Patterns Boosting Adaptivity in ACM

Helle Frisak Sem, Thomas Bech Pettersen, Steinar Carlsen, and Gunnar John Coll

Computas AS, Lysaker torg 45, N-1327 Lysaker, Norway
{Helle.Frisak.Sem,Thomas.Bech.Pettersen,Steinar.Carlsen,
Gunnar.John.Coll}@computas.com

Abstract. Adaptivity is the ability to adjust behavior to changes in context. In enterprise ACM, the need for control, uniformity, and analysis meet end-user's needs to adapt to situations at hand. The interplay of different layers of declaratively defined templates can meet these needs in a flexible and sustainable manner. A data centric architecture with a library of process snippets building on standard activities and actions and a model of the organization with roles, provides an operational quality system. By combining this with a standardized declarative modeling of the case objects with codes and soft typing, we achieve a flexible platform for domain specific enterprise ACM systems. The result is a case- and goal-driven approach to adaptivity in ACM, taking into account knowledge workers' autonomy and the information context of the work. This paper discusses means of achieving adaptivity in ACM solutions and presents examples of practical realization by these means.

Keywords: Adaptive Case Management (ACM), Domain Specific ACM, Templates, Declarative Knowledge Representation, Task support.

1 Background

This paper is based on three real-world projects that have delivered award-winning operational ACM solutions - the Norwegian Food Safety Authority's MATS application [1, 2, 3], the freight train transporter CargoNet's GTS system [4, 5] and Norwegian Courts Administration's LOVISA solution [6]. All these are developed on the FrameSolutions ACM application framework.

This approach focuses on tasks and task support instead of end-to-end processes. Knowledge workers execute tasks specified as a set of steps where each step represents collaboration between the system and the user. The knowledge worker is always in charge. He may initiate new tasks in a goal-driven manner, with a high degree of freedom regarding sequencing of steps and tasks. Essentially, there is no process model or process model templates. Instead, there are goals and tasks with their steps. These steps may have several conditions associated, specifying the relevant and applicable constraints. The process starts with the selection of a goal and the following flow of work between tasks emerges. Prescriptive process models reflecting all possible paths from start to end is replaced by a set of flexible tasks selected and executed by the knowledge worker, composing a valid and unique "process" – on the fly.

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 102–111, 2013.

We use a *declarative representation* of task logic without a predefined, strict control flow [11]. The use of declarative definitions, separated from programming code, provides flexibility and opens for design-time configuration by the knowledge workers or their experts. Declaratively modeled tasks, libraries, rules, code sets and code relations are all explicitly represented as important business logic. In our approach, these definitions are stored in xml format, separated from the application code, in the form of definition base files. By deploying a new version of the relevant definition files and hot-deploying them, the new definitions are made operational on the fly.

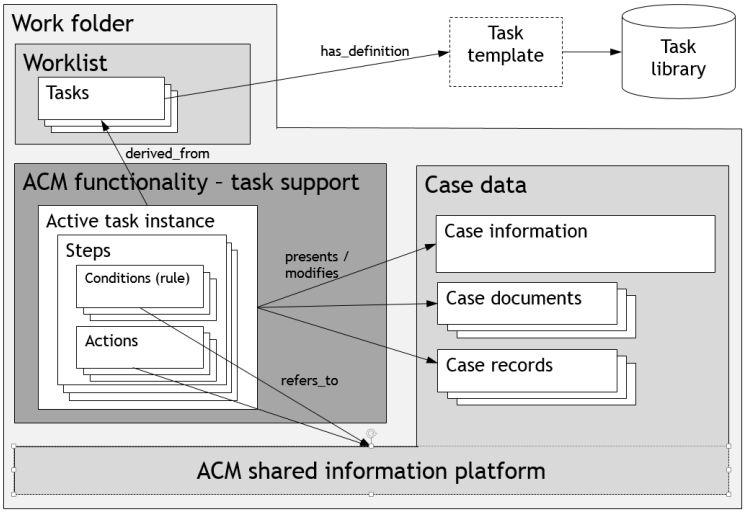

Fig. 1. Work folders providing task support

2 Motivation – ACM and Patterns

The "holy grail" of BPM has been to control behavior by means of process models, providing adaptability through changing these process models. Such changes may ideally come from end-user configuration activity, and the requirements of ACM typically include the possibility for the knowledge worker to perform configurations on the fly without support from professional system developers (as a developer configuration activity). Although important, we think this kind of model-driven change is overrated. In practice, the complexity of current model-based approaches requires implementation by developers understanding both programming languages and the detailed semantics of process modeling notations like BPMN. Consequently, BPMN models often end up as models of IT system behavior, instead of organizational or business behavior.

