# Do We Still Need This?
# Managing Variability in Modern Software Systems

Jacob Krüger

Eindhoven University of Technology, The Netherlands

June 24, 2024

# Modern systems: driven by software and variant-rich

Figures: https://github.com/SoftVarE-Group/Course-on-Software-Product-Lines

# Modern cars: driven by software and variant-rich

Figures: https://github.com/SoftVarE-Group/Course-on-Software-Product-Lines

# Modern cars: driven by software and variant-rich

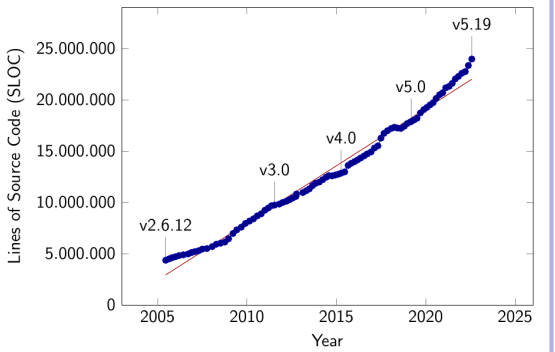
Figures: https://github.com/SoftVarE-Group/Course-on-Software-Product-Lines



TU/e

# Modern Linux: driven by software and variant-rich

## Size of the Code Base



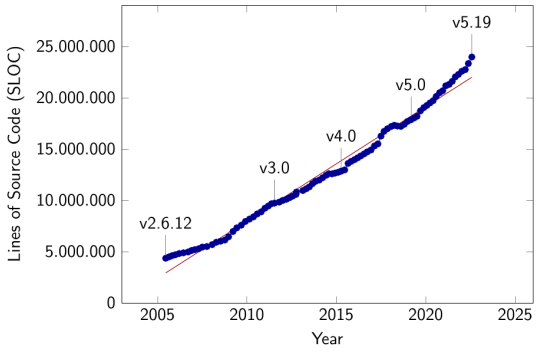

Figures: `https://github.com/SoftVarE-Group/Course-on-Software-Product-Lines`



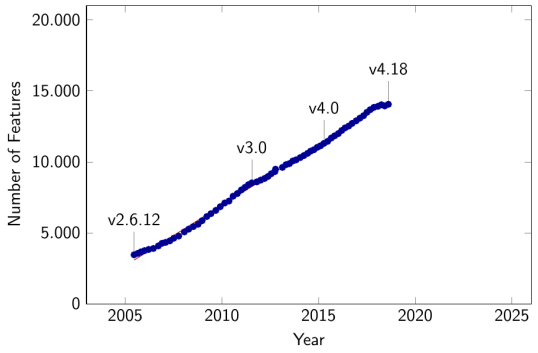
TU/e

# Modern Linux: driven by software and variant-rich

## Size of the Code Base




## Number of Features




Figures: https://github.com/SoftVarE-Group/Course-on-Software-Product-Lines



TU/e

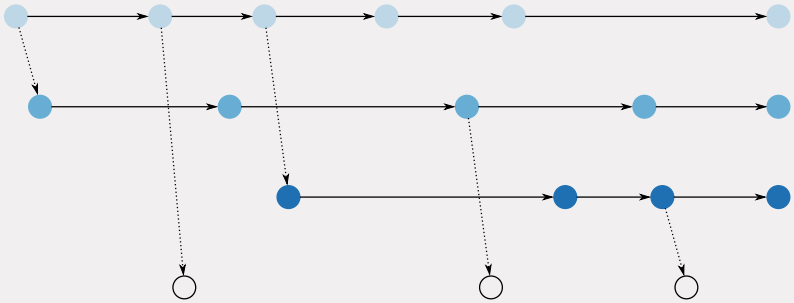
# But organizations often start with one system and cloning



**Krüger**: *Understanding the Re-Engineering of Variant-Rich Systems: An Empirical Work on Economics, Knowledge, Traceability, and Practices*. Dissertation, 2021

