DSM TP 2017

8th International Summer School
on Domain-Specific Modeling
Theory and Practice

Montreal, Canada
10-14 July 2017

Variability / Product Families

Kacper Bąk

Currently: The MathWorks, Inc.
Previously: GSD Lab, University of Waterloo

Acknowledgments

slides based on tutorials by

Andrzej Wąsowski, IT University of Copenhagen

Michał Antkiewicz, University of Waterloo

Krzysztof Czarnecki, University of Waterloo

Software-intensive Products Come in Many Variants

Domain Engineering

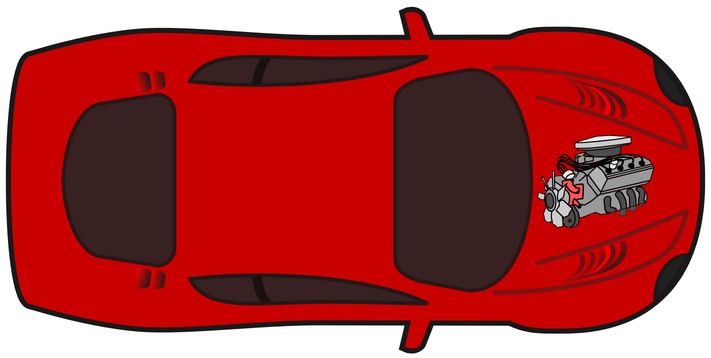
aka Product Line Engineering

aka Product Family Engineering

Application Engineering

Development *with* Reuse

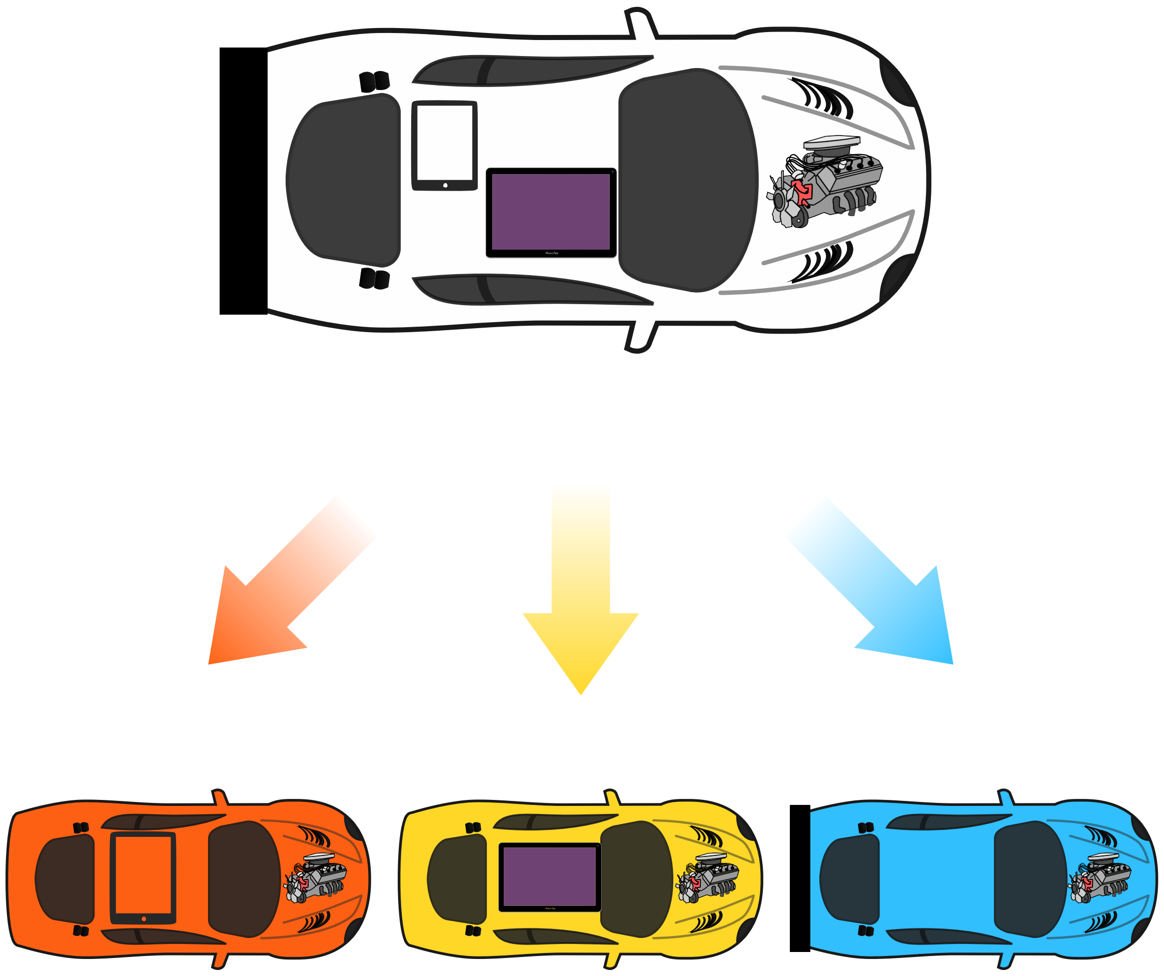
Single Product



Domain Engineering

Development *for* Reuse

Product Family



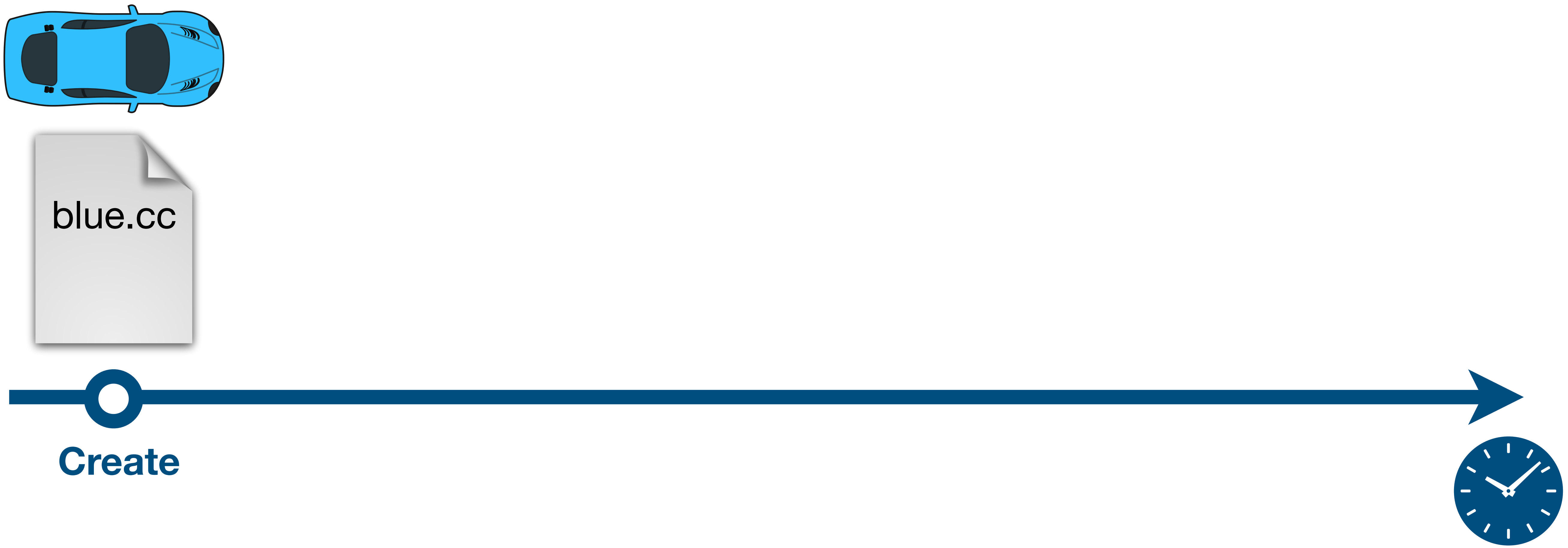
Why Domain Engineering?

Why Domain Engineering?

