4.2 Families of Programs
Overview of Chapter 4.2

- basic idea of families of programs
- . . . and what to do if the first version is due yesterday

First paper to introduce families of programs explicitly. Presents the essentials very clearly.

Best current book on how to do software product line engineering (families of programs) in practice.
Definition of Program Family

Definition 20 (Program family)

A set of programs constitutes a family whenever it is worthwhile to study programs from the set by first studying the common properties of the set and then determining the special properties of the individual family members.

• examples:
  ○ the set of versions of an embedded software for different environments
  ○ the set of versions of a software over time
The "Classical" Method of Producing Program Families

- set of initial possibilities
- incomplete program
- working program

Diagram:

0 → 1 → 2 → 3 → 4 → 6 → 7

5

9 → 8
Newer Techniques

- set of initial possibilities
- incomplete program
- working program
Stepwise Refinement

- intermediate stages:
  - complete programs
  - except: certain operators and operand types
    - only specified, not yet implemented

- next step: provide some more implementation,
  using more, newly introduced specifications as necessary

- linear sequence of steps towards one program
  - if a step must be taken back, all subsequent steps are lost
Module Specification

• intermediate stages:
  ◦ black-box specifications of modules
  ◦ not complete programs

• next step: add design decisions for a module, using newly introduced sub-modules as necessary

• steps taken in different modules are independent
  ◦ any step taken back affects its sub-modules only
  ◦ order of steps: more important
  ◦ independent further development of modules
Discussion of Both Development Approaches

- both based on same basic ideas:
  - represent intermediate stages precisely
  - postpone certain decisions
- extra effort to design first family member:
  - stepwise refinement: none
  - module specification: significant
- effort to design next family members:
  - stepwise refinement: high, if early step taken back
  - module specification: low, as long as low uses-level modules affected
Dilemma: Careful Engineering vs. Rapid Production

- careful engineering:
  - attractive functionality
  - ease of use
  - reliability
  - easy to enhance

- rapid production:
  - market it ahead of competition
A Solution in Other Fields

- **fields:**
  - aerospace
  - automotive
  - computer hardware
  - ...

- **idea:** _a family of products_
  
  *produced with a single production facility*
• family: set of items
  ○ common aspects (e.g., chassis)
  ○ predicted variabilities (e.g., engine)

• def. product line: a family of products
Family-Oriented Abstraction, Specification, and Translation (FAST)
Applications of FAST

• developed and in use within Lucent Technologies (development: Bell Labs)
• many product lines already created there
Basic Assumptions

- redevelopment hypothesis
- oracle hypothesis
- organizational hypothesis
Stages Towards an Engineered Family

1. potential family
   ○ one suspects sufficient commonality

2. semifamily
   ○ common and variable aspects identified

3. defined family
   ○ semifamily + economic analysis of exploiting it

4. engineered family
   ○ defined family + investment in processes, tools, resources
FAST Strategies

- identify collections of programs that can be considered families
- design the family for producibility
- invest in family-specific tools
- create a family-specific way to model family members
  - for validating the requirements by exploring the behaviour
  - for generating code and documentation
Outputs from Domain and Application Engineering

domain engineer

creates and uses

creates

aplication engineering environment

family definition

domain engineering

application engineering process

aplication engineering

uses

creates

family members

Predicting Change

- is critical
  - but is not all-or-nothing

- confidence should rule size of investment

- FAST: explicitly bounds change (oracle hypothesis)
  - allows for common abstractions

- good guides for future change:
  - past change
  - your marketing organization
  - early adopters
  - experienced developers
Organizational Considerations

- Reorientation of software development around domains may need change in organization of development
  - e.g., one sub-organization for each domain
  - e.g., a product line composed out of several sub-domains
    ▶ Example: protocol stack
Example: FAST Applied to Commands and Reports

- C&R: part of Lucent’s 5ESS telephone switch
- technicians monitor and maintain running switch
  - issue commands
  - receive status reports
- voluminous documentation
- command set: thousands of commands and report types
Example: FAST Applied to Commands and Reports

define C&R family and develop production facility

C&R domain engineering:
define C&R family and develop production facility

C&R application engineering environment:
SPEC language
ASPECT toolset
C&R application engineering process

C&R application engineering:
produce C&R family members

command & report descriptions and customer documentation

Feedback

investment

payback
Defining the C&R Family

- identify potential family members, characterize commonalities and differences
- 5ESS command:
  - always command code followed by parameters
    - command code: action and an object
      - example: report status of a line connected to the switch
  - the particular actions, objects, parameters vary
    - over reasonably well-defined sets
    - certain combinations not included in family
      - example: remove clock is not included, but set clock is included
C&R Commonality Analysis Document