## But organizations often start with one system and cloning

- A successful system gets adapted by cloning



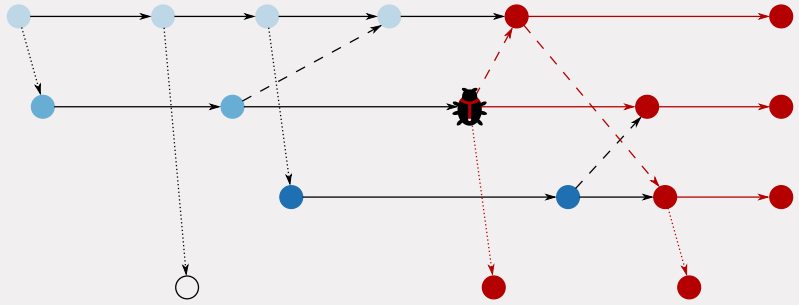
**Krüger**: *Understanding the Re-Engineering of Variant-Rich Systems: An Empirical Work on Economics, Knowledge, Traceability, and Practices*. Dissertation, 2021



TU/e

## But organizations often start with one system and cloning

- A successful system gets adapted by cloning
- Maintenance challenges lead to decision to re-engineer a platform



**Krüger**: *Understanding the Re-Engineering of Variant-Rich Systems: An Empirical Work on Economics, Knowledge, Traceability, and Practices*. Dissertation, 2021



TU/e

# Re-engineering into or between platforms is (still) common


## Getting Rid of Clone-And-Own: Moving to a Software Product Line for Temperature Monitoring

Elias Kuiter
Otto-von-Guericke-University
Magdeburg, Germany
kuiter@ovgu.de

Jacob Krüger
Harz University of Applied Sciences
Otto-von-Guericke-University
Wernigerode & Magdeburg, Germany
jkrueger@ovgu.de

Sebastian Krieter
Harz University of Applied Sciences
Otto-von-Guericke-University
Wernigerode & Magdeburg, Germany
skrieter@hs-harz.de

Thomas Leich
Harz University of Applied Sciences
METOP GmbH
Wernigerode & Magdeburg, Germany
tleich@hs-harz.de

Gunter Saake
Otto-von-Guericke-University
Magdeburg, Germany
gunter.saake@ovgu.de

**ABSTRACT**

Due to its fast and simple applicability, clone-and-own is widely used in industry to develop software variants. In cooperation with different companies for thermoelectric products, we implemented multiple variants of a heat monitoring tool based on clone-and-own. After encountering redundancy-related problems during development and maintenance, we decided to migrate towards a software product line. Within this paper, we describe this case study of migrating cloned variants to a software product line based on the extractive approach. The resulting software product line encapsulates variability on several levels, including the underlying hardware systems, interfaces, and use cases. Currently, we support monitoring hardware from three different companies that use the same core system and provide a configurable front-end. We share our experiences and encountered problems with cloning and migration towards a software product line—focusing on feature extraction and modeling in particular. Furthermore, we provide a lightweight, web-based tool for modeling, configuring, and implementing software product

**ACM Reference Format:**
Elias Kuiter, Jacob Krüger, Sebastian Krieter, Thomas Leich, and Gunter Saake. 2018. Getting Rid of Clone-And-Own: Moving to a Software Product Line for Temperature Monitoring. In *SPLC '18: 22nd International Systems and Software Product Line Conference, September 10–14, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3233027.3233050

### 1 INTRODUCTION

Software product lines are a systematic approach to reuse and manage software artifacts [2, 36]. These artifacts correspond to *features* – user-visible functionalities of a set of variants – that are modeled within variability models [15, 42] to define their dependencies. A selection of features that fulfills all these dependencies is a *valid* configuration. Based on such a configuration, a tool can automatically instantiate a variant from the implemented artifacts.

Using software product lines promises several benefits, for instance, reduced costs for development and maintenance, faster

TU/e

# Re-engineering into or between platforms is (still) common


## Getting Rid of Clone-And-Own: Moving to a Software Product Line for Temperature Monitoring

Elias Kuiter
Otto-von-Guericke-University
Magdeburg, Germany
kuiter@ovgu.de

Tho
Harz Universit
ME
Wernigerode &
tleich

**ABSTRACT**

Due to its fast and simple applicability, cl
used in industry to develop software variat
different companies for thermoelectric pro
multiple variants of a heat monitoring tool
After encountering redundancy-related pr
ment and maintenance, we decided to mig
product line. Within this paper, we describ
grating cloned variants to a software prod
tractive approach. The resulting software p
variability on several levels, including the u
tems, interfaces, and use cases. Currently,
hardware from three different companies th
tem and provide a configurable front-end. W
and encountered problems with cloning a
software product line—focusing on feature
in particular. Furthermore, we provide a
tool for modeling, configuring, and implem

