A Framework for Adaptive Process Modeling and Execution (FAME)

Perakath Benjamin
Knowledge Based Systems, Inc.
pbenjamin@kbsi.com
Madhav Erraguntla
Knowledge Based Systems, Inc.
merraguntla@kbsi.com
Richard Mayer
Knowledge Based Systems, Inc.
rmayer@kbsi.com
Michael Painter
Knowledge Based Systems, Inc.
mpainter@kbsi.com
Charles Marshall
Knowledge Based Systems, Inc.
cmarshall@kbsi.com

Abstract

This paper describes the architecture and concept of operation of a Framework for Adaptive Process Modeling and Execution (FAME). The research addresses the absence of robust methods for supporting the software process management life cycle. FAME employs a novel, model-based approach in providing automated support for different activities in the software development life cycle including project definition, process design, process analysis, process enactment, process execution status monitoring, and execution status-triggered process redesign. FAME applications extend beyond the software development domain to areas such as agile manufacturing, project management, logistics planning, and business process reengineering.

1. Introduction

The information revolution of the last decade has led to significant technology developments in computer based systems. These developments have produced faster, cheaper, and easier-to-use computing technology, both for business and home users. The rapid pace of growth, however, has led to an increased need for innovative methods, tools, and infrastructures for the enterprises that design and produce these computer-based systems. The prolific growth of the software industry in particular has created intense demands for new and improved production systems, processes, and infrastructures. This paper focuses on technology that can support a critical aspect of software production: the cost effective management of the *software development process*.

This paper presents a novel, life cycle-integrated approach to managing software development projects. There are four main activities in the process life cycle: process modeling, process analysis, process enactment, and process execution status monitoring and control [2], [5]. Previous approaches to providing automated support for the process lifecycle have focused attention on subsets of the lifecycle [8], [6], [4]. The SMART architecture represents the first attempt to provide comprehensive support for the entire process life cycle [2]. SMART, however, does not provide robust support for execution status monitoring and process redesign based on feedback status-analysis driven feedback.

This paper presents the architecture and the concept of operation of FAME, a *Framework for Adaptive Process Modeling and Execution*. FAME can be used to support the following life cycle software development project management activities: i) defining the project objectives and metrics, ii) designing the software development process, iii) analyzing the alternative processes, iv) enacting the software development process (i.e., providing automated support for process execution), v) monitoring the process execution, and vi) facilitating rapid redesign of the software process based on deviations detected from process execution status data analysis.

Section 2 describes the FAME architecture. The FAME Concept of Operation is presented in Section 3. Section 4 summarizes the paper and enumerates potential applications and benefits of the research.

2. Architecture

FAME employs an open architecture that provides life cycle support for the software development process (Figure 1).

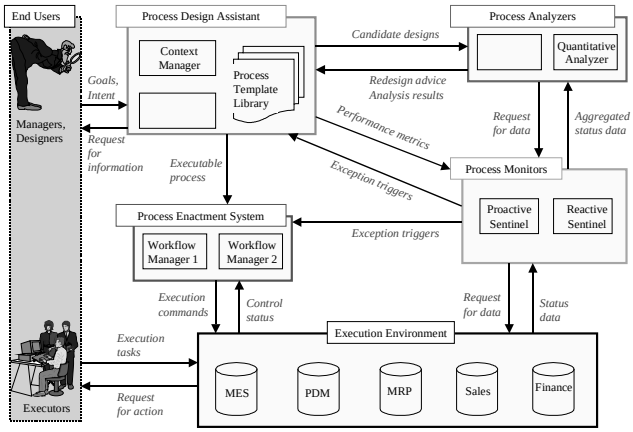

Figure 1. FAME architecture

FAME implements a model-based approach to managing software development projects. "Living" models of the software development process are used to predict the performance of the process before it is implemented; the process models are used to generate executable process enactment specifications which are then used to execute the software development process. Process monitors are used to gather execution status data; process exceptions and deviation from performance targets are used to redesign the process. The redesigned process models are then used for further analysis, execution, and monitoring. FAME is thus a *closed-loop feedback control architecture* and provides adaptive capabilities through proactive status monitoring and rapid process redesign. FAME therefore enables model-based "continuous process improvement" of software development projects. The main FAME subsystems and their functions are described in the following sections.

