Combining and Adapting Process Patterns for Flexible Workflow

Jonathan Moore
Computer Science Department
Loughborough University, UK
J.P.Moore@lboro.ac.uk

Robert Inder
AI Applications Institute
University of Edinburgh, UK
R.Inder@ed.ac.uk

Paul Chung
Computer Science Department
Loughborough University, UK
P.W.H.Chung@lboro.ac.uk

Ann Macintosh
International Teledemocracy Centre
Napier University, Edinburgh, UK
A.Macintosh@napier.ac.uk

Jussi Stader
AI Applications Institute
University of Edinburgh, UK
Jussi@aiai.ed.ac.uk

Abstract

To provide intelligent process management support in complex engineering domains, considerable advances in the flexibility of current workflow systems are necessary. We describe an approach to developing such flexibility based on the capture of process patterns within a particular domain, and the dynamic composition of such patterns to determine the structure of an overall process. The role of a formal ontology of the domain in maintaining internal consistency of processes being managed in this way is emphasized.

1. Introduction.

The highly flexible process structure of engineering projects presents a major challenge to current workflow management technology. Workflow systems have been highly successful in providing effective, streamlined management of certain classes of process, described by Alonso et al. [1] as "administrative" processes, and characterized by their well-defined structure and constant, predictable form. However, the inflexibility of currently available systems means that their application is limited in more dynamic and uncertain working environments, such as characterize most engineering endeavours.

At the same time, many engineering projects would greatly benefit from computerized process management support—not, as in many workflow applications, simply for efficiency's sake, but to help manage the complexity typical of such processes, and ensure that everything necessary is done, and done correctly. Research suggests, for example, that poor management of the product innovation process—such as a lack of communication and the inappropriate cutting of corners—contributes to many failures of new products [2].

For these reasons, there has been interest in recent years in the development of "adaptive" workflow systems, able to cope with the more flexible process structures, and to operate in more dynamic contexts. The work described in this paper forms part of the Task-Based Process Management (TBPM) project, currently ongoing as a collaboration between Loughborough University Computer Science Department and the University of Edinburgh's Artificial Intelligence Applications Institute.

Fundamental to the project's approach to the problem is the observation that, while much domain knowledge and process management expertise goes into the initial setting up of a workflow system, the same knowledge and expertise is not available to the system during its operation. It is exactly this knowledge that it is essential be used if the system is to adapt to changes in its context and environment.

We are attempting to address this deficiency of workflow systems by endowing them with knowledge about processes in general, and about the domain in which they are deployed in particular. This knowledge is then used to help to adapt the structure of a process to the circumstances within which any particular task is being carried out.

As a test-case for the approach adopted, we are addressing the scale-up process—a combination of experimentation and design by which potentially promising new products are brought from the laboratory to full-scale production in the chemical process industries.

2. Process Patterns.

As indicated above, it is the flexible, dynamic nature of most engineering projects which presents the process management challenge. Even within a fairly tightly specified domain, there is no single process model which can be applied to all projects. Many factors conspire to make the process followed unique to each project: the nature of the product, the customer, the history and structure of the business, the availability of people and resources, etc. Even for a single project, the nature of the process followed cannot be known a priori: it emerges over time, being affected by events and discoveries which occur during the projects lifetime.

On the other hand, many recognizable similarities do exist between different projects. The broad structure of most product innovation processes in any one business may be essentially the same (Cooper [3] suggests an outline process intended to be broadly applicable); very many engineering design projects contain an identifiable progression from requirements, through specification, to design (again, canonical forms for such processes exist in the literature); many engineers, when presented with similar tasks, will conduct them in essentially the same way.

The nature and use of this type of structural similarity correspond well with the concept of design patterns in software engineering (see Gamma et al [1] for the seminal work). They represent widely applicable solutions to classes of problem commonly encountered in a range of different contexts. These solutions are outlines only, requiring specialization for the context in question.

The availability and applicability of such process patterns is one of the key aspects of domain knowledge which must be available to an adaptive workflow system if it is to offer effective process management support. When confronted with a task, a user must be able to identify suitable canonical processes for achieving it, select the one most appropriate to the current situation, and, if necessary, customize it for that situation.

2.1. The Plan Library.

A key component of the TBPM system is therefore a *plan library*, which maintains a database of process structures, relating each structure to the types of tasks for which it is a suitable method. Each plan specifies a set of tasks, together with the ordering constraints and object flows between them. Thus, a plan represents one possible way of achieving a given type of task by breaking it down into a particular structure of sub-tasks.

Each plan specifies only a single level of structural decomposition. However, the decomposition is into a further set of tasks, for each of which further plans may exist in the library. These plans may in turn be selected to specialize the sub-tasks, and so a multi-level hierarchical process structure may be generated by composition of many plans.