## Decision Making for Managing Automotive Platforms: An Interview Survey on the State-of-Practice

Philipp Zellmer
philipp.zellmer2@volkswagen.de
Volkswagen AG & Harz University
Wolfsburg & Wernigerode, Germany

Jacob Krüger
j.kruger@tue.nl
Eindhoven University of Technology
Eindhoven, The Netherlands

Thomas Leich
tleich@hs-harz.de
Harz University
Wernigerode, Germany

**ABSTRACT**

The automotive industry is changing due to digitization, a growing focus on software, and the increasing use of electronic control units. Consequently, automotive engineering is shifting from hardware-focused towards software-focused platform concepts to address these challenges. This shift includes adopting and integrating methods like electrics/electronics platforms, software product-line engineering, and product generation. Although these concepts are well-known in their respective research fields and different industries, there is limited research on their practical effectiveness and issues—particularly when implementing and using these concepts for modern automotive platforms. The lack of research and practical experiences challenges particularly decision makers, who cannot build on reliable evidence or techniques. In this paper, we address this gap by reporting on the state-of-practice

**ACM Reference Format:**
Philipp Zellmer, Jacob Krüger, and Thomas Leich. 2024. Decision Making for Managing Automotive Platforms: An Interview Survey on the State-of-Practice. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion '24), July 15–19, 2024, Porto de Galinhas, Brazil.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3663529.3663851

### 1 INTRODUCTION

To remain competitive, automotive manufacturers must continuously enhance their product portfolios by incorporating novel features into their vehicles. Traditionally, the focus was on hardware components, but technological advancements, new customer preferences, and legal standards demand the integration of a rising number of software features into the existing hardware platforms [5, 9]. This shift is evident from the prevalence of software

TU/e

# Re-engineering into or between platforms is (still) common


## Getting Rid of Clone-And-Own: Moving to a Software Product Line for Temperature Monitoring

Elias Kuiter
Otto-von-Guericke-University
Magdeburg, Germany
kuiter@ovgu.de

Tho
Harz Universit
ME
Wernigerode &
tleich

**ABSTRACT**

Due to its fast and simple applicability, cl
used in industry to develop software varia
different companies for thermoelectric pro
multiple variants of a heat monitoring tool
After encountering redundancy-related pr
ment and maintenance, we decided to mig
product line. Within this paper, we describ
grating cloned variants to a software prod
tractive approach. The resulting software p
variability on several levels, including the u
tems, interfaces, and use cases. Currently,
hardware from three different companies th
tem and provide a configurable front-end. W
and encountered problems with cloning a
software product line—focusing on feature
in particular. Furthermore, we provide a
tool for modeling, configuring, and implem

## Decision Making for Managing Automotive Platforms: An Interview Survey on the State-of-Practice

Philipp Zellmer
philipp.zellmer2@volkswagen.de
Volkswagen AG & Harz Universit
Wolfsburg & Wernigerode, Germa

**ABSTRACT**

The automotive industry is changing due
ing focus on software, and the increasing
trol units. Consequently, automotive engi
hardware-focused towards software-focus
address these challenges. This shift include
ing methods like electrics/electronics platf
line engineering, and product generation.
are well-known in their respective resea
industries, there is limited research on t
ness and issues—particularly when implen
concepts for modern automotive platform
and practical experiences challenges par
ers, who cannot build on reliable evidenc
paper, we address this gap by reporting

## Insights into Transitioning towards Electrics/Electronics Platform Management in the Automotive Industry

Lennart Holsten
lennart.holsten@volkswagen.de
Volkswagen AG & Harz University
Wolfsburg & Wernigerode, Germany

Jacob Krüger
j.kruger@tue.nl
Eindhoven University of Technology
Eindhoven, The Netherlands

Thomas Leich
tleich@hs-harz.de
Harz University
Wernigerode, Germany

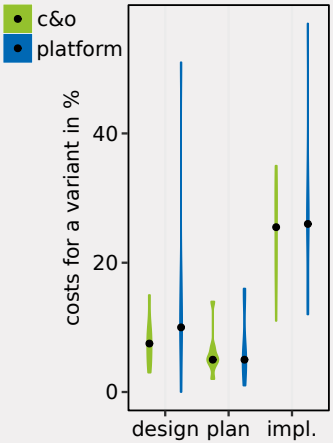
**ABSTRACT**

In the automotive industry, platform strategies have proved effective for streamlining the development of complex, highly variable cyber-physical systems. Particularly software-driven innovations

