Model-Based Testing of Software Product Lines

Prof. Dr. Holger Schlingloff
Fraunhofer FOKUS & Humboldt Universität

holger.schlingloff@fokus.fraunhofer.de

Fraunhofer FOKUS

Fraunhofer FOKUS

Fraunhofer Institute for
Open Communication Systems

- formed July 2012 by a merger of three institutes (FOKUS, FIRST, ISST)
- located in the center of Berlin
- staff: approx. 500 computer scientists
- main topic: smart cities

Competence Centers:

- Embedded Systems Quality Management
- Modelling and Testing for Systems and Service Solutions
- Next Generation Network Infrastructures
- Electronic Safety and Security Systems for the Public and Industries
- Future Applications and Media
- ....

Structure of this Talk

1. **Software product lines, feature modeling**
2. **Testing software product lines**
3. **A testing theory based on CSP-CASL**
4. **Implementation and applications**

I. Software Product Lines (SPL)

Concept emerged from CMU SEI in the 1990s

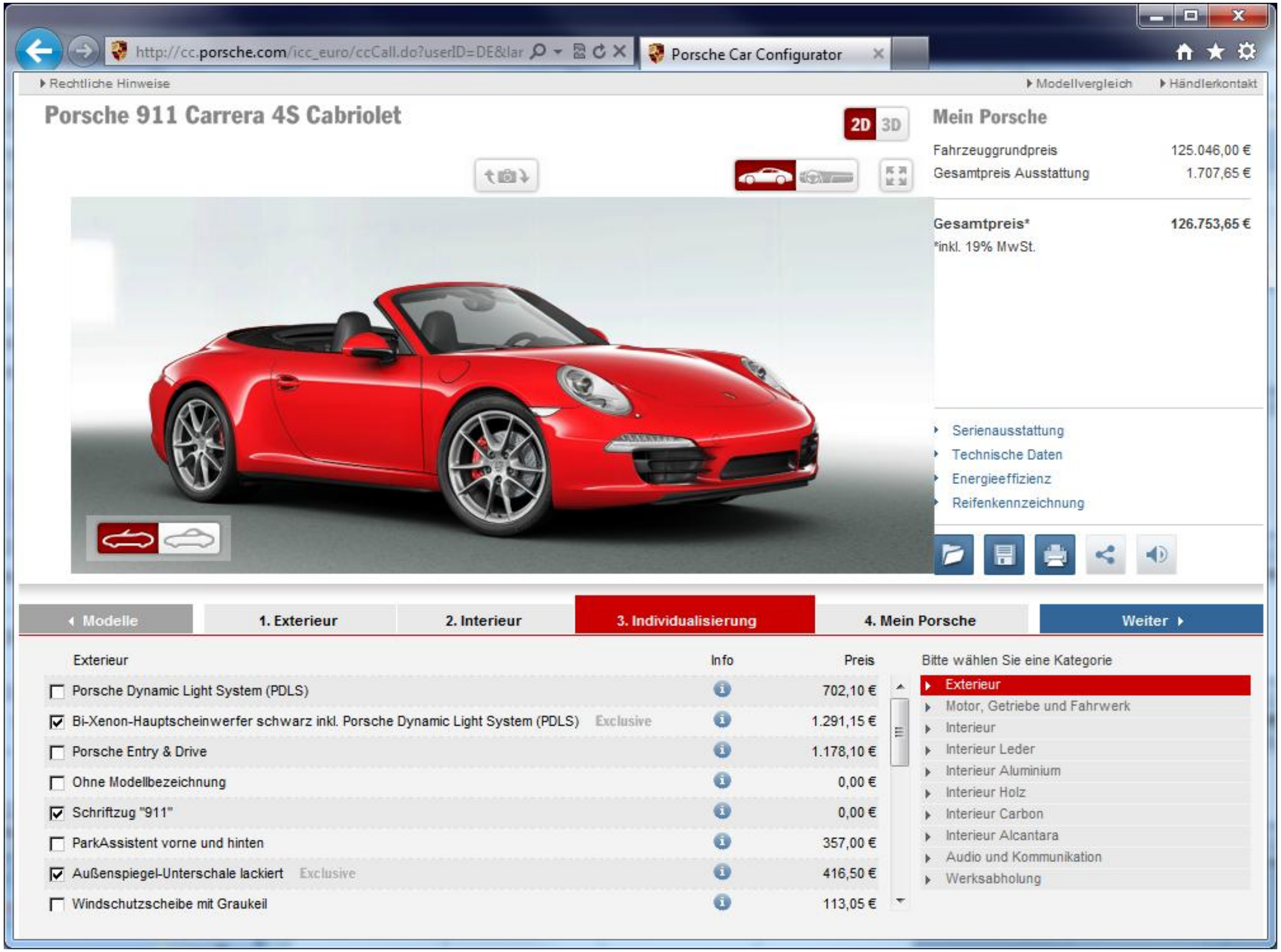
Goal: Managing variability in software

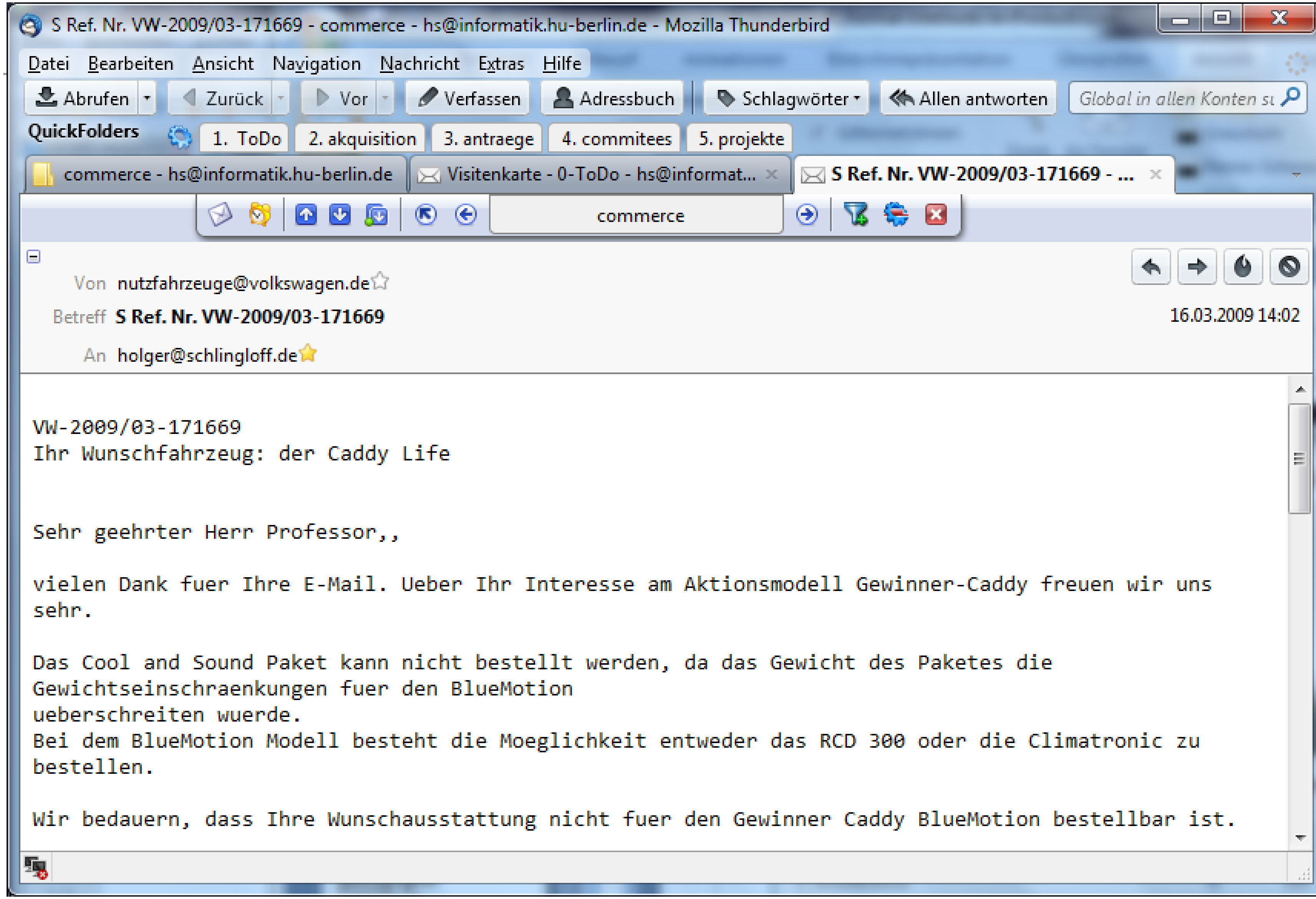
Sources of variability

- planned diversity (“basic/professional/ultimate edition”)
- user alternatives (“for MAC/Win/Linux”)
- evolution (“version 8”)
- re-use (different histories)

CMU SEI: „ A software product line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way“

A Car Configurator




S Ref. Nr. VW-2009/03-171669 - commerce - hs@informatik.hu-berlin.de - Mozilla Thunderbird

