WHEN WORKFLOW DOESN'T WORK

Issues in Managing Dynamic Processes

DANIEL D. CORKILL
Knowledge Technologies International
Amherst, Massachusetts

Abstract. Dynamic processes, such as those exemplified by design activities, present substantial problems for individuals charged with managing their execution. The design mantra of "better, cheaper, faster" is forcing design processes to operate with shorter time scales, fewer available resources, and increased concurrency and complex interactions among process steps—all exacerbating process-management difficulties.

Traditional business-process automation, such as workflow, is often suggested as a technological solution for managing dynamic processes. Such suggestions fail to appreciate the substantial differences between traditional "steady-state" business processes and the dynamics of design processes. We discuss why traditional process-execution systems cannot address the management requirements for dynamic processes and describe the need for a new category of dynamic-process management software; software developed specifically to address the management of dynamic processes. We close with an overview of KPM,™ a commercial software suite that is representative of this new category of dynamic-process management support software.

Process is the essence of every enterprise. The importance of process is demonstrated by the many software categories that have emerged to support processes, including: process modeling, business process re-engineering, workflow, and project management.

Traditional business activities, such as those in the insurance and banking industries, were the initial targets for computational support. These business processes were ideal candidates for automation as they involve moving large numbers of "cases" through the activity steps of relatively stable processes. The software tools that were developed for business processes handle the movement of cases in much the same way as material handling is managed in manufacturing processes. For example, analytical process-modeling software is used to engineer an efficient "steady-state" case flow, and then workflow software is used to implement the engineered process.

1. Dynamic Processes

Dynamic processes, such as those associated with design, are substantially different than traditional steady-state business processes. Where an insurance company is concerned with efficiently routing thousands of claims through its claims-processing process, an aerospace manufacturer is concerned with efficiently creating *one* design for the next-generation aircraft. Aggregate analysis and execution of the many becomes efficient management of the few,

Appeared in the *Proceedings of the Design Project Support using Process Models Workshop*, Sixth International Conference on Artificial Intelligence in Design, Worcester, Massachusetts pages 1-13, June 2000.

and the process-management goal changes from performing well on average to performing effectively on *each* process instance. *Dynamic processes are situational—not statistical.*

Dynamic processes often involve very limited and expensive resources, whose use must be carefully coordinated. For example, in aerospace the use of a single, highly expensive test fixture must be scheduled, not only for the design variants explored within a single design process, but also for the tests required by other concurrent design programs. Similarly, a landing-gear design specialist is a scarce resource that cannot be quickly supplemented with additional hiring or a temporary placement, should a bottleneck situation be encountered.

The duration of dynamic processes is typically much longer than traditional business processes, executing for months or even years. Many attributes of the process can change during its execution. The resources, design techniques, available materials and suppliers, and even the design requirements may change during the course of a design program. A few years ago, high mileage was a major engine design goal in the automotive industry. This goal was abruptly changed to high performance as gas prices dropped and "muscle cars" became fashionable (only to be changed again due to recent gas-price increases!). The engine-design programs that were underway had to be adapted quickly to the new requirements.

The duration of individual activities in a dynamic process may also vary significantly from one invocation to the next. Entering the data for an insurance claim may take a consistent seven minutes in a traditional business process. However, the time required for an engineering analyst to develop a feasible analytical model for a component assembly can vary from hours to days depending on the characteristics of the assembly.

Dynamic processes have numerous decision points that further complicate predictability. How many design variants can be explored? How many iterations of a generate-design, analyze, adjust loop will be required to obtain an acceptable component? How far must a candidate design be developed to discover whether it is feasible? These uncertainties make adhering to predefined schedules unrealistic and make strategic process decision making and resource management very difficult.

Finally, the dynamic process itself may need to be developed or elaborated in conjunction with its execution. For example, the downstream activities of a design process may be significantly different if a choice is made to use new composite materials for a structural element rather than traditional metals.