**1 INTRODUCTION**

Similar to other industries, innovations in the automotive domain are driven more and more by digital features that build on software. The consequent trends emerging in the automotive industry (e.g.,

# Moving to a systematic platform is useful

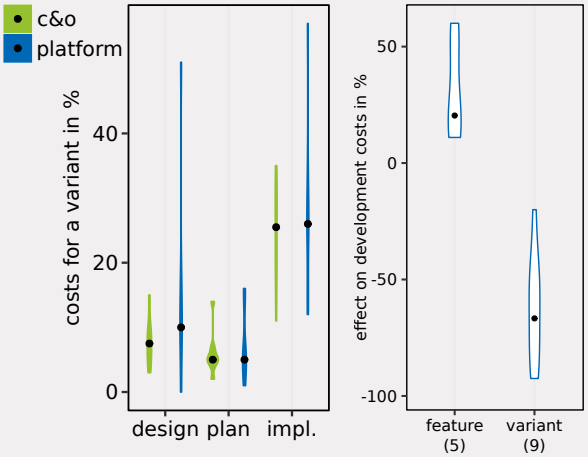



Åkesson, Nilsson, **Krüger**, Berger: *Migrating the Android Apo-Games into an Annotation-Based Software Product Line*. SPLC'19
Debbice, Lignell, **Krüger**, Berger: *Migrating Java-Based Apo-Games into a Composition-Based Software Product Line*. SPLC'19
**Krüger**, Berger: *An Empirical Analysis of the Costs of Clone- and Platform-Oriented Software Reuse*. ESEC/FSE'20
**Krüger**, Berger: *Activities and Costs of Re-Engineering Cloned Variants Into an Integrated Platform*. VaMoS'20



TU/e

# Moving to a systematic platform is useful

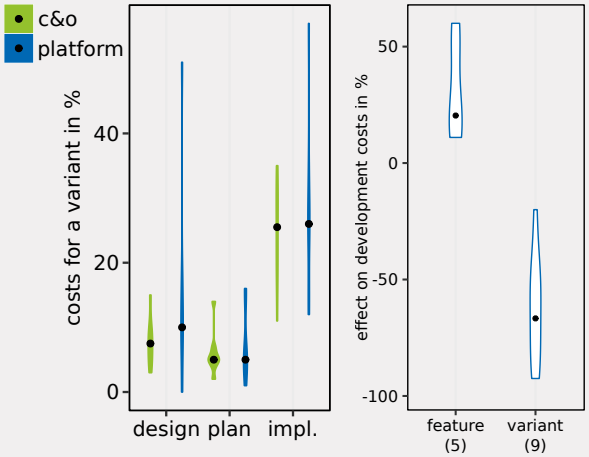





Åkesson, Nilsson, **Krüger**, Berger: *Migrating the Android Apo-Games into an Annotation-Based Software Product Line*. SPLC'19
Debbiche, Lignell, **Krüger**, Berger: *Migrating Java-Based Apo-Games into a Composition-Based Software Product Line*. SPLC'19
**Krüger**, Berger: *An Empirical Analysis of the Costs of Clone- and Platform-Oriented Software Reuse*. ESEC/FSE'20
**Krüger**, Berger: *Activities and Costs of Re-Engineering Cloned Variants Into an Integrated Platform*. VaMoS'20

TU/e

# Moving to a systematic platform is useful (but challenging)




| activity type | team 1 | team 2 |
|---|---|---|
| product-line training | 16.00 | 90.00 |
| domain analysis | 18.00 | 82.00 |
| preparatory analysis | 49.25 | 40.00 |
| feature identification | 22.25 | 22.00 |
| architecture identification | 2.00 | 5.00 |
| feature location | 50.00 | 7.00 |
| feature modeling | 7.00 | 10.00 |
| transformation | 103.50 | 180.00 |
| quality assurance | 103.50 | 60.00 |