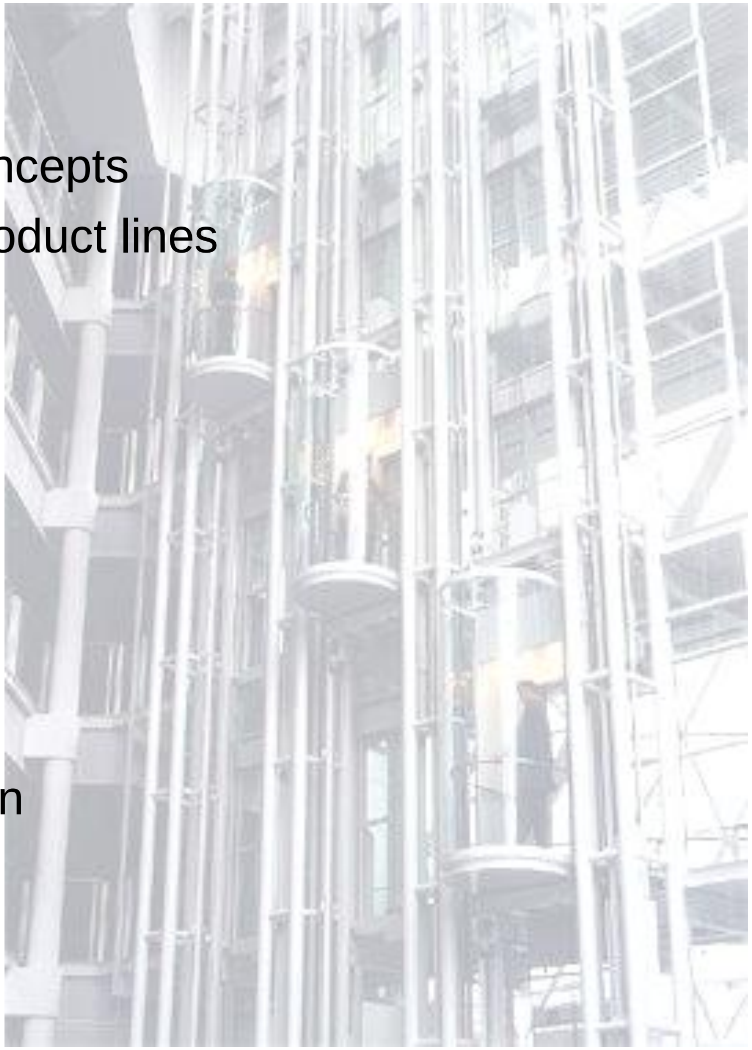
Von nutzfahrzeuge@volkswagen.de

Betreff **S Ref. Nr. VW-2009/03-171669** 16.03.2009 14:02

An holger@schlingloff.de

```
VW-2009/03-171669
Ihr Wunschfahrzeug: der Caddy Life


Sehr geehrter Herr Professor,,

vielen Dank fuer Ihre E-Mail. Ueber Ihr Interesse am Aktionsmodell Gewinner-Caddy freuen wir uns
sehr.

Das Cool and Sound Paket kann nicht bestellt werden, da das Gewicht des Paketes die
Gewichtseinschraenkungen fuer den BlueMotion
ueberschreiten wuerde.
Bei dem BlueMotion Modell besteht die Moeglichkeit entweder das RCD 300 oder die Climatronic zu
bestellen.

Wir bedauern, dass Ihre Wunschausstattung nicht fuer den Gewinner Caddy BlueMotion bestellbar ist.
```

Examples for Software Product Lines



Schindler elevator controls division

- mergers, different devices, new user concepts
- >20 different controllers reduced to 3 product lines

Thales ETCS RBC

- one product (line), highly diverse market
- 700 features, $>10^{100}$ variants

AUTOSAR ECUs

- 1000s of configuration parameters
- feature model for automated configuration

Fraunhofer FOKUS

Features of Product Lines

A *feature* is the description of a designated functionality

– a special added value for the customer
– a characteristics of the product which is interesting for a stakeholder
– e.g., a requirement, a function or function group, or a quality criterion