For any given task, there may be multiple possible plans, expressing different ways of breaking the task down which may be suitable for different situations. Figure 1 shows a simple plan for the task of acquiring some artifact. The plan has two sub-tasks: "specify" the artifact required, then "obtain" it. Two additional plans are shown, each a possible method for achieving the "obtain" task: one by designing and constructing the artifact in question, the other by purchasing it.

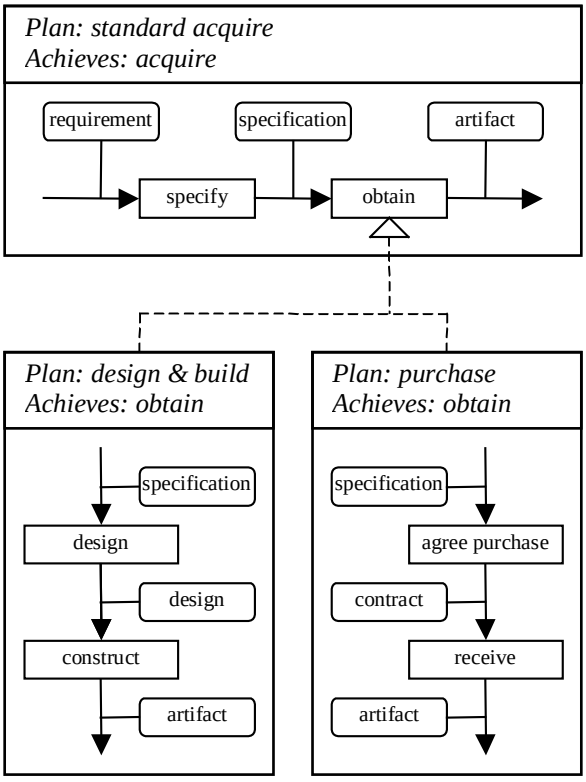

Figure 1. Examples of plans from the plan library. (Tasks are shown as squared boxes, object flow by rounded boxes.) Two possible alternative plans are shown for achieving the "obtain" task.

A manager of a process can put together such plans dynamically, deciding based on the current situation which of the alternative breakdowns is more appropriate. A similar process of specialization may then be applied to structure each of the sub-tasks within the chosen breakdown. For example, the "agree purchase" sub-task may have several different available plans; one might cover putting large orders out to tender, another simply ordering a part from a manufacturer's catalogue.

2.2. Development of Process Patterns.

The example described above is taken from the scale-up scenario developed as a test case for TBPM. The development process consisted of knowledge acquisition sessions involving members of the project team and domain experts who are routinely involved in the scale-up process. Normative descriptions of the common processes involved in scale-up were developed using an abbreviated form of the IDEF process capture method, and expressed in IDEF 3 notation. Subsequent work by the project team concentrated on refining the models obtained, identifying similarities between processes and attempting to further structure and generalize the models.

2.3. Process Adaptation.

The ability to construct process models on the fly by combining process plans as described above introduces a high level of flexibility to the process management system. However, it was recognized early in the project that even more flexibility would be necessary.

Any library of plans embodies a particular corpus of experience within the domain. Such a body of experience will almost inevitably be incomplete, may get out of date, or may simply not be appropriate to a novel situation. If the existing plans are the only options available, then there is likely to be a significant proportion of processes for which the system is not a suitable management tool.

To obviate this difficulty, it was decided to integrate the process-modelling tool used to generate the process plans with the run-time process enactment system, the task manager. Integration of the process editor with the task manager allows plans retrieved from the plan library to be edited both before and during their enactment. This enables a user to select the most appropriate of the plans available for a given task, and then specialize them for the current situation, by adding and removing tasks, constraints, and object flows to the process structure. To deal with novel situations may therefore require more work than standard cases, but it is at least possible. For further discussion of the interaction between the process modeller and the task manager, see [5].

3. Ontologies.

3.1. Representing the Process Context.

One problem introduced by the flexible modelling approach described above is of maintaining consistency within and between the plans making up the overall process model. How, for example, to prevent a user from putting together a process where the output of a "specify" task—a "specification"—is used as input to a "construct" task, which requires a "design"? While "specification" and "design" are certainly related, they are by no means interchangeable—a specification for an artifact will not normally contain the detailed instructions necessary to construct the artifact, information which *is* given by the design.

Any process management system would be expected to manage and co-ordinate the inputs and outputs of the different tasks involved. But in order to maintain the internal coherence and consistency of the process as a whole, in the sort of dynamic process structure envisaged herein, the system needs also to have some representation of the different natures of the various inputs and outputs.