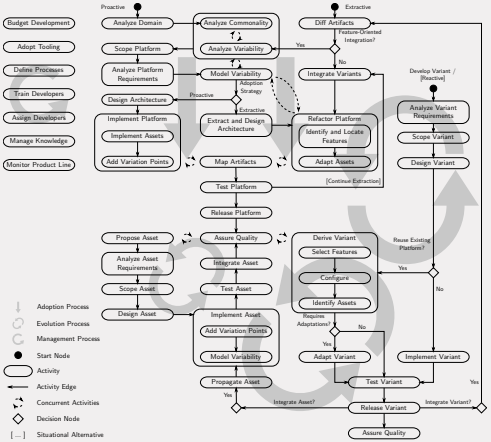


Åkesson, Nilsson, **Krüger**, Berger: *Migrating the Android Apo-Games into an Annotation-Based Software Product Line*. SPLC'19
Debbiche, Lignell, **Krüger**, Berger: *Migrating Java-Based Apo-Games into a Composition-Based Software Product Line*. SPLC'19
**Krüger**, Berger: *An Empirical Analysis of the Costs of Clone- and Platform-Oriented Software Reuse*. ESEC/FSE'20
**Krüger**, Berger: *Activities and Costs of Re-Engineering Cloned Variants Into an Integrated Platform*. VaMoS'20

TU/e

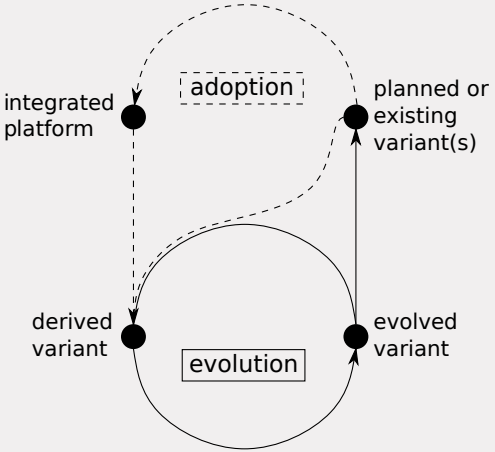
# Recommendations and guides can facilitate re-engineering

# Recommendations and guides can facilitate re-engineering




**Krüger**, Mahmood, Berger: *Promote-pl: A Round-Trip Engineering Process Model for Adopting and Evolving Product Lines*. SPLC'20

# Recommendations and guides can facilitate re-engineering



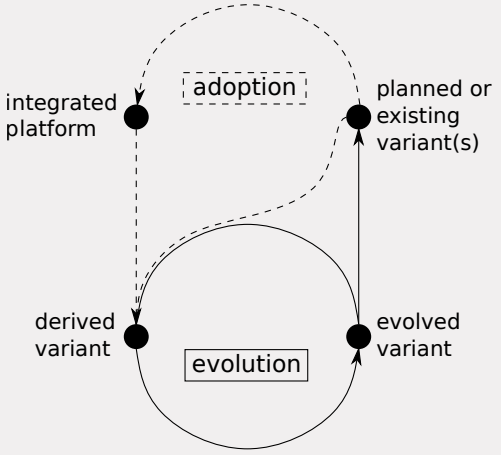

**Krüger**, Mahmood, Berger: *Promote-pl: A Round-Trip Engineering Process Model for Adopting and Evolving Product Lines*. SPLC'20



TU/e

# Recommendations and guides can facilitate re-engineering

- Support for feature modeling
- Process for maturity assessment



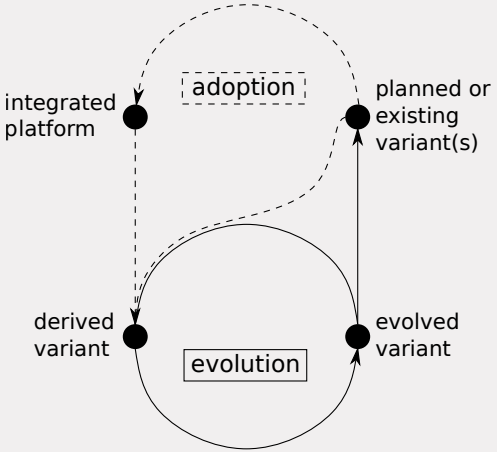



**Krüger**, Mahmood, Berger: *Promote-pl: A Round-Trip Engineering Process Model for Adopting and Evolving Product Lines*. SPLC'20
Lindohf, **Krüger**, Herzog, Berger: *Software Product-Line Evaluation in the Large*. Empirical Software Engineering, 2020
Nešić, **Krüger**, Stănciulescu, Berger: *Principles of Feature Modeling*. ESEC/FSE'19

TU/e

# Recommendations and guides can facilitate re-engineering

- Support for feature modeling
- Process for maturity assessment
- Studies built on industry collaborations/cases
  - VW
  - Axis
  - Saab
  - Danfoss
  - pure-systems
  - ABB
  - ...