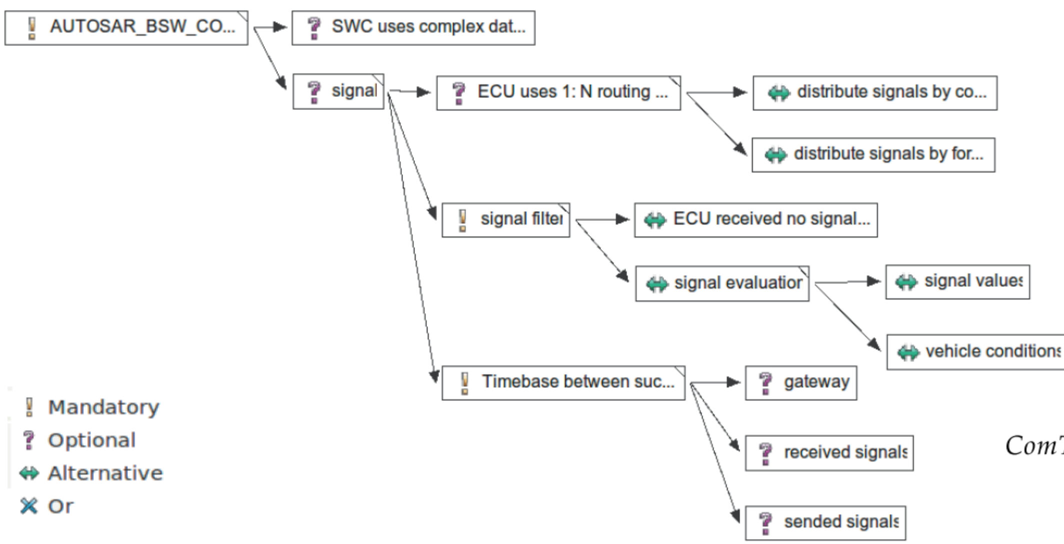
Typically, products are evolved by adding new features

e.g., Galaxy S4 phone with new IR-emitter diode
to work as a remote control for your home TV

Feature Modelling

For multi-variant systems (e.g., product lines), features are represented in a feature model

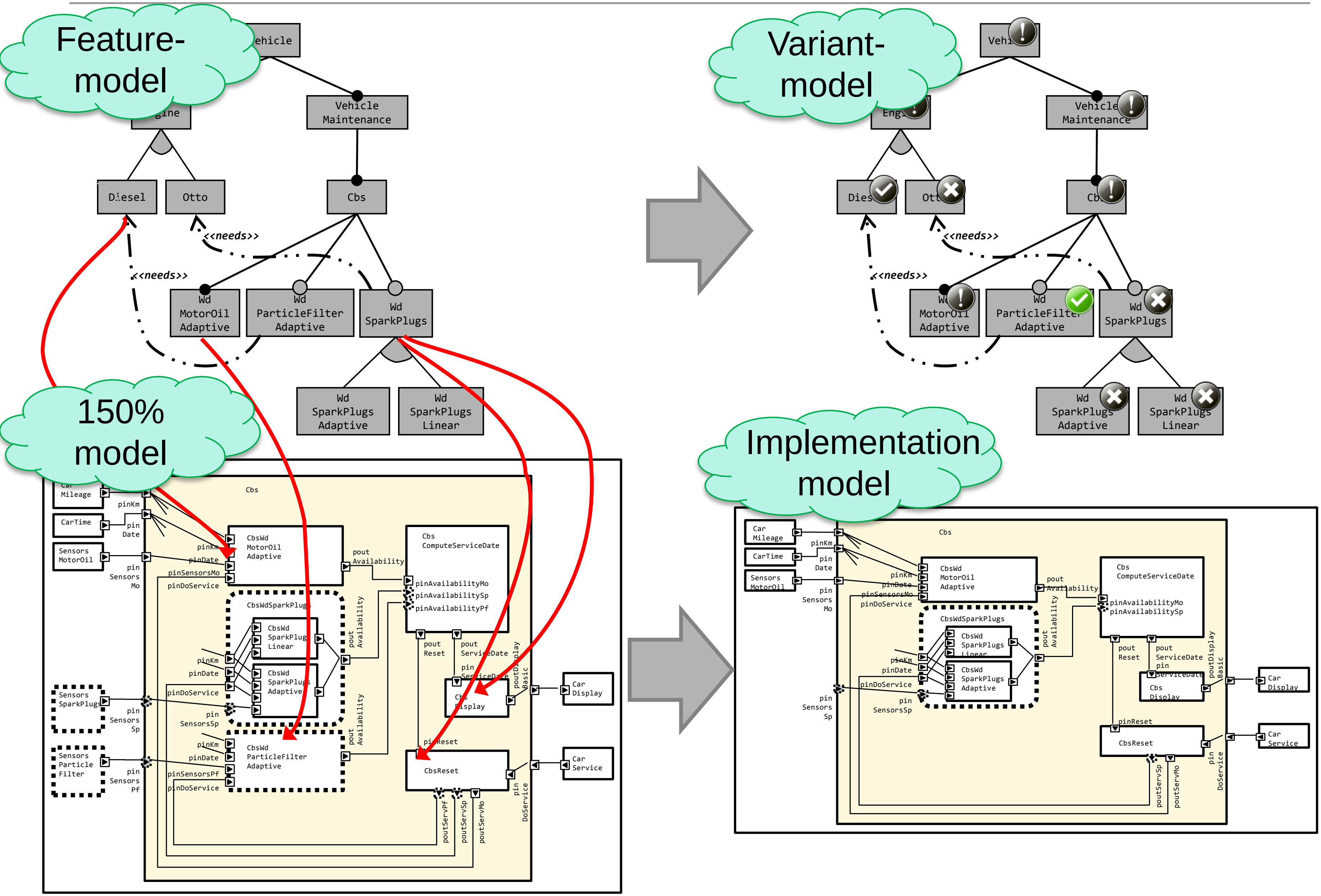
– usually in a tree-like manner (and / or – tree)
– additional constraints (mandatory, optional, includes, excludes)
– nothing more than the graphical representation of a Boolean formula



$$
\begin{aligned}
ComTimeBase \Leftrightarrow & (ComTimeBaseGW \wedge \neg ComTimeBaseGW) \wedge \\
& (ComTimeBaseRX \wedge \neg ComTimeBaseRX) \wedge \\
& (ComTimeBaseTX \wedge \neg ComTimeBaseTX) \wedge \\
& (ComTimeBaseGW \Rightarrow ComGwMapping) \wedge \\
& (ComTimeBaseRX \Rightarrow \neg ComFilterSendOnly)
\end{aligned}
$$

Fraunhofer FOKUS

Instantiation („Materialization“)



Model-Based SPL Testing Strategies

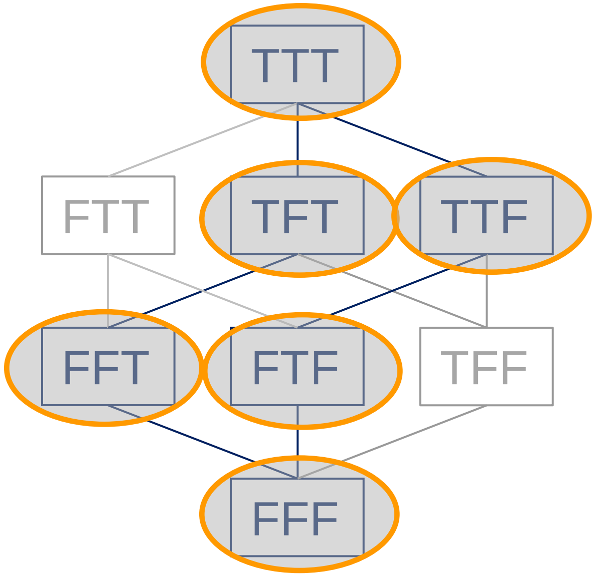
Main problem: which variants to choose? (*n* features ➔ 2^n instances)

➔ The feature model represents a Boolean condition on the set of features

➔ The set of satisfying interpretations forms a partial order

Reasonable choices for variant selection (feature coverage):

- All possible product variants
- All features
- Pairwise
- Minimal and maximal elements
- Selected and unselected
- ...

