4. Automating Software Production

The two preceding chapters describe two related approaches to software-based product development without assuming automation beyond the continued use of conventional tools. A primary limitation of many such tools is their lack of support for explicitly building and evaluating alternative versions or multiple variants of a product (much less instances of a product family), incurring additional manual effort to address these.

Customers can have differing needs even for products serving similar purposes and those needs change over time. To account for this, a capability is needed to build products in a way that allows them to be progressively changed over time, whether to address a single or multiple customers' needs, without undue uncertainty or effort.

Achieving automation that properly supports the described approaches would provide significant advances in productivity and product quality over conventional practices. This chapter describes a progression of increased automation for streamlining of software production with either of the two described approaches, from an enhanced approach for component reuse to concepts for generalized or customized environments (i.e., factories) for rapidly building complete customized software-based products.

This automation has two objectives, building on the objectives of a DsE approach: leveraging developer competence to build higher quality products with less effort and exploiting the reality of market-based diversity and change. The basis for automation is a concrete realization of the concept of a family that enables the repeated mechanical derivation of customized instances. Adaptability of abstraction-based assets is the means for this automation.

Automation through adaptability progresses through three levels:

- The Systematic Reuse of Software Assets

  Previously built software assets can be opportunistically reused as-built or modified to fit differing needs. Alternatively, assets can be built in anticipation of reuse, with explicit guidance concerning changes that can safely be made. A particular form of the latter, developed as an enabler of DsE, is the use of a metaprogramming approach for building adaptable assets. An adaptable asset is a concrete realization of some “abstraction” (i.e., an envisioned family of similar assets), with an associated mechanism for deriving customized instances.

- An Abstraction-Based Environment

  An abstraction-based environment provides a generic descriptive notation for specifying a product model for a needed product based on its composition in terms of a set of computational abstractions. Each abstraction is represented as an adaptable asset from which customized instances are derived and combined to create a particular product realization. Any product whose realization can be described as a composition of the provided abstraction-based elements can be built.

- Domain-specific Environments for Producibility

  A domain-specific environment, pursuing the producibility vision, uses a constrained descriptive notation. This notation reduces development efforts by being limited to derivable instances of a specified product family. This leverages the increased degree of commonality that characterizes similar products, requiring resolution only of deferred decisions that are sufficient to discriminate among those products. Only instances of the specified product family can be built.