- introduction
  - purpose of the commonality analysis

- overview
  - brief overview of C&R domain

- dictionary
  - defines technical terms for the C&R domain used

- commonalities
  - assumptions true for every member

- variabilities
  - assumptions about how members may vary
• parameters of variation
  ○ the value space for each variability
  ○ the time at which the value must be fixed

• issues
  ○ issues that arose during analysis / how resolved
# Excerpts from Dictionary Section

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command code</strong></td>
<td>Unique identifier of an <em>input command</em>, consisting of a <em>verb</em> and an <em>object</em>.</td>
</tr>
<tr>
<td><strong>Input command</strong></td>
<td>A command entered by an office technician that acts as a stimulus to the 5ESS to perform tasks. Such tasks include changing the state or reporting the state of the 5ESS.</td>
</tr>
<tr>
<td><strong>Input command definition</strong></td>
<td>A specification of all the information needed to identify and produce an <em>input command</em> or a set of <em>input commands</em> with common structure and contents.</td>
</tr>
<tr>
<td><strong>Input command manual page</strong></td>
<td>Documentation of an <em>input command</em> for the customer’s use.</td>
</tr>
<tr>
<td><strong>Output report</strong></td>
<td>An information message that is printed on an output device.</td>
</tr>
<tr>
<td><strong>Output report definition</strong></td>
<td>A specification of all the information needed to identify and produce an <em>output report</em> or a set of <em>output reports</em> with common structure and contents.</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Customer documentation that describes the use of an input command.</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Verb</strong></td>
<td>The name of the action indicated by an input command.</td>
</tr>
</tbody>
</table>
COMMONALITIES

The following are basic assumptions about the domain of input commands, output reports, and customer documentation.

INPUT COMMANDS

C1. Each input command is uniquely determined by its command code. When an input command definition is used to define more than one input command, it defines multiple command codes, all of which share the same set of input parameters.

C2. Each input command is described on exactly one input manual page.

C3. The following administrative data are required in an input command definition: msgid, process, ostype, schedule, and auth. Each input command has exactly one value for each of these fields.

C4. A verb is an alpha-string with a maximum length.

C5. There is a fixed maximum number of input parameters permitted for input commands.
C6. An input parameter description consists of a parameter name and a value specification. The value specification defines the range of values that an office technician may use for the input parameter.

OUTPUT REPORTS

C7. Output reports appear in three different contexts as follows.
   a. Runtime: At runtime an output report may appear on an output device, such as the printer.
   b. Report definition: The set of output reports that a 5ESS switch may produce at runtime, and the meaning of each possible output report, must be defined before building the software for the switch.
   c. Output report documentation: Each output report must be documented for customer use. The documentation of output reports must include all the information that the office technician needs to know to understand the report and determine the reason for its appearance at runtime.


C9. There is a fixed maximum number of characters in a line of an output report.
C10. Each output report is described on exactly one output manual page; however, an output manual page may describe more than one output report.

C11. An output report definition is a sequence of text block definitions.

DOCUMENTATION

C12. An (input command or output report) manual page consists of several fixed sections. It may also reference an appendix.

C13. An (input command or output report) manual page documents one or more input commands or output reports.

SHARED COMMONALITIES

C14. All the information needed to define an input command, the associated solicited output report, and the associated manual pages must be describable as one specification. It must be possible to generate from such a specification all the files and data needed to process input commands and produce output reports at runtime and to generate either (1) the input command and output manual pages or (2) files and data that can be used to generate the input command manual pages and output manual pages.
The following statements describe how input commands, output reports, and customer documentation may vary.

VARIABILITIES

INPUT COMMANDS
V1. The maximum length of a verb, object, parameter name, or enumeration value.
V2. The domain for verbs.
V3. The maximum number of input parameters.
V4. The Csymbol used to designate a msgid.