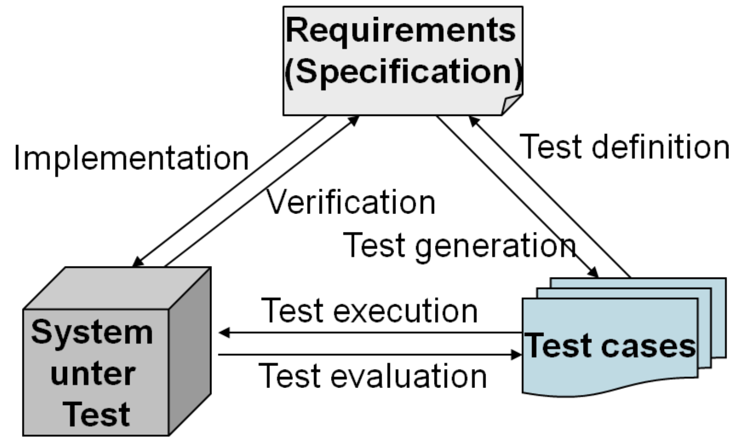


Structure of this Talk

1. Software product lines, feature modeling
2. **Testing software product lines**
3. A testing theory based on CSP-CASL
4. Implementation and applications

Fraunhofer FOKUS

Application Scenario



The **validation triangle**

Test cases are first-class artefacts
(manually created, engineered, ...)

Furthermore, systems are not
designed “from scratch”

Often, both product and test cases pre-exist
and make good money

Then, the product is enhanced by new features

And the regression tests fail

Q.: Is this due to a bug in the product (line) or due to a bad test case?

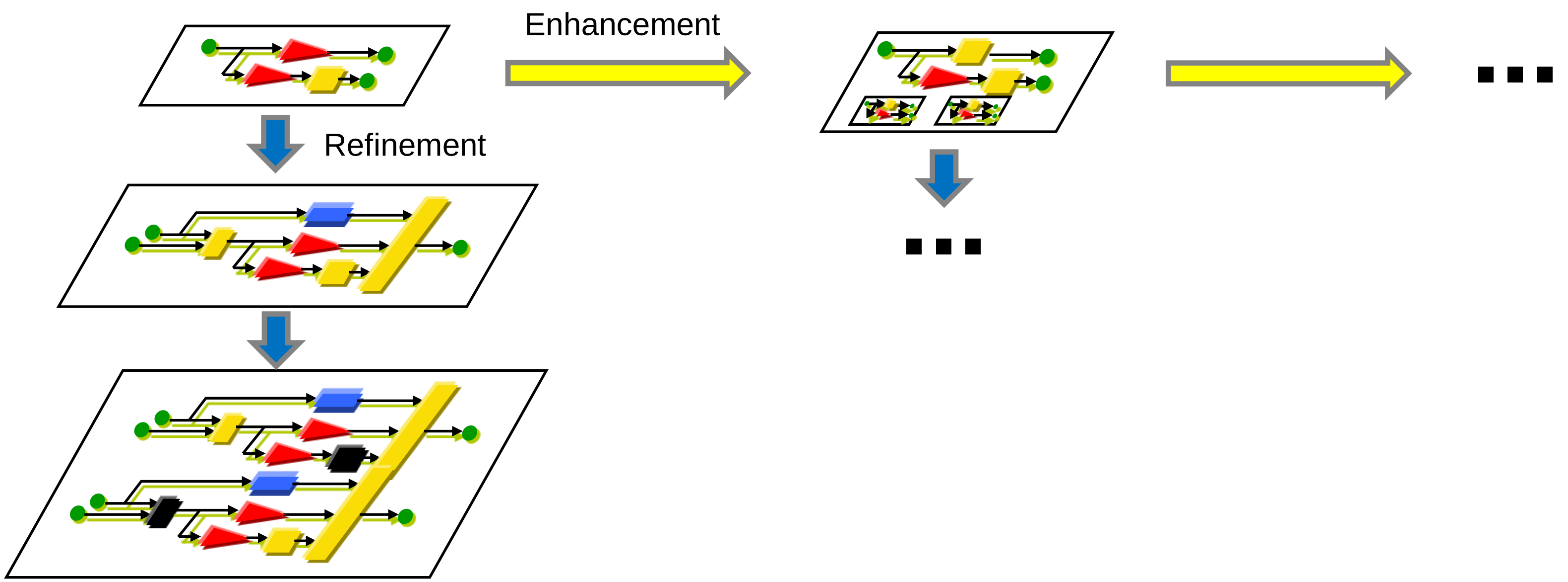
Vertical and Horizontal Systems Development

There are many systems development models.

Refinement: means vertical development (top-down)

Product lines are almost never conceived in one go.

Enhancement means horizontal development (left-right)



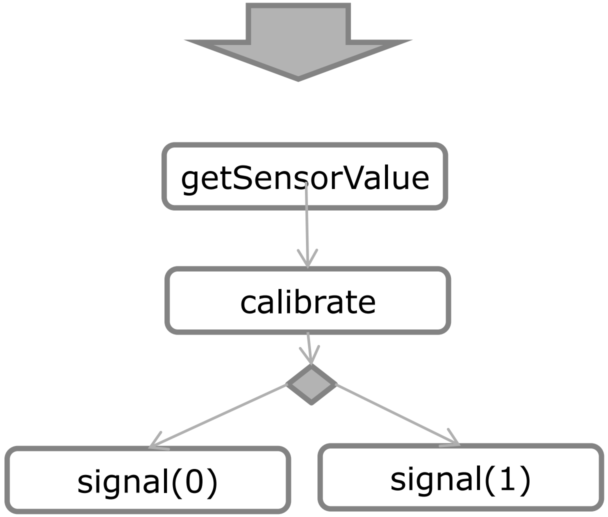
Abstraction and Nondeterminism

Task: Given a (150%-) model and a test suite,
characterize those test cases which should pass.

```
m:=getSensorValue (x);
calibrate(m);
if (m>17) signal(0)
else signal(1)
```

What does “should pass” mean?

(Model used as a test oracle)



In contrast to concrete implementations,
models are abstract

and, therefore, in general nondeterministic

Test outcome may not be decided yet

➔ Test result “undecided” at this stage

Three-Valued Test Oracles

Task: Given a (150%-) model and a test suite,
characterize those test cases which should pass.

What does “should pass” mean?