…because

opportunistic reuse *does not scale*

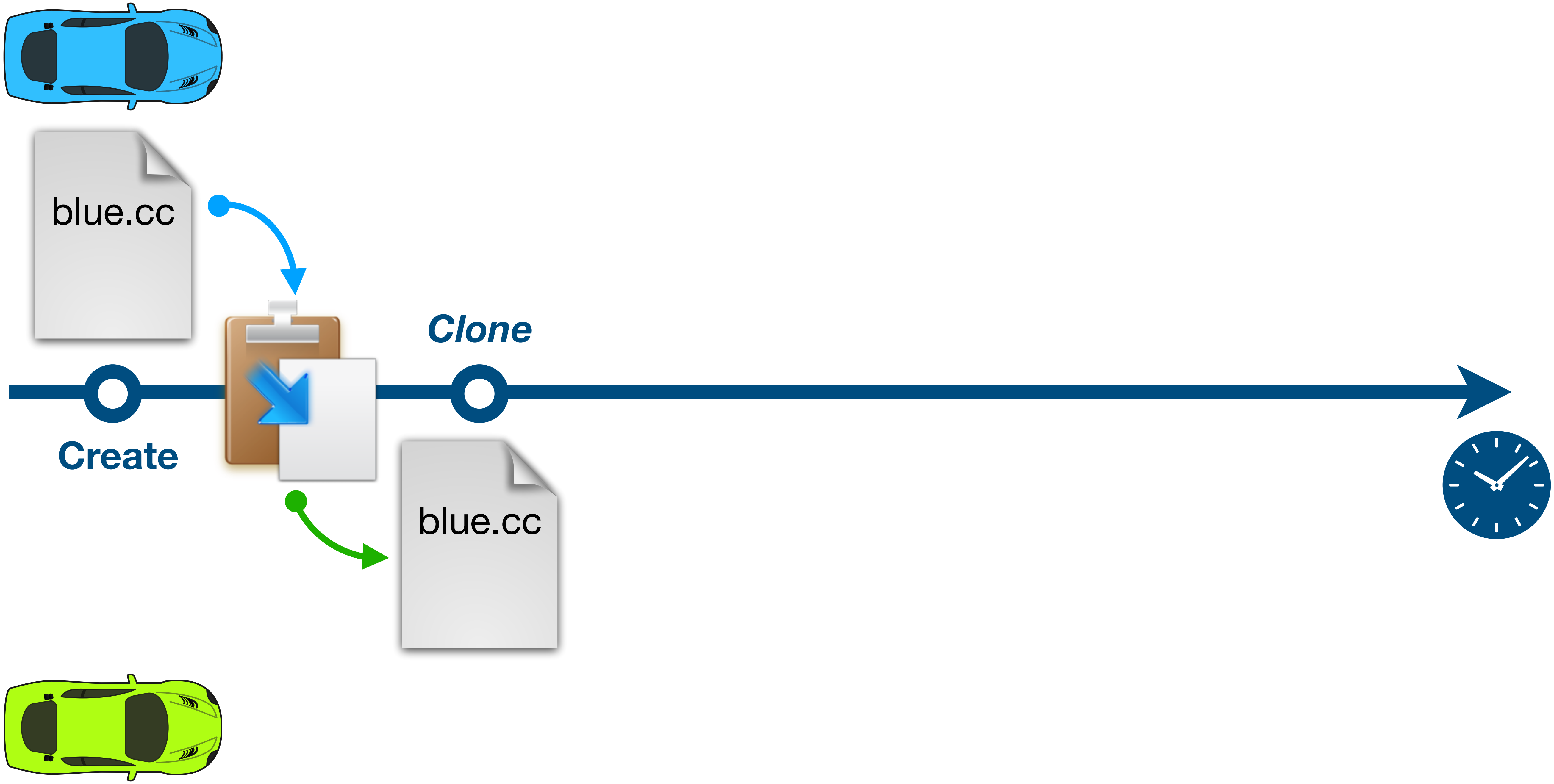
Clone and Own



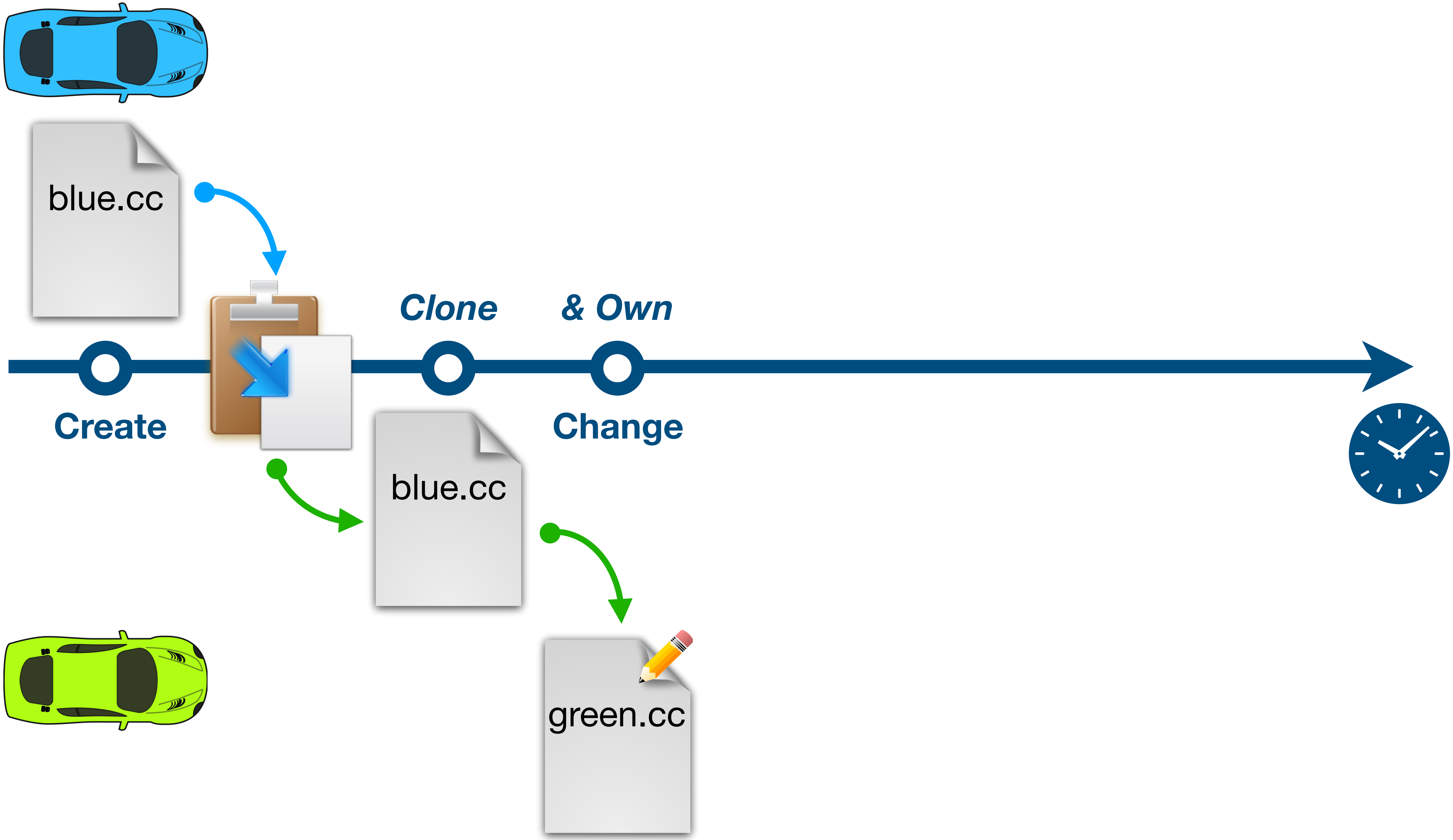
blue.cc

Create

Clone and Own



Clone and Own



blue.cc

Clone

& Own

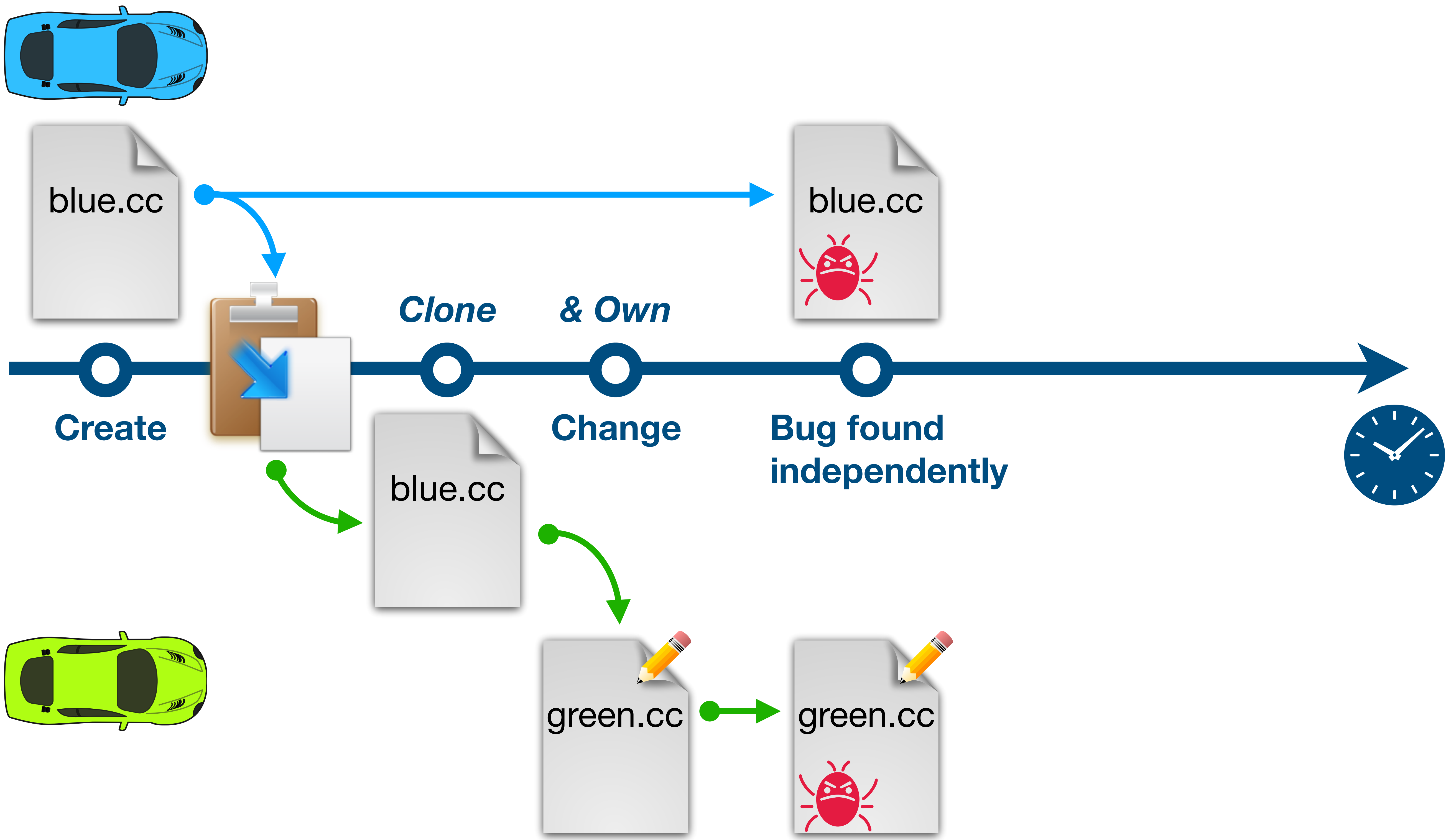
Create

Change

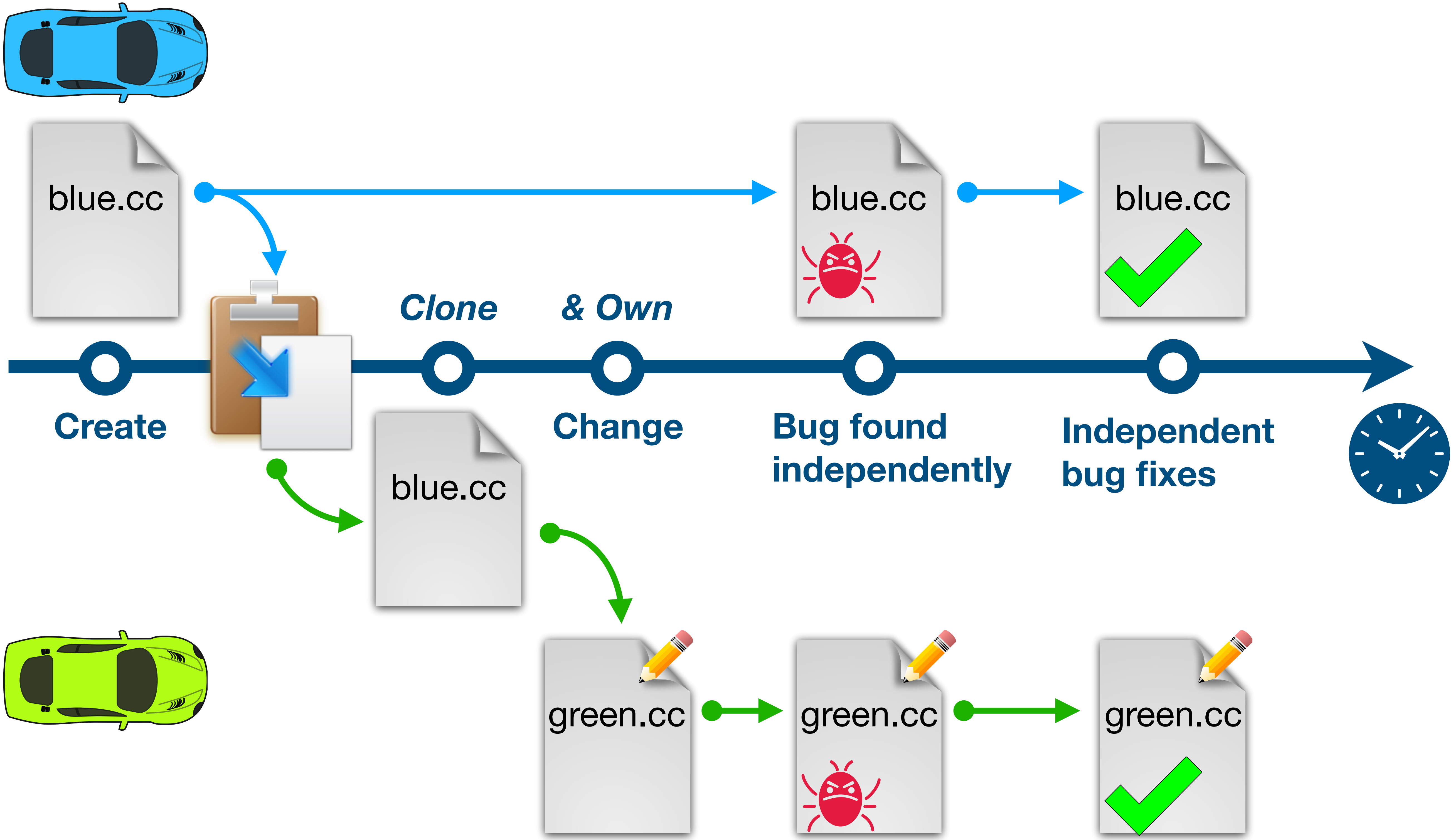
blue.cc

green.cc

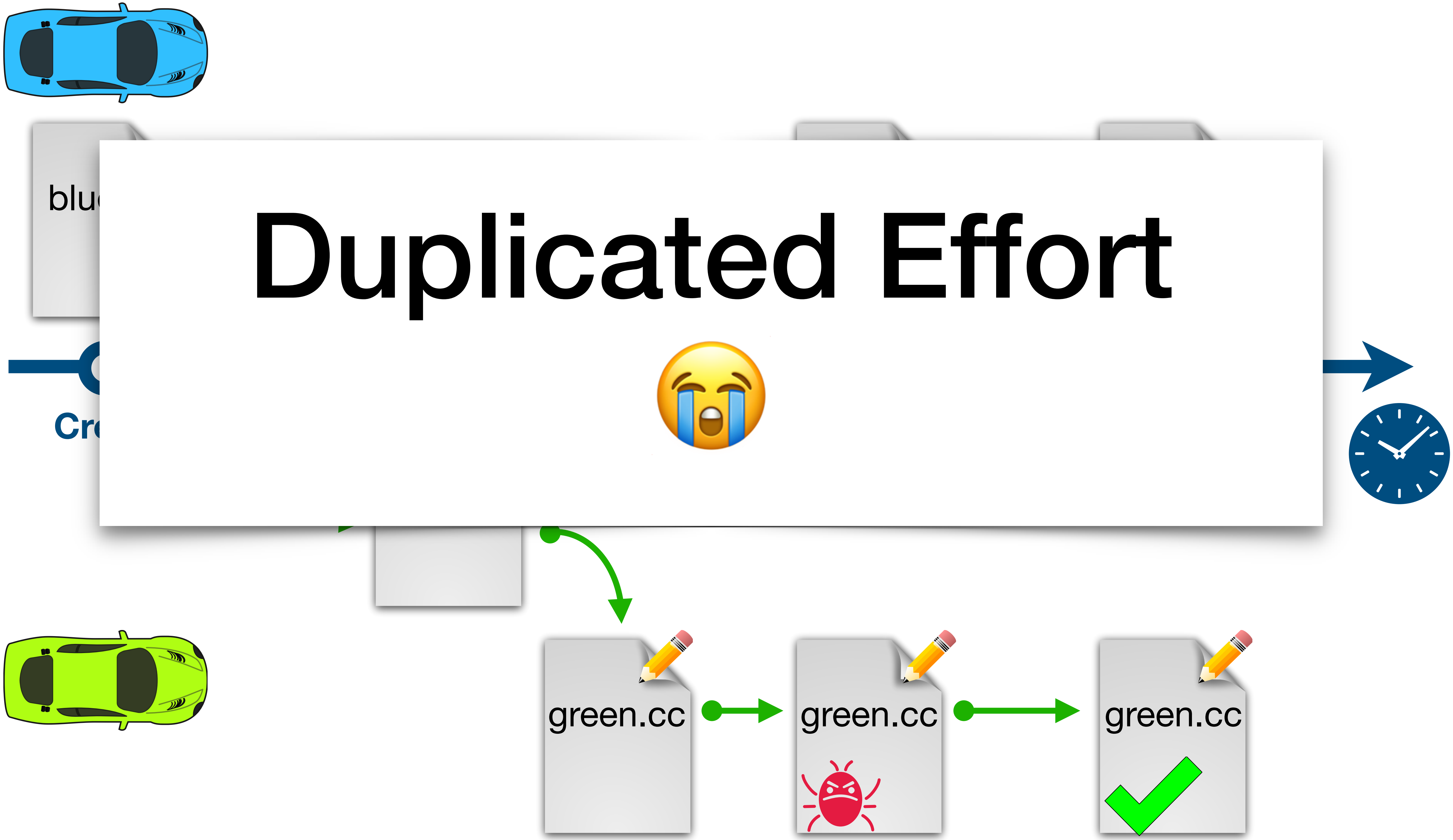
Clone and Own



Clone and Own



Clone and Own



Duplicated Effort

😭

Cloning as Opportunistic Reuse

Dubinsky et al., Exploratory Study of Cloning in Industrial SPLs, CSMR 2013

+ **Easy**, no special tooling required

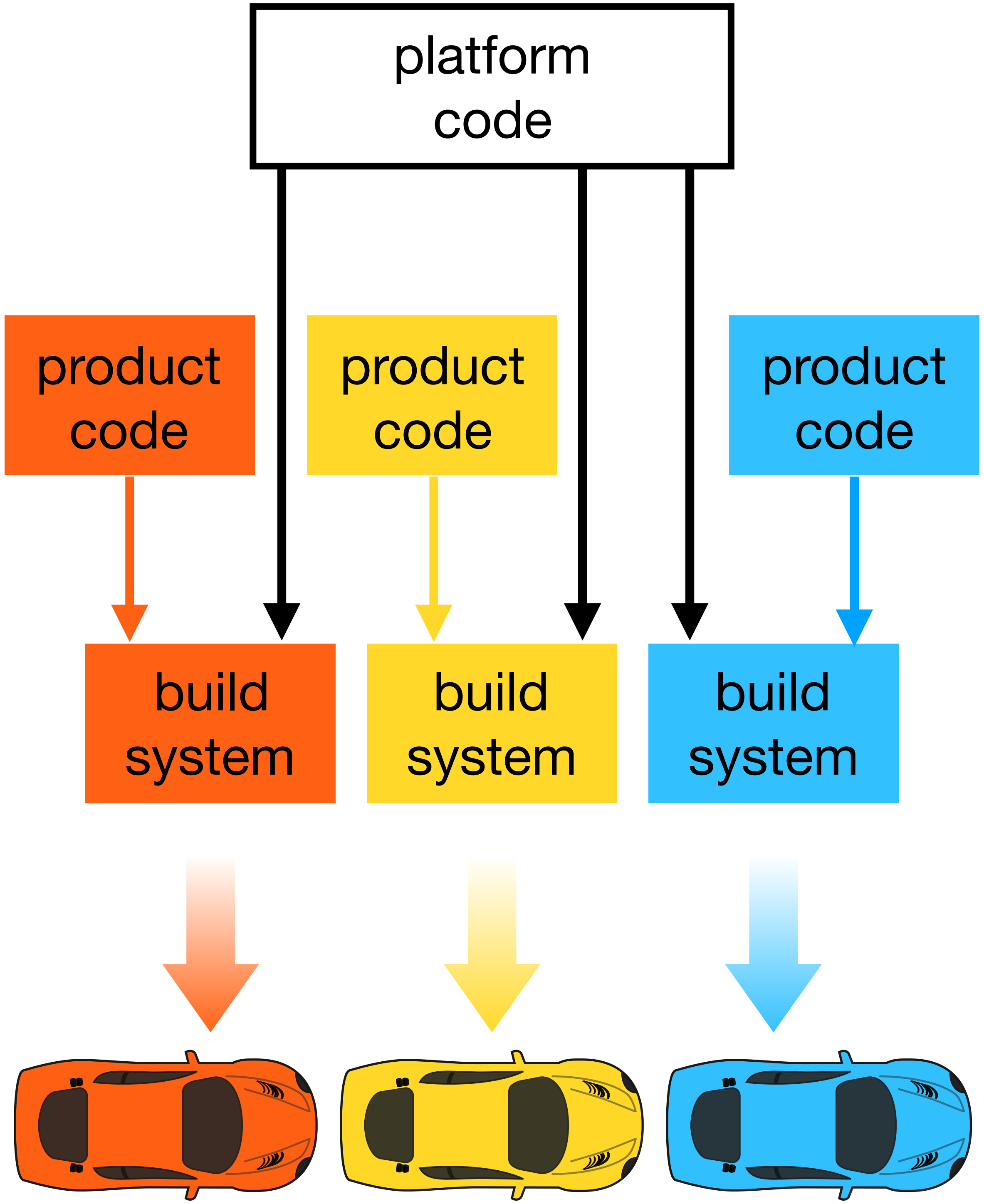
+ **Quickly** available functionality

- **No** sharing (fixes & features)

- Maintain **yourself** (test, debug, change)

- **Product specific** code grows

- Platform code **diminishes and degrades**



Successful Reuse

Proactive

Planned

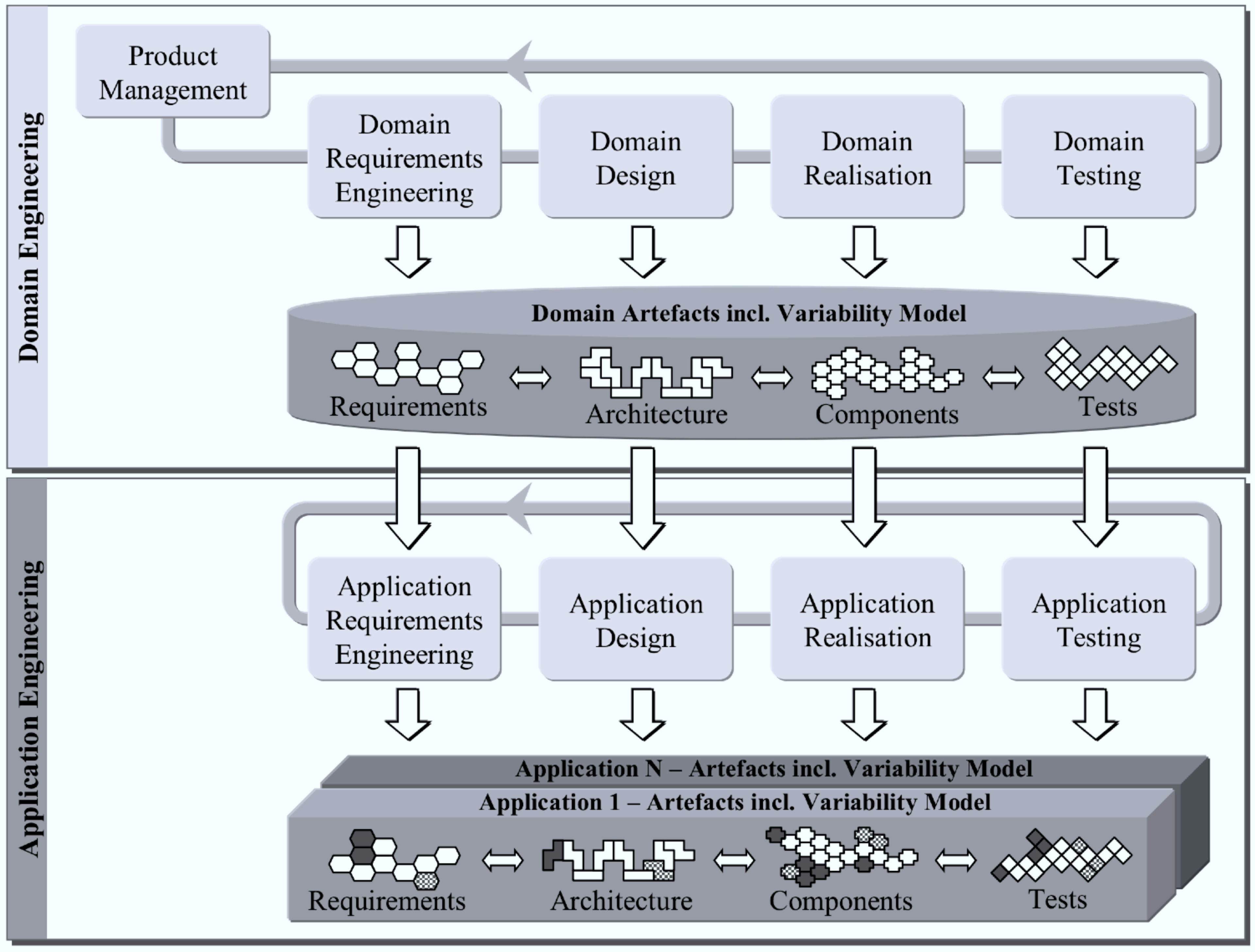
Managed

Domain vs Application Engineering

Pohl et al., Software Product Line Engineering, Springer Science, 2005

product independent

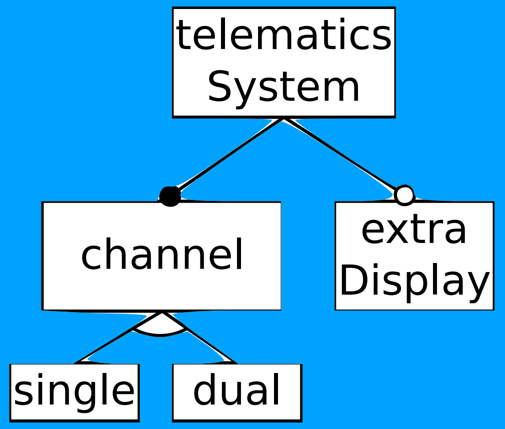
product specific



features

Problem Space

domain-specific abstractions

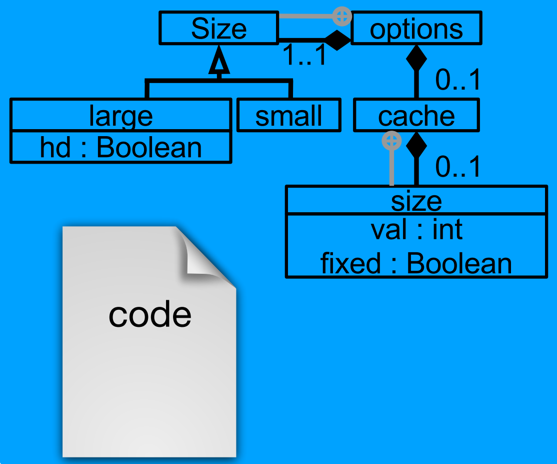


Mapping

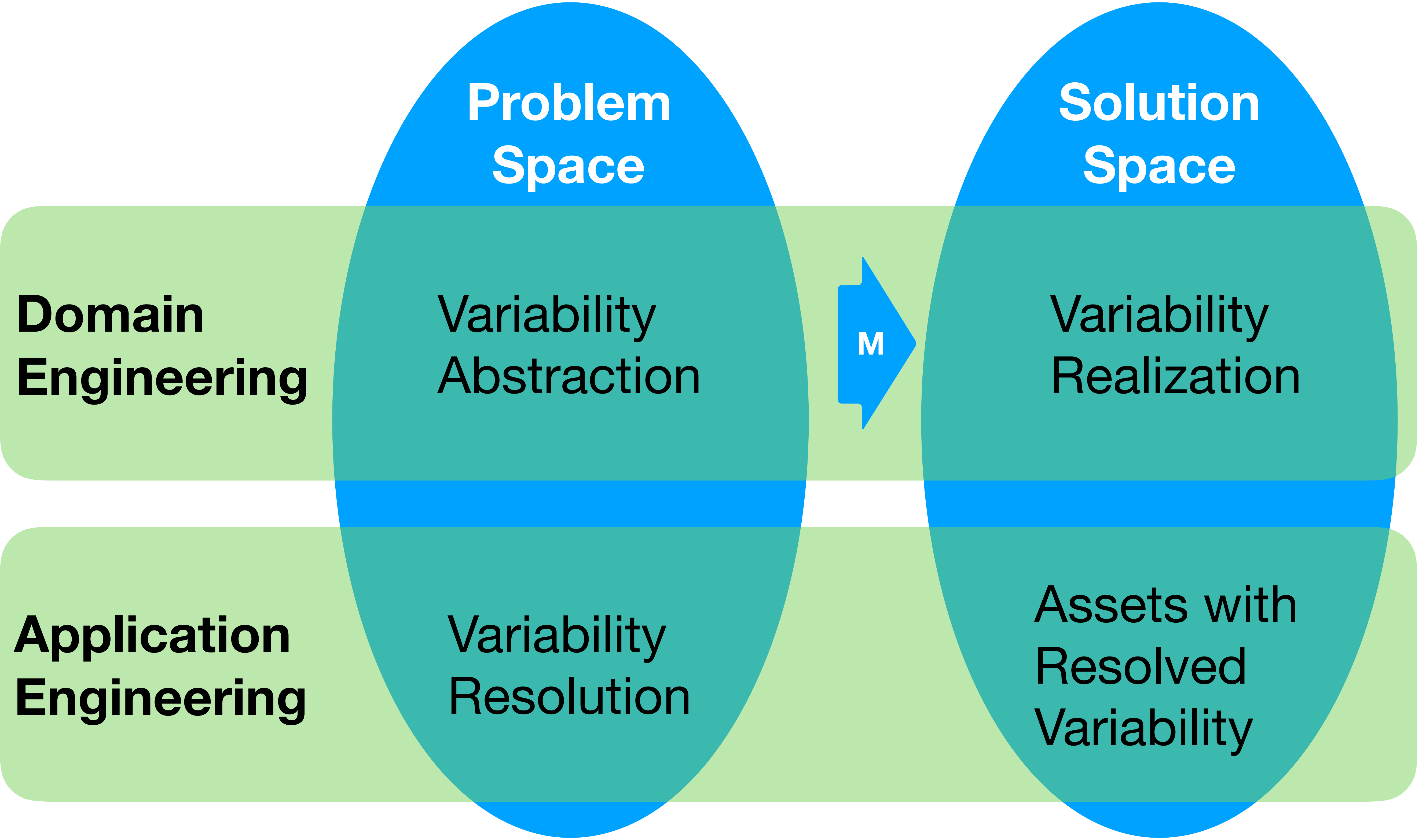
variation points

Solution Space

implementation oriented abstractions

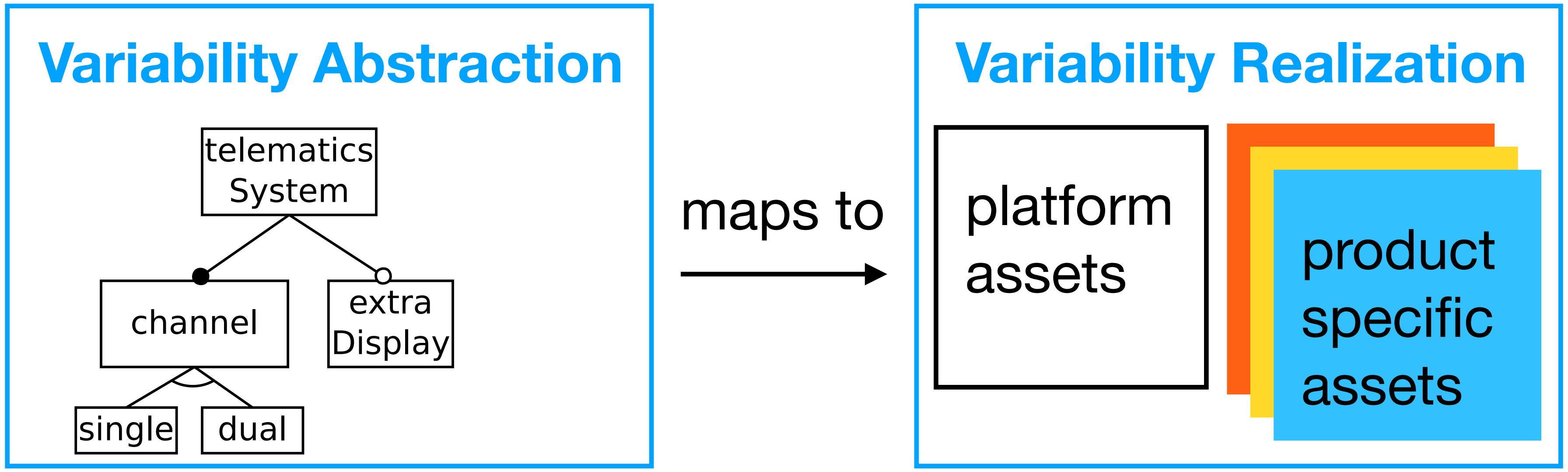


Software Product Line



	Problem Space	Solution Space
Domain Engineering	Variability Abstraction	Variability Realization
Application Engineering	Variability Resolution	Assets with Resolved Variability