In order to reduce costs, shorten design cycles, and reduce time to market, long-standing design processes have been improved by:

- **Reducing the duration of individual activities** by performing localized improvements. An example of this is replacing a multiple-week manual CAD process with a knowledge-based, generative system that achieves the same results in minutes.
- **Re-engineering the process for increased concurrency** by looking for ways to subdivide and restructure activities. For example, a lengthy activity that produces results for Task A early on, but doesn't produce the results needed by Task B until it completes, could be split in two, allowing Task A to be performed concurrently with the rest of the lengthy activity.
- **Increasing the efficiency of task interfaces** such as by replacing manual exchange of data among activities with direct interfaces.

These improvements lead to better processes, which become the new baseline for further process improvement—and still there remains the competitive need to design better products faster and cheaper.

Unfortunately, the easiest process improvements are now implemented, and the greatest impediment to (and candidate for) further improvement lies with better understanding and

control of the detailed activities and resources used in the dynamic process. Surprisingly, the prior process improvements have made the process-management task much more difficult. With increased process concurrency and complexity, there is less reaction time and the scope and consequences of management decisions have increased, both for the executing process and for other dynamic processes that share resources with it. Further performance gains can be obtained by operating with fewer resources and less contingency (slack) allowance, making managing dynamic processes even more difficult.

2. Managing Dynamic Processes

Today, the detailed management of dynamic processes is performed largely without computational assistance. All too often, reactive decisions are made in response to immediate problems using managers' best intuition and experience.[1] Time and effort is spent in gathering information about the state of the process, the available options, and, once decisions have been made, in having those decisions carried out.

Process modeling techniques used for traditional business processes in advance of their execution are unsuitable for dynamic processes, because the decisions become quickly outdated by changes in the dynamic process. Similarly, periodic project reporting cannot present an accurate view of the process, as it has likely changed by the time the data is gathered and the report prepared.

What is needed is automated assistance in the form of an informed, knowledge-intensive, proactive, highly responsive, and flexible environment supporting dynamic-process management decisions. This decision-support environment needs to provide an instantaneous view of process activity and the current best estimate of downstream activities and potential problems. It must deal with the coordination of multiple dynamic processes and allow responsive execution of management decisions. The goal is to provide the support needed to keep management activities ahead of the rate of change in the dynamic process.

To achieve this goal, dynamic-process management support software must meet the following five requirements:

1. **Complete process representation**—Support for dynamic-process management must begin with a full model of the dynamic process, with sufficient detail to allow automated execution of the model and to allow reasonable expectations of future activities to be made. This representation must represent the *entire* process, not just data transitions. Process models must be process—not data—oriented.
2. **Direct model execution**—The process representation must be "live," so that what is represented is exactly what is executed. Direct model execution is important for validation of the process representation and to ensure that changes in the representation will be reflected in the executing process. Additionally, all activities must be integrated into the execution, so that their execution status is known. (It's hard to manage what you don't know about. . .)
3. **Integrated dynamic scheduling**—Scheduling of activities and resources is needed to provide the on-line situational equivalent of off-line process modeling. The dynamic scheduler uses process knowledge and current process information to generate and maintain the downstream expectations needed for proactive intervention. The dynamic scheduler must balance the need for process flexibility with the need to maintain process stability

[1]The noun "managers" is used here to indicate the people responsible for managing an executing process, no matter what their organizational role or job title. In many design settings, these managers are senior engineers rather than administrative managers.

whereever possible. It must be able to make timing and resourcing decisions across multiple dynamic processes (such as across multiple design programs).

4. **Live presentation of execution status, history, and expectations**—It is essential that crucial decision-making details are provided to those responsible for a process. These details must include instantaneous process state and the latest expectations for the future. The presentation should be tailored for quick understandability, relate directly to the process representations, and quickly focus attention on problem areas.
5. **On-the-fly process modification**—The software must allow proactive response to unanticipated situations and problems, ensuring that execution and scheduling adapts immediately to modifications. Managers can directly manipulate the live model to change process structure and execution. These changes are immediately propagated throughout the model, keeping managers and process participants in sync with process changes.