We may go further: there must be some structure to such a representation, since different plans are expressed at different levels of abstraction, and so also, therefore, will be the nature of their inputs and outputs. A plan describing a generic design process, for example, may have inputs and outputs expressed in terms of an "artifact", which at a lower level of abstraction turns out to be a pump. The system should recognize "pump" as a valid example of "artifact".

Another problem introduced by the added flexibility described above is again caused by the potential genericity of the plans developed. Both the design of an experiment and the design of an item of chemical plant equipment may be carried out using the same high-level "design" plan. However, the appropriate set of more detailed plans will be different in each case: it would be desirable not to present the scientist planning an experiment with a list of methods suitable for designing pumps, or the engineering designer with a list of possible experimental procedures.

In order to support process-planning decisions effectively, we therefore need to be able to parameterize the basic "design" plan with some elements of the present context, to be used as constraints when selecting the set of possibly appropriate lower-level plans to offer. As with the representation of inputs and outputs, it is necessary to capture some of the structure of the domain—to know, for example, that the task "design a pump" is a valid example of the more abstract task "design an item of chemical plant equipment".

The approach taken in TBPM has been to develop *ontologies* of the domain in order to capture, structure, and reason with knowledge about the process context. (Although there is conceptually a single "domain ontology", there were several distinct areas of knowledge identified as being necessary to capture, and these were developed separately as a related set of ontologies. Moore et al [6] gives further details.)

3.2. Informal Ontology Development.

Following the outline ontology development method described by Uschold & Gruninger [8], the ontologies were developed at first in an informal form. Using the results of the plan library development as a starting point, and again in collaboration with domain experts, a core set of important terms and concepts from the scale-up domain was identified. For each term or concept, a concise definition in natural language was agreed, bringing out the relationships and distinctions between the ontology terms.

3.3. Formal Ontology Development.

An informal ontology is a useful tool in itself, and may be used to harmonize understanding and permit unambiguous communication between participants in a multidisciplinary domain. However, in order to allow the level of automated reasoning required for process management, it is necessary to encode the ontologies' knowledge in a more formal way.

Selecting a formalism to use to encode the ontology required a trade-off between expressive power and simplicity. Languages such as KIF and Ontolingua provide much expressive power, but it was recognized that users of the TBPM system would be directly exposed to, and expected to work with, the ontology formalism. A simpler, if less powerful, formalism was therefore sought, in order to make the system more accessible to its potential users.

The major uses of the ontologies in support of process management were all recognized to involve providing answers to questions of the form "Is X an example of Y?" (as in the "design a pump" example given above). The formalism adopted therefore centres on the arrangement of terms in a generalization hierarchy, such that any term is a specialization of any of its parents. In addition, each term may be defined to possess a number of named parameters, whose values can be constrained to be terms drawn from a particular sub-tree of the ontology. A particular term's parameters, and their associated constraints, are inherited by all that term's children, which may add further specialization to the constraint (in object orientated programming terms, varying in a covariant manner).

Recursively testing the base terms and all parameter values against the generalization hierarchy allows unambiguous determination of whether one ontological expression "subsumes" another.

To continue the example above, given an ontology hierarchy of terms and associated parameters such as that illustrated in Figure 2, it is possible to determine that

```
specification (
    system: pump (
                type: centrifugal))
```

is both an example of

```
specification (
    system: chemical-plant-equipment)
```

and an example of:

```
design-information (
    system: pump)
```

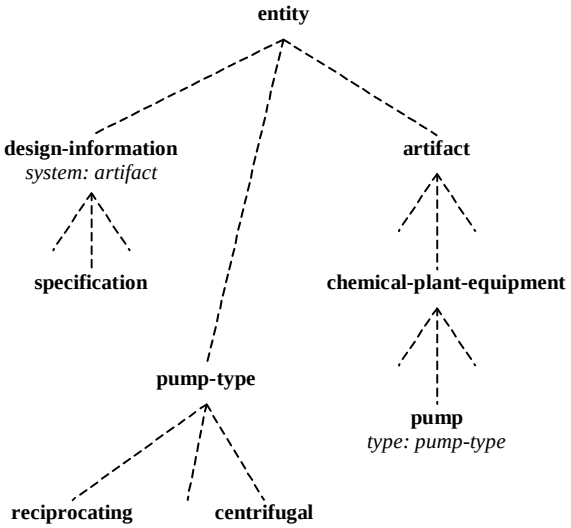

Figure 2. Some terms from a formal ontology for the domain of the scale-up process. All types of design information are parameterized by the class of system to which they relate, while all pumps are parameterized by their type, which must be a valid "pump-type".

