Domain Engineering

Operating System Engineering
What is Domain Engineering?
What is a Domain?

An area of knowledge (2, p. 34):

- scoped to maximize the satisfaction of the requirements of its stakeholders
- includes a set of concepts and terminology understood by practitioners in that area
- includes the knowledge of how to build software systems or parts of software systems
- defined by the consensus of its stakeholders, i.e., people having interest in a given domain: managers, marketing people, developers, vendors, contractors, standardization bodies, investors, customers, and end-users.
Domain Engineering as a Three-Phase Process

**Domain Analysis**

- Systematic organization of the existing (i.e., recorded) domain knowledge in a way that enables and encourages extensions to be made in creative ways:
  - Select and define the domain of focus
  - Collect all relevant domain information
  - Integrate the domain information into a coherent domain model

**Domain Design**

Development of an architecture for the family of systems in the domain and to devise a production plan.

**Domain Implementation**

Involves implementing the architecture, the components, and the production plan using appropriate technologies.
A domain model is an explicit representation of the common and the variable properties of the systems in a domain, the semantics of the properties and domain concepts, and the dependencies between the variable properties. ([2], p. 23)
Constituents of a Domain Model

Domain definition

- Define the scope of a domain and characterize its contents.
- Define a set of reusable and configurable requirements for specific systems in the domain.

Domain lexicon

- Define the domain vocabulary.

Concept models

- Describe the concepts in the domain in some appropriate mode.
- Define formalism (e.g., object, interaction, state-transition, entity-relationship).
- Define informal text and diagrams (e.g., flow diagrams).

Feature models

- Define a set of reusable and configurable requirements for specific systems in the domain.

Domain engineering

- Define the domain vocabulary.

System or capability

- Define generic rules (including the rationale) of inclusion or exclusion of a given system in the domain.
- Define examples of existing systems in the domain, counterexamples, and counterexamples in the domain.
- Describe the scope of a domain and characterize its contents.
Feature and Feature Model

Feature

Are reusable and configurable requirement.

Feature Model

Used in domain analyses to capture the commonalities and variabilities of systems in a domain:

- Feature A Reusable and Configurable Requirement
- Feature B Reusable and Configurable Requirement

Application Engineering

Is the basis for developing the means of ordering concrete systems during software development. And in the end, the configuration aspect of the whole reusable model represents the configuration aspect of the concept models previously described.● preferred under which conditions and why● prescribes which feature combinations are meaningful, which of them are

Feature Model Used in domain analyses to capture the commonalities and

Feature and Feature Model
Activities of Domain Analysis

- data collection and analysis, taxonomic classification, evaluation

**domain modeling**

Yields the domain model:

- data collection and analysis, taxonomic classification, evaluation

**domain scoping**

Identities the domain of interest, the stakeholders and their goals, and defines the scope of the domain of interest, the stakeholders and their goals.

**domain selection and description**

Aims at finding a domain scope that is economically viable and promises business success.
When developing a software architecture, we have to consider not only functional requirements but also non-functional requirements, such as performance, robustness, fault tolerance, extendability, adaptability, and reusability. Indeed, one of the purposes of software architecture is to be able to quickly tell how a piece of software satisfies its requirements. ([2], p. 27)
An arrangement where several specialized subsystems assemble their knowledge to build a partial or approximate solution to a problem for which no deterministic solution strategy is known.

Pipes and Filters  An arrangement that processes a stream of data, where a number of processing steps are encapsulated in filter components. Data is passed through pipes between adjacent filters, and the filters can be recombined to build related systems or system behavior.

Layers  An arrangement into groups of subtasks in which each group of subtasks is at a particular level of abstraction.

Architectural Patterns (1)
Architectural Patterns (2)

A broker is an arrangement where decoupled components interact by remote service invocations. A broker component is responsible for coordinating communication and for transmitting results and exceptions. A change-propagation mechanism ensures consistency between user interface and model.

Model-view-controller A decomposition of an interactive system into three components: A model containing the core functionality and data, one or more views displaying information to the user, and one or more controllers that handle user input. A change-propagation mechanism ensures consistency.

Microkernel An arrangement that separates a minimal functional core from extended functionality and custom-specific parts. The microkernel also serves as a socket for plugging in these extensions and coordinating their collaboration.

Model-view-controller A decomposition of an interactive system into three components: A model containing the core functionality and data, one or more views displaying information to the user, and one or more controllers that handle user input. A change-propagation mechanism ensures consistency.

Broker An arrangement where decoupled components interact by remote service invocations. A broker component is responsible for coordinating communication and for transmitting results and exceptions. A change-propagation mechanism ensures consistency between user interface and model.
Multi-Architecture Design

Preferences for specific architectural patterns must never be a dogma. Preferences for specific architectural patterns must never be a dogma.

- on the domain
- on the specific platform
- on the specific application

which (and how many) of the patterns to apply depends...

- on the domain
- on the specific platform
- on the specific application

Real architectures are usually based on more than one of these and other parts, views, and at different levels of an architecture ([2], p. 28).

no single architectural pattern is superior to the others.
The skeleton has to be modified sometimes. ([2], p. 28) The skeleton and keep the rest flexible and easy to evolve. But even if the skeleton has to be modified and still leaves a large degree of freedom for the implementation, its goal is to come up with a flexible structure that satisfies all important requirements and still leaves a large degree of freedom for the high-level design. As a rule, we use the most stable parts to form the "skeleton."
Architectures and their Amount of Flexibility

A generic architecture can be thought of as a fixed frame with a number of sockets where we can plug in some alternative or extension components.

Highly flexible architecture supports structural variation in its topology.

The resulting system has a fixed topology and fixed interfaces.

Components and sockets must clearly specify their interfaces.

It allows us to negotiate and configure interfaces.

Even the skeleton has been componentized – it can be configured to yield a particular generic architecture.

An architecture for a system family has to include an explicit representation of the variability (i.e., configurability) it covers.

Components and sockets must clearly specify their interfaces.

It can be thought of as a fixed frame with a number of sockets where we can plug in some alternative or extension components.
Production Plan

A production plan describes how to build a system from the common architecture and components. It makes up the second artifact to be developed during domain design.

- The measuring, tracking, and optimizing of the production process
- The process of handling change requests and custom development
- The process of assembling the components
- The interface to the customers ordering concrete systems

That is to say, a production plan describes...
Automation Levels of the Assembly Process

1. Excluded therefrom are parts requiring custom development.

2. Generative Programming

3. Order Record

4. A set of tools supports the process of assembling an application.

5. Various tools including component browsing, search tools, and generators automatically assemble selected aspects of application development.

6. The assembly of components is supported with a set of tools.

7. A set of tools supports the process of assembling an application.

8. Operating System Engineering—Domain Engineering

9. Generative Programming
Initially, the application is assembled manually or generated automatically — to be able to refine and extend the reusable assets.

Finally, the new customer requirements should be fed back to domain engineering.

(2[p.30])

During the requirements analysis for a new concrete application, we take advantage of the existing domain model and describe customer needs using the features (i.e., reusable requirements) from the domain model.

The process of building systems based on the results of domain engineering.
Conventional software engineering concentrates on satisfying the requirements for a single system, whereas Domain Engineering concentrates on providing reusable solutions for a family of systems. ([2], p. 21)
Bibliography