These five requirements form a closed process-management loop, allowing managers to quickly comprehend the current process status and potential downstream problems, make proactive interventions to address problems, and have those interventions immediately reflected in the executing processes. If *any* of these requirements are not tightly integrated in the management-support software, it becomes impossible to perform proactive management at the speed dictated by process dynamics.

3. Dynamic Processes and Workflow

Because a workflow system executes "processes," workflow is often suggested as a technological solution for improving the management of dynamic processes. Considering the five software requirements for assisting dynamic process management, it is clear that *workflow is the wrong tool* for the job.

3.1. COMPLETE PROCESS REPRESENTATION

A workflow system uses a "process definition" as the basis of its operation. On the surface, this would seem to satisfy the requirement for a complete process representation. However, the standardized workflow representation is based on the data transitions among the atomic activities of a process, rather than on the structure of the process itself. This emphasis is not surprising, given the data-centric nature of workflow engines. Workflow is defined as "the automation of procedures where documents, information, or tasks are passed between participants according to a defined set of rules to achieve ... an overall business goal" [1].

The limitations of the workflow representations can be illustrated with a typical design-process example. Figure 1 shows a workflow-style representation for an iterative design loop, where a design is elaborated or modified, then analyzed by one or more concurrent analysis activities, and then evaluated as to suitability. Depending on the evaluation, the loop is exited or another iteration of modification and analysis is performed.

Using the workflow representation, an engineer would like to specify transition rules indicating which of the seven analysis tasks should be performed, based on the characteristics and level of detail of the design for the current iteration. Initially, only two of the analysis tasks may be appropriate. If the evaluation results are good, the next iteration might refine the design, followed by analysis using four of the analysis tasks. Eventually an iteration using all seven analysis tasks might be performed on a fully detailed design candidate. At the end of any iteration, the evaluation might indicate that a modification is needed, in which case the level of detail and the number of analysis tasks might be reduced in the next iteration.

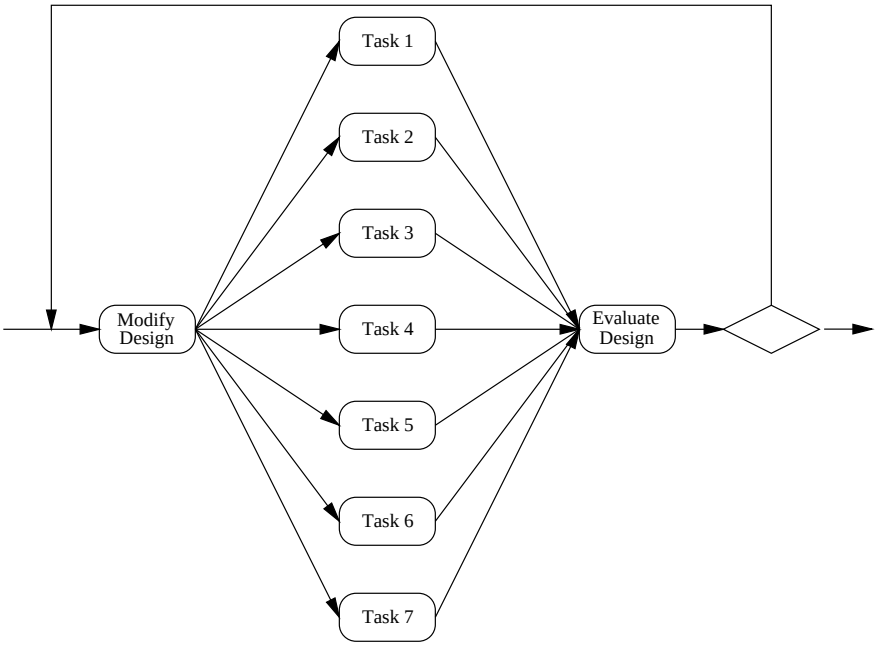

Figure 1. Workflow "Process" Representation

Fully representing this process in the workflow model is actually very difficult, and the reason is deeply rooted in the data-transition-centric representation. In a workflow engine, threads of control are split and joined according to data transition rules. The workflow model provides two forms: AND-splits/joins and OR-splits/joins [2]. With an OR-split, only one outgoing transition is followed. An AND-split creates control threads for all outgoing transitions. OR-joins are paired with OR-splits and are unsynchronized, as there are no concurrent threads created by an OR-split. AND-joins are synchronized rendezvous for AND-splits that wait until all threads have been completed before initiating the next activity.[2]