Contrasting the model-driven approach, we are in search of a *case- and goal-driven* approach, where the result is a dynamic combination of *process snippets*, all contributing to an emergent behavior and workflow. This provides more direct means

for achieving adaptivity, than allowing users to change process models. There simply is no (large) process model available to change, instead there are (small) tasks associated with task definitions which can be puzzled and combined in any manner the knowledge worker sees as appropriate – subject to the prevailing constraints, such as deadlines, sequence, state and milestone dependencies. Instead of explicitly representing valid process paths, any path is allowed as long as constraints are adhered to. These constraints reflect means for necessary work performance control.

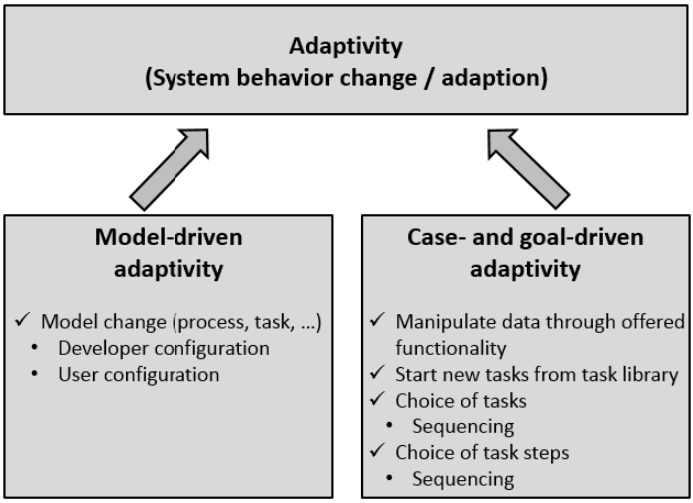

Fig. 2. Model- and Case-driven adaptivity

We are trying to establish a case-driven approach to adaptivity in ACM, taking into account the information context of the work, to provide adaptivity related to case data as well as user behavior. This strongly complements the traditional process model-driven approach.

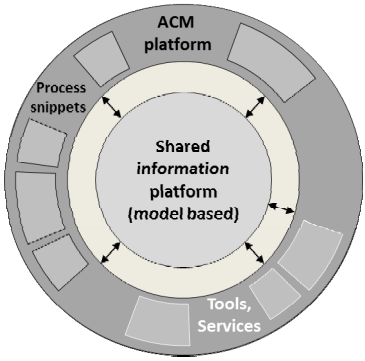

Fig. 3. Data-centric ACM [4]

In this paper, we are concerned with ACM systems with a built-in representation of their domain. The representation of the domain itself provides context - a domain specific ACM may be considered a subclass of a data-centric BPM, where there is a bundle of processes (process snippets in our case) all referring to the same domain model. In most of the BPM literature, this important property of the system is ignored. We think this is a consequence of traditional BPM putting the processes in the center, with data as an afterthought, whereas ACM is inherently data-centric with a shared information model as the core.

The tasks performed by knowledge workers in any organization are often variations of task themes or patterns. Patterns are a well-established metaphor for describing a family of similar phenomena. Patterns are defined by templates, and each individual instance can be described as a specification of the template. The template / pattern concept had a breakthrough in the world of computer programming following the "Hillside Group" in the mid-nineties [8], heavily inspired from Christopher Alexander's work in the field of building architecture in the seventies [7]. In the field of artificial intelligence the same concept was the key result from the European Union Research Project 5146 KADS in the eighties [9], where a catalog of generic reasoning patterns for problem solving is brought forward in order to facilitate the process of knowledge acquisition in complex problem solving scenarios.

Templates in the same tradition can increase the benefit of case management systems in general, and adaptive CM systems in particular, through techniques described below. The use of templates has proven to ease the initial formulation of the content of ACM applications, and more importantly – significantly decreases the cost and complexity of the long-term maintenance of applications and business logic.

3 Providing Adaptivity

For the knowledge worker, the need to be adaptive results from different sources, requiring different coping strategies. Unexpected events may occur in the case, new knowledge may turn up, other agendas may enter into the picture, or the case may have particular properties. A generic ACM approach must therefore include a range of adaptivity-enhancing features. This experience paper looks into the capability to create adaptivity, and outlines generic solution elements that can put the "A" into practical ACM solutions, in a broad sense:

- **Context-sensitive task patterns:** Templates for the functionality supporting the performance of tasks capable of adapting to data in its context; to case data, to system properties and to case state.
- **User-sensitive task patterns:** Templates that open for the user to choose as freely as possible under only required restrictions in order to execute the task in a regular way.
- **Goal-oriented task pattern libraries:** The possibility for the knowledge worker to select freely between goals – starting points of task sequences – in accordance to what he judges fit.