(Model used as a test oracle)

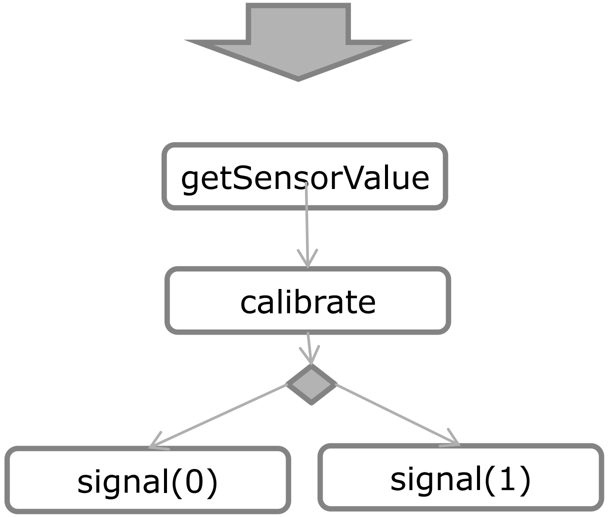
Usual test generation algorithms only produce tests that “should pass”

How to test that “it must not happen”?

Test case may check for unwanted behavior

➔ Test result “expected fail”

```
m:=getSensorValue (x);
calibrate(m);
if (m>17) signal(0)
else signal(1)
```



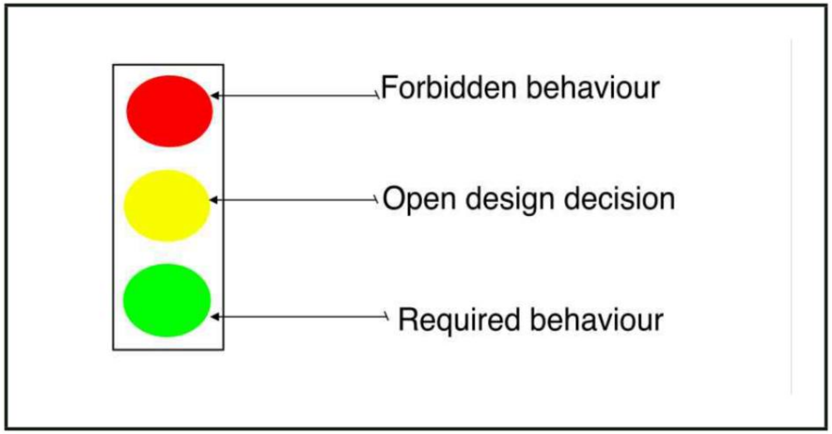
Fraunhofer FOKUS

Defining Expected Results

A test case T is a sequence or tree of i/o-events, potentially with variables

We define the *colour of a test with respect to a specification*

$$colour(T, SP) \in \{\mathrm{green}, \mathrm{yellow}, \mathrm{red}\}$$



A test case T is **green** if

- all of its traces are possible system runs according to the specification,
- whose execution can’t be refused

T is **red**, if

- not all of its traces are possible system runs

T is **yellow**, otherwise

Structure of this Talk

1. Software product lines, feature modeling
2. Testing software product lines
3. **A testing theory based on CSP-CASL**
4. Implementation and applications

Example: A Product Line of Remote Control Units

Fraunhofer FOKUS

An Abstract RCU

ccspec ABSRCU =
data
  sort $Button, Signal$
  op $codeOf : Button \rightarrow Signal;$
process
  $\text{AbsRCU} = ?x : Button \rightarrow codeOf(x) \rightarrow AbsRCU$
end

$A_0 : u : Button \rightarrow codeOf(u) \rightarrow Stop$

$A_1 : u : Button \rightarrow v : Signal \rightarrow Stop$

$A_2 : u : Button \rightarrow w : Button \rightarrow Stop$

	A_0	A_1	A_2
AbsRCU	*G*	*Y*	*R*

Fraunhofer FOKUS

A Concretization of AbsRCU (Refinement)

ccspec BRCU =
data
  sort $Button, Signal$
  ops $b_0, b_1, \ldots, b_9, b_{OnOff}, b_{Mute} : Button;$
  free type Bit ::=0 | 1
  then LIST[**sort** Bit]
    then
      sort $Signal = \{l : List[Bit] \bullet \#l = 16\}$
      op $codeOf : Button \rightarrow Signal;$
        $prefix : List[Bit] = [0000] + +[01010]$
      axioms
        $codeOf(b_0) = prefix ++[0000000];$
        $\ldots$
        $codeOf(b_9) = prefix ++[0001001];$
        $codeOf(b_{Mute}) = prefix ++[0001111];$
        $codeOf(b_{OnOff}) = prefix ++[1111111];$
        $\forall b : Button \bullet \ \exists l : List[Bit] \bullet$
        $codeOf(b) = prefix \ ++l$
process
  BRCU $= ?x : Button \rightarrow codeOf(x) \rightarrow \ BRCU$
end

$A_0 : u : Button \rightarrow codeOf(u) \rightarrow Stop$
$A_1 : u : Button \rightarrow v : Signal \rightarrow Stop$
$A_2 : u : Button \rightarrow w : Button \rightarrow Stop$

	A_0	A_1	A_2
$AbsRCU$	G	Y	R

$T_0 : Stop$
$T_1 : b_1 \rightarrow Stop$
$T_2 : b_1 \rightarrow codeOf(b_1) \rightarrow b_6 \rightarrow codeOf(b_6) \rightarrow Stop$
$T_3 : b_1 \rightarrow b_6 \rightarrow Stop$
$T_4 : b_0 \rightarrow (prefix ++[0000101]) \rightarrow Stop$

	T_0	T_1	T_2	T_3	T_4
$BRCU$	G	G	G	R	R

Extending the Abstract RCU (Data Enhancement)

ccspec ABSERCU =
data
  sorts $Button < EButton;\ Signal$
  op $codeOf : Button \to Signal;$
    $codeOf : EButton \to Signal;$
process
  $\mathrm{AbsERCU} = ?x : EButton \to codeOf(x) \to AbsERCU$
end

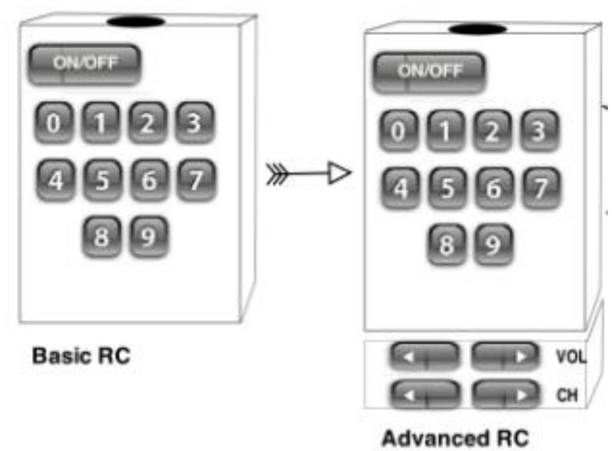
$$A_0 : u : Button \to codeOf(u) \to Stop$$
$$A_1 : u : Button \to v : Signal \to Stop$$
$$A_2 : u : Button \to w : Button \to Stop$$

	A_0	A_1	A_2
$AbsRCU$	G	Y	R

NOT a refinement:

Signature is extended by supersort EButton;

operation codeOf is extended by overloading



Are the test verdicts still valid for this modified specification?