In our design-process example, some number of analysis tasks (not necessarily one or all) will be executed based upon the design data. This situation is problematic for a workflow engine, as the control threads needed to fulfill a rendezvous are not known because the full process structure is not represented in the workflow model.

From a process viewpoint, some number of analysis tasks are executed in parallel, sandwiched between a serial execution of a Modify-design task and an Evaluate-design task. An example process-based representation used in KPM, Knowledge Technologies International's dynamic-process management tool, is shown in Figure 2. In this representation, the non-atomic Analysis task doesn't complete until all the selected subtasks tasks have completed, at which point the Evaluate Design task can begin execution.

The representational limitations of workflow demonstrated in this simple design-process example become overwhelming when the process is scaled to include nested iterations, branches, quantified design operations, etc., making the workflow representation fundamentally deficient for modeling dynamic processes.

[2]Some workflow systems require a block structure where all the threads created at an AND-split must converge at a common AND-join point. Other *free-graph-structure* workflow systems allow threads to converge at different AND-join points.

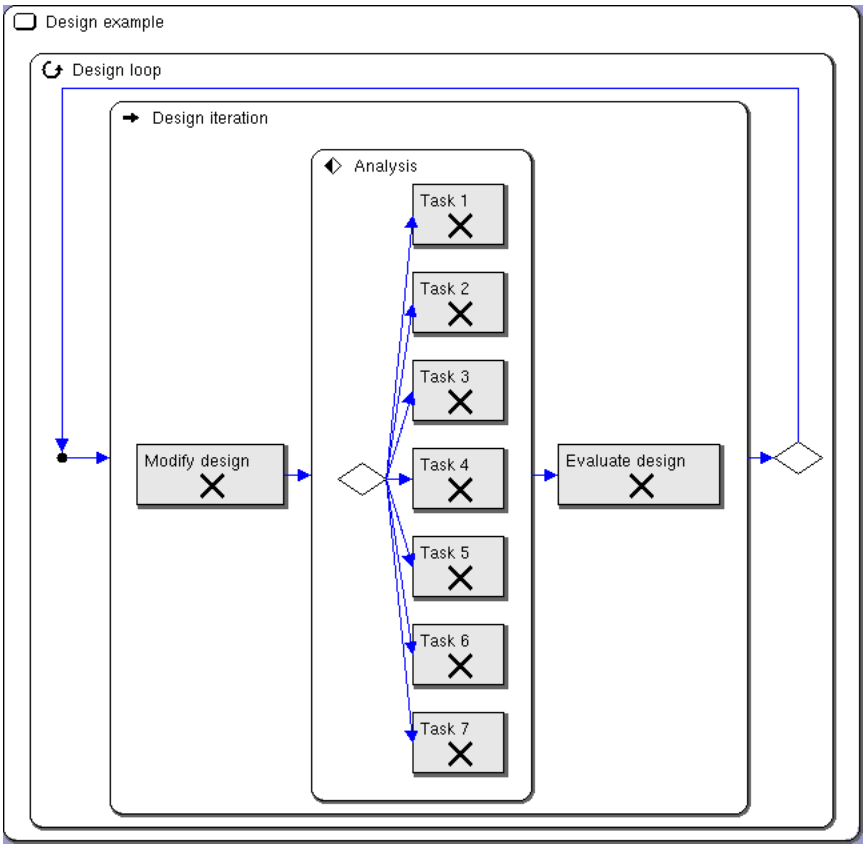

Figure 2. Hierarchical Process Representation (from KPM)

3.2. DIRECT MODEL EXECUTION

This requirement is satisfied by workflow tools and is the main reason workflow is suggested as an appropriate technology for dynamic processes. A workflow engine's job is to ensure that the activity model is properly interpreted and the data transitions are made according to the transition rules in the model. The problem with workflow is with the model that is being directly executed.