OUTPUT REPORTS
V5. The maximum number of characters in a line of an output report.
**DOCUMENTATION**

V6. The representation of an *input command* on an *input manual page*, particularly the following in the syntactic template for the *input command*:

a. The separators used between the *command code* and the list of *input parameters*
b. The terminator for the representation of the *input command*
c. The separator used between the *verb* and the *object*

Typical *input command* representations appear as follows:

```plaintext
<command code rep><separator1><input parameter rep><input terminator>
<command code rep><input terminator>
<verb><separator2><object>
```

V7. Typographic distinguishers for command templates.
Sample Command Template, Written in SPEC

COMMAND {
    TEMPLATE {
        abt-task:tlws;
        purpose: "Aborts an active trunk and line workstation (TLWS) maintenance task.";
        warning: "Once this command is entered, the consistency of all hardware states and data in use by the task is questionable.";
    }
}
ABT-TASK:TLWS=a;

Warning: Once this command is entered, the consistency of all hardware states and data in use by the task is questionable.

- **Purpose**
  Aborts an active trunk and line workstation (TLWS) maintenance task.

- **Explanation of Parameters**
  
  a = Task identifier given to active TLWS maintenance tasks by the OP-JOBST command.

- **Responses**
  Only standard system responses apply.
Sample Parameter Definition

COMMAND {
 .........

PARAM tlws {
    TYPE {
        domain: num;
        min: 0;
        max: 15;
        default: 0;
    }
    desc: "Task identifier given to active TLWS maintenance tasks by the OP-JOBST command."
    csymbol: task_id;
}
}
A Parameterized Version of TLWS

PARAM lib_tlws( x ) {
    TYPE {
        domain: num;
        min: 0;
        max: 15;
        default: 0;
    }
    desc: "Task identifier given to active TLWS maintenance tasks by the OP-JOBST command."
    csymbol: x;
}
Reuse of TLWS

COMMAND {
    TEMPLATE {
        abt-task:tlws;
        purpose: "Aborts an active trunk and line workstation (TLWS) maintenance task.";
        warning: "Once this command is entered, the consistency of all hardware states and data in use by the task is questionable.";
    }

    PARAM tlws use lib_tlws( task_id )
}

Producing Multiple Documentation Formats

SPEC source → ASPECT → TROFF, HTML, SGML, Postscript, text preview
Designing the Translators

• existing parser generator tools used
• principles of software family development applied
• combined with SCR design process
• minimal toolset:
  ○ command translator
  ○ report translator
  ○ command documentation generator
  ○ report documentation generator

• much overlap between translators expected (commonalities)
Using the SCR Design Process

• information hiding hierarchy
  ○ module guide
  ○ uses relation

• ASPECT:
  ○ external interface module
    ▶ . . .
  ○ behaviour hiding module
    ▶ . . .
  ○ software decisions module
    ▶ . . .

• result: substantial code reuse
ASPECT External Interface Module

- **output drivers module**
  - command format module
  - report format module
  - documentation format module

- **library reference module**

- **device drivers module**
  - text module
  - HTML module
  - formatter macros (TROFF) module
  - Postscript module
  - SGML module
ASPECT Behaviour Hiding Module

- tool builder module
- input command traversal module
- output report traversal module
- command documentation traversal module
- report documentation traversal module
- shared services module
ASPECT Software Decisions Module

- cross reference module
- database module
- domain translator module
- error recorder module
- global context module
- preprocessors module
  - alter structure module
  - alter syntax module
  - random access module
• semantic verification module
  ◦ completeness module
  ◦ consistency module
  ◦ placement module

• specification expander module

• symbol reference module

• text function module

• text translation module

• global language data module

• system interface module

• transversal module
The Economics of FAST

Modelling the FAST Process

- there is a precise model for the FAST process
  - see [WeLa99]

- description of process models: PASTA approach (Process and Artifact State Transition Abstraction)
  - see [WeLa99]
Finding Domains where FAST is Worth Applying

- usually apply to legacy systems
- look for domain with
  - frequent, continuing change
  - change at high cost
  - predictable change
  - (quick change needed)
- do an informal or formal economic analysis
Applying FAST Incrementally

• early activities of FAST:
  better understanding of market, customers, requirements
  ◦ facilitates communication, staff training, member design
  ◦ modest cost

• later activities of FAST:
  make effective use of information and understanding

• apply FAST iteratively, e.g.:
  1. commonality analysis only, to make design more flexible
  2. introduce a rudimentary language
     to generate data structures changing most often
  3. expand language to generate majority of code
Transitioning to a FAST Process

• FAST process allows for gradual introduction into company
• early: staff learns to think in terms of families
  ○ test: can they predict future changes?
• later: use this thinking
• one way to start:
  ○ pick a few, high-leverage, well-understood domains
  ○ apply a simple version of FAST
  ○ several iterations
  ○ if you understand it and if it works,
    spread to more domains and more parts of company
• you might have to reengineer your organization later