Program Families

OSE

OSE

OE

OSE
We were behind schedule and wanted to deliver an early release with only a proper subset of intended capabilities, but found that subset would not work until everything worked. We wanted to add simple capability, but to do so would have meant rewriting all or most of the current code. Therefore, we should take advantage of this simplification by removing the unneeded capability, but to simplify and speed up the system we would have had to rewrite major sections of the code. [5]
Software as Product Family

...understanding a program as a single product...

is a practice from the perspective of marketing development.

Single-product thinking goes back to the "Stone Age" of programming.

-- don't reinvent wheels, waste man power, and extend the time to market.

Provide an infrastructure of reusable (software) components.

-- aims at looking for already existing solutions.

Product-line development helps to let do even marketing a better job.

Good

Bad
what was (is) true for „hardware” over a long period is also true for software

software-manufacturing (e.g., operating systems) is an engineering discipline
a program family is a software-product line with programs being the products: ●

- M-lenses are upward compatible from the M3 to the M7 (i.e., 1954 – today)
- SIPS and WHIPS is available for the V/S 40, 60, 70, and 80 models resp.
- all products of the same line share the same (sub-) set of properties: ●

- the Leica-M system with its M3, M2, M1, M4, M5, M6, and M7 cameras
- the Volvo 40, 60, 70, and 80, with its V and S models and variations
- a program line is made of a number of individual products:

Product Line  ⇝  Program Family
We consider a set of programs to be a program family if they have so much in common that it pays to study their common aspects before looking at the aspects that differentiate them. [5]

We want to exploit the commonalities, share code, and reduce maintenance costs. [5]
For a function to be reusable, it needs to be “componentized”

- Remind the experiences made over time and reflect the lessons learned.
- Differentiate functional requirements from non-functional requirements.
- Design a complete and, yet, easy to employ function interface.
- Hide the function's implementation details as far as possible.
- Concentrate on the function's essentials.
- Provide a full specification and extensive documentation of the design.
- Free a function's implementation from any non-functional property.

Particularly, don't hard-code assumptions about the function’s use pattern.

Separation of Concerns

for a function to be reusable, it needs to be “componentized”

Reuse

Function

Separation of Concerns [2]
Some users may require only a subset of the services or features of the services. These "less demanding" users may demand that they are not be forced to pay for the resources consumed by the unneeded features. Therefore, "less demanding" users may need that they are not be forced to pay for the resources consumed by the unneeded features. Some users may require only a subset of the services or features that other users need. Therefore, "less demanding" users may demand that they are not be forced to pay for the resources consumed by the unneeded features. However, this may not be what you demand — WYDIVYG.
Family-Oriented Design

- Family-oriented design is a top-down approach, developed stepwise.
- It encapsulates strategic and application-specific design decisions.
- It provides mechanisms in the form of fundamental building blocks.
- It captures common functions that are useful to build specialized systems.
- A minimal subset of system functions is made of a distinct (i.e., small) number of reusable assets.
- It allows reuse and specialization or customization of fundamental system functions.
- Minimal system extensions are developed bottom-up, directed in a top-down manner.

Operating System Engineering | Program Families
program families tend to exhibit a distinguished, deep hierarchical structure.

Incremental System Design

...
Example of a Family of Operating Systems — FAMOS [3]
Rather than write programs that transform input to output data, we design software machine extensions that will be useful in writing many such programs. [5]
Summary

A major concern is what ideas to exclude from the design. Simple things nearly always work, and simple thinks are extensible. The challenge lies in doing it ‘right’, and ‘right’ often means staying simple. Keeping things as simple as possible surprisingly well.

[Liskov, 1981]

A major concern is what ideas to exclude from the design.

[Birrell, 1986]

Simple systems work surprisingly well.

[Svobodova, 1985]

Simple things nearly always work, and simple thinks are extensible.

[Lampson, 1983]

Keeping things as simple as possible.

[Needham, 1986]

Operating-System Engineering — Program Families