3.3. INTEGRATED DYNAMIC SCHEDULING

Workflow systems are reactive—not predictive—and do not provide expectations of downstream resource assignments and activity timings. Providing such expectations requires process knowledge about the expected duration of tasks, the branches likely to be taken at decision points, the expected number of iterations, etc. Workflow representations do not have a place for this knowledge, and workflow's data-transition structure prevents the scheduling of abstract process representations, which may be all that is present if a dynamic process is being elaborated on the fly.

An important aspect of dynamic scheduling is the ability to modify the schedules as expectations are violated. With a hierarchical, process-oriented representation, it is clear which tasks need to be removed from the schedule, which tasks need to be added, and which tasks need to be reassigned or rescheduled. Even if expected control threads were able to be scheduled from the workflow representation, the task of identifying which scheduled control threads should be terminated, which should now be scheduled, and which should be modified would be very

difficult.

3.4. LIVE PRESENTATION OF EXECUTION STATUS, HISTORY, AND EXPECTATIONS

Workflow products support varying degrees of current-state presentation and reports (audit trails) of past execution. Without predictive capabilities, however, workflow provides no help in identifying potential downstream problems. Even in the presentation of the current state, the workflow model limits the ability to present arbitrary abstractions of the dynamic process, forcing the presentation to be comprehended at the detailed level of the workflow representation.

3.5. ON-THE-FLY PROCESS MODIFICATION

Although static workflow systems require that the process representation remain fixed during execution, dynamic and ad-hoc workflow systems allow transition rules and even activity structures to be changed on the fly. These dynamic and ad-hoc workflow systems meet this requirement, but, because of the limits in the workflow representation and workflow's inability to provide managers with expectations, this modification capability provides minimal benefit. Consider the difficulties associated with adding several additional analysis tasks at execution time to our simple design-process example or moving the design loop (and all of its activities) to a different point in an encompassing process representation. The need to implement such changes at the transition-rule level makes the modification tedious and error prone. On the other hand, a hierarchical process-based representation permits these modifications to be made with a simple drag-and-drop of abstract process components.

3.6. THE REPORT CARD ON WORKFLOW

Although workflow is a suitable technology for traditional business processes, it fails to satisfy the majority of requirements needed for dynamic-process management. The limitations of workflow are integral in the workflow standard, and there are no simple fixes or extensions that can remove these limitations and extend workflow's suitability to the domain of dynamic processes.

Instead of looking for solutions in technologies created for other purposes, it is time to acknowledge that a new class of software for supporting dynamic-process management is needed.

4. KPM: A dynamic-process management tool

KPM is a management-support product suite, comprising the following six components, developed specifically to meet the five requirements for dynamic-process management software:

- *KPM Server*—An enactment server for maintaining process definitions and libraries and for instantiating and executing a KPM process instance. The KPM Server includes a dynamic scheduler and can cooperate with other KPM Servers in coordinating the use of common resources and information.
- *KPM Developer*—A graphical development client for creating, editing, and maintaining KPM process definitions and libraries.
- *KPM User Assistant*—A browser-based graphical desktop "agent" for people performing tasks in a KPM process execution. The User Assistant notifies the user of task requests, maintains a to-do list of assigned tasks and tasks in process, allows users to launch desktop applications and update and complete tasks.
- *KPM Execution Manager*—A graphical monitoring and control client for managing an executing KPM process. The Execution Manager allows users to stop, edit, restart and redefine an executing process instance.

- *KPM Execution Monitor*—A read-only version of the Execution Manager used by individuals who need to track the progress of an executing process but don't need to change it.
- *KPM Task Valet*—A desktop tool that allows a user to specify what activities a KPM Server can perform on her account. The Task Valet runs these fully-automated activities as requested by the KPM Server without requiring user supervision.