2.1 Process Design Assistant

The Process Design Assistant facilitates project definition and template-based process design. The Context Manager facilitates the definition of software development process intent: the definition of objectives, scope, and the establishment of project performance metrics. The Process Composer facilitates the design of software processes. Design is typically done by selecting a process template (a "blueprint") from the Process Template Library (PTL) and then customizing the template to address the objectives of the focus software development project. The Context Manager also assists in the selection of "good" starting templates. The PTL templates are organized taxonomically and the process designer can browse/navigate these taxonomies during the design effort.

2.2 Process Analyzers

The Process Analyzers provide support for qualitative and quantitative process analysis. The analyzers are invoked in two different life cycle phases of the software development process: i) during process design (to evaluate the relative merit of alternative designs of the software development process) and ii) during execution status monitoring (to evaluate process execution status data compiled by the status monitoring "Sentinels"). The analysis results are used to redesign the process to rectify undesirable execution performance deviations (see Section 2.4).

The Qualitative Analyzer (QA) is a rule-based expert system that diagnoses potential problems with the software development process or the execution status, and suggests alternative ways to rectify the problems with the designed process or with the executing process.

The QA rules support three analysis items: i) diagnostic checking for potential problems in the designed process, ii) consistency and completeness checking, and iii) process improvement—advice to move from a current design state to a desired target design state.

The Quantitative Analyzer (QNA) supports different kinds of quantitative analyses. Quantitative process analysis techniques supported by QNA include i) Systems Simulation, ii) Activity Based Costing (ABC), iii) PERT/CPM, and iv) Life Cycle Costing (LCC). These techniques permit more detailed and sophisticated analysis of the process and thus yield more comprehensive data for decision making.

2.3 Process Enactment System

The Process Enactment System is a collection of workflow management tools that facilitate the execution/enactment of the software development process. Because the software development process is highly knowledge intensive, several development tasks must be performed by humans (i.e., some tasks cannot be automated). The workflow management tools serve to route and coordinate the flow of work as specified in the designed process. The workflow enactment specifications are automatically generated from the designed processes.

2.4 Process Monitors

The Process Monitors are an agent-based system that monitors the process execution status, gathers compiled process status information, and submits this information to the Process Analyzers for detecting and correcting process exceptions and deviations. Reactive Sentinels are software agents designed to collect execution status data. These sentinels are programmed to detect process execution problems *after* they occur. The type of performance metrics that need to be reported to the software development project managers guides the selection of the type data to be monitored. For example, if the manager would like to look at the actual activity execution duration time for Activity A, then the sentinel gathers start and finish times for all activities that are part of A's decomposition. This execution data is obtained by the Sentinels through the Process Enacters and are sent to the Process Analyzers for further analysis and interpretation.

Proactive Sentinels are anticipatory software agents that detect process execution problems *before* they occur. Proactive sentinels use a variety of different techniques including neural networks, fuzzy logic, simulation, and scheduling to forecast, based on the current execution status, undesirable process deviations that are expected to occur in the future. For example, the occurrence of a certain kind of software bug, coupled with the time taken to develop the module in which the bug occurred, may be indicative of a certain type of programming style that needs to be rectified. Sentinels may then be trained to look for such patterns in the process execution data. Similarly, simulation based scheduling using process execution time data may be used to predict shortages of software development resources in the near term.

The FAME Execution Environment (see Figure 1) shows examples of applications and data sources that could reside in the environment where the software development process is being managed. The current implementation of FAME runs on Windows™-based environments; it is currently being migrated to a multi-platform implementation. The FAME tools include commercial software and advanced prototypes.

3. FAME Concept of Operation

This section describes the FAME concept of operation: in other words, the activities supported by FAME from an end users perspective. The FAME end-user activities and the relationships between these activities are shown in Figure 2.

