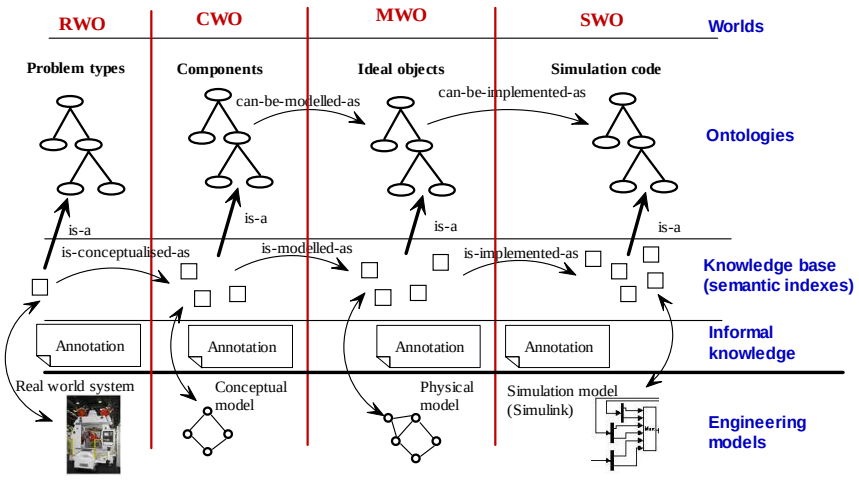

Fig. 2. Four-world simulation and modeling framework

Developing knowledge models

Each world includes additional information needed as a context for model development: The question to be answered by the modeling experiment, is associated with the Real World Object, converted into the modeling objective at the conceptual level and further refined as input and output specification for the physical model. Results of the simulation experiment are interpreted in terms of the physical and conceptual models and the real world system. Assumptions under which the model has been elaborated are associated with each world.

Fig. 2 also shows relations between pairs of knowledge models. Relation *is-conceptualised-as* is one-to-many, relations *is-modelled-as* and *is-implemented-as* can be in general many-to-many, though the model refinement usually means that one object of conceptual/physical model is modeled/implemented as several objects of physical/simulation world and therefore these relations are one-to-many. Other predefined relations included are *is-part-of* for membership of an object to the model and *has-status* to indicate whether the model was a success or failure. The user can define additional relations for problem description.

Relations across worlds capture important modeling knowledge. For example, we may express that a shaft of the conceptual model *is-modelled-as* three masses and two ideal springs in the physical model. When the designer builds a model of any of the worlds, he/she selects the object which carries interesting design knowledge, gives it a name and associates it with the corresponding class in the ontology, i.e. creates a semantic index. Then the object becomes a part of the knowledge model, can be included in relations and can be used by the reasoning algorithm. An example of the annotation of a revolute joint in a physical model is shown in Fig. 3.

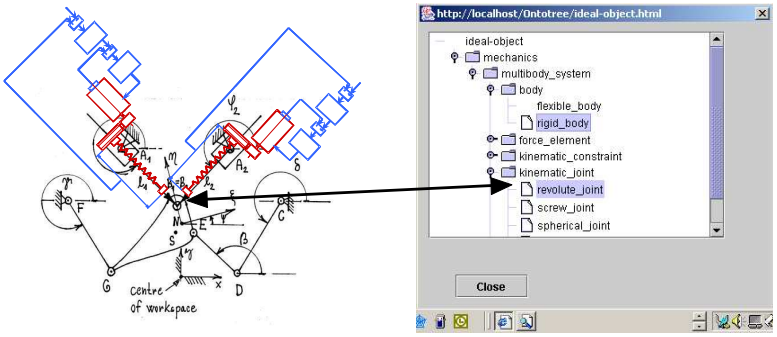

Fig. 3. Defining semantic index in the physical model

In accord with Fig.1 and 2, CKMT also allows the designer to provide informal annotation in the form of plain text or sketches. The annotation is included as one of the documents associated with each world and text can be accessed by keyword search.

Querying knowledge models

CKMT query engine offers a number of options which can be combined. The simplest query is the keyword search which can be applied to all annotations and assumptions. However, this mechanism is usually not powerful enough and therefore does not provide good results. For this reason CKMT introduces semantic search which exploits ontologies, semantic indexes and inter-world relations described above.

