

An Open Architecture for Software Process Asset Reuse

Barry Boehm,

University of Southern California

Steven Wolf,

Rockwell Collins

Abstract

The development and reuse of software engineering processes within an organization can be impeded by the lack of a solid process framework. An open process architecture provides a framework through the identification of architectural elements and the specification of element interfaces. This paper introduces one open process architecture and examines some architectural element interfaces.

1. Introduction

Beginning with the paper, "Software Processes Are Software Too," [1], an attractive line of software process research has developed involving the exploitation of a duality between software products and software processes. This duality can be expressed as, "If a given approach (disciplined programming, requirements definition and validation, reuse, risk management, performance modeling) is good for software products, then its process counterpart is good for software processes."

Given that differences exist between software products and software processes (software products are executed by machines; software processes are executed by combinations of people and machines), this duality cannot be pushed too hard or applied unthinkingly. But, in a number of papers in the series of International Conferences on the Software Process, this duality has provided useful insights and results in such process areas as process life-cycles [2], process requirements [3], process validation [4], and process evolution [5].

This paper addresses the issue, "If open architectures are good for software product reuse, then their process counterparts will be good for software process reuse." The lack of such open architecture specifications has been a major impediment to the full development of Process

Asset Libraries: ensembles of reusable plug-and-play software process assets.

2. A "Toaster Model" Architecture

The open architecture for software process asset reuse presented here is a counterpart of the HP-NIST-ECMA "toaster" model for interoperability of tools within and across software environment frameworks [6]. It is shown in Figure 1. The product toaster model provides slots into which various instances of software tools can be placed. It also provides interface specifications along the slot boundaries which ensure that tools developed in compliance with the interface specifications will interoperate with each other.

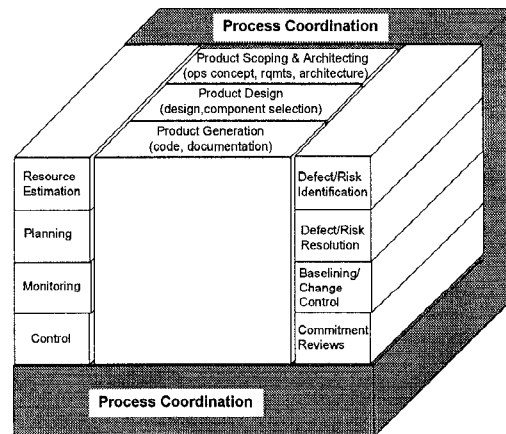


Figure 1. Open Process Architecture: Toaster Variant

The software process architecture in Figure 1 operates on the same principle. For example, the interface specifications between the Product Design slot and the Defect/Risk Identification slot are identified in Table 1 below in terms of preconditions and postconditions. If these preconditions and postconditions are satisfied by a design of any form (Booch, Coad-Yourdon, DeMarco, Jackson, Martin, Rumbaugh, Shlaer-Mellor, etc.) and by

any form of design defect/risk identification activity (Fagan inspection, structured walkthrough, quality metrics analyzer, automated consistency checker, SEI team risk assessment, corporate risk management group assessment, automated risk assessment advisor, etc.), then any of the forms of design can undergo any of the defect/risk identification activities in a predictable and controllable manner.

Table 1. Interface Specification Between Product Design and Defect/Risk Identification

Preconditions

1. Design artifact objectives and constraints: artifact requirements, artifact architecture, standards, quality attribute requirements, other objectives and constraints.
2. Design artifact.
3. Design artifact rationale: evidence that the artifact as designed would satisfy the objectives and constraints.
4. Defect/risk identification objectives and criteria: selection and prioritization among such criteria as completeness, consistency, traceability, feasibility, testability, implementation risk, cost/schedule risk, and such quality attributes as usability, maintainability, safety, reliability, availability, security, accuracy, etc. [7].
5. Necessary defect/risk identification resources (budget, schedule, qualified personnel, tools and techniques) or constraints.
6. Defect/risk resolution plan, including consistent specification of objectives, milestones, schedules, responsibilities, approaches, resources, and assumptions.
7. Commitment of the plan’s participants to the plan.

Postconditions

1. Satisfaction of defect/risk identification objectives and criteria, including resource constraints.
2. Plans for defect removal and risk resolution: these can be as simple as a list of actions, agents, and due dates; or they may take the form of plans with consistent specification of objectives, milestones, schedules, responsibilities, approaches, resources, and assumptions.
3. Commitment of the defect removal and risk resolution plan’s participants to the plan.

Table 2. Interface Specification Between Product Scope & Architecture and Resource Estimation

Preconditions

1. Product definition: operational concepts, product description, application environment.
2. Product architecture: decomposition of product into architectural elements, work breakdown structure(s), identification of potential reuse elements, etc.

3. Product size estimates: SLOCs (new, modified, reused).
4. Product constraints: special requirements, schedule, assumptions, compliance standards, process specifications, etc.
5. New technologies: hardware, software, development techniques applied.

Postconditions

1. Skill levels required to meet technology requirements.
2. Personnel assessment: determination of existing people with skills required, potential subcontracting organizations, etc.
3. Hardware: development and test resources currently available, shortcomings in existing equipment, special facility requirements.
4. Software /tools: development environment assessment, software upgrades to meet technology needs, etc.

An initial definition of the toaster process architecture and one example of the interface specification was provided in [8]. Subsequently, one paper [9] worked out sets of preconditions and postconditions for the 24 interfaces between the 3 Product Creation slots and the 8 slots for Planning and Control and Project Management. This paper [9] also provides more detailed definitions for each of these process categories.