Product Line Architecture Overview



Variability Abstraction

telematics System

channel

extra Display

single

dual

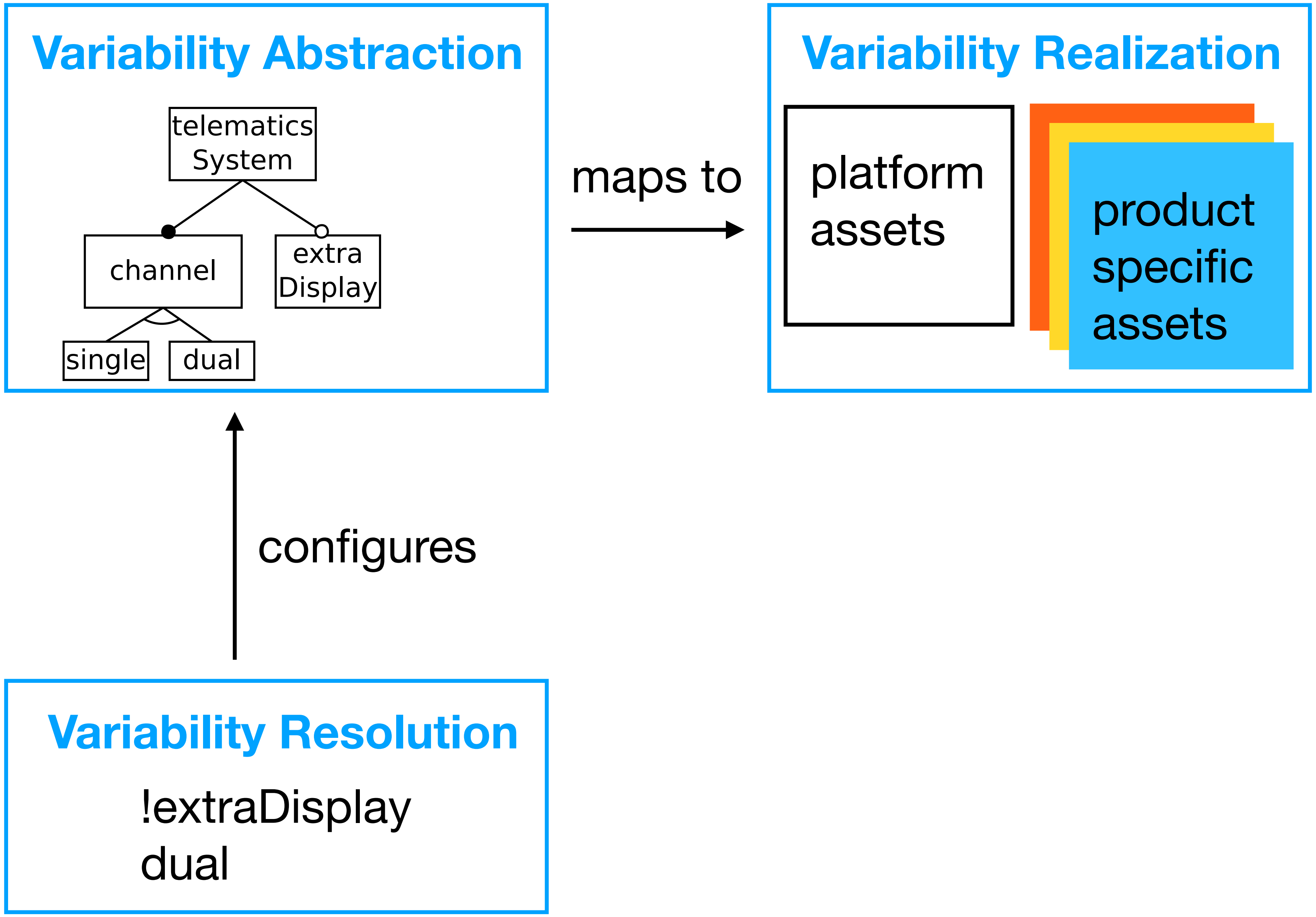
maps to

Variability Realization

platform assets

product specific assets

Product Line Architecture Overview



Variability Abstraction

maps to

Variability Realization

platform assets

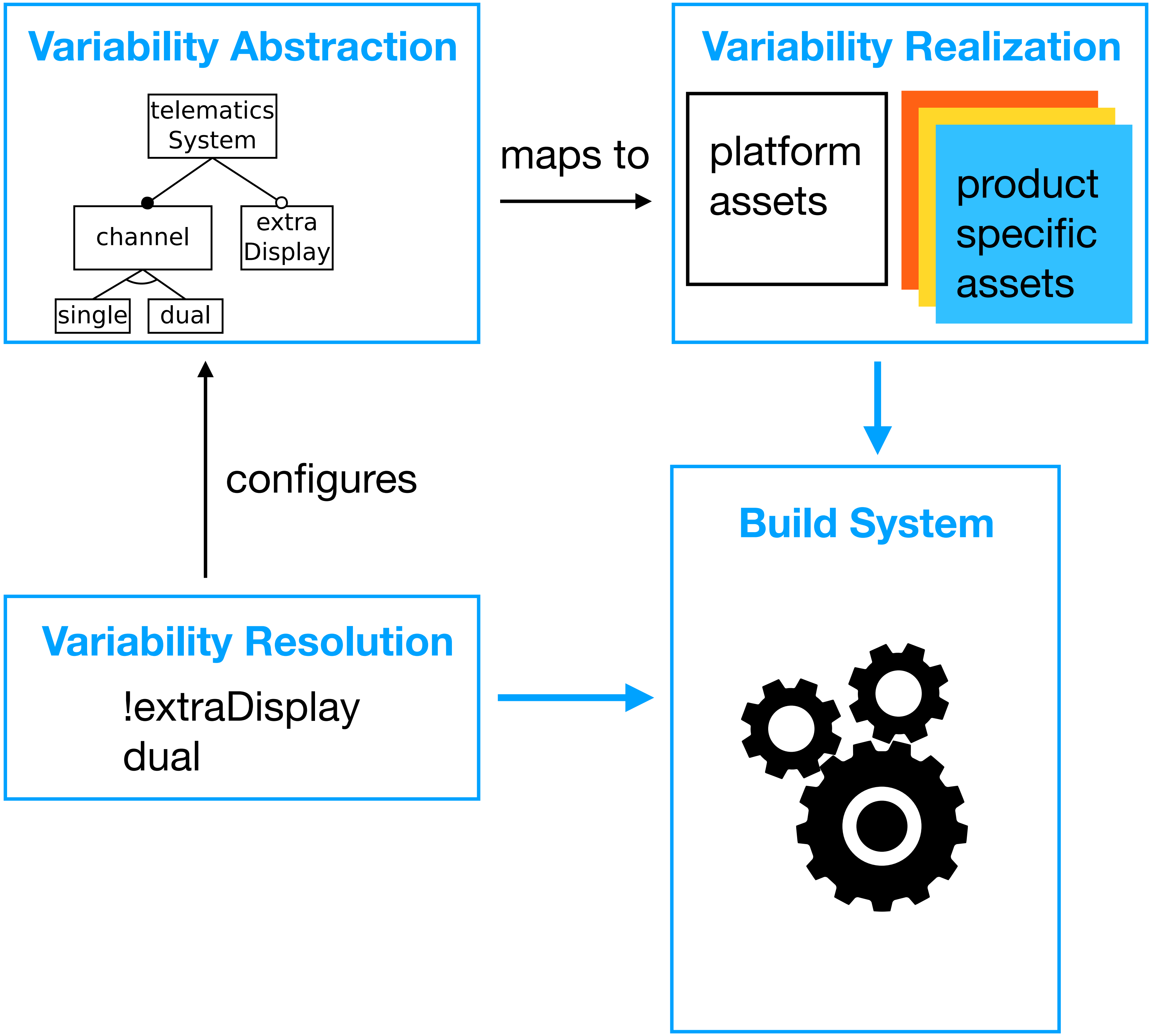
product specific assets

configures

Variability Resolution

!extraDisplay
dual

Product Line Architecture Overview



Variability Abstraction

maps to

Variability Realization

platform assets

product specific assets

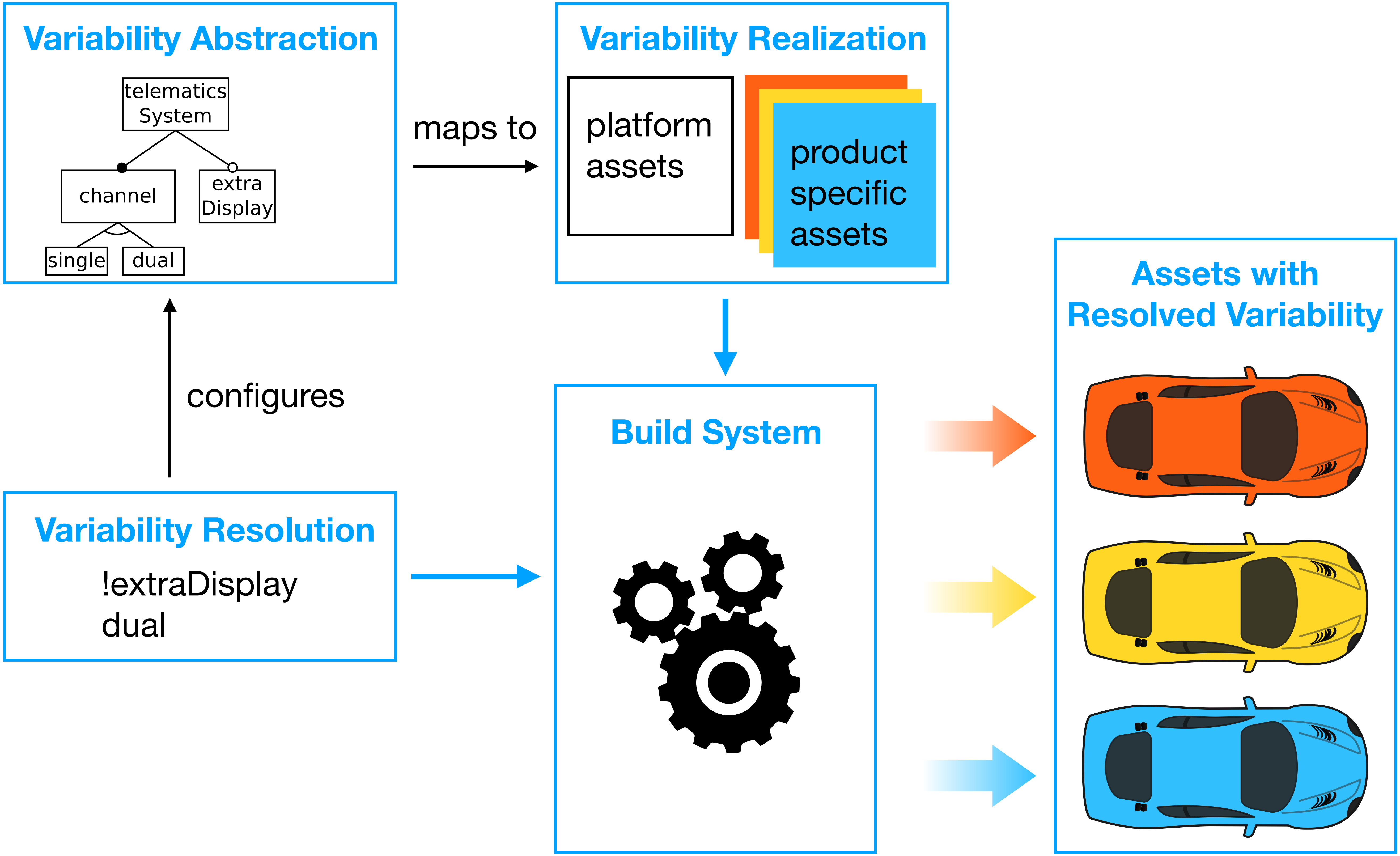
configures

Variability Resolution

!extraDisplay
dual

Build System

Product Line Architecture Overview



Variability Abstraction

telematics System

channel

extra Display

single

dual

maps to

Variability Realization

platform assets

product specific assets

configures

Variability Resolution

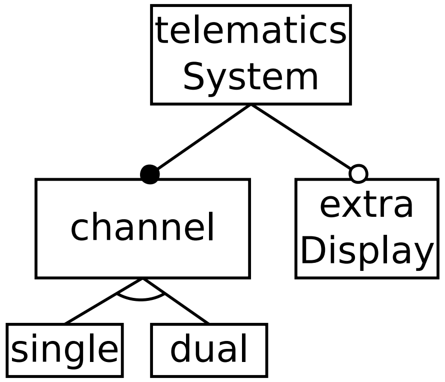
!extraDisplay
dual

Build System

Assets with Resolved Variability

Implementation Technologies

Variability Abstraction



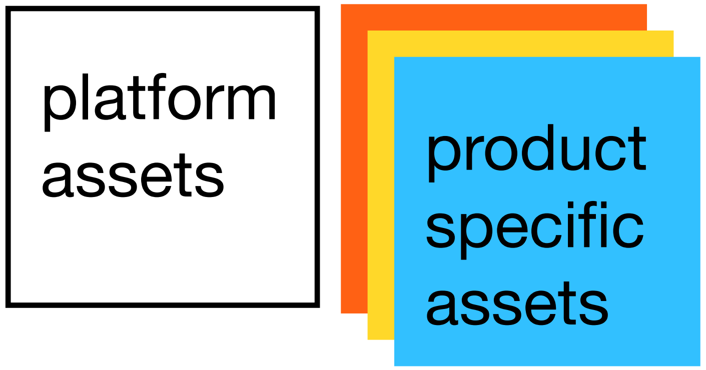
- **Feature models**
- Domain Specific Languages
- *none*

Variability Resolution

!extraDisplay
dual

- **Feature model configuration, constraints**
- Domain specific model
- XML, JSON, custom text format, …