Semantic indexes and ontologies allow the query engine to reason in class hierarchies and use relations as executable rules. For example, the query “Show me an example of modelling kinematic joint” will retrieve the model of DynaM shown in Fig.3, because the semantic index for a revolute joint has been created (see the arrow in Fig.3) and a revolute joint is a kind of kinematic joint. The underlying representation of knowledge models in OCML allows for defining complex relations. For example, the user can define a new relation *successful-simulation-of-component* for models ?x and component ?y as follows:

?x *is-successful-simulation-of-component* ?y
if

component ?y *is-part-of* conceptual model ?z, and
?z *is-modelled-as* ?w, and
?w *is-implemented-as* ?x, and
?x *has-status* success.

If the variable ?y is instantiated to an object describing a shaft, the relation returns all available successful simulation models of a shaft. If ?y is left uninstantiated, the query returns all successful simulation models of all components. The CKMT query tool is shown in Fig. 4.

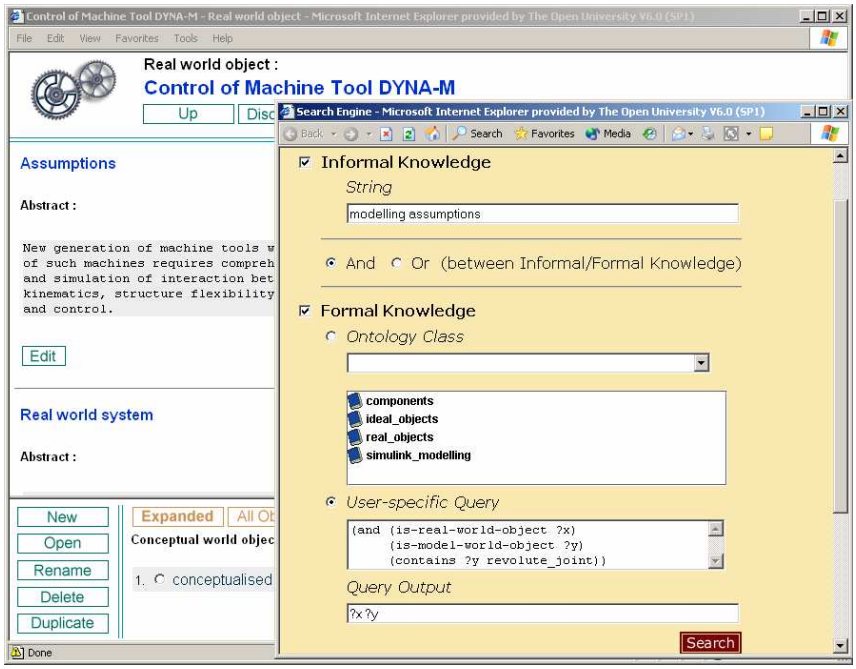

Fig. 4. Query tool

5 Developing New Clockwork Applications

Though the underlying processes of design problems have much in common, the applications differ depending on the engineering domain, goals, the range of activities which need to be supported and other factors. In order to seamlessly support existing work practices the new application must include existing knowledge management tools and legacy tools of the company. The same knowledge modeling technology used for semantic search can support the development of a new application. The analysis of the design process identifies the number of worlds and their structure needed to represent the problem. A model of the design process is constructed in terms of a *backbone ontology* and serves as an input for the *CKMT generator* which

designs the client and server component of the CKMT tool. An application developer then integrates these components with existing knowledge management and legacy tools. The overall architecture is shown in Fig. 5.

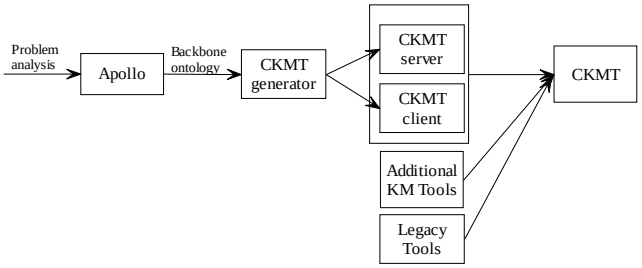

Fig. 5. Ontology driven generation of CKMT application

By introducing the modeling approach this methodology significantly simplifies and speeds up the application development. This approach has been used to develop customized CKMT's in a number of applications including the one described in Case Study 2.