3. Discussion

The preconditions and postconditions in Tables 1 and 2 and their 22 counterparts are sufficient to mediate the interoperation of human-executable process elements. They are not sufficient to mediate interoperation of computer-executed process elements. However, they provide a framework within which more detailed interface specifications for executable process elements could interoperate.

The refinement of the toaster process architecture proceeds in three major steps. The first step involves obtaining review feedback on the current preconditions and postconditions. Another step is to reorient the product creation process elements with the three “anchor point” milestones recently developed in [10]. These three milestones cover the achievement of a system’s Life Cycle Objectives (LCO), Life Cycle Architecture (LCA), and Initial Operation Capability (IOC). The components of the LCO and LCA milestones are given in Table 3. They indicate that the most important property of these milestones is the achievement of a compatible set of operational concepts, requirements architecture definitions, development plans, and stakeholder

commitments, along with a rationale substantiating their compatibility.

Milestone Element	Life Cycle Objectives (LCO)	Life Cycle Architecture (LCA)
Definition of Operational Concept	<ul style="list-style-type: none"> • Top-level system objectives and scope <ul style="list-style-type: none"> - System boundary - Environment parameters and assumptions - Evolution parameters • Operational concept <ul style="list-style-type: none"> - Operations / maintenance scenarios and parameters - Organizational life-cycle responsibilities 	<ul style="list-style-type: none"> • Elaboration of system objectives and scope of increment • Elaboration of operational concept by increment
Definition of System Requirements	<ul style="list-style-type: none"> • Top-level functions, interfaces, quality attribute levels including: <ul style="list-style-type: none"> - Growth vectors - Priorities • Stakeholders' concurrence on essentials 	<ul style="list-style-type: none"> • Elaboration of functions, interfaces, quality attributes by increment <ul style="list-style-type: none"> - Identification of To Be Determined items • Stakeholders' concurrence on their priority concerns
Definition of System and Software Architecture	<ul style="list-style-type: none"> • Top-level definition of at least one feasible architecture <ul style="list-style-type: none"> - Physical and logical elements and relationships - Choices of COTS and reusable software elements • Identification of unfeasible architecture options 	<ul style="list-style-type: none"> • Choice of architecture and elaboration by increment <ul style="list-style-type: none"> - Physical and logical components, connectors, configurations, constraints - COTS, reuse choices - Domain-architecture and architectural style choices • Architecture evolution parameters
Definition of Life Cycle Plan	<ul style="list-style-type: none"> • Identification of life-cycle stakeholders <ul style="list-style-type: none"> - Users, customers, developers, maintainers, interoperators, general public, others • Identification of life-cycle process model <ul style="list-style-type: none"> - Top-level stages, increments • Top-level WWWWWHH* by stage 	<ul style="list-style-type: none"> • Elaboration of WWWWWHH* for Initial Operational Capability (IOC) <ul style="list-style-type: none"> - Partial elaboration, identification of key TBD's for later increments
Feasibility Rationale	<ul style="list-style-type: none"> • Assurance of consistency among elements above <ul style="list-style-type: none"> - via analysis, measurement, prototyping, simulation, etc. - Business case analysis for requirements, feasible architectures 	<ul style="list-style-type: none"> • Assurance of consistency among elements above • All major risks resolved or covered by risk management plan

*WWWWWHH: Why, What, When, Who, Where, How, How Much

Table 3. Elements of Critical Front End Milestone

These milestones can indeed serve as common anchor points across mixed combinations of waterfall, evolutionary, incremental, spiral, design-to-cost, rapid application development, applications composition, and other process styles. Further, the use of these milestones to anchor a project's life cycle helps avoid many of the problems involved with critical milestones in previous process styles such as waterfall and evolutionary development [10].

The third major step experimentally applies the refined toaster model to software projects and iterates the refinement process based on the experiences. Several organizations affiliated with the University of Southern California - Center for Software Excellence are researching projects for the refinement of the toaster process architecture.

References

- [1] L. Osterweil, *Software Processes Are Software Too*, *Proceedings, ICSE 9*, IEEE, March 1987, pp. 2-13.
- [2] P. Feiler and W. Humphrey, *Software Process Development and Enactment: Concepts and Definitions*, *Proceedings, ICSP 2*, IEEE, February 1993.
- [3] M. Dowson, *Software Process Themes and Issues*, *Proceedings, ICSP 3*, IEEE, February 1993.
- [4] J. Cook and A. Wolf, *Toward Metrics for Process Validation*, *Proceedings, ICSP 3*, IEEE, October 1994.
- [5] S. Bandinelli, E. DiNitto, and A. Fuggetta, *Policies and Mechanisms to Support Process Evolution in PSEE's*, *Proceedings, ICSP 3*, IEEE, October 1994.
- [6] *NIST Special Publication 500-211, Reference Model for Frameworks of Software Engineering Environments* (Technical Report ECMA TR/55, 3rd ed.), National Institute of Standards and Technology, August 1993.
- [7] B. Boehm, *Verifying and Validating Software Requirements and Design Specifications*, *Software*, January 1984, pp. 75-88.
- [8] B. Boehm, *Software Process Architectures*, *Proceedings, ICSE 17 Architecture Workshop*, April 1995.
- [9] S. Wolf, *Open Process Architecture: Toaster Variant*, University of Southern California-Center for Software Excellence Report, December 1995.
- [10] B. Boehm, *Anchoring the Software Process*, *IEEE Software*, July 1996.