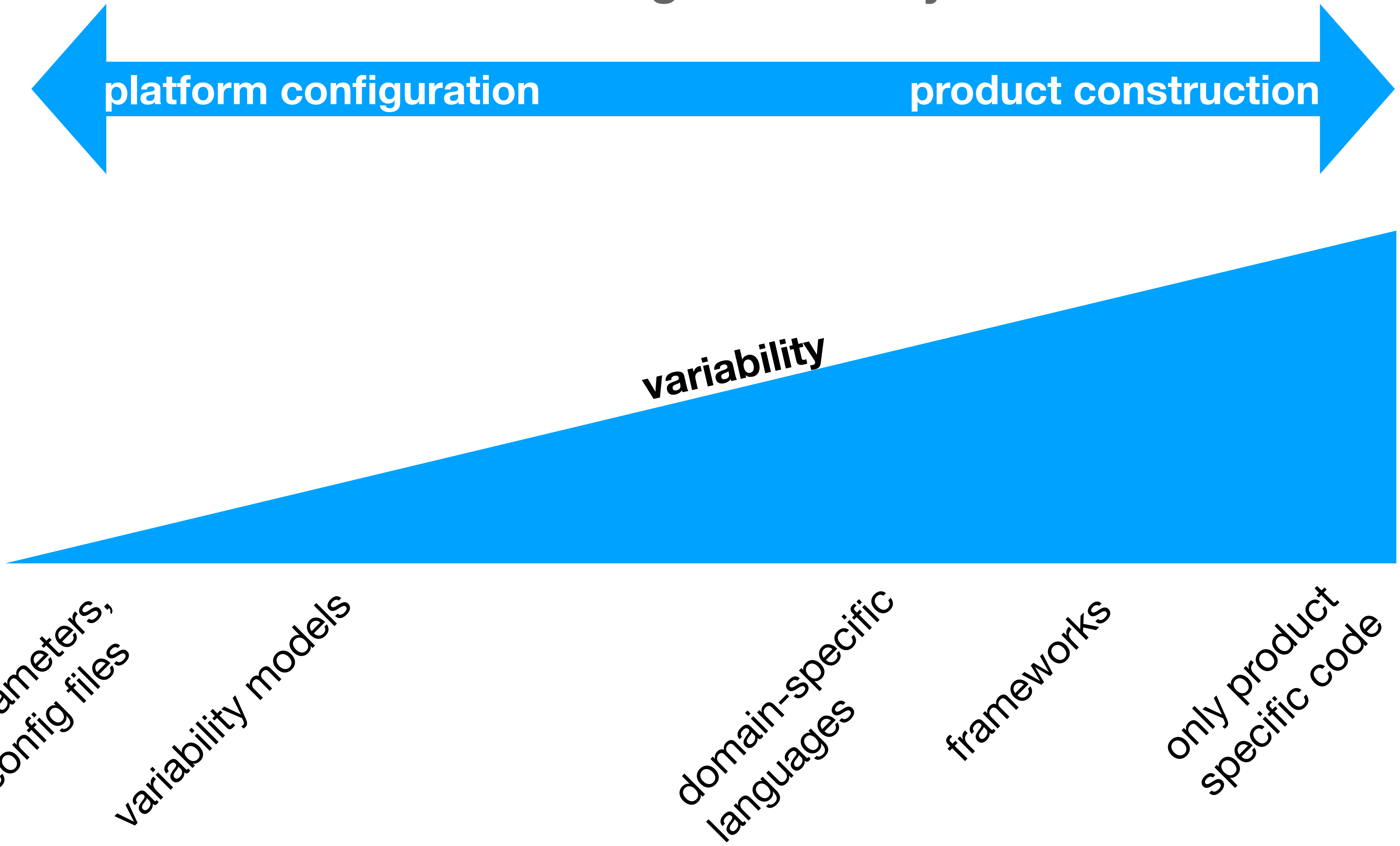
Variability Realization



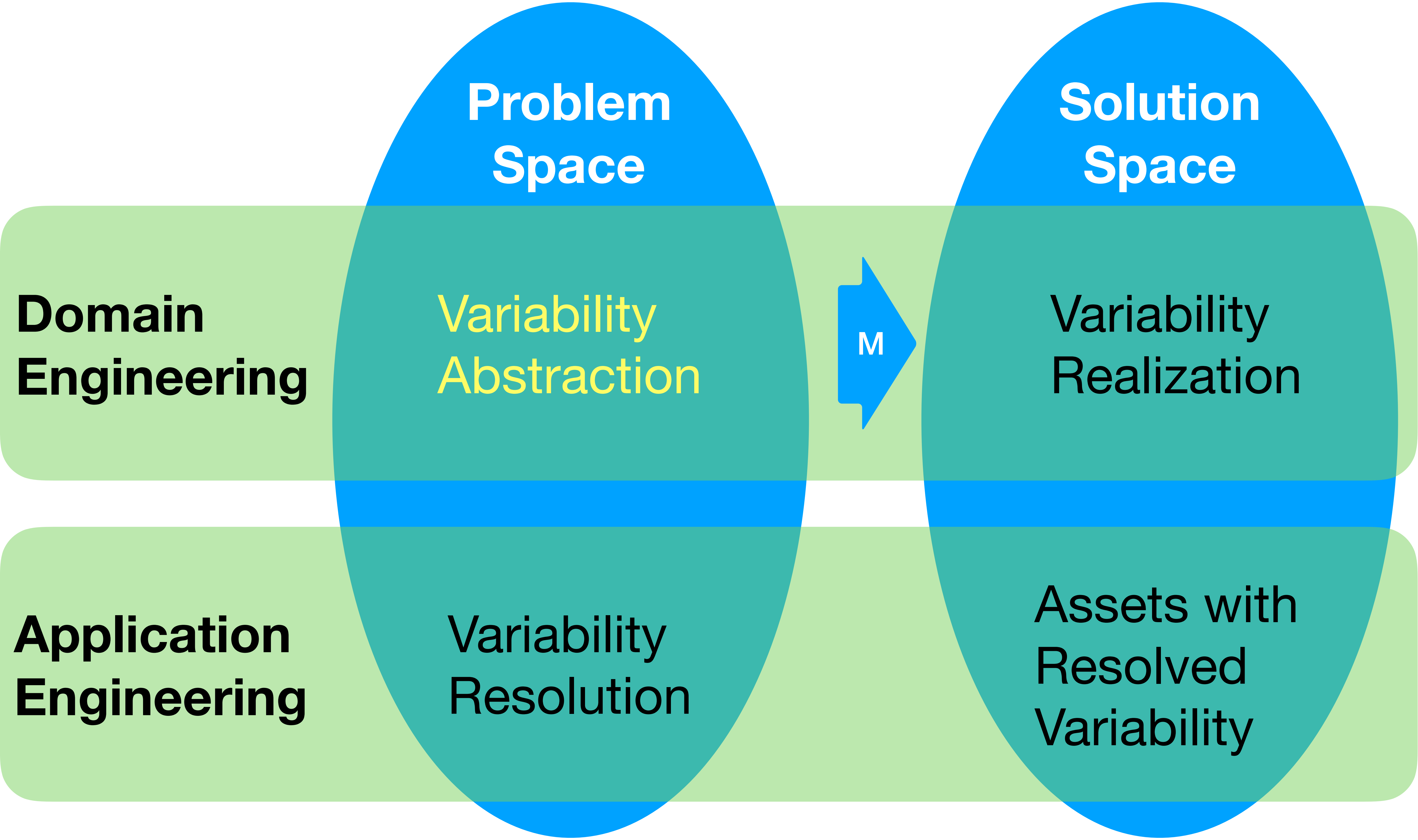
- Code (with variability techniques)
- Code generators
- Model transformers
- Parts may use DSLs

Spectrum of Variability Architectures

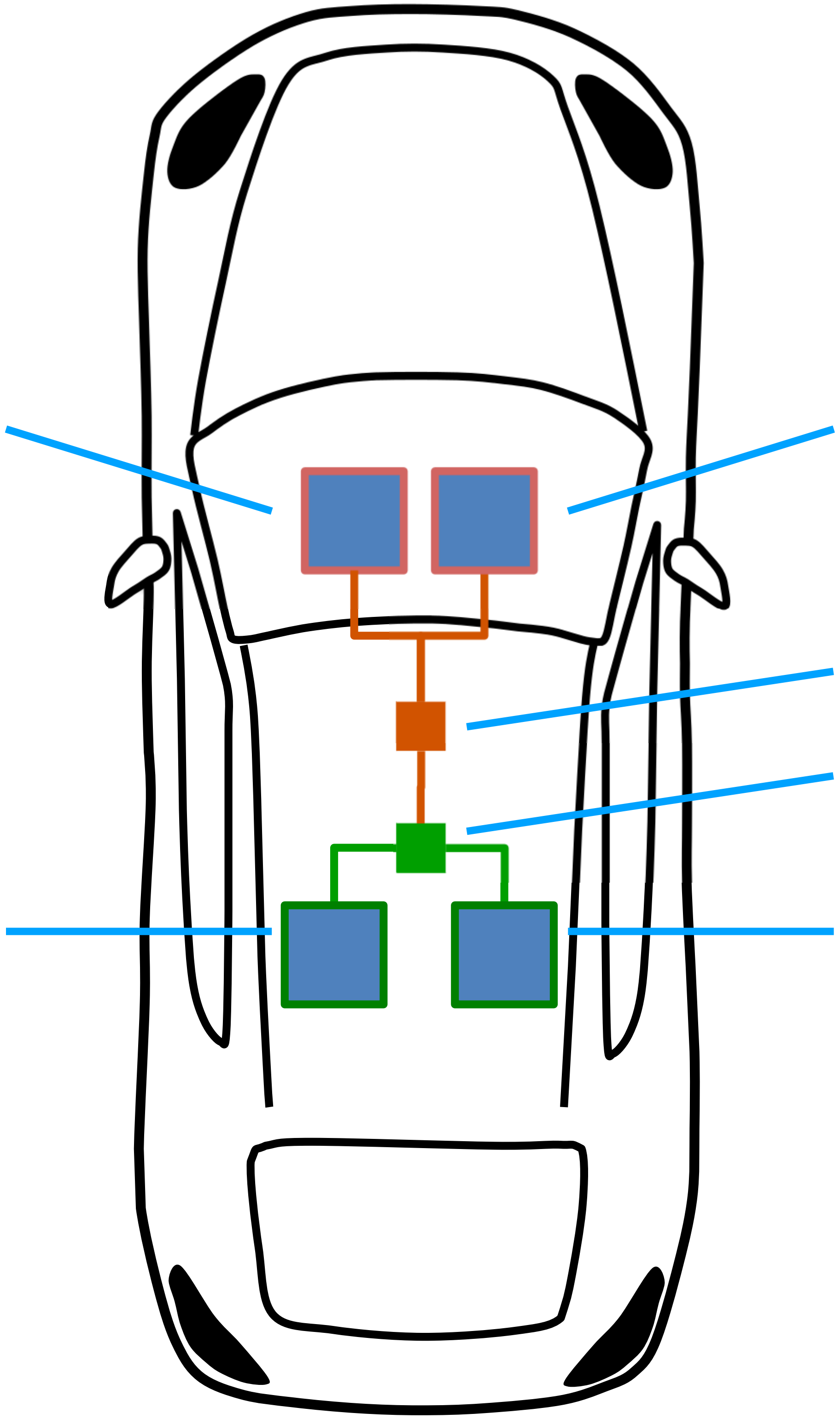
Stay as Close to the Left as Possible
Exploit Commonality
Manage Variability



Software Product Line



Example: Telematics System



Primary display (front)

Extra display (front)

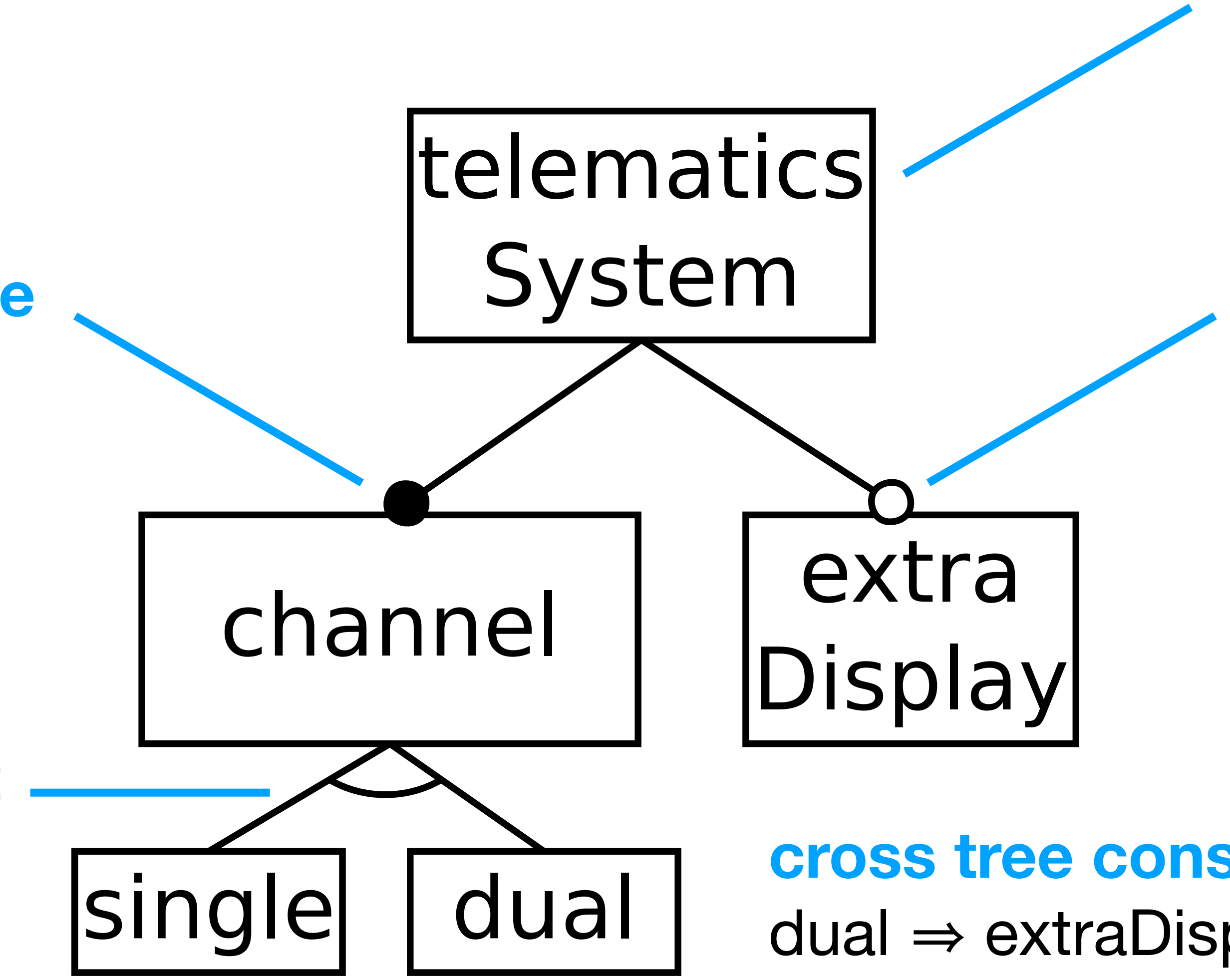
ECU (front channel)

ECU (rear channel)

Primary display (rear)

Extra display (rear)

Feature Model



features form a hierarchy

mandatory feature

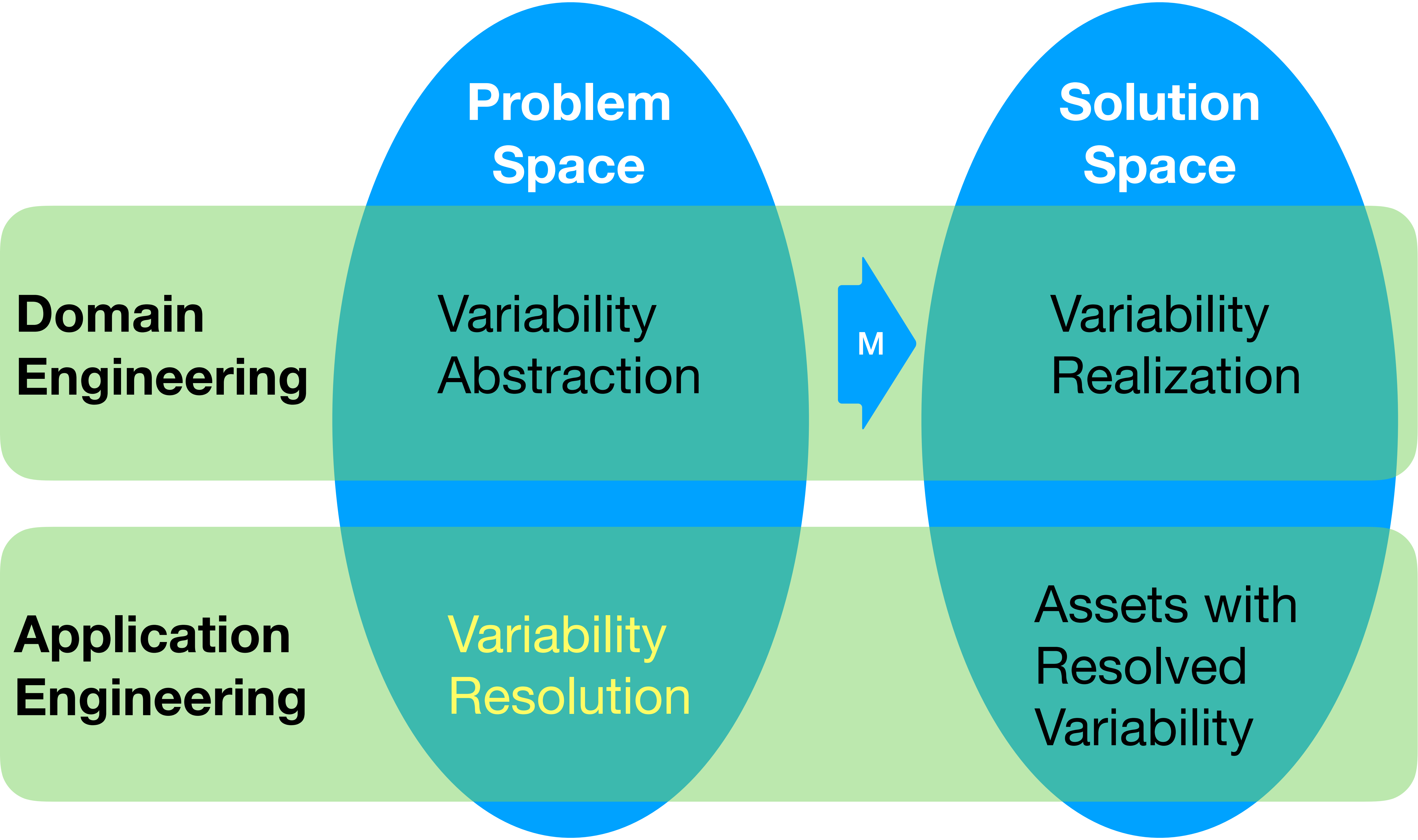
optional feature

group constraint (xor)

cross tree constraint, e.g.,
dual ⇒ extraDisplay

single kind of relationship: *subfeature*
meaning: *implication*

Software Product Line



Problem Space

Solution Space

Domain Engineering

Variability Abstraction

M

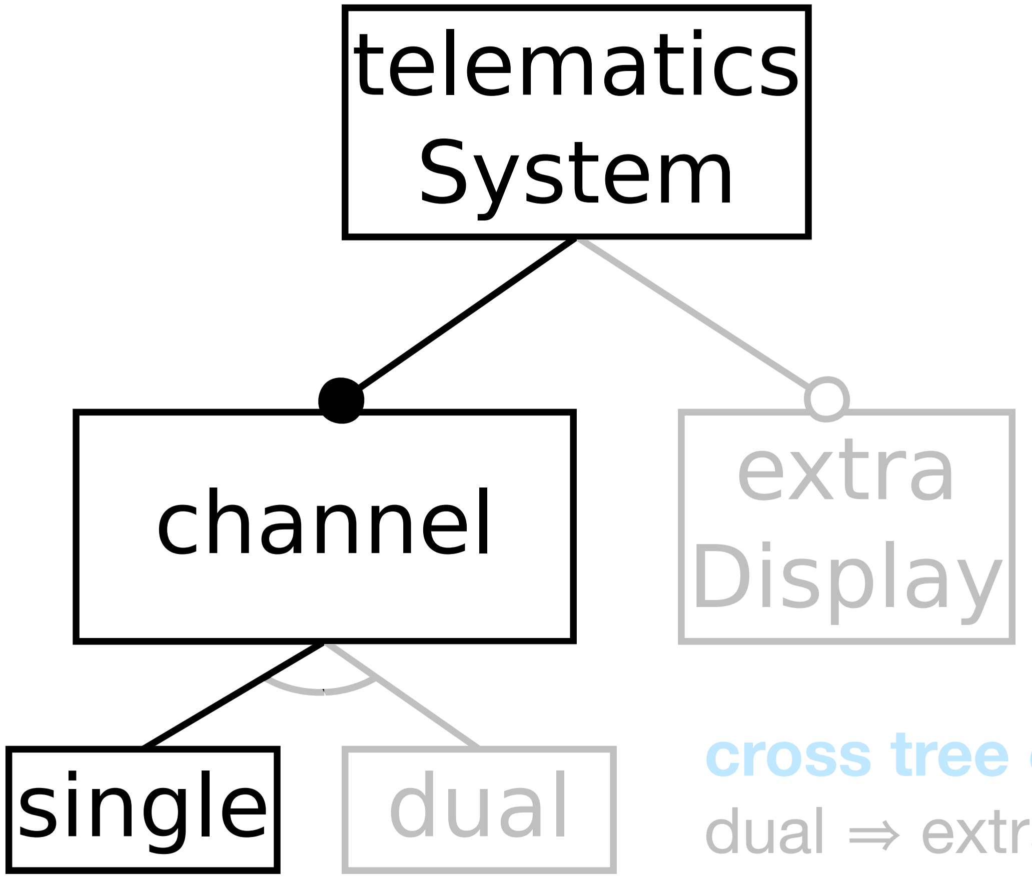
Variability Realization

Application Engineering

Variability Resolution

Assets with Resolved Variability

Feature Configuration



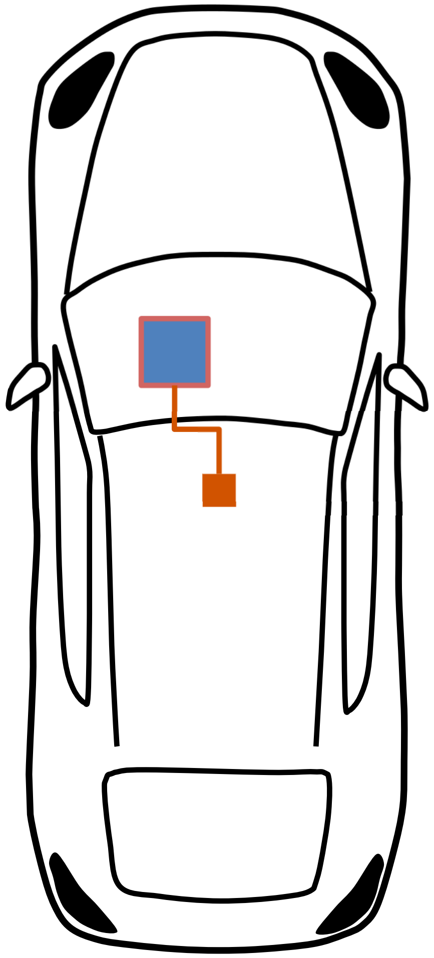
cross tree constraint, e.g.,
dual ⇒ extraDisplay