6 Case Study 2: Design and After-Sale Support of Thermal Technology

This application has been developed at the site of Clockwork industrial partner ELOTHERM. The company's products are industrial thermal technology, developed for customers, and installed and maintained at customer sites. The company use simulation and modelling to support the design, selection and tuning of these products prior to product implementation.

The ELOTHERM application can be described by the following four worlds:

Requirements world: ELOTHERM in collaboration with the customer develop requirements for the thermal machine, covering productivity, cost and relevant technological parameters.

Initial design: Requirements are used to construct or select a conceptual model for the design. The conceptual model takes the form of an abstracted physical model of the machine. For example, the designer may think, a machine is required of the type '20 coil inductor'.

Detailed design: For the detailed design components and their parameters are selected. This step may raise further questions (e.g. acceptable range in the rate of heating) to be discussed with the customer. ELOTHERM have a set of available models and simulation and modelling software. The proposed model is then tested. A range of parameters are then tried and the values which ensure that the model functions satisfactorily are selected.

Product world: The thermal technology has been designed, manufactured and installed at the customer site. Details of the installation procedure are recorded in the Product World. ELOTHERM provide maintenance for the machine in the long term.

Fault reports are used to document this process and provide additional feedback to the design department.

7 Conclusions

Design is characterized as comprising a number of phases from requirement specifications to detailed design. In Clockwork, these phases are described as "worlds". Crucial design knowledge supports transitions between the worlds. The Clockwork toolkit makes it possible to record important knowledge concerning the construction of each world and its transformations. In addition, the toolkit allows the user to analyze the design process, describe it in terms of a backbone ontology and use the knowledge model to generate the structure of the tool.

The Clockwork knowledge modelling approach produces a minimal design history but provides a sufficient level of description to support its reuse. Engineering models such as sketches, diagrams and executable simulation models are integrated with knowledge models described in terms of knowledge bases, ontologies and text annotations. Recorded knowledge is used to guide semantic search in retrieving relevant engineering models which can be reused in a new design task.

Clockwork can be contrasted with many previous design rationale approaches which impose too high an annotation burden to the designer. The methodology and tools have been successfully tested in two industrial applications.

References

1. Cross N.: Engineering Design Methods. John Wiley & Sons. (1989).
2. Hubka V. and Eder E.W.: Design Science : Introduction to the Needs, Scope and Organization of Engineering Design Knowledge. Springer-Verlag Berlin and Heidelberg. (1995)
3. Louridas P. and Loucopoulos P.: A generic Model for Reflexive Design. ACM Transactions on Software Engineering and Methodology, Vol.9. No.2. pp. 199 – 237, (2002)
4. Myers K.L., Zumel N.B. and Garcia P.: Automated Capture of Rationale for the Detailed Design Process. In Proceedings of the Eleventh Conference on Innovative Applications of Artificial Intelligence (IAAI99). Pp. 1-8. (1999)
5. Newell A.: Artificial Intelligence and the Concept of Mind. In Computer Models of Thought and Language (R.C.Schank and K.M.Colby eds.). W.H.Freeman and Company. San Francisco. 1973. pp. 1-60. (1973)
6. Oltmans M. and Davis R.: Naturally Conveyed Explanations of Device Behavior. PUI 2001 Orlando FL, USA (2001)
7. Ormerod T.C., Mariani J., Ball L.J. and Lambell N.: Desperado: Three-in-one indexing for innovative design. Proceedings of Interact '99. Edinburgh. (1999)
8. Schön, D. A.: The Reflective Practitioner: How Professionals Think in Action. New York, Basic Books. (1983)
9. Simon H. A.: The Structure of Ill Structured Problems. Artificial Intelligence 4 (1973). pp. 181-201. (1973)
10. Yeung J.: SSPARCY: A Sofrware Integration and Design Rationale Capture Tool. Space System Policy Architecture Research Center. MIT. Bitstream (2002)