OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Databases
and
Software
Engineering

RUHR
UNIVERSITÄT
BOCHUM

Iterative Development and Changing Requirements

Drivers of Variability in an Industrial System for Veterinary Anesthesia

Elias Kuiter, Jacob Krüger, Gunter Saake

Otto-von-Guericke-University Magdeburg, Ruhr-University Bochum

VariVolution@SPLC 2021
September 6–11 | Leicester, United Kingdom

Motivation

Embedded Systems

Ubiquitous in our lives

Motivation

Embedded Systems

Ubiquitous in our lives

- Internet of things

Motivation

Embedded Systems

Ubiquitous in our lives

- Internet of things
- Vehicles

Motivation

Embedded Systems

Ubiquitous in our lives

- Internet of things
- Vehicles
- Medical domain

Motivation

Embedded Systems

Ubiquitous in our lives

- Internet of things
- Vehicles
- Medical domain

Correctness? Safety?

⟹ Financial risk

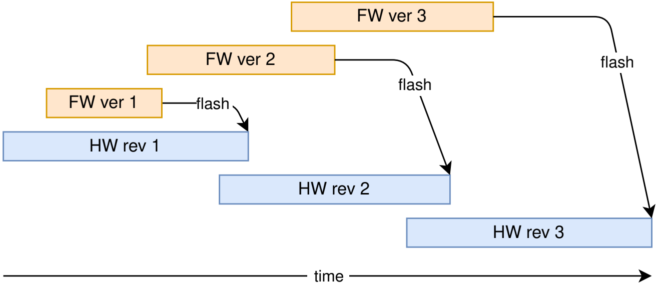
× *burn-and-pray*
✓ *waterfall/V model*
✓ *iterative development*

Motivation

- **Iterative development** with **prototyping** for novel products based on “living standards”
- Often hardware (HW) and software/**firmware** (FW) are developed in parallel

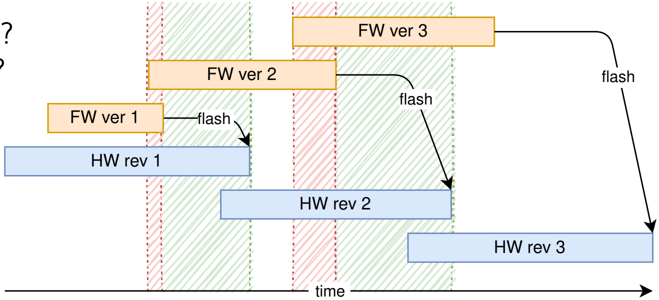
Motivation

- **Iterative development** with **prototyping** for novel products based on “living standards”
- Often hardware (HW) and software/**firmware** (FW) are developed in parallel
- **Problem**: HW/FW **development cycles** have different duration/cost
  Firmware versions are finished long before their targeted hardware revisions are produced



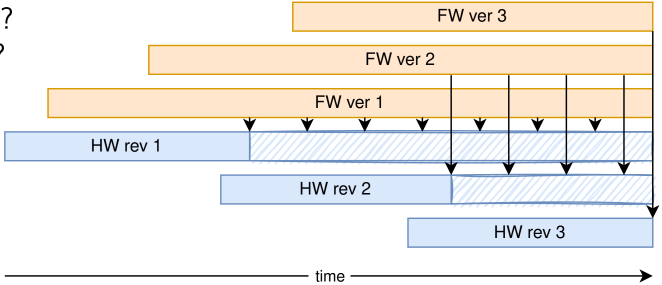
Motivation

- **Iterative development** with **prototyping** for novel products based on “living standards”
- Often hardware (HW) and software/**firmware** (FW) are developed in parallel
- **Problem**: HW/FW **development cycles** have different duration/cost
  Firmware versions are finished long before their targeted hardware revisions are produced
- How to handle ...
  $\Longrightarrow$ Overlaps in FW development?
  $\Longrightarrow$ Waiting for HW to catch up?