- **Domain model patterns:** A mechanism to set up parts of the case data structure in a declarative way separated from application programming code.
- **User-driven combination of process snippets:** The possibility for the user to build the work connected to the case dynamically without being predetermined. The case is what binds the different work tasks together, not a predefined process.
- **Real time composition of process snippet sequences:** Instead of full processes with all possibilities and choices predefined, the process emerges in a non-predictable way.
- **Business rule base:** business rules separated from application programming code and executed by a rule engine covering situations where the business logic frequently changes.

4 Elements of a Practical Approach

In this chapter, we provide more detail about the various knowledge representations utilized in our approach – task templates, task libraries, business objects and business rules.

The variety of knowledge representations described below has been very stable over the years, while underlying technologies have gone through many changes and shifts. We use knowledge engineering techniques for creating such representations. This can also be combined with other more traditional techniques. We have successfully deployed solutions combining this with Cockburn use case descriptions [10], BPMN, user stories, user story grouping, etc. We consider all these valuable process discovery techniques from the architect's toolbox.

4.1 Task Templates

Instead of trying to support linear and static "end-to-end" processes, ACM solutions can be based on dynamically combinable process snippets of task support functionality. Each such snippet covers a task pattern that can typically be performed by one person within the scope of one day.

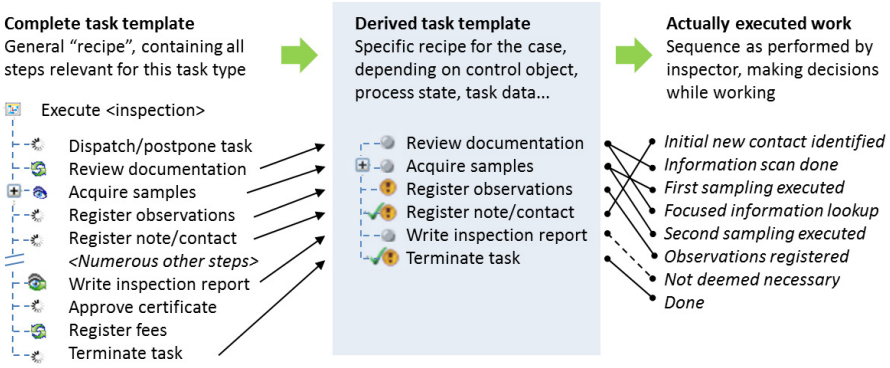

Fig. 4. Task template providing active task support in a particular context [3]

A task pattern is defined by means of a *task template*, which contains a set of *steps* as a general "recipe" for performing the task. These steps are subject to several *conditions*, defined using predicates and rules, which make it possible to derive a task template for the specific context as a unique recipe for a particular instance, always reflecting its state and data. This contextualized derived task template is the basis for offering active task support to the knowledge worker, who then selects and executes permitted steps, based on personal, professional judgment. Step conditions specify pre- and post-conditions in addition to repeat-conditions (step can be repeated), include-condition (step is made available dependent on context) and mandatory-conditions (step must be performed). The interplay between these conditions produces an appropriate set of steps for the user. Each such step will offer relevant functionality, information or tools, often including a user interface component for operating on relevant case data.

All available system services can be invoked from a step. Conditions and actions in the step definitions also have access to all available system functionality. By allowing a step to create new tasks, adaptive workflow can be accomplished.

Thus, a two-way flexibility is obtained. The task template is flexible with regard to the case and the system context – it is context-sensitive. The template also provides flexibility for the knowledge worker – it is user-sensitive. A main goal is to ensure correct practice, provide the right information and functionality, with full freedom to apply the knowledge worker's judgments and expertise. The task support functions as an operational quality system.

4.2 Goals and Goal-Oriented Task Libraries

In order to achieve a particular goal, the knowledge worker can select to create a new task instance. All the task types that the user may opt to start are organized in a runtime task library. Access to the tasks types is controlled by permissions linked to actively managed user roles. The knowledge worker may start any task available to him in the runtime task library, whenever he wants.

Goals may also be identified by the system, triggered by events. Such events may be the arrival of a new document in the document archive system, the submission of an application, the execution of a step in a task, or an event pattern. A typical event pattern is an expected reply from an external source not having arrived in time, giving rise to the need for some other action. When the system identifies the need for action, it can create and dispatch human knowledge worker tasks. Such automatically started tasks are being set up with available context information. Tasks that are created automatically may or may not be accessible from the runtime task library.