**Krüger**, Mahmood, Berger: *Promote-pl: A Round-Trip Engineering Process Model for Adopting and Evolving Product Lines*. SPLC'20
Lindohf, **Krüger**, Herzog, Berger: *Software Product-Line Evaluation in the Large*. Empirical Software Engineering, 2020
Nešić, **Krüger**, Stănciulescu, Berger: *Principles of Feature Modeling*. ESEC/FSE'19

TU/e

## Having a platform? Still, not everything is perfect

- Ensuring program comprehension?
- Assuring software quality?
- Analyzing variability?
- Aligning hardware and software releases?
- Deprecating (variable) features or a platform?
- Re-engineering variability safely?
- …




TU/e

# Comprehension and quality are key

- Insights:
  - Feature traces are helpful …

```
1     char_u  *
2  fix_fname(fname)
3     char_u  *fname;
4  {
5  #ifdef UNIX
6     return FullName_save(fname, TRUE);
7  #else
8     if (!vim_isAbsName(fname)
9             || strstr((char *)fname, "..") != NULL
10            || strstr((char *)fname, "//") != NULL
11 # ifdef BACKSLASH_IN_FILENAME
12            || strstr((char *)fname, "\\\\") != NULL
13 # endif
14 # if defined(MSWIN) || defined(DJGPP)
15            || vim_strchr(fname, '~') != NULL
16 # endif
17            )
18        return FullName_save(fname, FALSE);
19
20    fname = vim_strsave(fname);
21
22 # ifdef USE_FNAME_CASE
23 #  ifdef USE_LONG_FNAME
24    if (USE_LONG_FNAME)
25 #  endif
26    {
27        if (fname != NULL)
28            fname_case(fname, 0);
29    }
30 # endif
31
32    return fname;
33 #endif
34 }
```



Fenske, **Krüger**, Kanyshkova, Schulze: `#ifdef` *Directives and Program Comprehension: The Dilemma between Correctness and Preference*. ICSME'20
**Krüger**, Çalıklı, Berger, Leich, Saake: *Effects of Explicit Feature Traceability on Program Comprehension*. ESEC/FSE'19
Ludwig, **Krüger**, Leich: *Covert and Phantom Features in Annotations: Do They Impact Variability Analysis?*. SPLC'19

TU/e

# Comprehension and quality are key

- Insights:
  - Feature traces are helpful …
    … but should not be configurable

```
1     char_u  *
2  fix_fname(fname)
3     char_u  *fname;
4  {
5  #ifdef UNIX
6     return FullName_save(fname, TRUE);
7  #else
8     if (!vim_isAbsName(fname)
9             || strstr((char *)fname, "..") != NULL
10            || strstr((char *)fname, "//") != NULL
11 # ifdef BACKSLASH_IN_FILENAME
12            || strstr((char *)fname, "\\\\") != NULL
13 # endif
14 # if defined(MSWIN) || defined(DJGPP)
15            || vim_strchr(fname, '~') != NULL
16 # endif
17            )
18        return FullName_save(fname, FALSE);
19
20    fname = vim_strsave(fname);
21
22 # ifdef USE_FNAME_CASE
23 #  ifdef USE_LONG_FNAME
24    if (USE_LONG_FNAME)
25 #  endif
26    {
27        if (fname != NULL)
28            fname_case(fname, 0);
29    }
30 # endif
31
32    return fname;
33 #endif
34 }
```



Fenske, **Krüger**, Kanyshkova, Schulze: `#ifdef` *Directives and Program Comprehension: The Dilemma between Correctness and Preference*. ICSME'20
**Krüger**, Çalıklı, Berger, Leich, Saake: *Effects of Explicit Feature Traceability on Program Comprehension*. ESEC/FSE'19
Ludwig, **Krüger**, Leich: *Covert and Phantom Features in Annotations: Do They Impact Variability Analysis?*. SPLC'19