Modifier Button (Behavioural Enhancement)

ccspec MERCU =
data ERCUDATA **then**
  free type $MButton ::=$ **sort** $EButton \mid b_{alt}$
  sort $AltButton = \{x : MButton \bullet x = alt\}$
  op $codeOfAlt : EButton \to Signal$
process
  MERCU =
    $?x : EButton \to codeOf(x) \to MERCU$
    $\Box\ b_{Alt} \to ?x : EButton \to codeOfAlt(x)$
      $\to MERCU$
end

If arbitrary changes to data and process specification are allowed in the enhancement process, preservation of features cannot be guaranteed.

What are “sensible” restrictions on the evolution of software?

Fraunhofer FOKUS

Enhancement in CSP-CASL

Principle:

- Add "more data symbols"
- add "more behaviours"
- however, keep the "old" system

$$(D, P) \gg (D', P')$$

Data enhancement:

Let $\iota : \Sigma_D \to \Sigma_{D'}$ be a signature embedding.

1. $\mathbf{Mod}(D) = \mathbf{Mod}(D')|_\iota$

Process enhancement:

Let $\alpha : A \to A'$ be an injective map on event sets.

- Define reducts $traces(P')|_\alpha$, $failures(P)|_\alpha$.

2. for all $M' \in \mathbf{Mod}(D')$: $traces(P) = traces(P')|_\alpha$ and
3. for all $M' \in \mathbf{Mod}(D')$: $failures(P) = failures(P')|_\alpha$.

Data Enhancement

Data enhancement can be characterized in terms of a supersort.

Given a map $extend : S \to S'$ on sort symbols

If

1. Σ is embedded into Σ' with the mapping $extend$,
2. $\mathbf{Mod}(D) = \mathbf{Mod}(D')\,|_{\iota}$, and
3. $P' = extend(P)$,

then $Sp \gg Sp'$

Behavioral Enhancement

A possible pattern for behavioral enhancement is in terms of external choice

Let

$$\begin{array}{ll} Sp: & P = \ ?x : s \to P' \\ Sp': & P = \ ?x : s \to P' \ \ \Box \ ?y : t' \to Q' \end{array}$$

where

- Σ *is embedded into* Σ', $\boldsymbol{Mod}(D) = \boldsymbol{Mod}(D')|_\iota$, *and*
- *for all* $u \in \Sigma$ *it holds that* $D' \models \forall x : u \forall y : t' .\ x \neq y$.

Then $\quad Sp \gg Sp'$

Re-Use of Test Cases

Theorem:

Let Sp and Sp' be CSP-CASL specifications.

Let $Sp \gg Sp'$.

Let T be a test process over Sp.

Then

$$colour_{Sp}(T) = colour_{Sp'}(T).$$

That is, if the enhancement is done in a “controlled” way,
then test cases can be reused in the product line development.

Extending Test Suites

ccspec BRCU =
data
  sort $Button, Signal$
  ops $b_0, b_1, \ldots, b_9, b_{OnOff}, b_{Mute} : Button;$
  then LIST[**sort** Bit]
    then
      sort $Signal = \{l : List[Bit] \bullet \#l = 16\}$
      op $codeOf : Button \rightarrow Signal;$
        $prefix : List[Bit] = [0000] + +[01010]$
      axioms
        $codeOf(b_0) = prefix + +[0000000];$
        $\ldots$
process
  BRCU $= ?x : Button \rightarrow codeOf(x) \rightarrow BRCU$
end

$T_0 : Stop$
$T_1 : b_1 \longrightarrow Stop$
$T_2 : b_1 \longrightarrow codeOf(b_1) \longrightarrow b_6 \longrightarrow codeOf(b_6) \longrightarrow Stop$
$T_3 : b_1 \longrightarrow b_6 \longrightarrow Stop$
$T_4 : b_0 \longrightarrow (prefix + +[0000101]) \longrightarrow Stop$

	T_0	T_1	T_2	T_3	T_4
$BRCU$	G	G	G	R	R

ccspec ERCU =
data BRCUDATA **then**
  free type $EButton ::=$ **sort** $Button$
    $\mid b_{volup} \mid b_{voldn}$
    $\mid b_{chup} \mid b_{chdn}$
  op $codeOf : EButton \rightarrow Signal$
  axioms
    $codeOf(b_{volup}) = prefix + +[0010000];$
    $codeOf(b_{voldn}) = prefix + +[0100000];$
    $codeOf(b_{chup}) = prefix + +[1000000];$
    $codeOf(b_{chdn}) = prefix + +[1100000];$
process
  ERCU $= ?x : EButton \rightarrow codeOf(x) \rightarrow ERCU$
end

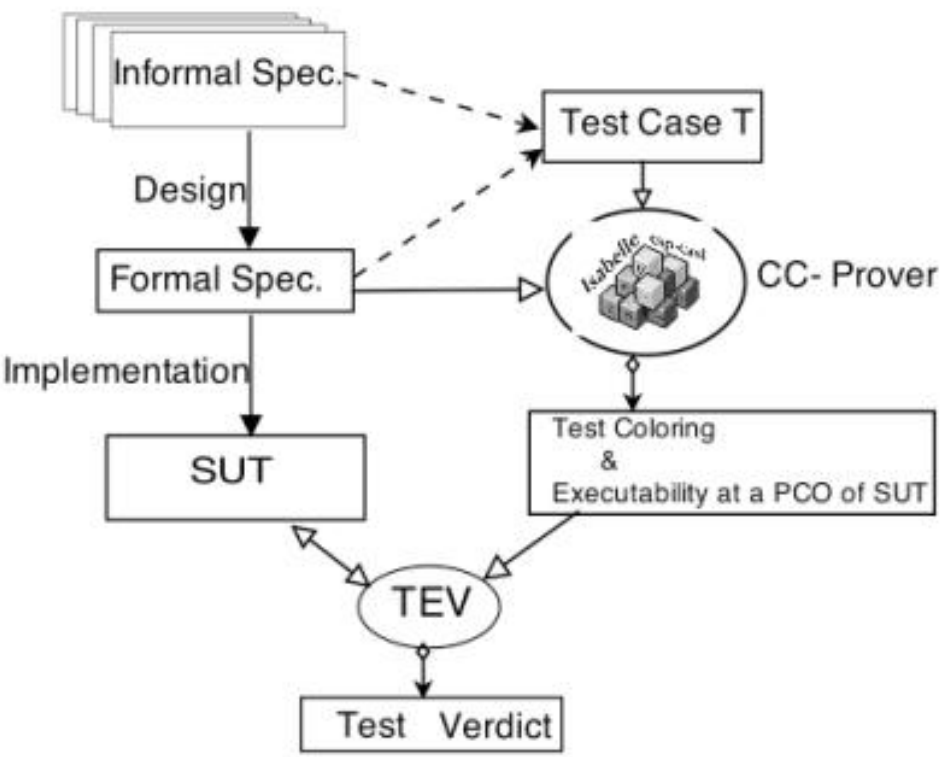
$T_5 : b_1 \longrightarrow codeOf(b_1) \longrightarrow b_{VolUp} \longrightarrow codeOf(b_{VolUp}) \longrightarrow Stop$
$T_6 : b_{ChUp} \longrightarrow codeOf(b_{ChUp}) \longrightarrow Stop$
$T_7 : b_{ChDn} \longrightarrow codeOf(b_{ChDn}) \longrightarrow b_1 \longrightarrow Stop$
$T_8 : b_{ChUp} \longrightarrow b_{VolDn} \longrightarrow Stop$
$T_9 : b_{ChUp} \longrightarrow codeOf(b_{VolUp}) \longrightarrow Stop$