4.3 Domain Model Patterns

The diversity of domain object types is one of the main challenges of making a system to help the different functions of an enterprise knowledge organization become unified, uniform, and flexible in its behavior.

Our solution has been to apply modeling in different ways. First, we applied classic Java and database modeling. This is used for those parts of the case types that are common or require special types of functionality, such as identity handling. Second, we have extensively used *soft typing*, applying code sets defined in xml, instead of object typing. Third, we have combined code sets by means of code relations defined in xml, in order to define which acts and regulations are relevant for which types of cases. We have found *codes and code relations* very useful for implementing *interrelated controlled vocabularies*.

A cost driver when creating enterprise applications is the design, development and testing of domain objects and functionality. We have used a pattern that we call *business object definitions* for defining parts of case types that are mainly informational and vary between otherwise equal case types. Soft-typed business objects are defined as xml structures, handling storage and display with generic methods. We have found *soft-typed business objects* useful for realizing generalized solutions that are easy to specialize for future system enhancements.

4.4 Business Rules

Rules can be used to define:

- pre-, post-, and other conditions in connection with task definitions
- permitted logical relationships between data in the system
- calculation rules for various types of complex calculations

For ACM user organizations, declaratively defined business rules provide a particularly useful format for implementing regulations that are subject to frequent change, such that changes can be achieved without modifying the application.

By separating out business rules driving calculations in a rule base, several advantages are achieved:

- The calculations are easy to maintain, they are represented by rules in text form and can be edited in a user friendly way.
- The rules are stored separately from application code and can be changed without having to create and deploy a new version of the application.
- Calculation rules allow for multiple versions of regulations to co-exist in the system, to ensure that every case is evaluated in accordance with the correct set of rules valid when the relevant events of the case occurred.

5 Putting the Elements to Use

In an ACM project, an important part of developing a flexible environment is to develop a common language, toolbox and organizational model. These correspond to the atomic or molecular operations and organizational functions that the system will support, spanning out the room of possibility for realizing the ACM solution. This flexible environment supports establishing local patterns for both the development and the use of the system.

5.1 Subject Matter Experts

In several of our practical ACM applications, subject matter experts (SMEs) from the customer have been engaged in describing a shared data model as well as growing their own vocabulary related to their processes, tasks, business rules and organization roles. The approach enables the combination of snippets, turning them into assets both for discussing and for implementing future system functionality. This helps maintain focus on the customer's knowledge assets in terms of value creation, not only on best practice use of IT technologies for performing "requirements engineering". Working in this way rapidly provides handles for the SMEs and business analysts to identify the key ingredients of a knowledge organization – *value-driven system development*: What are the different user groups? What tasks do they solve on a daily basis? What are their frustrations? How are the tasks related? Who is allowed to initiate which tasks? Who collaborates on the problem solving needed to perform these tasks? Which task components are reusable? Which tasks, steps and conditions are meaningful as standard elements referring to known vocabularies? What added task-support to my application will have the greatest business benefits? How can we benefit from web-based self-service in improving data quality?

Our experience is that growing an adapted vocabulary of standard template fragments makes collaboration between system developers and subject matter experts (SMEs) easier and places the SMEs closer to the driver's seat.

SMEs can theoretically make and maintain the declarative definitions themselves, at least for the more straightforward situations. This amounts to design-time adaptivity without the need for professional system developers. However, creating and changing these definitions does require specialist training. It does after all define the business logic for tasks and domain data for the enterprise in question. It may be wise to let SMEs concentrate on specifications, and let developers perform the actual changes in the configuration controlled declarations and subject the changes to normal test procedures before deploying the new definitions.

5.2 Organizational Roles

We have found the use of a flexible organizational model very useful in ACM systems. This model allows for the representation of several (possibly multi-root) organizational trees spanned out using named relations (like "belongs-to", "reports-to" etc.) where any kind of organizational matrices can be represented. As a declarative structure, it can also be maintained separately from program code. The ability to describe a richer set of functional roles in the organization than the average "directory" solution enables enhanced flexibility in role-based collaboration patterns and provision of task templates for different purposes/roles.

5.3 The Art of Templating - Recognizing Patterns

As the first definitions of tasks and steps emerge from the collaboration between SMEs and knowledge engineers, there will be little repeatability or familiarity between the

definitions. As the project team gains speed, a common vocabulary grows in the form of patterns of processes, tasks and steps. The FrameSolutions approach represents these building blocks as *standard elements* – standard steps, conditions, actions, subtasks and tasks.