Motivation

- **Iterative development** with **prototyping** for novel products based on “living standards”
- Often hardware (HW) and software/**firmware** (FW) are developed in parallel
- **Problem**: HW/FW **development cycles** have different duration/cost
  Firmware versions are finished long before their targeted hardware revisions are produced
- How to handle …
  ⟹ Overlaps in FW development?
  ⟹ Waiting for HW to catch up?
- (One) **solution**:
  ⟹ Embrace FW variability
  ⟹ Enables hardware reuse
- **Goal**: Reduce cost and risk



Contributions

- Analysis of **emerging variability** in an industrial system for veterinary anesthesia

Contributions

- Analysis of **emerging variability** in an industrial system for veterinary anesthesia
- Discussion of **drivers of variability** in our case study

Contributions

- Analysis of **emerging variability** in an industrial system for veterinary anesthesia
- Discussion of **drivers of variability** in our case study
- **Scenarios** for intertwined HW/FW evolution and their tradeoffs

Contributions

- Analysis of **emerging variability** in an industrial system for veterinary anesthesia
- Discussion of **drivers of variability** in our case study
- **Scenarios** for intertwined HW/FW evolution and their tradeoffs
- **Goal**: Improve understanding and resolve the HW/FW gap in embedded systems
  Give initial guidelines for project managers in such projects

Case Study

- PigNap[1] is an industrial system for veterinary anesthesia
- Developed with our industry partners HCP and BEG

HCP-Technology

BEG Schulze Bremer GmbH
Ihr Partner für Tierzuchtbedarf

[1] http://pignap.com

[2] https://github.com/ekuiter/pignap-case-study

Case Study

- PigNap[1] is an industrial system for veterinary anesthesia
- Developed with our industry partners HCP and BEG
- Purpose of device: **castration of piglets** (newborn pigs)
  - Piglets are castrated to improve meat quality
  - 2021: Anesthesia is mandatory to ensure animal well-being
  - 2019: Innovative, law-compliant devices had to be developed
  - High-risk/reward project, short timeframe, many stakeholders

HCP-Technology

BEG Schulze Bremer GmbH
Ihr Partner für Tierzuchtbedarf

[1] http://pignap.com

[2] https://github.com/ekuiter/pignap-case-study

Case Study

- PigNap[1] is an industrial system for veterinary anesthesia
- Developed with our industry partners HCP and BEG
- Purpose of device: **castration of piglets** (newborn pigs)
  - Piglets are castrated to improve meat quality
  - 2021: Anesthesia is mandatory to ensure animal well-being
  - 2019: Innovative, law-compliant devices had to be developed
  - High-risk/reward project, short timeframe, many stakeholders
- Developed over 1 year, financial success, 33% market share
- Embracing FW variability contributed to the project's success
- Case study published on GitHub[2]

HCP-Technology

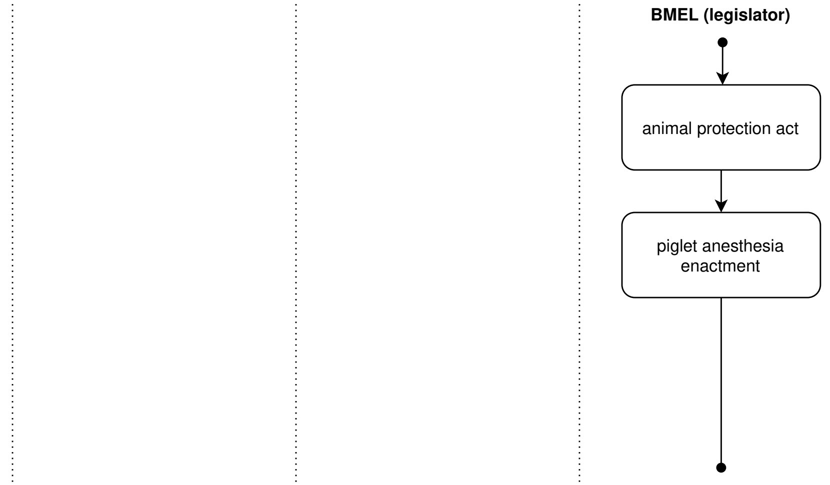
BEG Schulze Bremer GmbH
Ihr Partner für Tierzuchtbedarf