FAME supports the following main kinds of activities: i) Define Software Development Project, ii) Develop Process Design, iii) Analyze Process, iv) Enact Software Development Process, and v) Monitor Process Execution. Together these activities allow for a closed loop software development process plan design, analysis, execution, monitoring, and sentinel-driven redesign. The activities supported by FAME are described in more detail in the following sections.

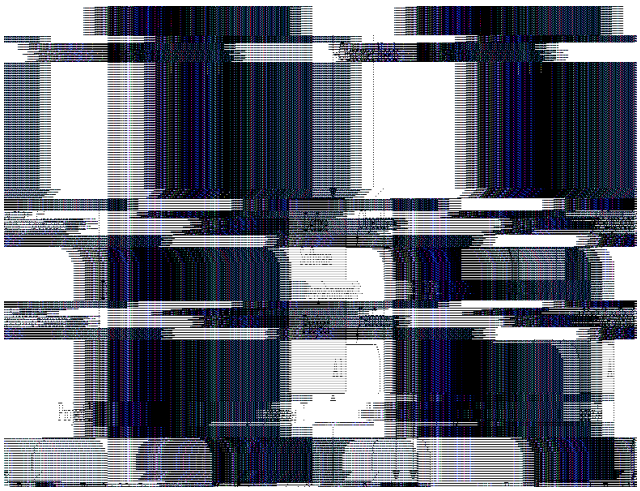

Figure 2. FAME concept of operation

3.1 Define Software Development Project

A preliminary activity associated with software development is defining the development project. Project definition involves five main activities: i) defining software development objectives, ii) establishing the scope of the development effort, iii) defining performance metrics, iv) defining major software development functions, and v) assigning roles and responsibilities for these functions. The FAME Context Manager supports these functions. The Context Manager maintains a library of useful project management objectives and metrics that help evaluate whether and how these objectives are obtained. This library allows the end user to browse and retrieve previously used objectives and metrics for re-use and specialization (if appropriate) on the current development project. An example taxonomy of generic software project management metrics is shown in Figure 3.

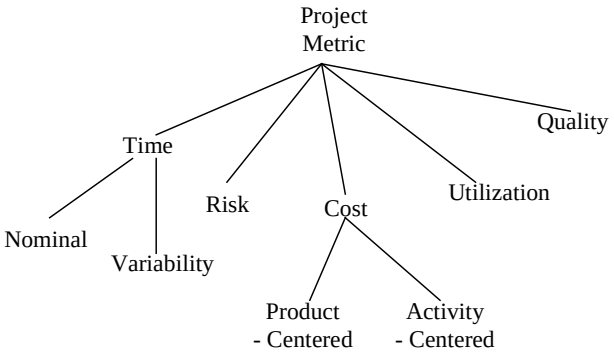

Figure 3. Partial taxonomy of project oversight metrics

3.2 Develop Process Design

Process Design is an inductive process. Rather than starting from scratch, real-world designers attempt to re-use knowledge of previous designs. Process design knowledge re-use is facilitated through the FAME Process Template Library (PTL). The PTL is a structured collection of content-rich, re-usable process templates. The FAME Context Manager (Figure 1) provides automated support for selection of a "good" starting template to help accomplish the current software development project objectives. The FMAE Process Composer provides a graphical browser that helps the user visually navigate process taxonomies within the PTL. In situations where none of the PTL templates can be re-used, the software development process is designed from scratch.

3.2.1 Process Decomposition

FAME supports a phased "decompose and aggregate" strategy to facilitate distributed process design and analysis. FAME supports scenarios of use in which the end users may be distributed in time and space. With FAME, design requirements are allocated/dis-aggregated and distributed to high level activities and functions. Detailed design activities are then performed by designers who are geographically distributed. Once the design activities of the different project participants is complete, FAME supports the composition and harmonization of these (piece-wise) process modules and fragments into end-to-end integrated processes.

FAME uses heuristics for two main activities: i) decomposing requirements and metrics and ii) aggregating performance information from detailed process plans to meaningful metrics at higher levels of abstraction. We'll describe the decomposition mechanism with the help of an example (the aggregation mechanism is described in Section 3.3).