These standard elements can be connected as *sources of inheritance* to other definition elements, making it easy to maintain a uniform and consistent definition set for an enterprise case management platform. These standard elements are easy to reuse through the inheritance scheme, and the elements that represent the reuse can have their properties and behavior defined as overrides to the standard element properties and behavior.

The ultimate success is obtained when the SMEs start reasoning and communicating using these standard elements as a living meta-language, gradually established through a domain oriented bottom-up approach, well founded in the enterprise's knowledge worker community. By focusing on process snippets the users can "put puzzle pieces together" without the traditional end-to-end processes. The organization can start small and achieve benefits early. Consolidated and suitable end-to-end processes – or even longer process fragments – are often unrealistic in today's dynamic and unpredictable enterprise environments.

5.4 Templating at Higher Levels of Abstraction

While our ACM framework offers various built in template types as described above, a particular ACM system may offer additional higher-level templates that form a package of the basic templates. In such a package, task templates typically are bundled (loosely coupled) with some of their relevant context, such as encoded legislation. This kind of packaging tends to be domain-specific, and may ultimately result in re-usable ACM solutions that can be put to use in similar knowledge organizations. We have successfully "exported" task templates for immigration administration including visa applications from an ACM solution made for UDI (The Norwegian Directorate of Immigration) to other countries also bound by the Schengen Agreement (Denmark, Greece, Macedonia, Kosovo).

6 Conclusion: Towards Case-and Goal-Driven Adaptivity

We have discussed how adaptivity can be achieved in a work context, using active task support, user-decided sequencing of task steps, and goal-driven task libraries [2, 3, 4, 5, 6]. The result is a dynamic combination of process snippets, all contributing to emergent behavior and workflow. This provides more direct means for achieving adaptivity, than allowing users to change models, replacing the traditional model-based policy of "what is allowed is only what is prescribed" with the knowledge worker policy of "anything not forbidden is allowed". Case-driven adaptivity takes the information context of tasks into account – adaptivity is related to case data. Goal-driven adaptivity is the result of putting the knowledge worker in charge, allowing dynamic sequencing of tasks and their steps.

Being *built-to-change*, through the ability to adapt to changes in business logic and the business environment, is a key to delivering successful ACM applications. This can be achieved by combining traditional systems development with a declarative approach to business logic representation, avoiding buried logic in the source code and enabling runtime changes to knowledge representations (task definitions, rules, codes, etc.) by simply reloading xml files.

Declarative business logic representations are essential for building adaptive operational case management solutions. Instead of describing complete and predefined end-to-end processes, the focus is on identifying activities of the organizational actors involved and how to support them in a best possible way. Goal-driven task libraries give access to process snippets as combinable resources for action. The knowledge workers themselves are empowered to combine these process snippets into emerging processes.

References

1. 2012 Winners Announced in Global Awards for Excellence in Adaptive Case Management, http://adaptivecasemanagement.org/awards_2012_winners.html
2. Sem, H.F., Carlsen, S., Coll, G.J.: Norwegian Food Safety Authority, Nominated by Computas AS, Norway. In: Fischer, L. (ed.) How Knowledge Workers Get Things Done - Real-world Adaptive Case Management. Future Strategies Inc., Florida (2012)
3. Sem, H.F., Carlsen, S., Coll, G.J.: On Two Approaches to ACM. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops 2012. LNBIP, vol. 132, pp. 12–23. Springer, Heidelberg (2013)
4. Sem, H.F., Carlsen, S., Coll, G.J., Pettersen, T.B.: ACM for railway freight operations. In: Intelligent BPMS. BPM and Workflow Handbook series. Future Strategies Inc., Florida (2013)
5. Sem, H.F., Carlsen, S., Coll, G.J., Holje, H., Landro, E., Mork, H., Pettersen, T.B.: GTS - Cargonet AS Norway, http://adaptivecasemanagement.org/awards_2013_winners.html
6. Pettersen, T.B., Carlsen, S., Coll, G.J., Matras, M., Moen, T.G., Sem, H.F.: LOVISA - Norwegian Courts Administration (NCA), http://adaptivecasemanagement.org/awards_2013_winners.html
7. Alexander, C.: The timeless way of building. Oxford University Press (1979)
8. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional (1994)
9. Schreiber, G., Wielinga, B., Breuker, J.: KADS: A Principled Approach to Knowledge-Based System Development (Knowledge-Based Systems). Academic Press (1993)
10. Cockburn, A.: Writing effective use cases. Addison-Wesley Professional (2000)
11. Rychkova, I., Regev, G., Wegman, A.: Using declarative specifications in business process design. International Journal of Computer Science and Applications 5(3b) (2008)