[1] http://pignap.com

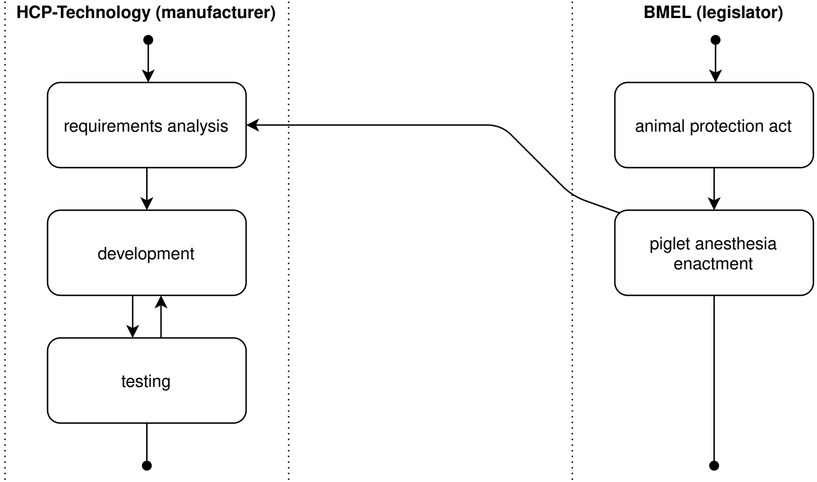
[2] https://github.com/ekuiter/pignap-case-study