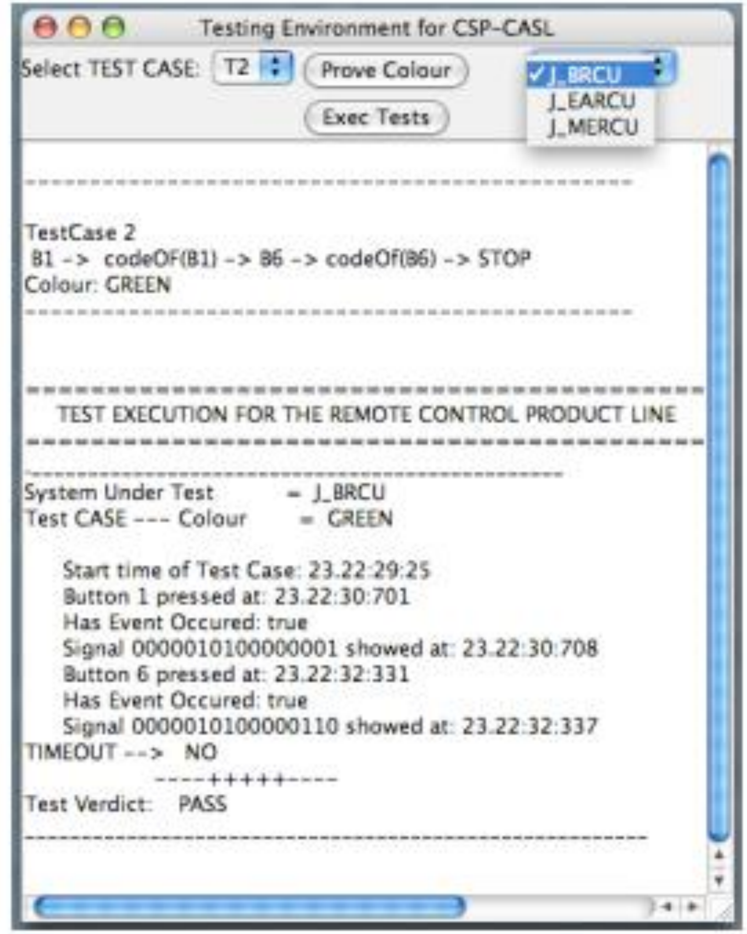
	T_5	T_6	T_7	T_8	T_9
$ERCU$	G	G	G	R	R

Structure of this Talk

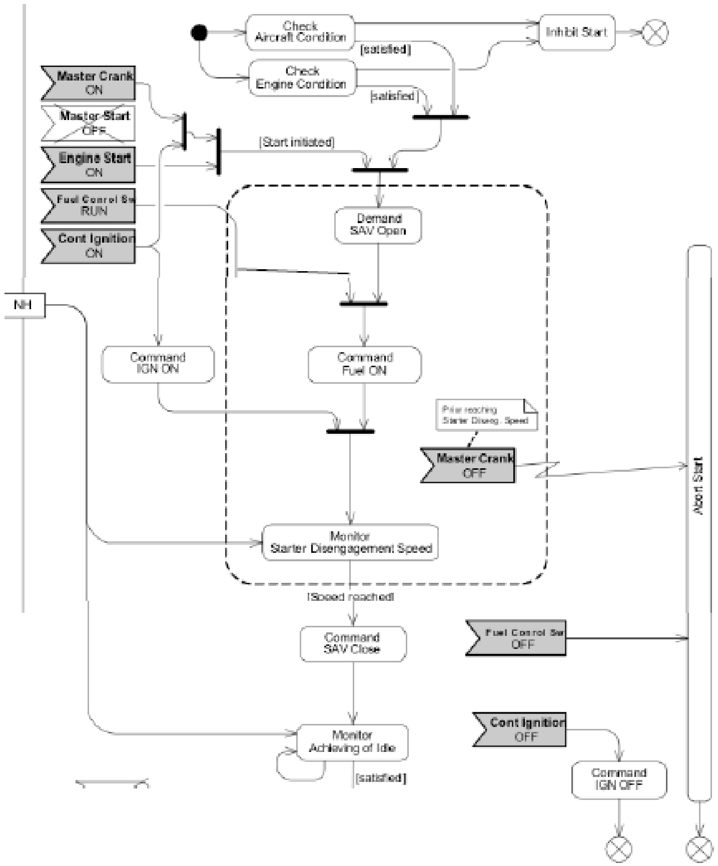
1. Software product lines, feature modeling
2. Testing software product lines
3. A testing theory based on CSP-CASL
4. **Implementation and applications**

Workflow and Tool





Outside the Ivory Tower




```
channel press, release
ButtonOFF = press -> ButtonON
ButtonON = release -> ButtonOFF

channel mc_press, mc_release
MasterCrank = ButtonOFF[press <- mc_press
                        release <- mc_release]

Buttons =  MasterCrank
        ||| MasterStart
        ||| EngineStartON

InteractEEC= (CheckConditions
               [|{synchStart}|]
             CrankAndIgnite)
              \ {synchStart}
            ; FuelAndSAV
            ; MasterIdle

MGS_Core = InteractEEC || Buttons

MGS_ErrorHandling = MGS_Core  /\  InterruptFC
```