TU/e

# Comprehension and quality are key

- Insights:
  - Feature traces are helpful …
    … but should not be configurable
  - Understanding variability is hard …

```
1     char_u  *
2 fix_fname(fname)
3     char_u  *fname;
4 {
5 #ifdef UNIX
6     return FullName_save(fname, TRUE);
7 #else
8     if (!vim_isAbsName(fname)
9             || strstr((char *)fname, "..") != NULL
10            || strstr((char *)fname, "//") != NULL
11 # ifdef BACKSLASH_IN_FILENAME
12            || strstr((char *)fname, "\\\\") != NULL
13 # endif
14 # if defined(MSWIN) || defined(DJGPP)
15            || vim_strchr(fname, '~') != NULL
16 # endif
17            )
18        return FullName_save(fname, FALSE);
19
20    fname = vim_strsave(fname);
21
22 # ifdef USE_FNAME_CASE
23 #  ifdef USE_LONG_FNAME
24    if (USE_LONG_FNAME)
25 #  endif
26    {
27        if (fname != NULL)
28            fname_case(fname, 0);
29    }
30 # endif
31
32    return fname;
33 #endif
34 }
```



Fenske, **Krüger**, Kanyshkova, Schulze: `#ifdef` *Directives and Program Comprehension: The Dilemma between Correctness and Preference*. ICSME'20
**Krüger**, Çalıklı, Berger, Leich, Saake: *Effects of Explicit Feature Traceability on Program Comprehension*. ESEC/FSE'19
Ludwig, **Krüger**, Leich: *Covert and Phantom Features in Annotations: Do They Impact Variability Analysis?*. SPLC'19

TU/e

# Comprehension and quality are key

- Insights:
  - Feature traces are helpful …
    … but should not be configurable
  - Understanding variability is hard …
    … and analyzing it even more

```
1     char_u  *
2 fix_fname(fname)
3     char_u  *fname;
4 {
5 #ifdef UNIX
6     return FullName_save(fname, TRUE);
7 #else
8     if (!vim_isAbsName(fname)
9             || strstr((char *)fname, "..") != NULL
10            || strstr((char *)fname, "//") != NULL
11 # ifdef BACKSLASH_IN_FILENAME
12            || strstr((char *)fname, "\\\\") != NULL
13 # endif
14 # if defined(MSWIN) || defined(DJGPP)
15            || vim_strchr(fname, '~') != NULL
16 # endif
17            )
18        return FullName_save(fname, FALSE);
19
20    fname = vim_strsave(fname);
21
22 # ifdef USE_FNAME_CASE
23 #  ifdef USE_LONG_FNAME
24    if (USE_LONG_FNAME)
25 #  endif
26    {
27        if (fname != NULL)
28            fname_case(fname, 0);
29    }
30 # endif
31
32    return fname;
33 #endif
34 }
```



Fenske, **Krüger**, Kanyshkova, Schulze: `#ifdef` *Directives and Program Comprehension: The Dilemma between Correctness and Preference*. ICSME'20
**Krüger**, Çalıklı, Berger, Leich, Saake: *Effects of Explicit Feature Traceability on Program Comprehension*. ESEC/FSE'19
Ludwig, **Krüger**, Leich: *Covert and Phantom Features in Annotations: Do They Impact Variability Analysis?*. SPLC'19

TU/e

# Comprehension and quality are key

- Insights:
  - Feature traces are helpful …
    … but should not be configurable
  - Understanding variability is hard …
    … and analyzing it even more
- Challenges:
  - Refactoring
  - Removing/simplifying variability
  - Comprehending software

```
1      char_u  *
2  fix_fname(fname)
3      char_u  *fname;
4  {
5  #ifdef UNIX
6      return FullName_save(fname, TRUE);
7  #else
8      if (!vim_isAbsName(fname)
9              || strstr((char *)fname, "..") != NULL
10             || strstr((char *)fname, "//") != NULL
11 # ifdef BACKSLASH_IN_FILENAME
12             || strstr((char *)fname, "\\\\") != NULL
13 # endif
14 # if defined(MSWIN) || defined(DJGPP)
15             || vim_strchr(fname, '~') != NULL
16 # endif
17             )
18         return FullName_save(fname, FALSE);
19
20     fname = vim_strsave(fname);
21
22 # ifdef USE_FNAME_CASE
23 #  ifdef USE_LONG_FNAME
24     if (USE_LONG_FNAME)
25 #  endif
26     {
27         if (fname != NULL)
28             fname_case(fname, 0);
29     }
30 # endif
31
32     return fname;
33 #endif
34 }
```