Development Process

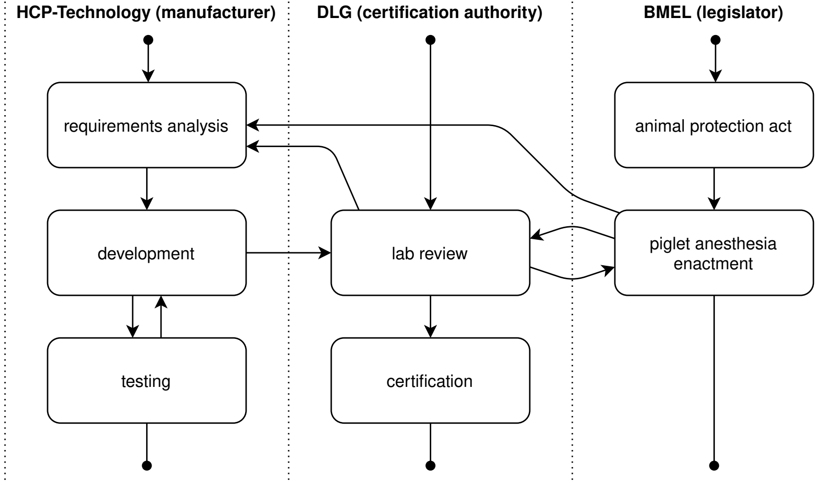
Development Process



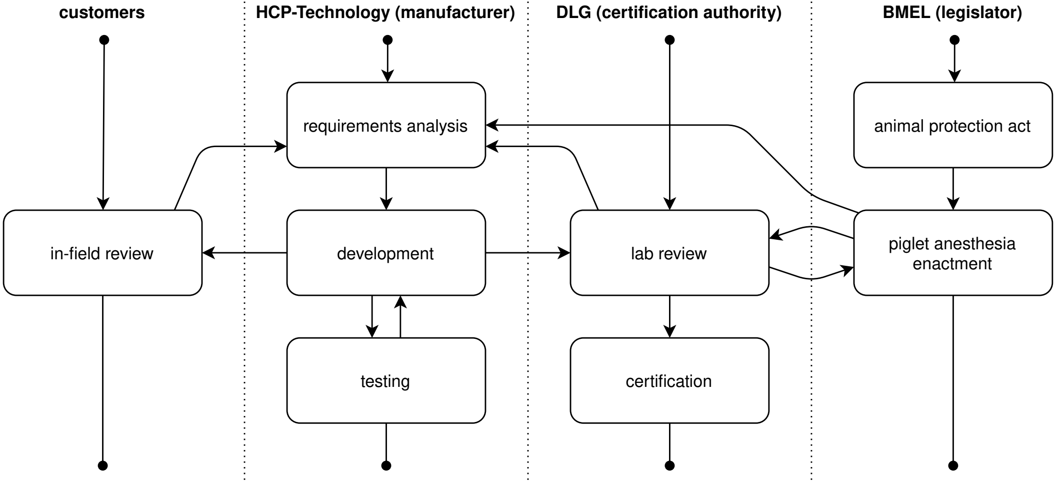
Development Process



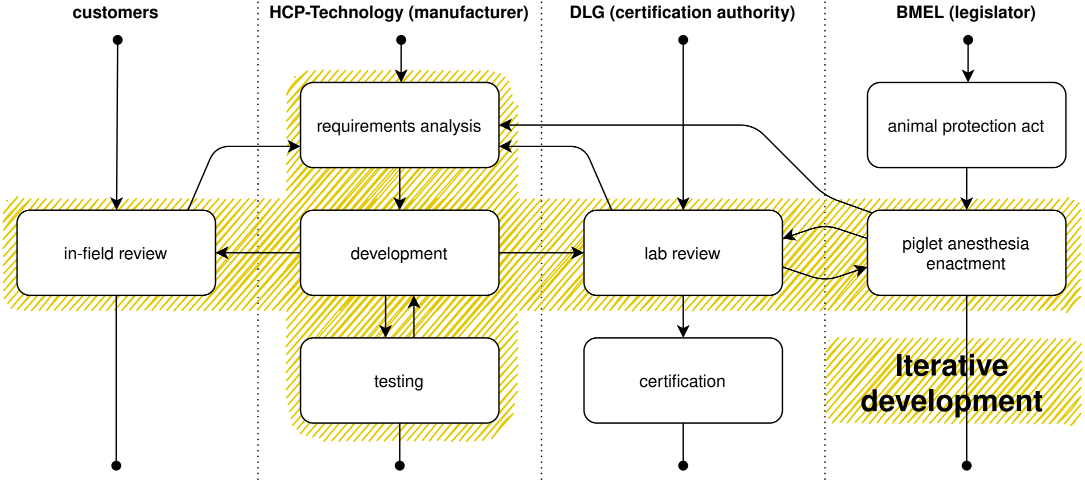
Development Process



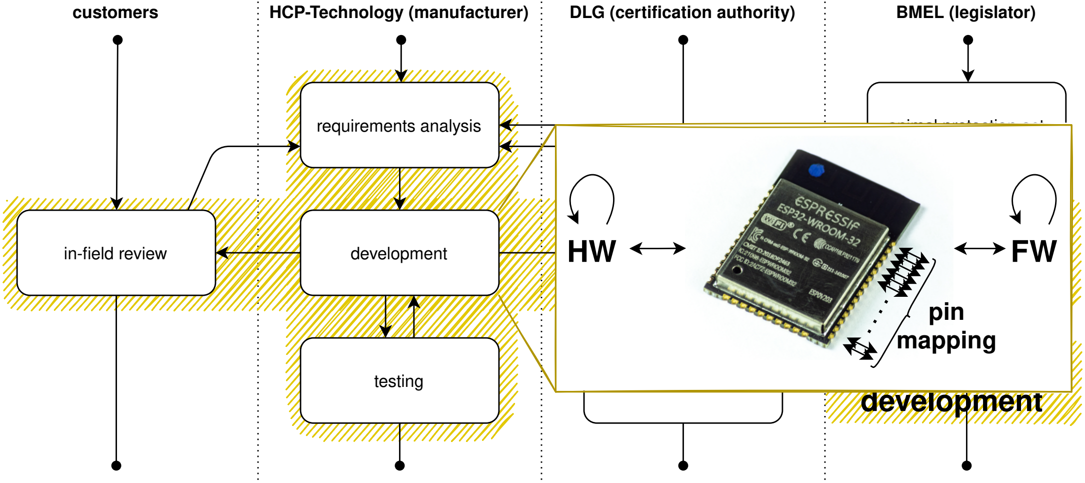
Development Process



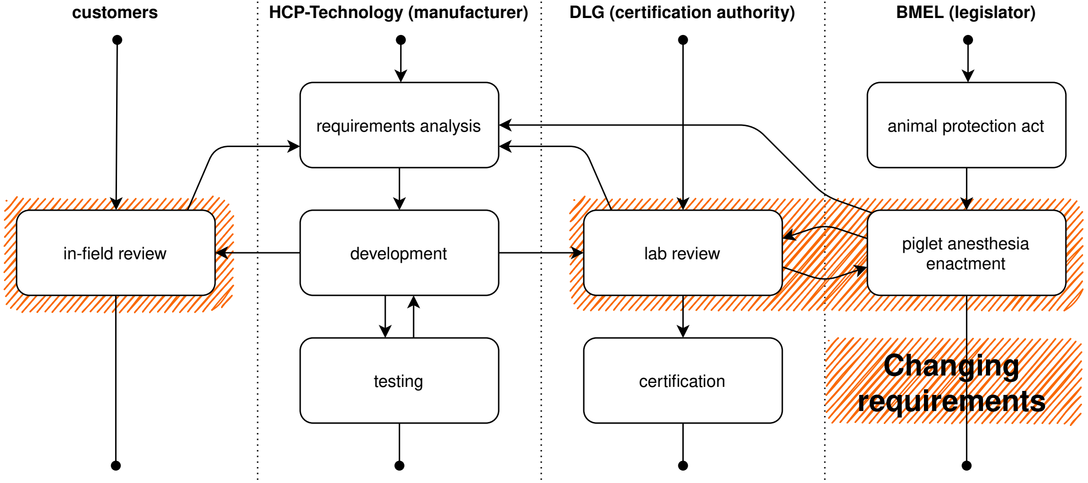
Development Process



Development Process

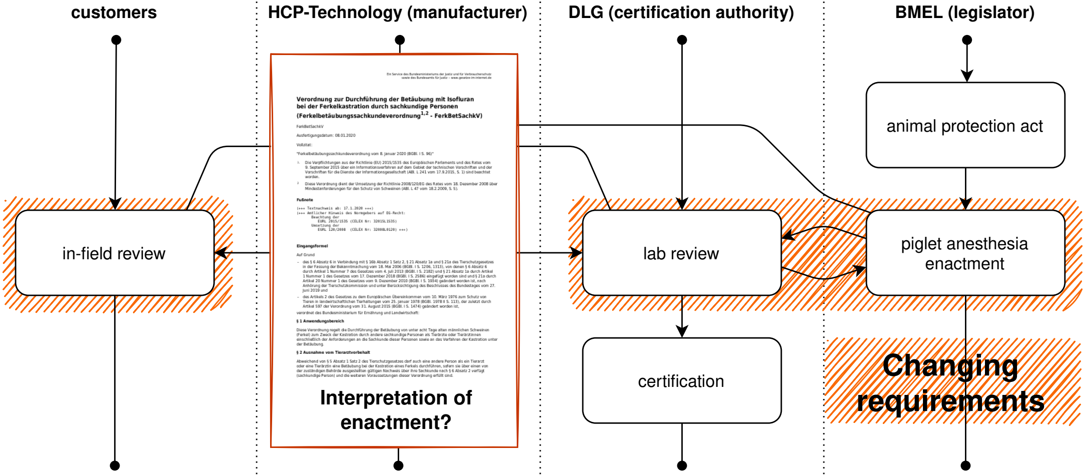


Development Process



Changing requirements

Development Process



Ein Service des Bundesministeriums der Justiz und für Verbraucherschutz
sowie des Bundesamts für Justiz – www.gesetze-im-internet.de

Verordnung zur Durchführung der Betäubung mit Isofluran bei der Ferkelkastration durch sachkundige Personen (Ferkelbetäubungssachkundeverordnung[1,2] - FerkBetSachkV)

FerkBetSachkV

Ausfertigungsdatum: 08.01.2020

Vollzitat:

"Ferkelbetäubungssachkundeverordnung vom 8. Januar 2020 (BGBl. I S. 96)"

[1] Die Verpflichtungen aus der Richtlinie (EU) 2015/1535 des Europäischen Parlaments und des Rates vom 9. September 2015 über ein Informationsverfahren auf dem Gebiet der technischen Vorschriften und der Vorschriften für die Dienste der Informationsgesellschaft (ABl. L 241 vom 17.9.2015, S. 1) sind beachtet worden.