{ telematicsSystem,
channel, **single** }

Configuration Semantics of an FM

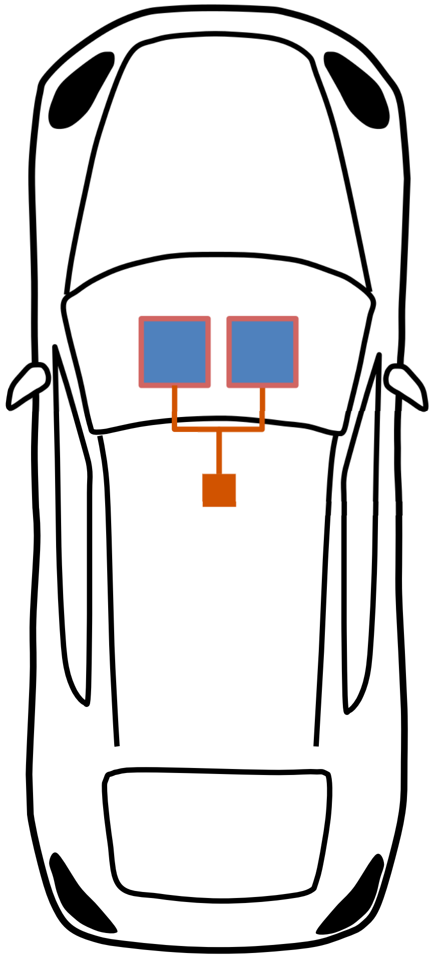
Set of Configurations

✓



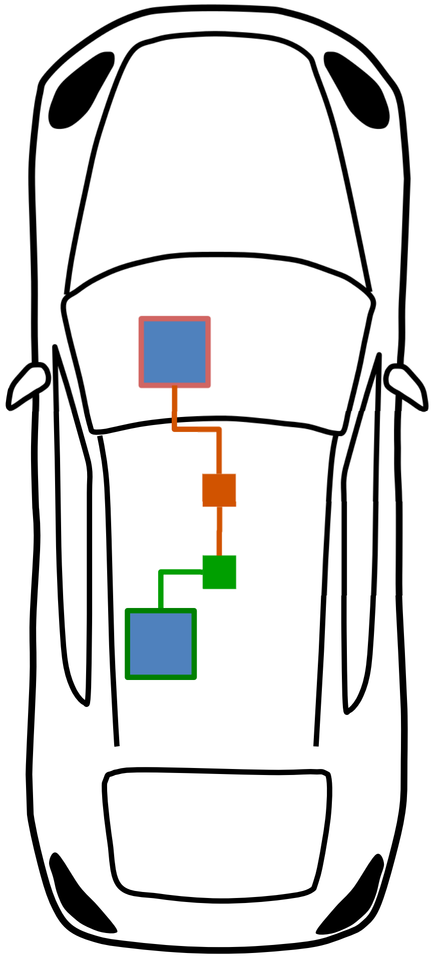
{ telematicsSystem,
channel, **single** }

✓



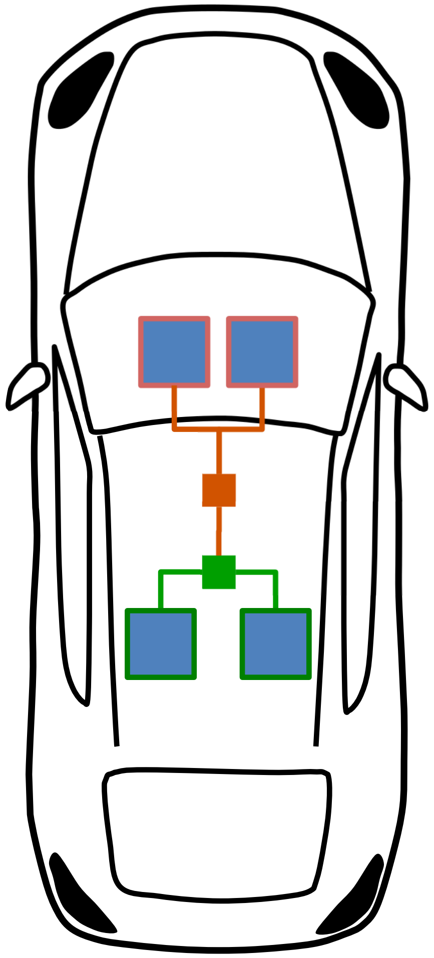
{ telematicsSystem,
channel, **single**,
extraDisplay }

✗



{ telematicsSystem,
channel, **dual** }

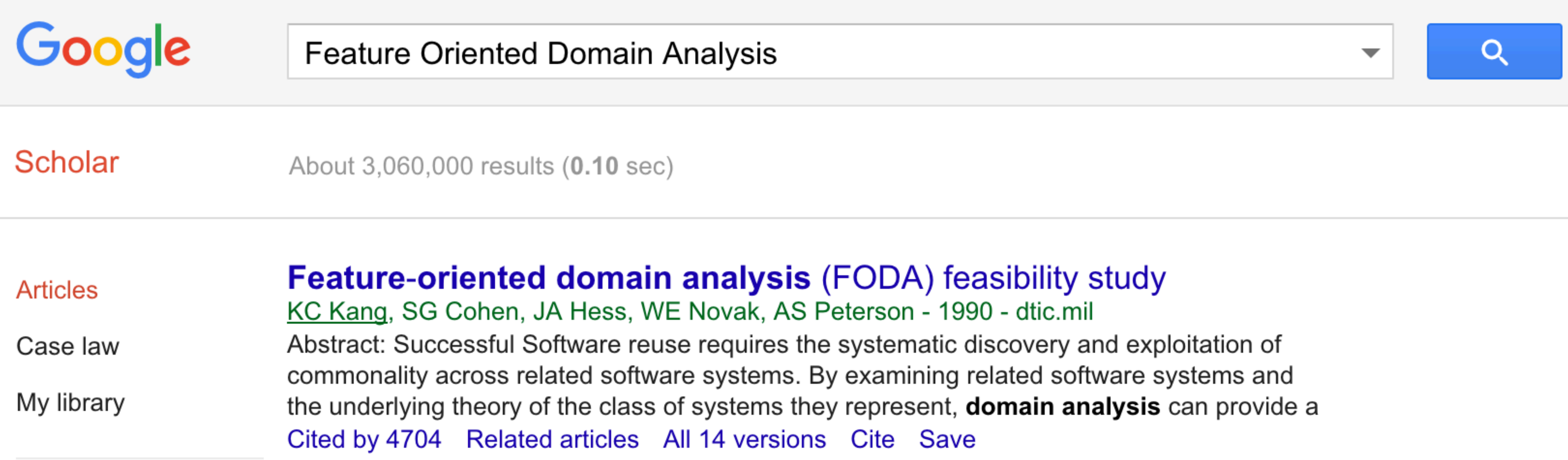
✓



{ telematicsSystem,
channel, **dual**,
extraDisplay }

Feature Modeling and FODA

- FODA succeeds for its **simplicity**
- Probably best intro in Czarnecki's *Generative Programming* (Ch. 4)
- 4700+ citations, **never formally published**



Feature Oriented Domain Analysis

Scholar

About 3,060,000 results (**0.10** sec)

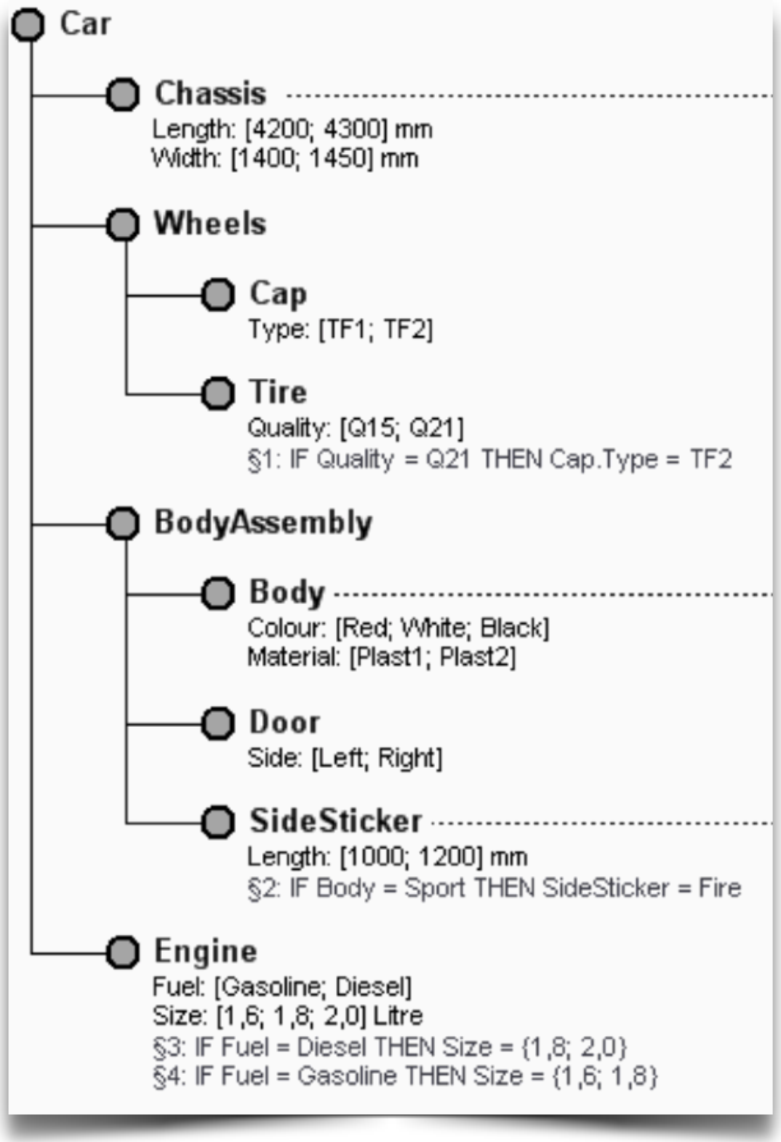
Feature-oriented domain analysis (FODA) feasibility study

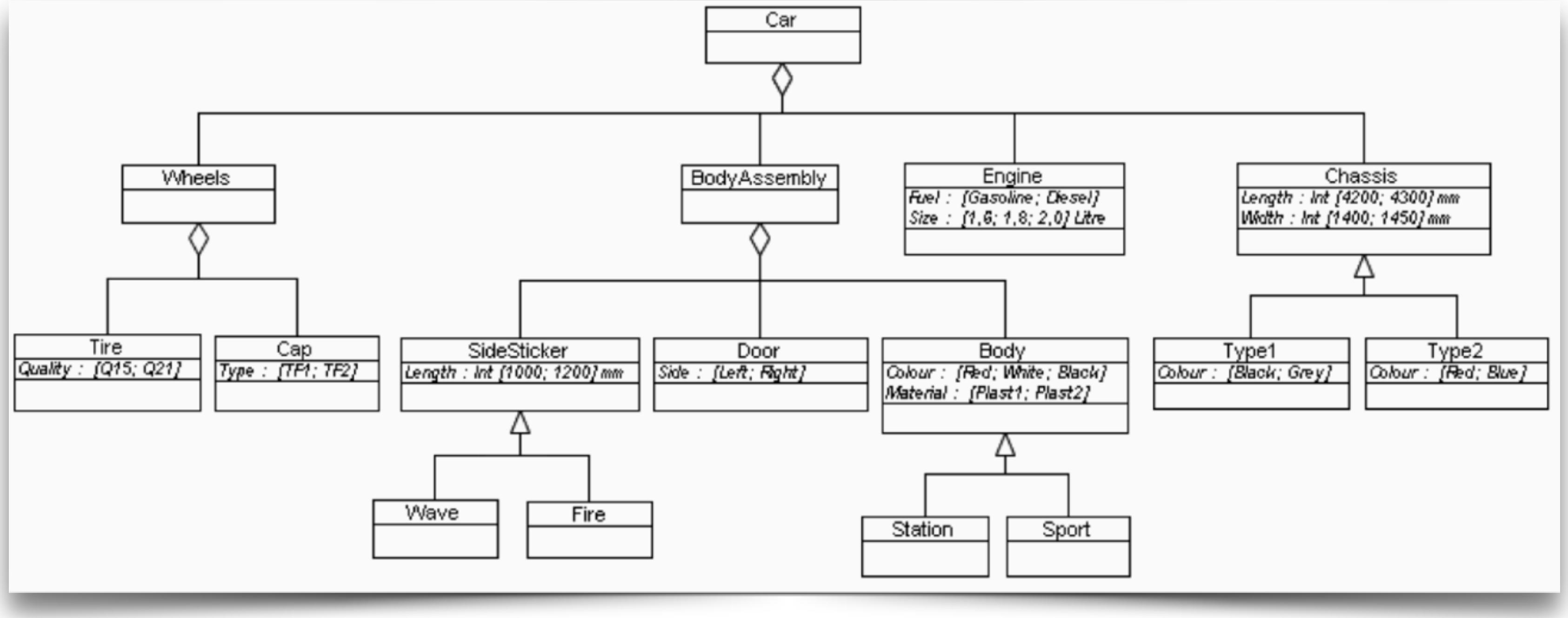
KC Kang, SG Cohen, JA Hess, WE Novak, AS Peterson - 1990 - dtic.mil

Abstract: Successful Software reuse requires the systematic discovery and exploitation of commonality across related software systems. By examining related software systems and the underlying theory of the class of systems they represent, **domain analysis** can provide a

Feature Models vs Class Models

A Feature Model in *Product Variant Master* Notation (Hvam)


Car
Chassis
Length: [4200; 4300] mm
Width: [1400; 1450] mm
Wheels
Cap
Type: [TF1; TF2]
Tire
Quality: [Q15; Q21]
§1: IF Quality = Q21 THEN Cap.Type = TF2
BodyAssembly
Body
Colour: [Red; White; Black]
Material: [Plast1; Plast2]
Door
Side: [Left; Right]
SideSticker
Length: [1000; 1200] mm
§2: IF Body = Sport THEN SideSticker = Fire
Engine
Fuel: [Gasoline; Diesel]
Size: [1,6; 1,8; 2,0] Litre
§3: IF Fuel = Diesel THEN Size = {1,8; 2,0}
§4: IF Fuel = Gasoline THEN Size = {1,6; 1,8}


Car
Wheels
BodyAssembly
Engine
Fuel : {Gasoline; Diesel}
Size : {1,6; 1,8; 2,0} Litre
Chassis
Length : Int {4200; 4300} mm
Width : Int {1400; 1450} mm
Tire
Quality : {Q15; Q21}
Cap
Type : {TF1; TF2}
SideSticker
Length : Int {1000; 1200} mm
Door
Side : {Left; Right}
Body
Colour : {Red; White; Black}
Material : {Plast1; Plast2}
Type1
Colour : {Black; Grey}
Type2
Colour : {Red; Blue}
Wave
Fire
Station
Sport

Haug et al., Creating a documentation system to support the development and maintenance of product configuration systems, WSEAS 2007

Feature Models vs Class Models

Concepts	**Few and simple**: feature, subfeature, group, constraint	**Many and complex**: class, generalization, composition, association, redefinition, refinement, property, multiplicity, package, data type, primitive type, enumeration, …
Use	**Variation** of *user-relevant* characteristics of product variants	**Concepts** representing more detailed aspects of products; *product line architectures*
Semantics	**Configuration** - selections from *predefined choices* within a fixed tree structure	**Instantiation** - making *new* structures that conform to predefined types, and *connecting* them via links

Bąk et al., Clafer: Unifying Class and Feature Modeling, SOSYM 2014

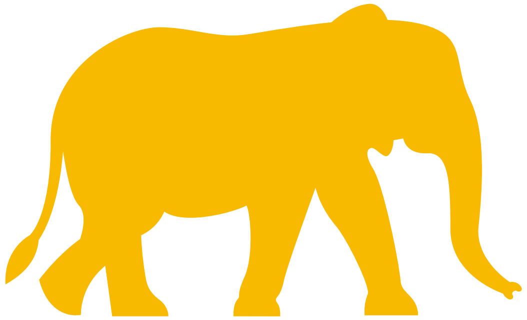
How to Build Feature Models?

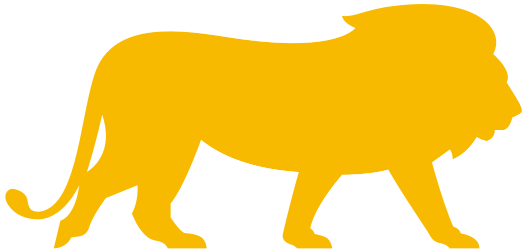
Bottom Up - Incremental Adoption

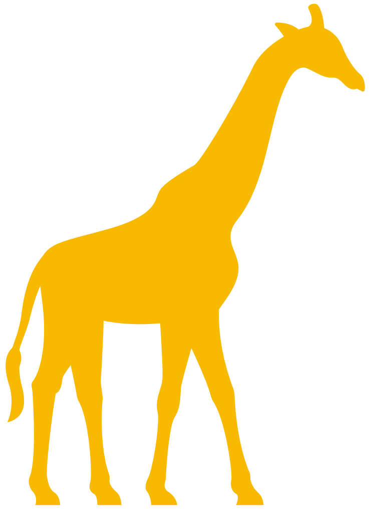
- Identify **cloned** code/functionality
- Find the **patches** that describe differences
- Diffs → **variation points**
- Aggregate variation points into hierarchical **features**

Jepsen et al., Minimally Invasive Migration to Software Product Lines, SPLC 2007
Berger et al., A survey of variability modeling in industrial practice, VAMOS 2013

Variability Modeling in the Wild

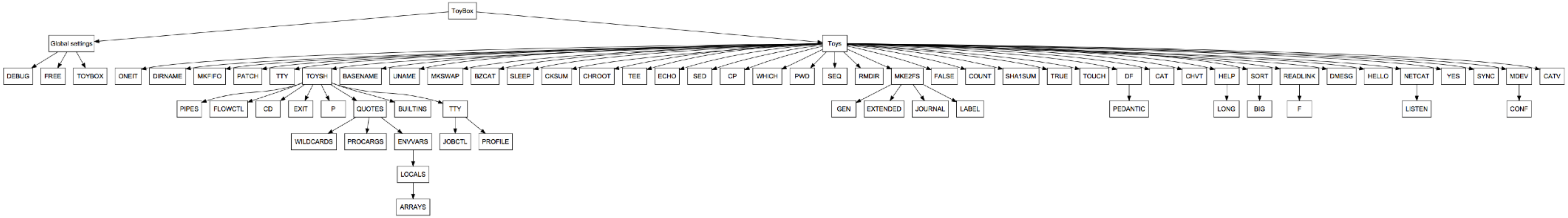






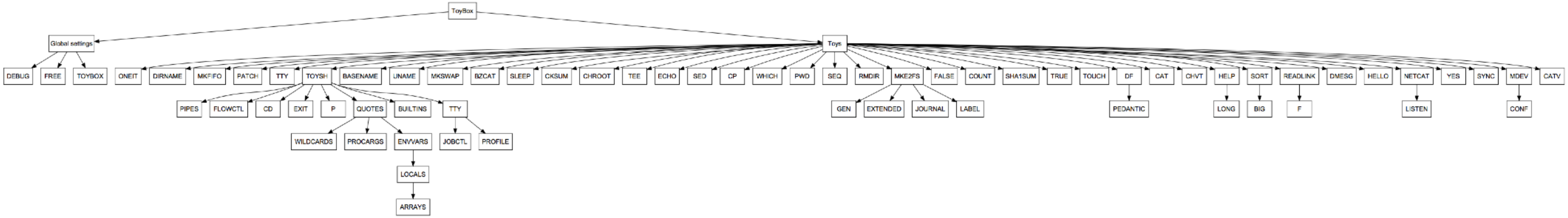
Healthy Wild Variability Model Club

ToyBox Project, 71 Features



Healthy Wild Variability Model Club

ToyBox Project, 71 Features



The Linux Kernel has 6-12k features, depending on how you count.