Suppose that the target completion time for Enterprise Z's software development effort is 200 days. It is also known from the process template library that the software development effort will require the activity types and the "proportional" times for completion shown in Table 1. FAME computes a "Target Completion Time" for these activities by decomposing the total time (200 days in this example) to the activities using the proportions given in the second column. Note that the proportional assignment of time targets assumes that these activities are performed sequentially. A different heuristic is used for parallel activities. Similar heuristics have been developed for cost, risk, resource utilization, and quality metrics.

Table 1. Example metric decomposition

Activity Name	Proportion of Time (%)	Target Completion Time (Days)
Define Requirements	15	45
Develop Conceptual Design	15	45
Develop Detailed Design	25	75
Develop Prototype	20	60
Develop Final Product	25	75

3.2.2 Detailed Design

Process design involves constructing a series of transformations that will produce a desired output starting from a set of inputs. Designing a software development process thus involves determining a collection of activities that will result in a software program that performs as desired by the end user. Designs are typically done using one or a combination of two strategies: i) from scratch ("greenfield design") or ii) through the specialization of a set of re-usable templates/patterns ("template-based design"). The latter approach is more popular in typical software development efforts.

Regardless of the strategy adopted, the following conceptual activities are involved in process design.

1. Design of inputs and outputs
2. Design of resources
3. Allocation of resources to activities

4. Design of activities that transforms a given set of inputs to a desired set of outputs
5. Design of precedence constraints between activities or sets of tasks
6. Design of inter- and intra-activity coordination mechanisms (for example, a management activity typically needs to be designed to help coordinate and resolve contentions for resources that are shared between different tasks)

The FAME Process Composer provides automated support for these tasks. The Process Composer is based on the IDEF3 process description capture method and the IDEF5 ontology description capture method [10], [9].

3.2.3 Process Composition

An important aspect of process design for large scale software development is process composition. Process composition is important because processes and process knowledge is becoming increasingly distributed. Process knowledge is distributed both within an organization and also across Virtual Enterprises (VEs)—collections of organizations that collaborate for a specific purpose [3]. We anticipate that organizations will increasingly use VEs for software development. FAME facilitates the composition of end-to-end integrated processes from process segments and process modules.

3.2.4 Process Harmonization

Once processes have been integrated, they need to be "harmonized." Process harmonization refers to the detection and rectification of mismatches that would lead to undesirable performance during process execution. A variety of qualitative and quantitative analysis mechanisms are provided by FAME to support process harmonization. A more detailed description of the FAME process design mechanisms is provided in [12]. The FAME process analysis functionality is described in the following sections.

3.3 Analyze Process

FAME supports a variety of different qualitative and quantitative analysis techniques. These analysis techniques are invoked during process design and to assess process execution status design as a precursor to process redesign.

3.3.1 Qualitative Analysis

The FAME Qualitative Analyzer (QA) provides knowledge-based support for qualitative process analysis. The QA helps the software process designer detect and diagnose potential problems in the process design models. Candidate process designs are submitted to the QA for qualitative analysis. The QA then "fires" the rules to produce a list of potential design problems that require action by the designer. Using the QA-generated diagnostic advice, the designer then modifies the process design model. These diagnosis and refinement activities are performed repeatedly until an "acceptable" process design is produced.

There are four main types of QA rules contained in FAME. These rules are briefly described in the following list.

1. **Input Output Mismatch Rule.** This rule matches input requirements for an activity with the outputs (and other sources) from activities servicing this activity. If one or more required inputs are not being provided to this activity, then this condition is tagged as a potential problem. A variant of this rule is to match the attributes of the inputs of an activity with the attributes of the corresponding outputs that are being delivered by other activities to service the input requirements. For example, if a design activity requires that a Requirements Document delivered to it conform to the "XYZ" format and the activity providing this delivers the Requirements Document using another format (say "PQR"), then this condition is tagged as a potential problem. Input output mismatches often occur at organizational boundaries and are especially relevant to VEs.
2. **Rate Mismatch Rule.** This rule compares the rate at which inputs arrive at an activity with the processing rate of that activity. If the rate at which inputs arrive at an activity exceeds the activity processing rate (that is, its capacity), then this condition is tagged as a potential problem.
3. **Ontology Mismatch Rule**. This rule detects problems associated with the terms used to describe the activities and the interpretation of these terms in different contexts. Specifically, the ontology mismatch rule attempts to detect terms of the same name that appear to have different meaning and terms with different name but appear to have the same or similar meaning. In addition to these rules, the FAME Process Composer supports ontological analysis, which allows end users that helps end user detect ontology mismatches by

browsing and navigating domain ontology descriptions relevant to the software project.