4. Discussion.

The challenge for adaptive workflow is to provide a good level of intelligent support for process management decisions, and to automate as far as possible the performance of common tasks using standard methods, while retaining sufficient flexibility to avoid users ending up fighting the system in an attempt to get not-so-standard tasks done.

The plan library provides good support for carrying out common tasks in any one of a number of standard ways. Collaboration in process management is enhanced by the provision of plans at different levels of abstraction, so that each user can work with plans expressed at the necessary level, without having to commit unnecessarily to particular lower-level details. A project manager can outline the structure of the overall project, leaving the

details of the experimentation and engineering involved to be fleshed out by the relevant scientists and engineers.

There was a notable tendency during development of the plan library for plans to become simpler, but more numerous, as similarities between processes were recognized, and the differences relegated to choices to be made at a lower level of abstraction. This is remarkably similar to the theme of many software design patterns of "encapsulating the concept that varies" [1]. In object oriented programming, such encapsulation is accomplished by deriving a common interface shared by all implementations of the varying concept. In the example illustrated in Figure 1, it is notable that, as far as the higher-level plan is concerned, both plans for the "obtain" task present the same interface, requiring a specification as input, and resulting in an artifact as output.

Such encapsulation tends to break down, however, as additional process management considerations are taken into account. In particular, planning the provision of resources—including human resources—for tasks depends crucially on the lower-level plans adopted: will we need an engineering designer, or a purchasing executive? (Moore et al. [7] outlines the approach being adopted to providing support for this class of decision.)

Additional, necessary flexibility is provided by the ability to edit plans both before and during their execution by the process enactment system. This enables the system to be used to manage processes for which no combination of the available standard plans is suitable. In a dynamic environment, it seems likely that the mismatch between the available set of plans and the tasks encountered will increase over time, as business conditions, legal context, technology, and other factors change and recognized best practices in the domain move on and improve. Any plan library will, in practice, have to be kept under continual review, and possibly considerable pains taken to keep the plans up-to-date. Whether there are some plans which will prove "timeless" is, for now, a moot question.

5. Conclusion.

Experience on the TBPM project suggests that process patterns can be identified within complex engineering domains, at varying levels of abstraction. Such patterns represent widely applicable outline solutions to ubiquitous classes of tasks.

To recognize such patterns, capture them, and use them in varying contexts and dynamic environments presents a challenge to the state-of-the art in workflow technology. In particular, it is vital that users be able to adapt any patterns so used; to customize them for a particular set of circumstances.

Ontologies play a crucial role in structuring knowledge of the domain, to enable intelligent process management systems to retain the internal coherence and consistency of processes, while promoting the level of flexibility needed to manage processes within complex, dynamic environments, such as most engineering organizations.

6. Acknowledgements.

The TBPM project is a joint project between the Computer Science Department at Loughborough University, and the Artificial Intelligence Applications Institute at The University of Edinburgh. The project is funded under the EPSRC Systems Engineering for Business Process Change programme, and has ICI and Unilever as industrial partners.

7. References.

[1] Alonso G, Agrawal D, El Abbadi A, and Mohan C, "Functionality and Limitations of Current Workflow Management Systems", *IEEE Expert*, **12**(5), 1997.

[2] Cooper R G, *Winning at New Products: Accelerating the Process from Idea to Launch (2nd Ed)*, Addison-Wesley, Reading MA, 1996.

[3] Cooper R G, "A Process Model for Industrial New Product Development", *IEEE Transactions on Engineering Management*, **30**(1), 1983.

[4] Gamma E, Helm R, Johnson R, and Vlissides J, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading MA, 1995.

[5] Jarvis P, Moore J, Stader J, Macintosh A, Casson-du Mont A, and Chung P, "Exploiting AI Technologies to Realise Adaptive Workflow Systems", *Proceedings of the Workshop on Agent Based Systems in the Business Context (held during AAAI-99)*, 1999.

[6] Moore J, Stader J, Chung P, Jarvis P, and Macintosh A, "Ontologies to Support the Management of New Product Development in the Chemical Process Industries", In: Lindeman U, Birkhofer H, Meerkamm H, and Vajna S (eds), *Proceedings of the International Conference on Engineering Design, ICED 99*. Munich, August 24-26, 1999, pp. 159-164.

[7] Moore J, Inder R, Chung P, Macintosh A, and Stader J, "Who Does What? Matching Agents to Tasks in Adaptive Workflow", submitted to: *International Conference on Enterprise Information Systems*, Stafford, UK, July 2000.

[8] Uschold M, and Gruninger M, "Ontologies: Principles, Methods and Applications", *The Knowledge Engineering Review*, **11**(2), 1996, pp. 93–136.