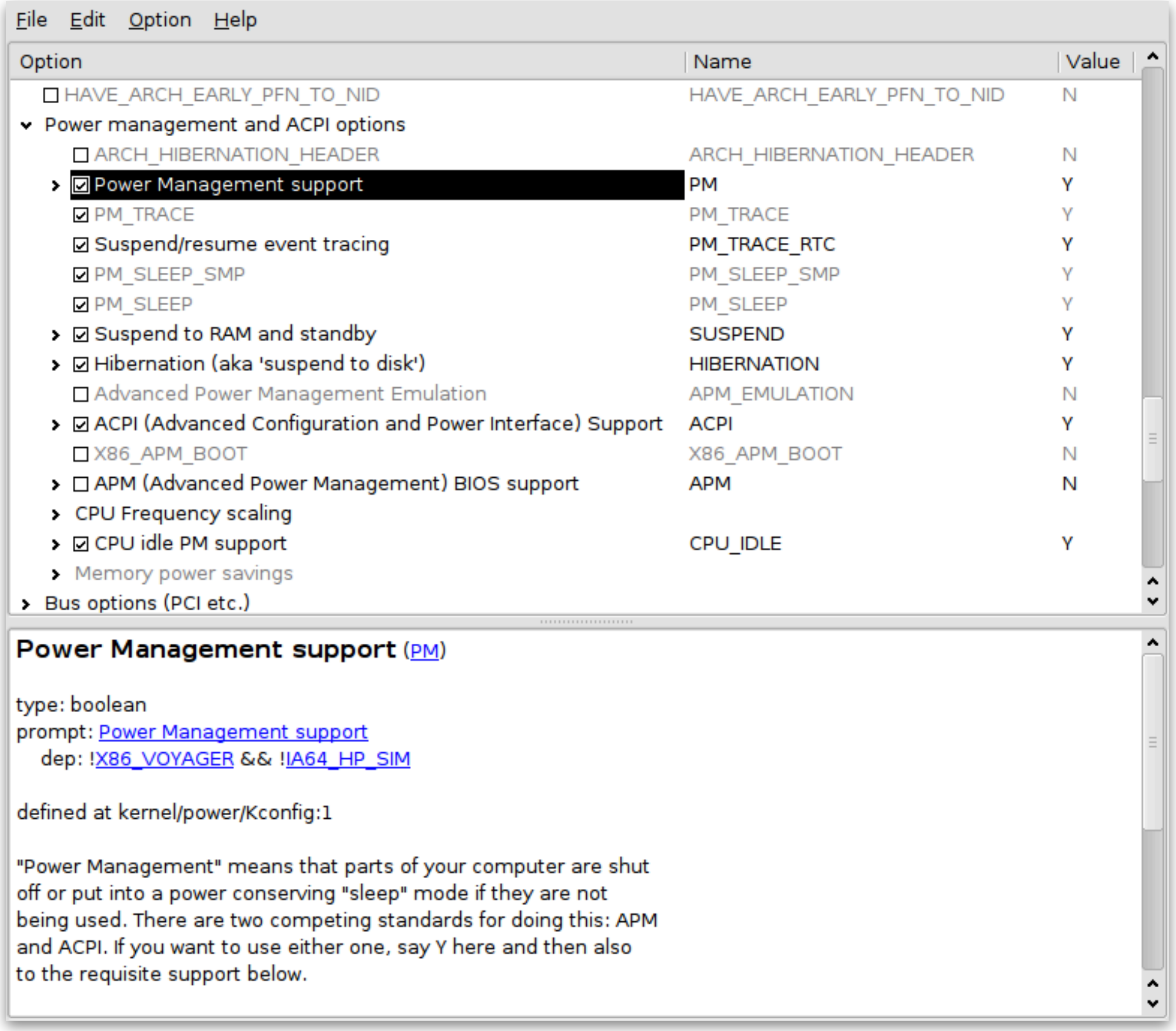
Max depth: 8. Most leaves are at 4! **Shallow**

↓ this is the Linux Kernel model fit to the slide width ↓

Berger et al., A Study of Variability Models and Languages in the Systems Software Domain, TSE 2013

KConfig & CDL

FM with Attributes, Defaults, Constraint Propagation, UI, ...



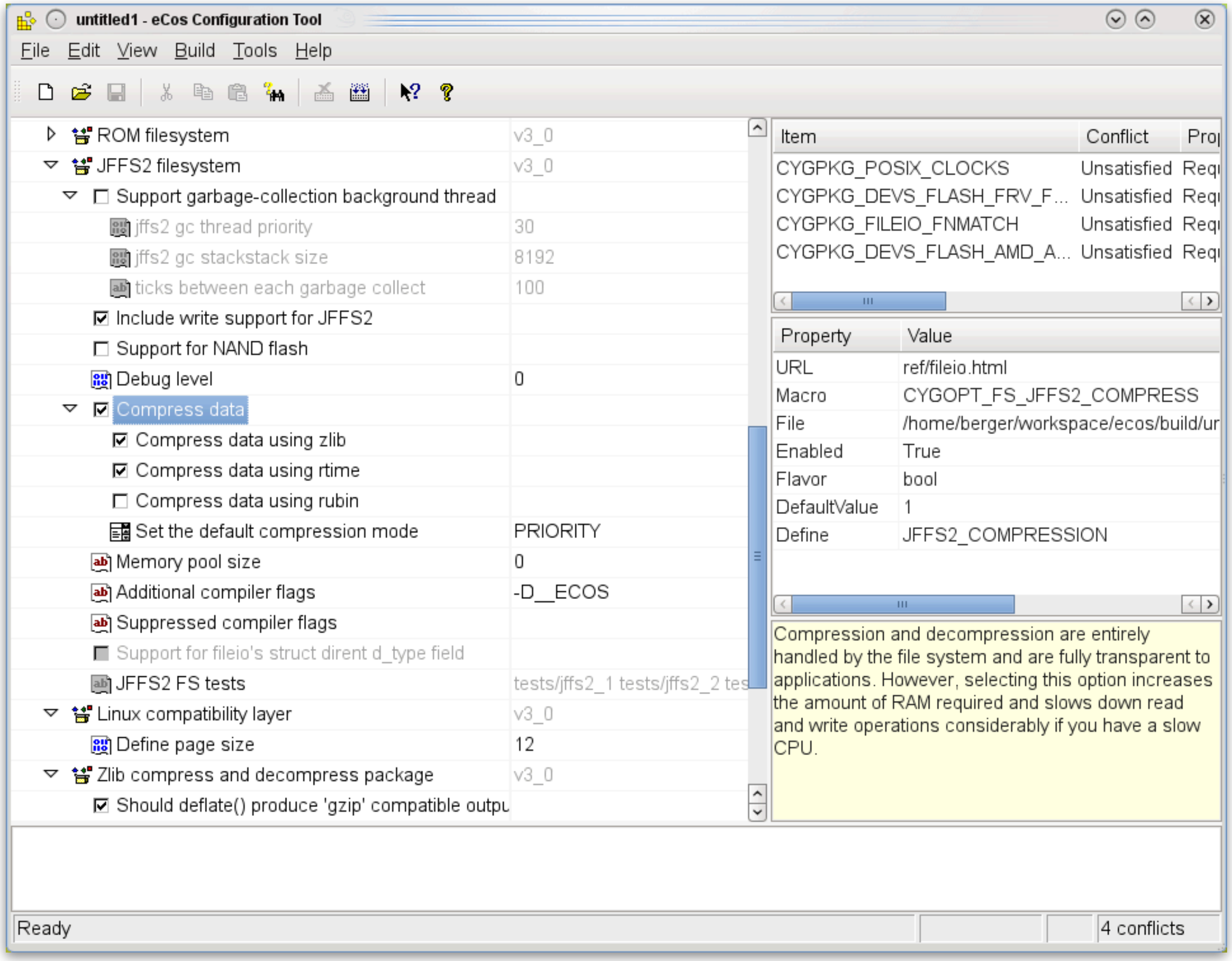
Option	Name	Value
□ HAVE_ARCH_EARLY_PFN_TO_NID	HAVE_ARCH_EARLY_PFN_TO_NID	N
⌄ Power management and ACPI options		
□ ARCH_HIBERNATION_HEADER	ARCH_HIBERNATION_HEADER	N
› ☑ Power Management support	PM	Y
☑ PM_TRACE	PM_TRACE	Y
☑ Suspend/resume event tracing	PM_TRACE_RTC	Y
☑ PM_SLEEP_SMP	PM_SLEEP_SMP	Y
☑ PM_SLEEP	PM_SLEEP	Y
› ☑ Suspend to RAM and standby	SUSPEND	Y
› ☑ Hibernation (aka 'suspend to disk')	HIBERNATION	Y
□ Advanced Power Management Emulation	APM_EMULATION	N
› ☑ ACPI (Advanced Configuration and Power Interface) Support	ACPI	Y
□ X86_APM_BOOT	X86_APM_BOOT	N
› □ APM (Advanced Power Management) BIOS support	APM	N
› CPU Frequency scaling		
› ☑ CPU idle PM support	CPU_IDLE	Y
› Memory power savings		
› Bus options (PCI etc.)		

Power Management support (PM)

type: boolean
prompt: Power Management support
  dep: !X86_VOYAGER && !IA64_HP_SIM

defined at kernel/power/Kconfig:1

"Power Management" means that parts of your computer are shut off or put into a power conserving "sleep" mode if they are not being used. There are two competing standards for doing this: APM and ACPI. If you want to use either one, say Y here and then also to the requisite support below.


untitled1 - eCos Configuration Tool
File Edit View Build Tools Help

▷ ROM filesystem	v3_0
▽ JFFS2 filesystem	v3_0
▽ □ Support garbage-collection background thread	
jffs2 gc thread priority	30
jffs2 gc stackstack size	8192
ticks between each garbage collect	100
☑ Include write support for JFFS2	
□ Support for NAND flash	
Debug level	0
▽ ☑ Compress data	
☑ Compress data using zlib	
☑ Compress data using rtime	
□ Compress data using rubin	
Set the default compression mode	PRIORITY
Memory pool size	0
Additional compiler flags	-D__ECOS
Suppressed compiler flags	
□ Support for fileio's struct dirent d_type field	
JFFS2 FS tests	tests/jffs2_1 tests/jffs2_2 tes
▽ Linux compatibility layer	v3_0
Define page size	12
▽ Zlib compress and decompress package	v3_0
☑ Should deflate() produce 'gzip' compatible outpu	

Item	Conflict	Pro
CYGPKG_POSIX_CLOCKS	Unsatisfied	Req
CYGPKG_DEVS_FLASH_FRV_F...	Unsatisfied	Req
CYGPKG_FILEIO_FNMATCH	Unsatisfied	Req
CYGPKG_DEVS_FLASH_AMD_A...	Unsatisfied	Req

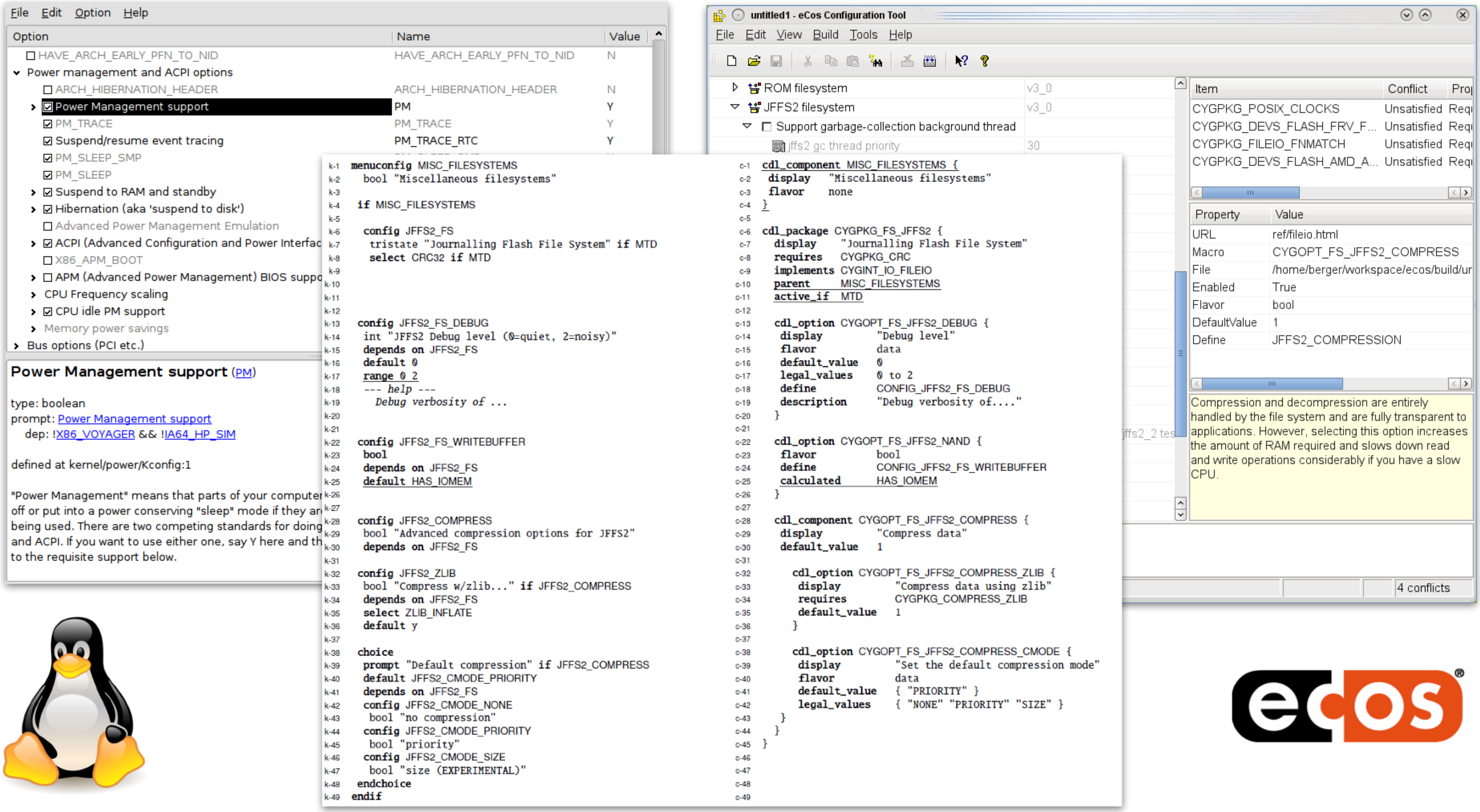
Property	Value
URL	ref/fileio.html
Macro	CYGOPT_FS_JFFS2_COMPRESS
File	/home/berger/workspace/ecos/build/ur
Enabled	True
Flavor	bool
DefaultValue	1
Define	JFFS2_COMPRESSION

Compression and decompression are entirely handled by the file system and are fully transparent to applications. However, selecting this option increases the amount of RAM required and slows down read and write operations considerably if you have a slow CPU.

Ready 4 conflicts

KConfig & CDL

Textual Variability Models



Option	Name	Value
□ HAVE_ARCH_EARLY_PFN_TO_NID	HAVE_ARCH_EARLY_PFN_TO_NID	N
⌄ Power management and ACPI options		
□ ARCH_HIBERNATION_HEADER	ARCH_HIBERNATION_HEADER	N
› ☑ Power Management support	PM	Y
☑ PM_TRACE	PM_TRACE	Y
☑ Suspend/resume event tracing	PM_TRACE_RTC	Y
☑ PM_SLEEP_SMP		
☑ PM_SLEEP		
› ☑ Suspend to RAM and standby		
› ☑ Hibernation (aka 'suspend to disk')		
□ Advanced Power Management Emulation		
› ☑ ACPI (Advanced Configuration and Power Interfac		
□ X86_APM_BOOT		
› □ APM (Advanced Power Management) BIOS suppo		
› CPU Frequency scaling		
› ☑ CPU idle PM support		
› Memory power savings		
› Bus options (PCI etc.)		

Power Management support (PM)

type: boolean
prompt: Power Management support
dep: !X86_VOYAGER && !IA64_HP_SIM

defined at kernel/power/Kconfig:1

"Power Management" means that parts of your computer
off or put into a power conserving "sleep" mode if they ar
being used. There are two competing standards for doing
and ACPI. If you want to use either one, say Y here and th
to the requisite support below.

untitled1 - eCos Configuration Tool

▷ ROM filesystem v3_0
▽ JFFS2 filesystem v3_0
▽ □ Support garbage-collection background thread
jffs2 gc thread priority 30
jffs2_2 tes

Item	Conflict	Pro
CYGPKG_POSIX_CLOCKS	Unsatisfied	Req
CYGPKG_DEVS_FLASH_FRV_F...	Unsatisfied	Req
CYGPKG_FILEIO_FNMATCH	Unsatisfied	Req
CYGPKG_DEVS_FLASH_AMD_A...	Unsatisfied	Req

Property	Value
URL	ref/fileio.html
Macro	CYGOPT_FS_JFFS2_COMPRESS
File	/home/berger/workspace/ecos/build/ur
Enabled	True
Flavor	bool
DefaultValue	1
Define	JFFS2_COMPRESSION

Compression and decompression are entirely handled by the file system and are fully transparent to applications. However, selecting this option increases the amount of RAM required and slows down read and write operations considerably if you have a slow CPU.

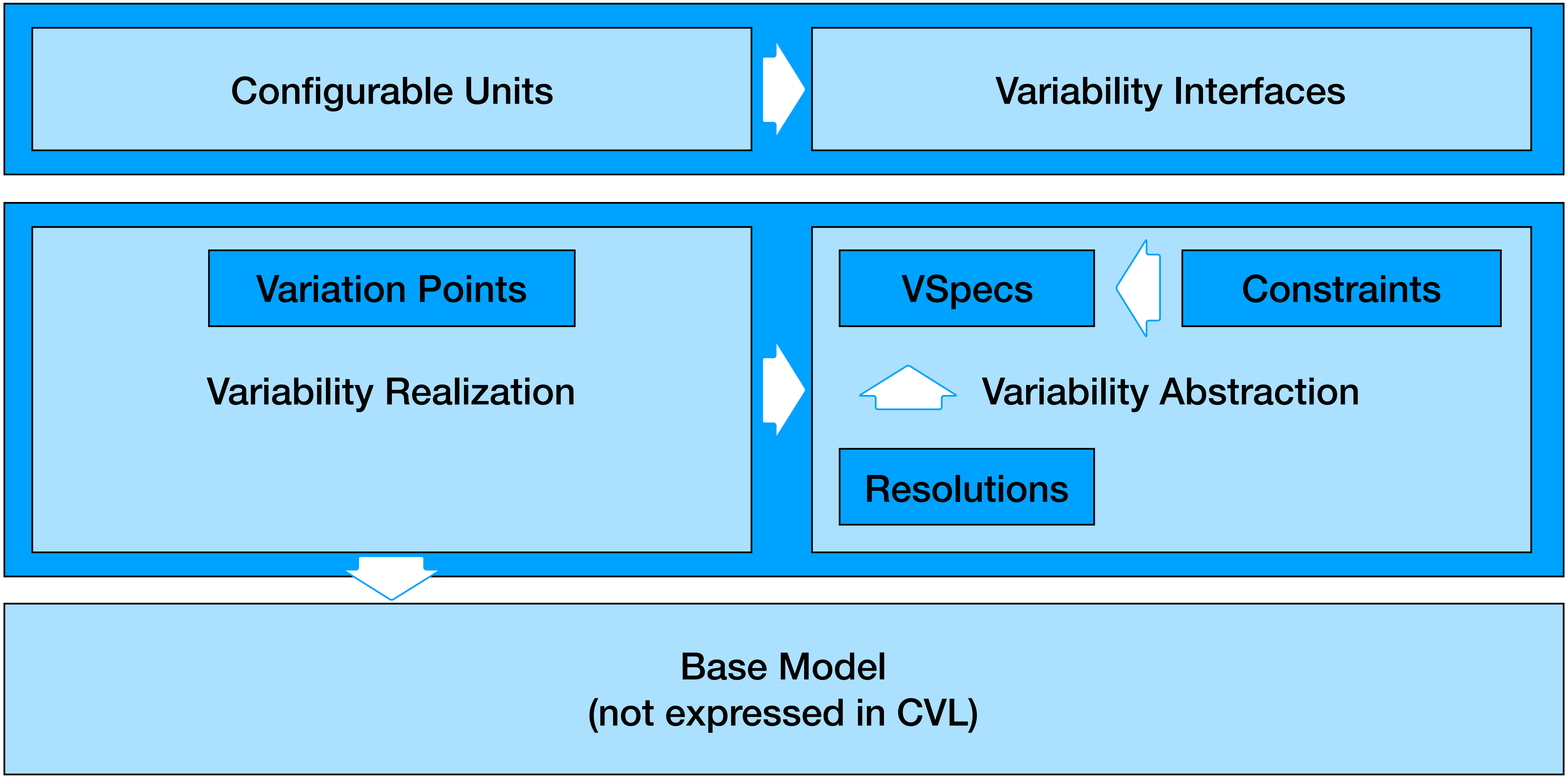
4 conflicts

```
menuconfig MISC_FILESYSTEMS
 bool "Miscellaneous filesystems"

if MISC_FILESYSTEMS

 config JFFS2_FS
  tristate "Journalling Flash File System" if MTD
  select CRC32 if MTD




config JFFS2_FS_DEBUG
 int "JFFS2 Debug level (0=quiet, 2=noisy)"
 depends on JFFS2_FS
 default 0
 range 0 2
 --- help ---
  Debug verbosity of ...


config JFFS2_FS_WRITEBUFFER
 bool
 depends on JFFS2_FS
 default HAS_IOMEM


config JFFS2_COMPRESS
 bool "Advanced compression options for JFFS2"
 depends on JFFS2_FS

config JFFS2_ZLIB
 bool "Compress w/zlib..." if JFFS2_COMPRESS
 depends on JFFS2_FS
 select ZLIB_INFLATE
 default y

choice
 prompt "Default compression" if JFFS2_COMPRESS
 default JFFS2_CMODE_PRIORITY
 depends on JFFS2_FS
 config JFFS2_CMODE_NONE
  bool "no compression"
 config JFFS2_CMODE_PRIORITY
  bool "priority"
 config JFFS2_CMODE_SIZE
  bool "size (EXPERIMENTAL)"
endchoice
endif
```