Fraunhofer FOKUS

Bringing It Onto a Test Rig

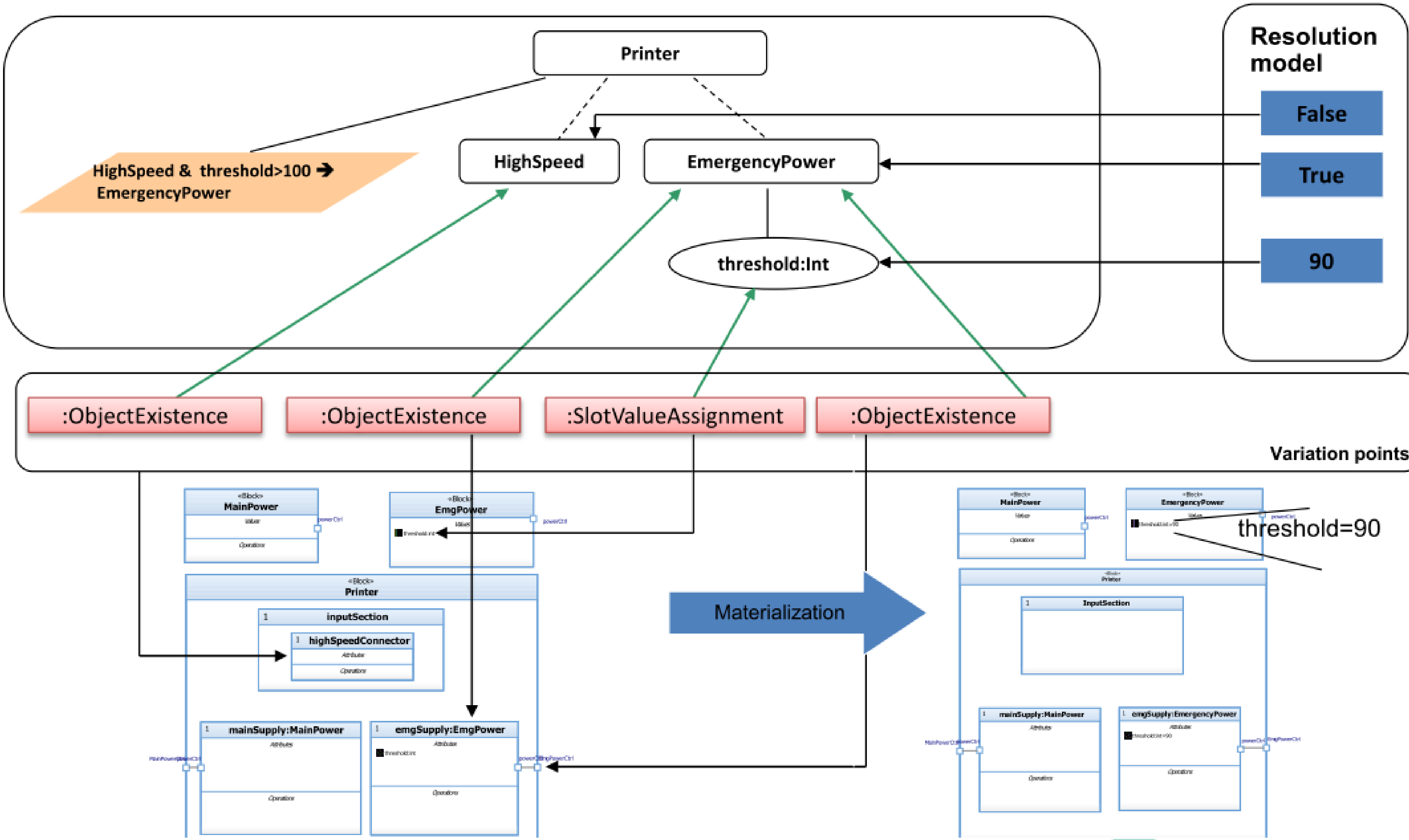

```
Set("MasterCrankCnd",1)
WaitTime(2)
Set("REngContinuousIGN",1)
WaitTime(2)
Set("REngStartCnd",1)
WaitTime(2)
Set("REngStartCnd",0)
WaitUntil("NHP>15")
Set("MasterLever",0)
WaitUntil("LIT==1")
Set("FlightStatus",1)
WaitUntil("NHP>65",60)
...
```

Simulation and verification of the model (FDR)

Manual test case definition, automated coloring

Automated execution and evaluation

UML CVL (Common Variability Language)



Fraunhofer FOKUS

Variability in CVL

CVL defines different kinds of variation points

- Object existence

  i.e., some model element is deleted or inserted

- Value assignment

  i.e., a variable is assigned a value

- Substitution

  i.e., one model fragment is replaced by another one

- Opaque variation point

  i.e., an arbitrary model transformation is applied

Even uncontrolled object existence can cause problems

Needed: rules (profile) for variation points

Structure of this Talk

1. Software product lines, feature modeling
2. Testing software product lines
3. A testing theory based on CSP-CASL
4. Implementation and applications

Summary, Remarks and Open Issues

- Software product lines, feature modeling,

- Theory of test case rating based on process algebraic specifications
- Preservation of test ratings under certain refinements and enhancements

- Work in progress (with interruptions)
- Semantical theory, some syntactic results

- Currently working on UML as a modeling language

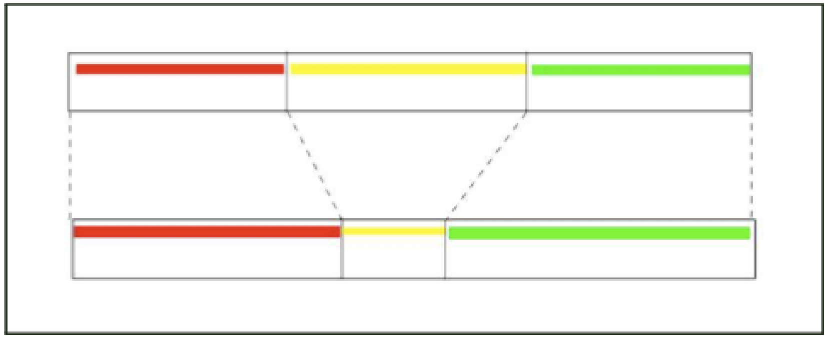
Thank you for your attention!

Backup Slides

Refinement and Testing

A refinement relation $\rightsquigarrow$ is called well-behaved (w-b) if – given $(Sp, P) \rightsquigarrow (Sp', P')$ for consistent Sp and Sp' – for all tests T:

1. $colour(T) = green$ with respect to (Sp, P) implies $colour(T) = green$ with respect to (Sp', P'), and
2. $colour(T) = red$ with respect to (Sp, P) implies $colour(T) = red$ with respect to (Sp', P').



Fraunhofer FOKUS

Mode Buttton

Re-use of specification modules can be handled by CC’s **and-** and **let-**concepts

$$
\begin{array}{lllll}
\textbf{ccspec}\ \mathrm{URCU} = & & & & \\
\textbf{data} & \{\ \mathrm{ERCUD{\scriptsize ATA}}\ \textbf{and}\ \mathrm{ERCUD{\scriptsize ATA}DVD} & & & \\
 & \qquad\qquad\quad \textbf{and}\ \mathrm{ERCUD{\scriptsize ATA}VCR}\ \ \} & & & \\
 & \textbf{then}\ \ \textbf{sort}\ NewButton & & & \\
 & \qquad\quad\ \textbf{op}\ \ \ mode : NewButton & & & \\
\textbf{process} & & & & \\
\quad \textbf{let}\ \mathrm{TV} & =?x : Button \rightarrow \mathit{codeOf}(x) \rightarrow \mathrm{TV} & & & \\
 & \ \ \square\ mode \rightarrow \mathrm{DVD} & & & \\
\quad\quad\ \ \mathrm{DVD} & =?x : Button \rightarrow \mathit{codeOfDVD}(x) \rightarrow \mathrm{DVD} & & & \\
 & \ \ \square\ mode \rightarrow \mathrm{VCR} & & & \\
\quad\quad\ \ \mathrm{VCR} & =?x : Button \rightarrow \mathit{codeOfVR}(x) \rightarrow \mathrm{VCR} & & & \\
 & \ \ \square\ mode \rightarrow \mathrm{TV} & & & \\
\quad \textbf{in}\ \ \mathrm{TV}\ \ \textbf{end} & & & &
\end{array}
$$