These components are shown in Figure 3. A delivered system consists of a custom configuration of these components, distributed across the organization (potentially including suppliers as well) [3].

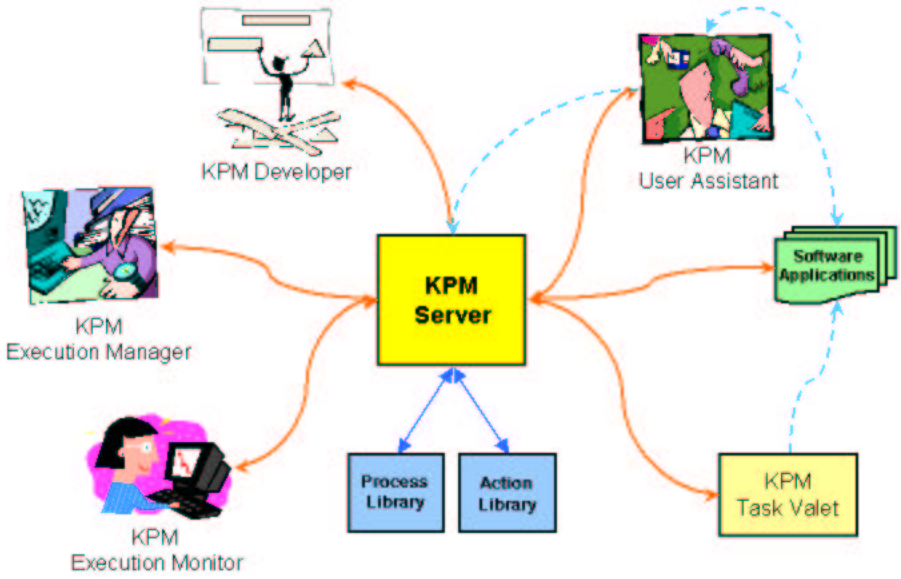

Figure 3. The Components of KPM

5. Status and Future Work

KPM was launched in November 1999, and since then initial KPM-based projects have been completed at major automotive and aerospace manufacturers. Follow-on projects are now underway. The response to KPM has been enthusiastically positive, and users are obtaining the following benefits from KPM's dynamic-process-management approach:

- **Improved process understanding**—KPM requires that the full process structure be fully captured in order to be executed. KPM ensures that the process that is represented is the process that is executed, and any mismatches in the way processes are viewed and the way they actually perform are quickly discovered.
- **Capture and formalization of best practices**—Many organizations are emphasizing the development and use of "best practice" standardized work packages. KPM's process-based representation is a natural environment for this activity.
- **Improved process visibility**—The instantaneous view of process status, presented using KPM's process-oriented representation, makes it easy to see how a dynamic process is behaving.
- **Advanced notice of problems**—KPM's dynamic scheduler provides early warning of downstream resource-scheduling and timing problems in executing dynamic processes.

KPM's dynamic-scheduling capability is particularly helpful in managing resources that are shared among multiple dynamic processes.

- **Capture of execution history**—KPM captures the dynamic-process-execution history (in addition to the data transitions).

It is too early to report case-study measures of dynamic-process cost and duration reduction, as KPM is just being introduced to production use. Given the interest shown in the initial KPM releases, we believe that KPM shows great promise in addressing dynamic-process management issues.

References

[1] David Hollingsworth. *The Workflow Reference Model*, volume WfMC-TC00-1003. Workflow Management Coalition, http://www.aiim.org/wfmc/standards/docs/tc003v11.pdf, January 1995.

[2] Workflow Management Coalition. *Workflow Management Coalition Terminology & Glossary*, volume WFMC-TC-1011, 3.0. Workflow Management Coalition, http://www.aiim.org/wfmc/standards/docs/glossy3.pdf, February 1999.

[3] Susan Lander, Daniel Corkill, and Zachary Rubinstein. KPM: A tool for intelligent project management and execution. In *Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business, Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, August 1999.