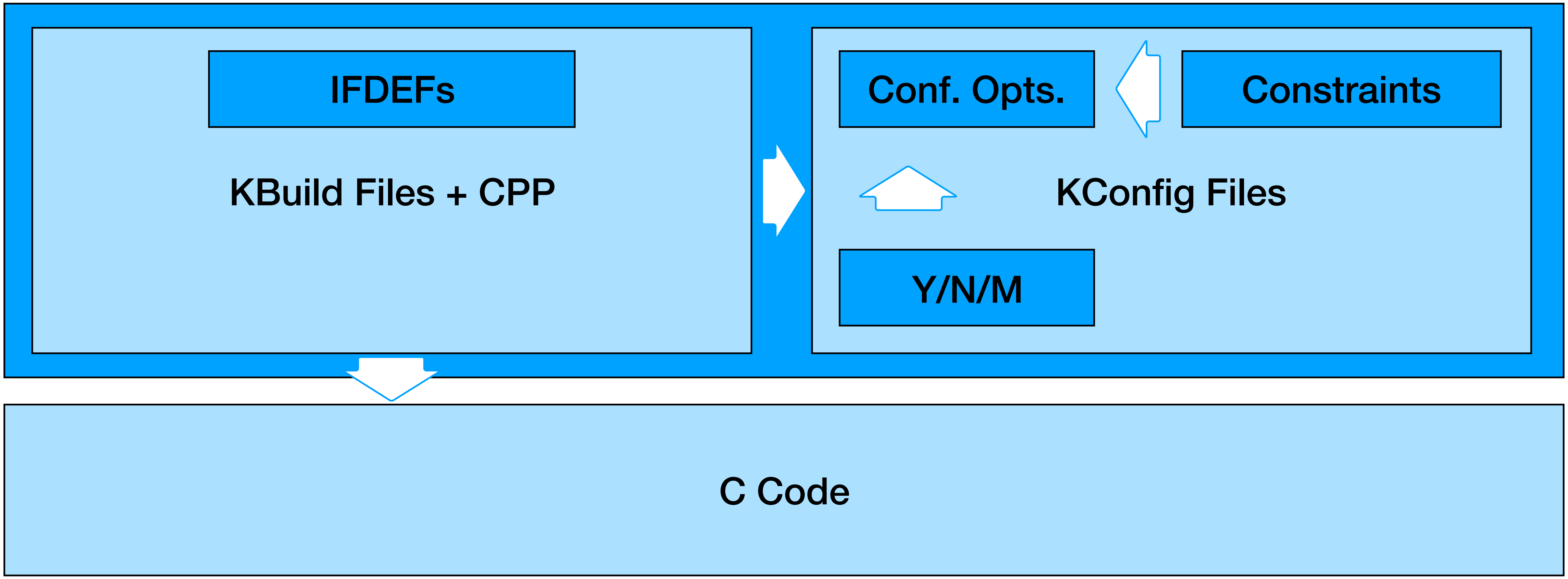
```
cdl_component MISC_FILESYSTEMS {
 display   "Miscellaneous filesystems"
 flavor    none
}

cdl_package CYGPKG_FS_JFFS2 {
  display    "Journalling Flash File System"
  requires   CYGPKG_CRC
  implements CYGINT_IO_FILEIO
  parent     MISC_FILESYSTEMS
  active_if  MTD

  cdl_option CYGOPT_FS_JFFS2_DEBUG {
   display         "Debug level"
   flavor          data
   default_value   0
   legal_values    0 to 2
   define          CONFIG_JFFS2_FS_DEBUG
   description     "Debug verbosity of...."
  }

  cdl_option CYGOPT_FS_JFFS2_NAND {
   flavor          bool
   define          CONFIG_JFFS2_FS_WRITEBUFFER
   calculated      HAS_IOMEM
  }

  cdl_component CYGOPT_FS_JFFS2_COMPRESS {
   display         "Compress data"
   default_value   1

     cdl_option CYGOPT_FS_JFFS2_COMPRESS_ZLIB {
      display         "Compress data using zlib"
      requires        CYGPKG_COMPRESS_ZLIB
      default_value   1
     }

     cdl_option CYGOPT_FS_JFFS2_COMPRESS_CMODE {
      display         "Set the default compression mode"
      flavor          data
      default_value   { "PRIORITY" }
      legal_values    { "NONE" "PRIORITY" "SIZE" }
   }
  }
}
```

Common Variability Language (CVL)

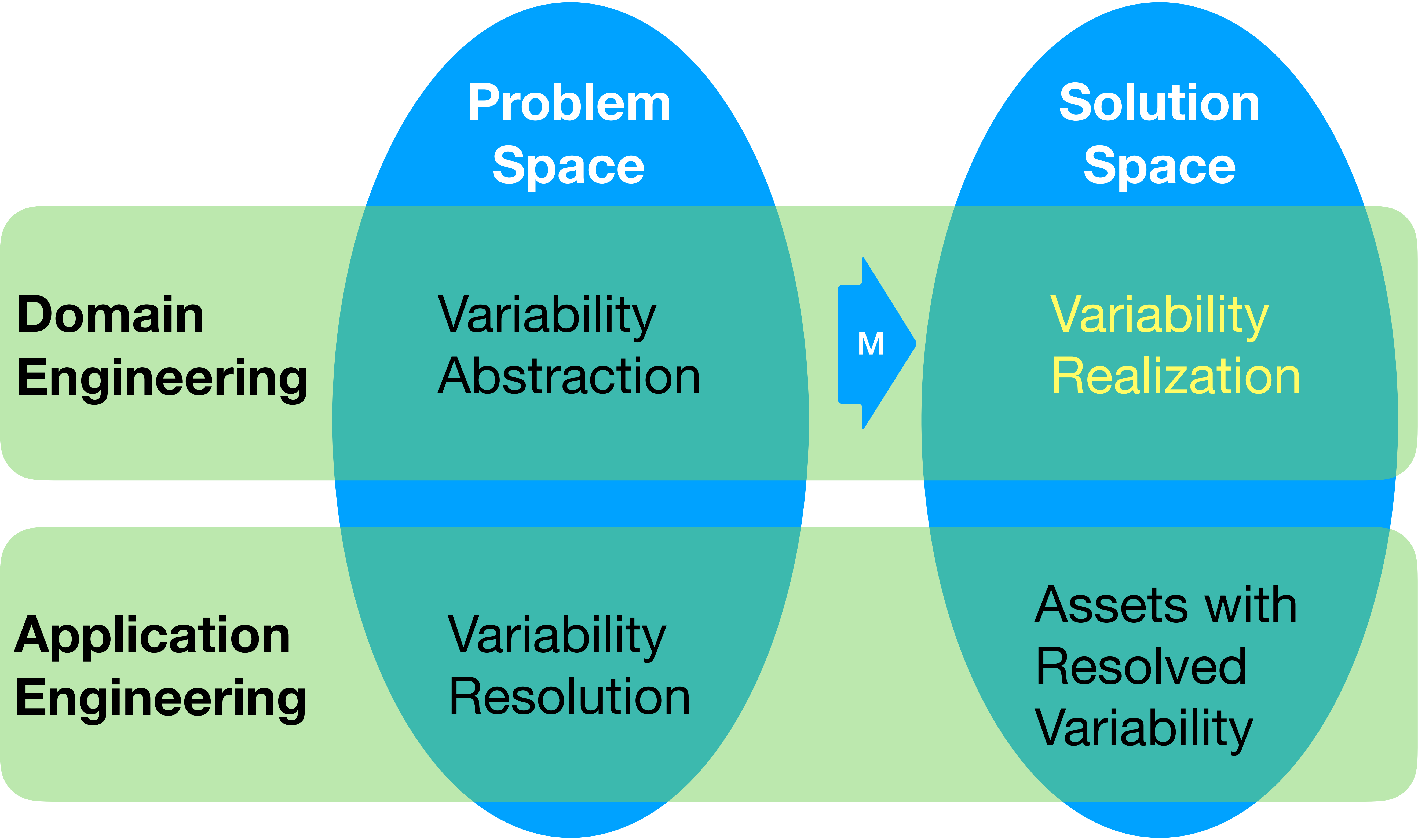
IBM et al., Proposal for CVL Revised Submission, 2012



CVL Architecture for Linux Junkies



Software Product Line



Problem Space

Solution Space

Domain Engineering

Variability Abstraction

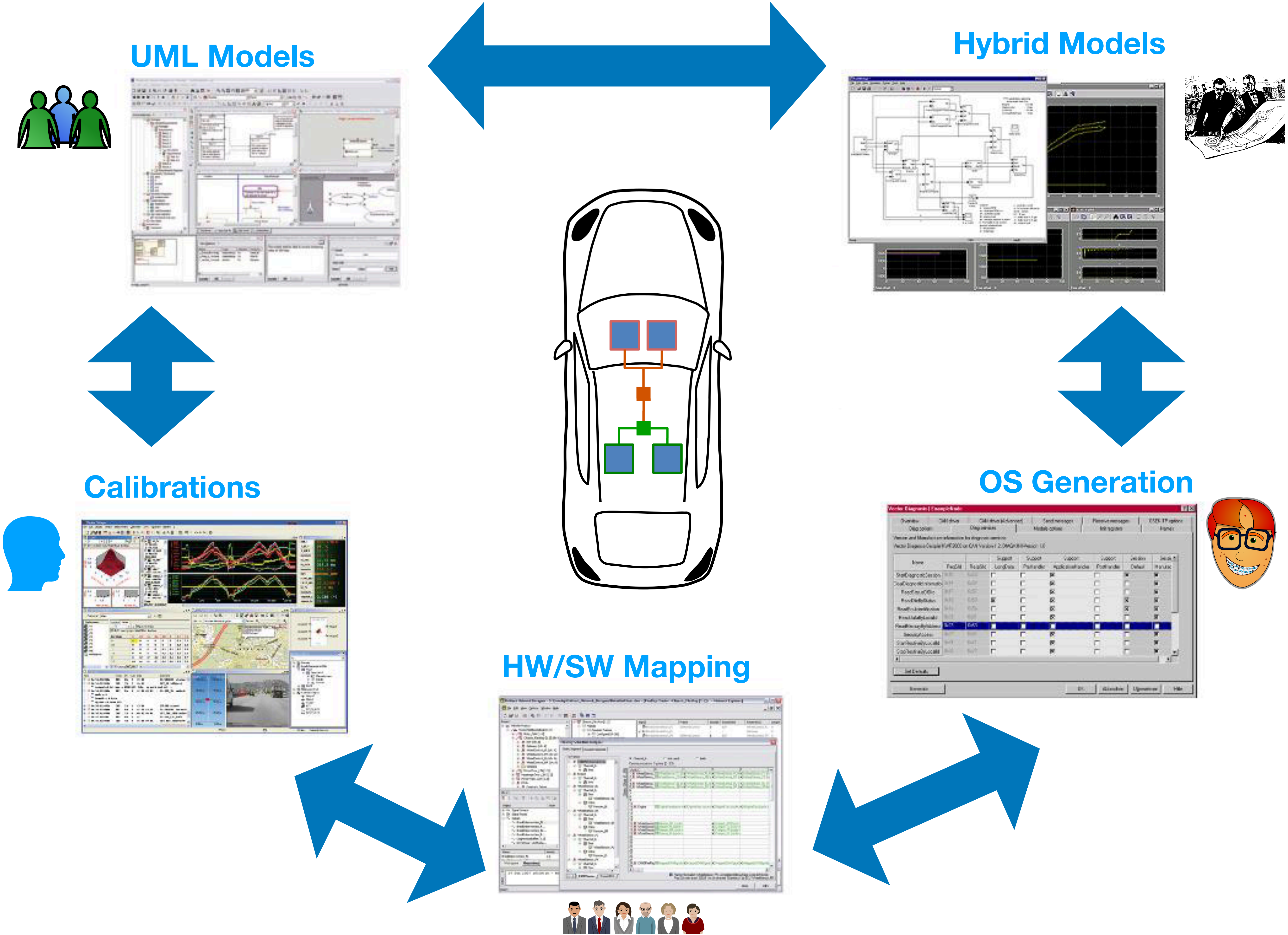
M

Variability Realization

Application Engineering

Variability Resolution

Assets with Resolved Variability



UML Models

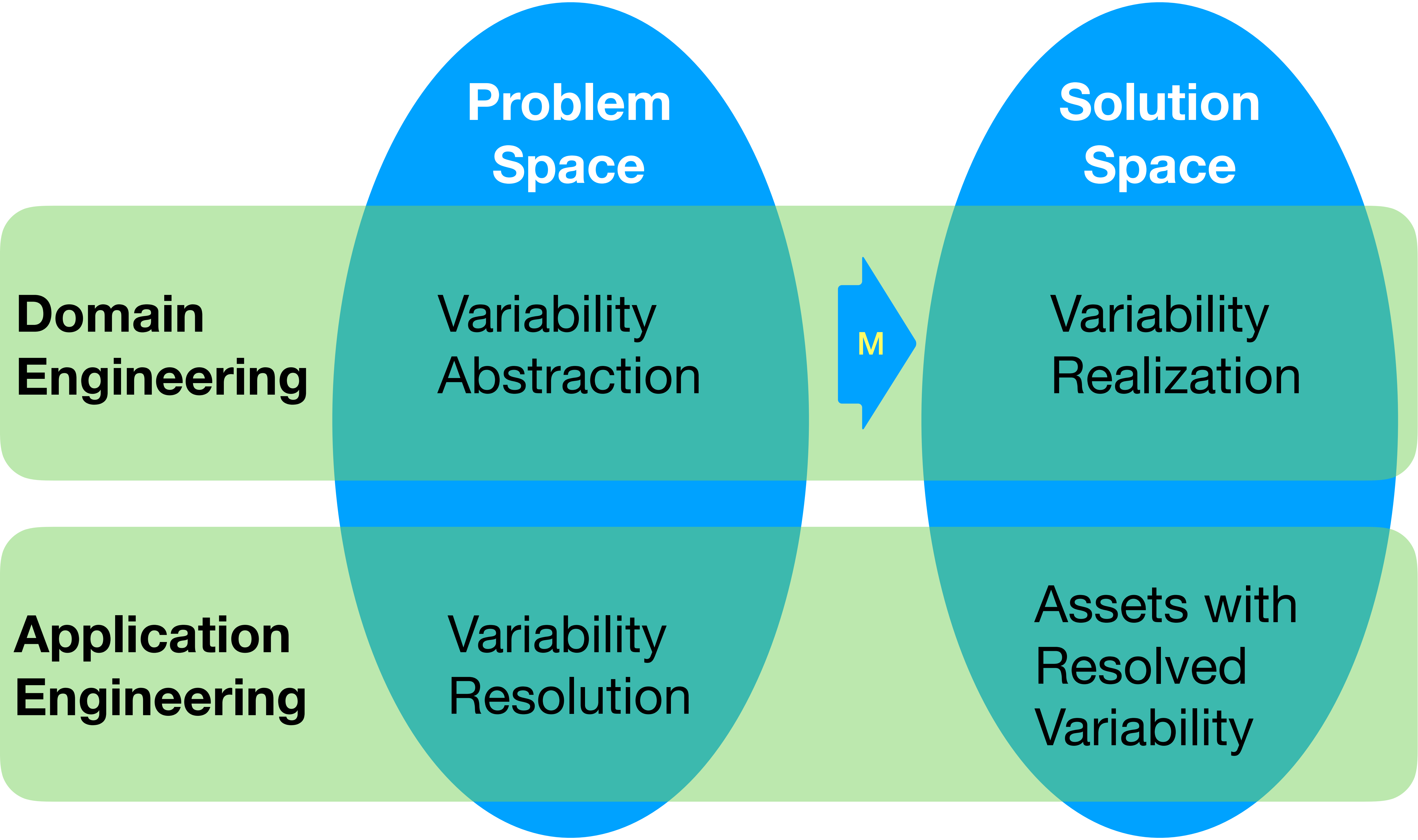
Hybrid Models

OS Generation

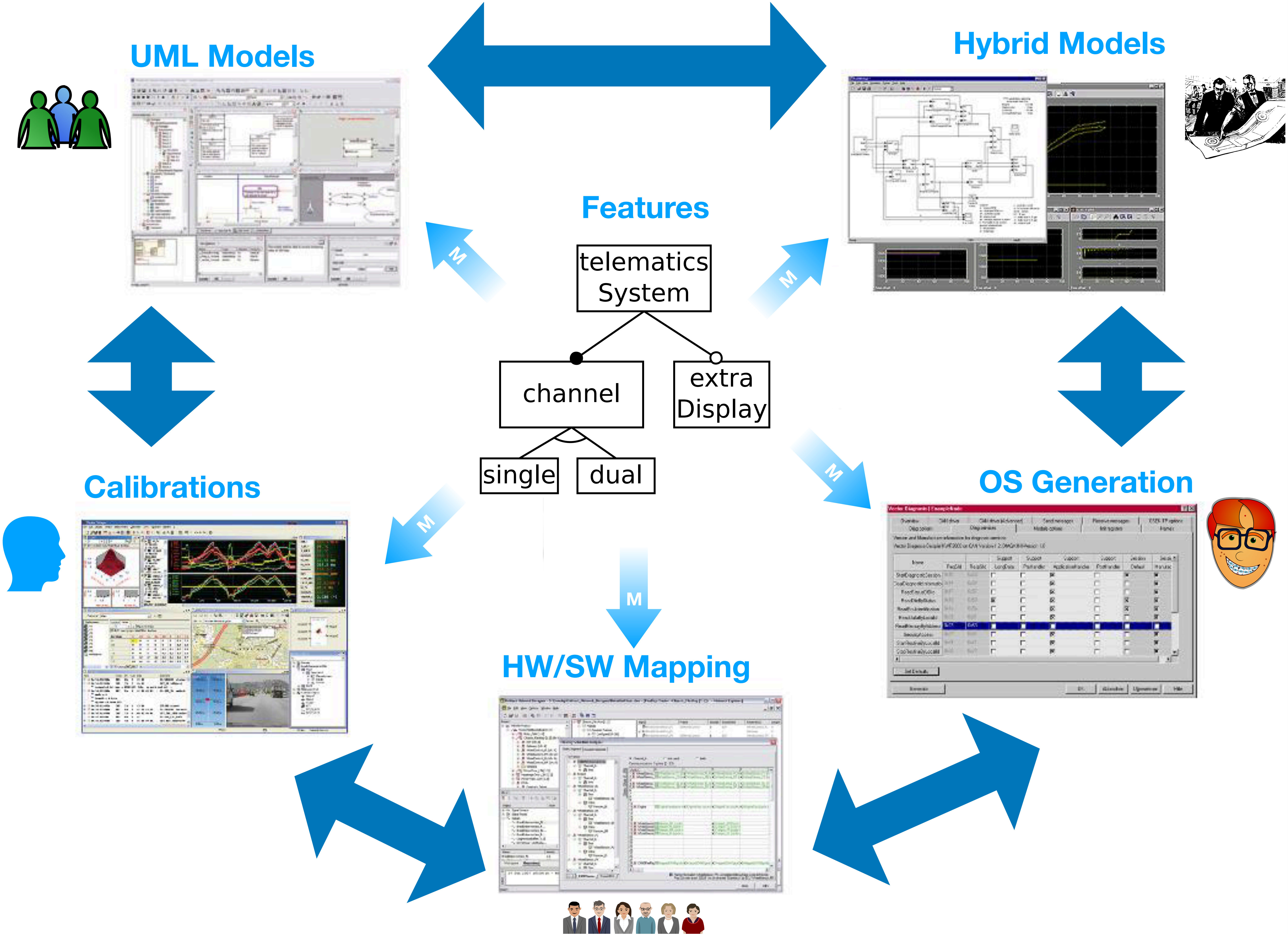
HW/SW Mapping

Calibrations

Software Product Line



	Problem Space		Solution Space
Domain Engineering	Variability Abstraction	M	Variability Realization
Application Engineering	Variability Resolution		Assets with Resolved Variability