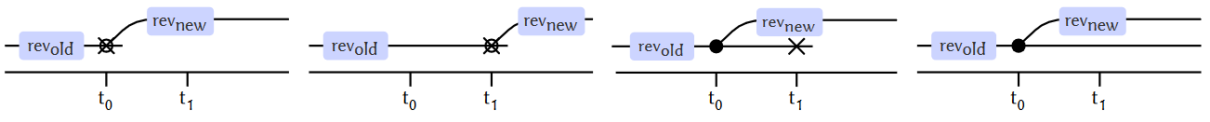


Fenske, **Krüger**, Kanyshkova, Schulze: `#ifdef` *Directives and Program Comprehension: The Dilemma between Correctness and Preference*. ICSME'20
**Krüger**, Çalıklı, Berger, Leich, Saake: *Effects of Explicit Feature Traceability on Program Comprehension*. ESEC/FSE'19
Ludwig, **Krüger**, Leich: *Covert and Phantom Features in Annotations: Do They Impact Variability Analysis?*. SPLC'19

TU/e

# Aligning hardware/software releases is hard

- Insights:
  - Different options for aligning hardware/software changes
  - Over-the-air updates become more important



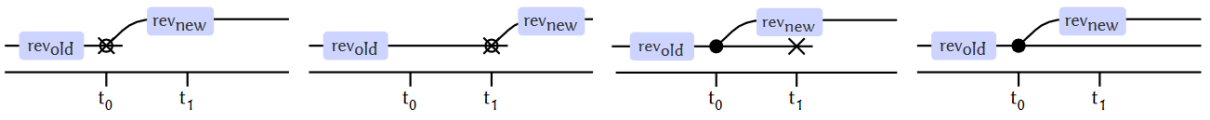



Holsten, Frank, **Krüger**, Leich: *Electrics/Electronics Platforms in the Automotive Industry: Challenges and Directions for Variant-Rich Systems Engineering*. VaMoS'23
Kuiter, **Krüger**, Saake: *Iterative Development and Changing Requirements: Drivers of Variability in an Industrial System for Veterinary Anesthesia*. SPLC'2019

TU/e

# Aligning hardware/software releases is hard

- Insights:
  - Different options for aligning hardware/software changes
  - Over-the-air updates become more important
- Challenges:
  - Understanding pros and cons of strategies
  - Deciding when to use what strategy




(a) Early Shift (ES) (b) Late Shift (LS) (c) Transition Phase (TP) (d) Hardware Reuse (HR)



Holsten, Frank, **Krüger**, Leich: *Electrics/Electronics Platforms in the Automotive Industry: Challenges and Directions for Variant-Rich Systems Engineering*. VaMoS'23
Kuiter, **Krüger**, Saake: *Iterative Development and Changing Requirements: Drivers of Variability in an Industrial System for Veterinary Anesthesia*. SPLC'2019

TU/e

## Some features may become dated

- Insights:
  - Features or whole systems may become outdated
  - Features may move into commodity to reduce complexity



Cortiñas, **Krüger**, Lamas, Luaces, Pedreira: *How to Retire and Replace a Software Product Line*. SPLC'23
Holsten, Frank, **Krüger**, Leich: *Electrics/Electronics Platforms in the Automotive Industry: Challenges and Directions for Variant-Rich Systems Engineering*. VaMoS'23
Ludwig, **Krüger**, Leich: *Covert and Phantom Features in Annotations: Do They Impact Variability Analysis?*. SPLC'19

## Some features may become dated

- Insights:
  - Features or whole systems may become outdated
  - Features may move into commodity to reduce complexity
- Challenges:
  - Deciding what features are not needed anymore
  - Ensuring safe re-engineering operations
  - Designing automated analyses/operations



Cortiñas, **Krüger**, Lamas, Luaces, Pedreira: *How to Retire and Replace a Software Product Line*. SPLC'23
Holsten, Frank, **Krüger**, Leich: *Electrics/Electronics Platforms in the Automotive Industry: Challenges and Directions for Variant-Rich Systems Engineering*. VaMoS'23
Ludwig, **Krüger**, Leich: *Covert and Phantom Features in Annotations: Do They Impact Variability Analysis?*. SPLC'19

## Also challenges in practice? Starting points for collaboration?

- Ensuring program comprehension?
- Assuring software quality?
- Analyzing variability?
- Aligning hardware and software releases?
- Deprecating (variable) features or a platform?
- Re-engineering variability safely?
- …




TU/e

## Want more information or get in touch?

Jacob Krüger

Eindhoven University of Technology
Department for Mathematics and Computer Science
Software Engineering and Technology
MetaForum 6.096 (likely to change to 6.089)

j.kruger@tue.nl
https://jacobkrueger.github.io/