4. **Bottleneck Resource Condition Detection Rules**. This rule detects the existence of resources for which demand exceeds capacity (we refer to such resources as "bottleneck" resources). Bottleneck resources will likely cause delays in the enactment of the process being designed and are thus tagged as a potential problem.

3.3.2 Quantitative Analysis

The FAME Quantitative Analyzer (QNA) supports different kinds of quantitative analysis. Quantitative analysis techniques are used for providing additional (and often detailed) information about the process performance. Quantitative process analysis techniques supported by QNA include i) Systems Simulation, ii) Activity Based Costing (ABC), iii) PERT and CPM (schedule analysis techniques), and iv) Life Cycle Costing (LCC).

Quantitative analysis techniques are often used to compare the relative merit of alternative process designs [13]. The analysis results provide the information needed to select a "best" process design from a set of competing alternatives.

1. **Process Performance Aggregation.** FAME aggregates performance information from detailed activities, performs "roll-up" calculations, and reports the aggregated information at a higher level of abstraction. Metrics that are aggregated include time, cost, and risk. A variety of different heuristics have been developed to perform the aggregation computations [11]. Abstraction mechanisms are useful both for large scale process and project management: decision makers operate at an abstract level, whereas performance information is derived from information or data at a finer level of granularity. An example "qualitative" time aggregation calculation is shown in Figure 4.

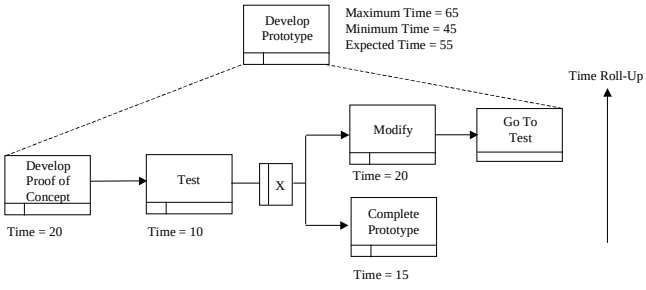

Figure 4. Example process performance roll-up heuristic

2. **Analyze Execution Status Data.** Once the execution status data has been collected and compiled by the Sentinel Agents, it is sent to the FAME Process Analyzers for assessment and diagnostic analysis (see Figure 1). The Process Analyzers apply different qualitative and quantitative analysis techniques to determine whether i) current performance has deviated from the process targets and ii) there are early warning signals that indicate poor performance in the future (anticipatory diagnostics). The analysis results are used to automatically redesign the software development process wherever possible. In situations where an automated process repair strategy is not obvious, the Process Analyzers present a set recommendations to the user along with the analysis results.

3.4 Enact Software Development Process

After the software development process has been designed it needs to be executed/enacted. Enactment is facilitated by the FAME Process Enactment System. Workflow Management Tools are used to perform process enactment. The workflow enactment specifications are automatically generated from the development process models produced by the Process Design Assistant (see Figure 1). Because the kind of workflow tool that is appropriate for a given enterprise depends on the complexity of the software development effort [1], FAME supports the generation of code to different kinds of workflow tools, as needed. This is facilitated through the use of a "neutral" process representation language called the Enhanced Process Interchange Format (EPIF) (see [7] and [12]). In situations where the work is distributed across multiple enterprises (that is, the software development project is performed by a virtual enterprise), it is often necessary to utilize a collection of different workflow tools in concert.

3.5 Monitor Process Execution

As the process executes, it needs to be monitored in order to ensure compliance with performance targets. FAME uses two kinds of status monitoring agents to perform process execution status monitoring: i) Reactive Sentinels and ii) Proactive Sentinels (see Section 2.4). The Sentinels gather the appropriate status data from the execution environment and send their output to the Process Analyzers. Analysis of the sentinel-driven data is used to determine what changes must be made to the process to correct poor current or past performance ("post mortem"

change) or to prevent any incidence of future problems or undesirable performance (proactive change).