UML Models

Hybrid Models

Features

telematics System

channel

extra Display

single

dual

M

M

M

M

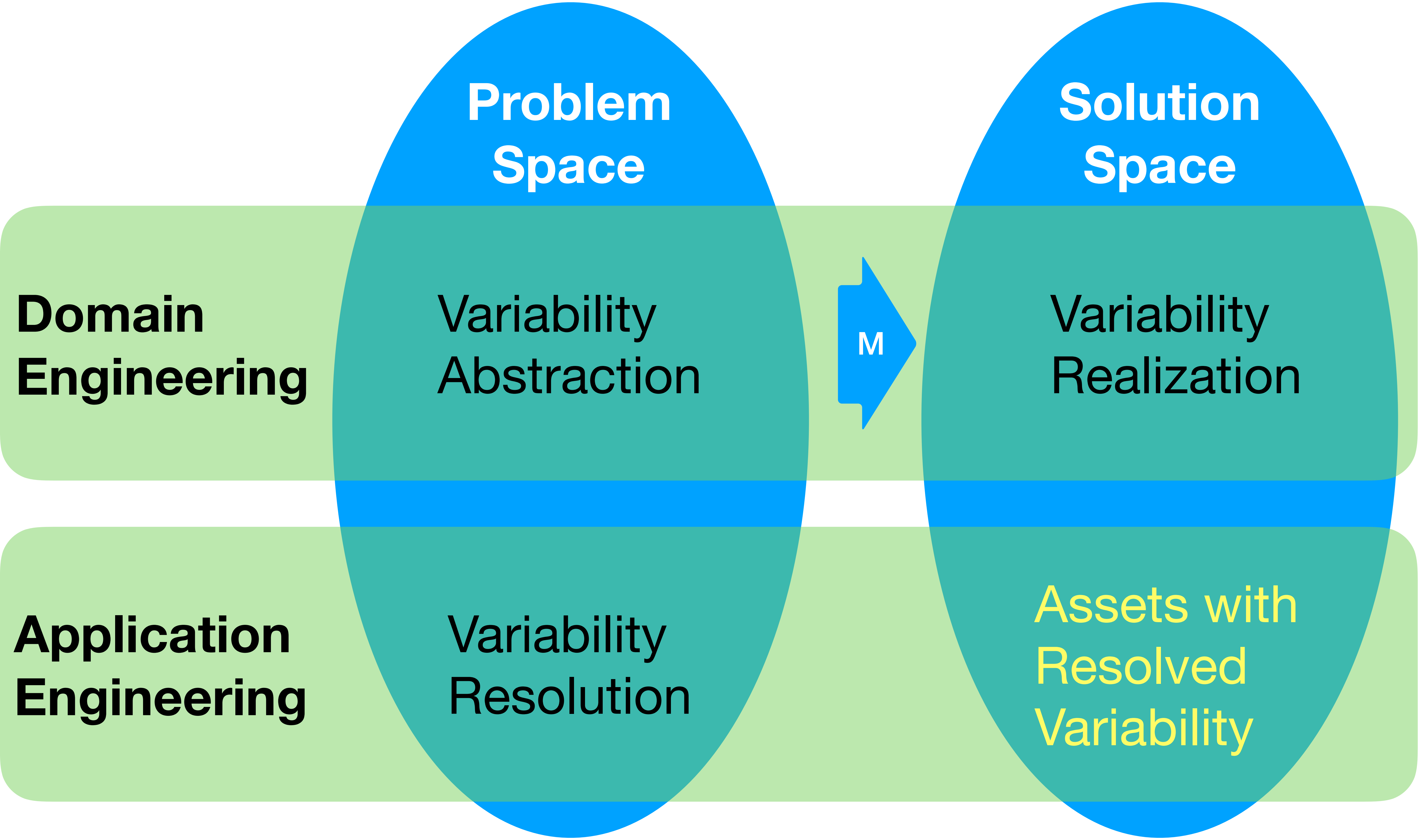
M

Calibrations

OS Generation

HW/SW Mapping

Software Product Line



Problem Space

Solution Space

Domain Engineering

Variability Abstraction

M

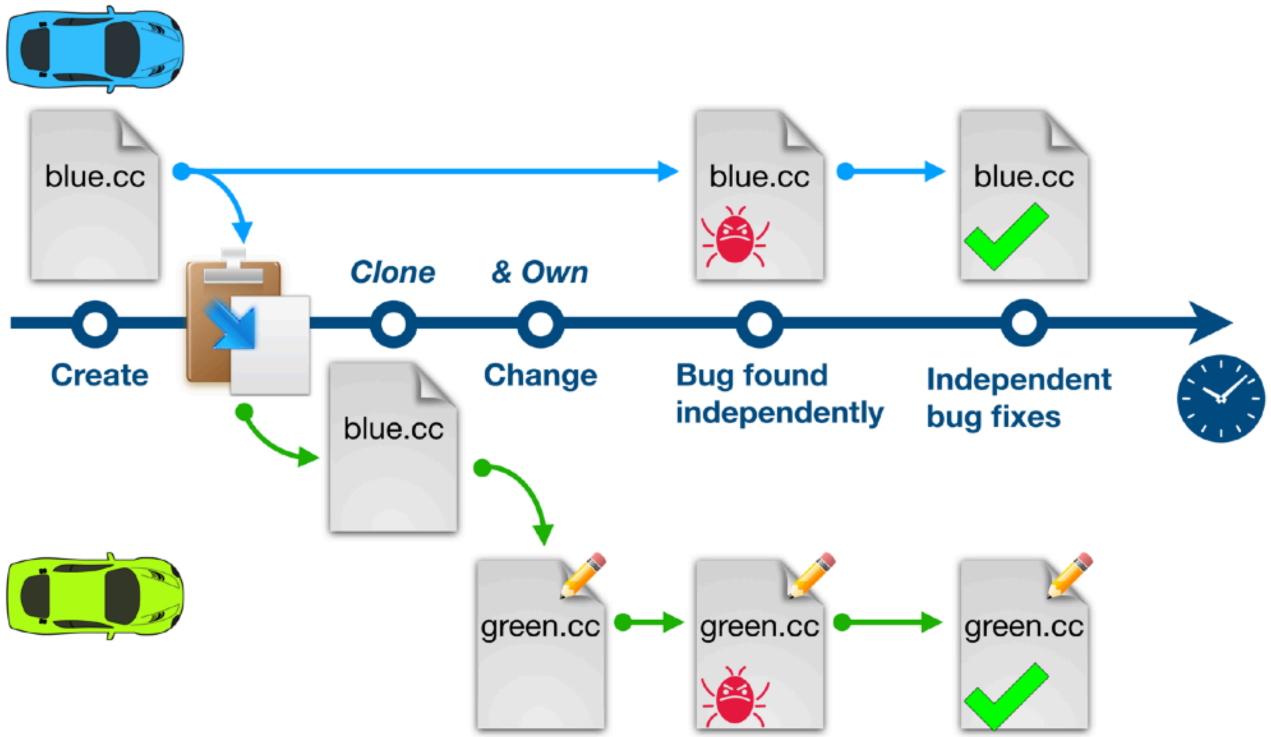
Variability Realization

Application Engineering

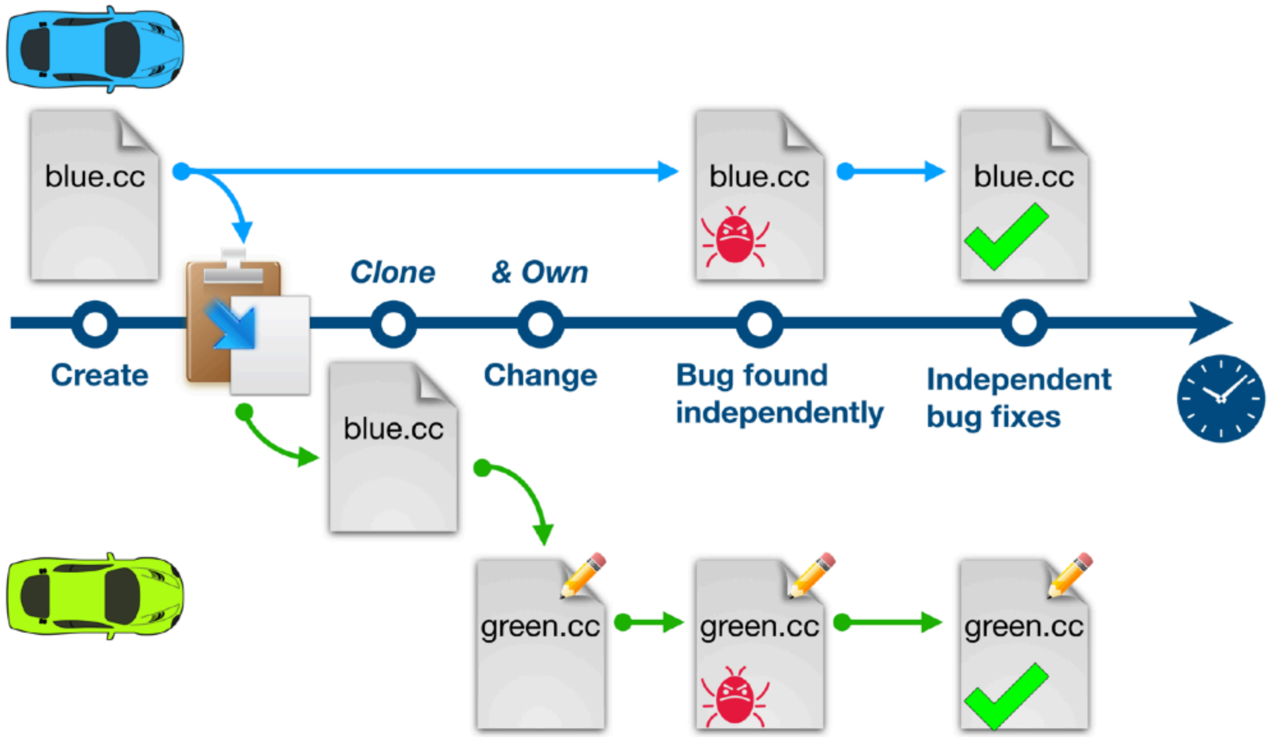
Variability Resolution

Assets with Resolved Variability

Is Clone and Own Always Bad?

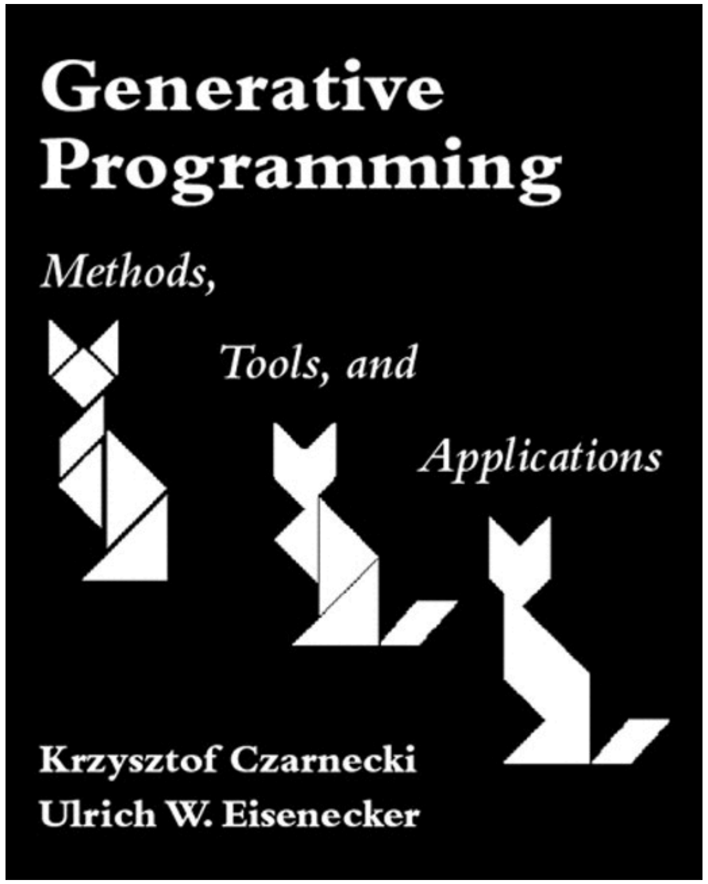


Is Clone and Own Always Bad?



not if the cost of cloning is less

than the cost of an SPL 😉



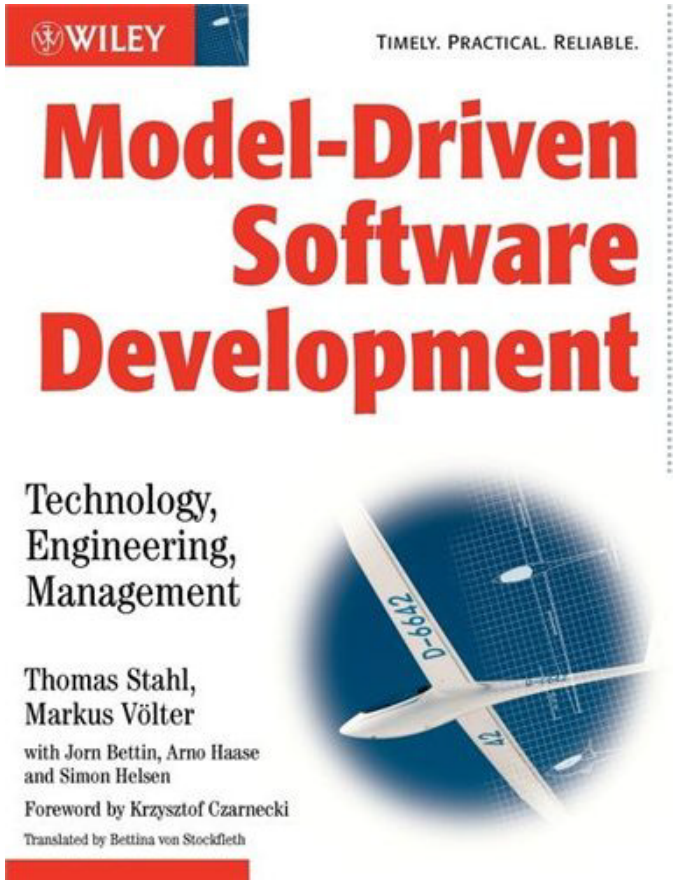
Generative Programming

Methods,

Tools, and

Applications

Krzysztof Czarnecki
Ulrich W. Eisenecker



WILEY

TIMELY. PRACTICAL. RELIABLE.

Model-Driven Software Development

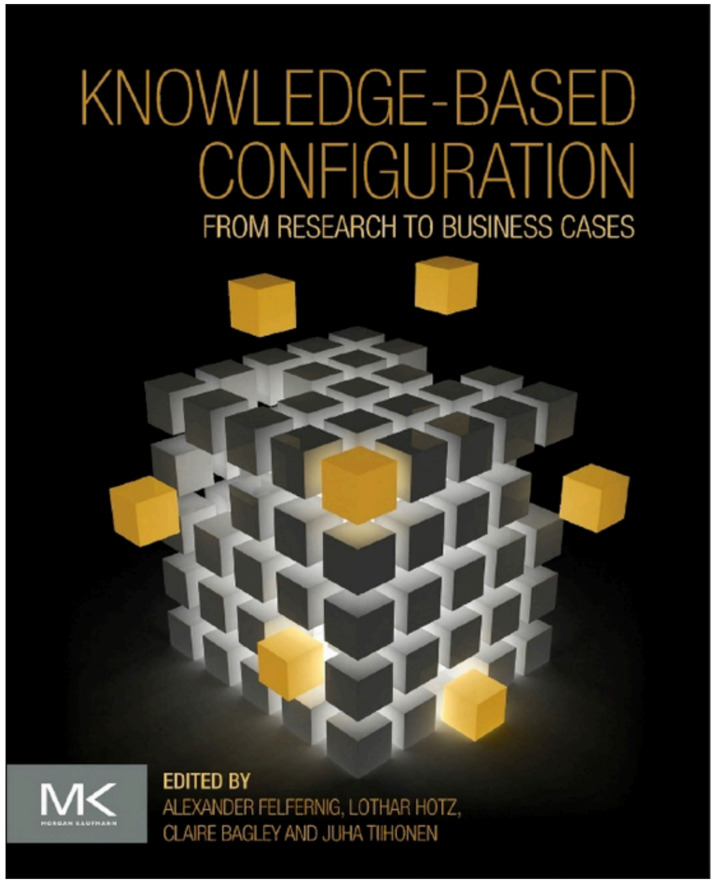
Technology,
Engineering,
Management

Thomas Stahl,
Markus Völter

with Jorn Bettin, Arno Haase
and Simon Helsen

Foreword by Krzysztof Czarnecki

Translated by Bettina von Stockfleth

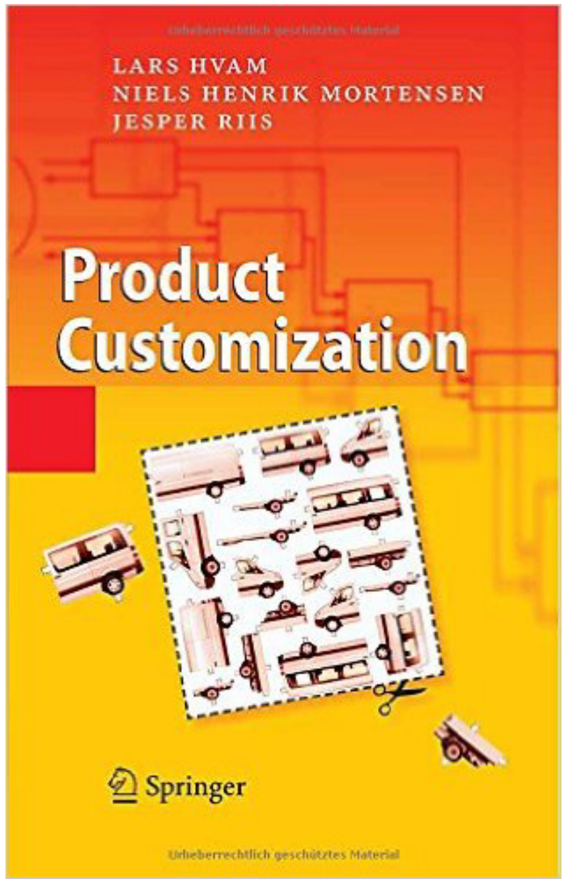


KNOWLEDGE-BASED CONFIGURATION

FROM RESEARCH TO BUSINESS CASES

MK

EDITED BY
ALEXANDER FELFERNIG, LOTHAR HOTZ,
CLAIRE BAGLEY AND JUHA TIIHONEN



LARS HVAM
NIELS HENRIK MORTENSEN
JESPER RIIS

Product Customization

Springer

Exercise

- Example Domain: *Traffic Lights*
  - Feature-oriented commonality/variability analysis
  - Domain concept analysis
  - Application configuration
- Apply *Example-Driven Modeling*
- Use Clafer & Web Tools
  - Tutorial style
  - Hands-on
  - Small exercises

Interactive Tutorial

http://t3-necsis.cs.uwaterloo.ca:8098/

Use Chrome or Firefox

Indent code with spaces, not tabs