Based on the analysis results, the processes are redesigned, the execution specifications are regenerated, and the process improvement cycle repeats. The concept of operation thus shows how the FAME system supports the different phases of the software development process life cycle.

4. Summary

This paper described the architecture and the concept of operation of FAME, a Framework for Adaptive Process Modeling and Execution. FAME is a model-based system than can be used by software development project teams to plan, analyze, schedule, execute, monitor, and control their work. The FAME architecture is general enough to be used effectively for process management in a variety of different application areas including agile manufacturing, project management, logistics planning, and business process reengineering. A variant of the FAME architecture was successfully deployed on two U.S. military process improvement initiatives in the manufacturing and logistics application domains [12]. FAME benefits include i) agile software development capabilities (the ability to respond quickly and cost-effectively to an unpredictably changing environment), ii) reduced time and cost for software development through robust process engineering and control, and iii) reduced life cycle costs through enhanced support for the software development process life cycle over extended periods of time.

5. Acknowledgements

The research described in this paper was partly funded by the U.S. Defense Advanced Research Project Agency, Contract Number F33615-96-C-5601.

6. References

[1] P. C. Benjamin, C. Marshall, and R. J. Mayer, "A Workflow Analysis and Design Environment (WADE)", *Proceedings of the 1996 Winter Simulation Conference*, 1996.

[2] P. K. Garg., P. Mi, T. Pham, W. Scacchi, and G. Thunquest, "The Smart Approach for Software Process Engineering", *Proceedings of the 16th International Conference on Software Engineering*, IEEE CS Press, Los Alamitos, CA, 1994, pp.341-350.

[3] S. Goldman, R. Nagel, and K. Preiss, *Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer*, Van Rostrand Reinhold, New York, 1995.

[4] K. E. Huff and V. R. Lesser, "A Plan-based Intelligent Assistant that Supports the Software Development Process", *Proceedings of the 3rd ACM SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments*, ACM Press, New York, 1988, pp. 97-106.

[5] N. H. Madhavji, "The Process Life Cycle", *Software Engineering Journal*, September 1991, pp. 234-242.

[6] N. H. Madhavji, V. Gruhn, W. Deiters, and W. Schafer, "Prism = Methodology + Process-oriented Environment", *Proceedings of the 12th International Conference on Software Engineering*, IEEE CS Press, Los Alamitos, CA, 1990, pp. 277-288.

[7] C. Menzel, F. Fillion, and J. Hwang, "The Enhanced Process Interchange Format (EPIF) Summary", http://www.kbsi.com/research/projects/epif.html.

[8] P. Mi. and W. Scacchi, "A Knowledge-Based Environment for Modeling and Simulating Software Engineering Processes", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 2, No. 3, Sept. 1990, pp. 283-294.

[9] Knowledge Based Systems, Inc. (KBSI), "IDEF5 Ontology Description Capture Method Report", Information Integration for Concurrent Engineering (IICE) Project, WL/WPAFB, Dayton, OH, 1994.

[10] Knowledge Based Systems, Inc. (KBSI), "IDEF3 Process Description Capture Method Report", Information Integration for Concurrent Engineering (IICE), WL/WPAFB, Dayton, OH, 1996.

[11] Knowledge Based Systems, Inc. (KBSI), "Cost benefit Analysis Support Environment (CBASE)", Phase II Final Report to the Air Force, Contract Number F19628-95-C0045, 1997.

[12] Knowledge Based Systems, Inc. (KBSI), "Virtual Enterprise Engineering Environment (VE3)" Final Report to DARPA, Contract Number F33615-96-C-5601, 1997.

[13] Mayer, R. J., P. Benjamin, B. E. Caraway, , and M. K. Painter, "A Framework and a Suite of Methods for Business Process Reengineering", in B. Kettinger & V. Grover (Eds.), *Business Process Reengineering: A Managerial Perspective,*1995, pp. 245-90.