[2] Diese Verordnung dient der Umsetzung der Richtlinie 2008/120/EG des Rates vom 18. Dezember 2008 über Mindestanforderungen für den Schutz von Schweinen (ABl. L 47 vom 18.2.2009, S. 5).

Fußnote

```
(+++ Textnachweis ab: 17.1.2020 +++)
(+++ Amtlicher Hinweis des Normgebers auf EG-Recht:
       Beachtung der
          EURL 2015/1535 (CELEX Nr: 32015L1535)
       Umsetzung der
          EGRL 120/2008  (CELEX Nr: 32008L0120) +++)
```

Eingangsformel

Auf Grund

- des § 6 Absatz 6 in Verbindung mit § 16b Absatz 1 Satz 2, § 21 Absatz 1a und § 21a des Tierschutzgesetzes in der Fassung der Bekanntmachung vom 18. Mai 2006 (BGBl. I S. 1206, 1313), von denen § 6 Absatz 6 durch Artikel 1 Nummer 7 des Gesetzes vom 4. Juli 2013 (BGBl. I S. 2182) und § 21 Absatz 1a durch Artikel 1 Nummer 1 des Gesetzes vom 17. Dezember 2018 (BGBl. I S. 2586) eingefügt worden sind und § 21a durch Artikel 20 Nummer 1 des Gesetzes vom 9. Dezember 2010 (BGBl. I S. 1934) geändert worden ist, nach Anhörung der Tierschutzkommission und unter Berücksichtigung des Beschlusses des Bundestages vom 27. Juni 2019 und
- des Artikels 2 des Gesetzes zu dem Europäischen Übereinkommen vom 10. März 1976 zum Schutz von Tieren in landwirtschaftlichen Tierhaltungen vom 25. Januar 1978 (BGBl. 1978 II S. 113), der zuletzt durch Artikel 597 der Verordnung vom 31. August 2015 (BGBl. I S. 1474) geändert worden ist,

verordnet das Bundesministerium für Ernährung und Landwirtschaft:

§ 1 Anwendungsbereich

Diese Verordnung regelt die Durchführung der Betäubung von unter acht Tage alten männlichen Schweinen (Ferkel) zum Zweck der Kastration durch andere sachkundige Personen als Tierärzte oder Tierärztinnen einschließlich der Anforderungen an die Sachkunde dieser Personen sowie an das Verfahren der Kastration unter der Betäubung.

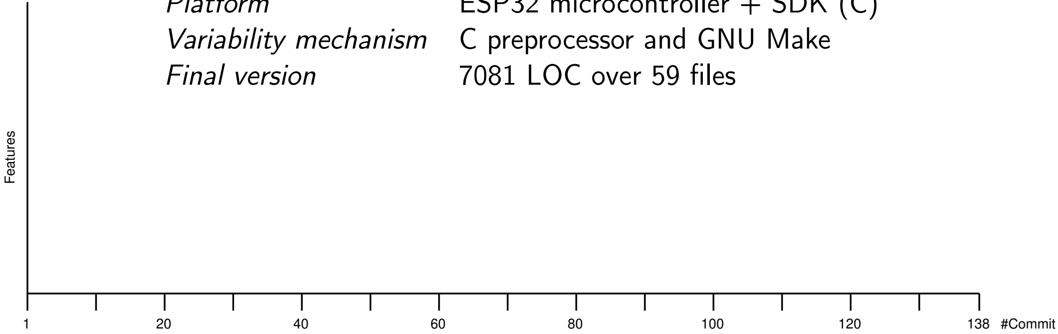
§ 2 Ausnahme vom Tierarztvorbehalt

Abweichend von § 5 Absatz 1 Satz 2 des Tierschutzgesetzes darf auch eine andere Person als ein Tierarzt oder eine Tierärztin eine Betäubung bei der Kastration eines Ferkels durchführen, sofern sie über einen von der zuständigen Behörde ausgestellten gültigen Nachweis über ihre Sachkunde nach § 6 Absatz 2 verfügt (sachkundige Person) und die weiteren Voraussetzungen dieser Verordnung erfüllt sind.

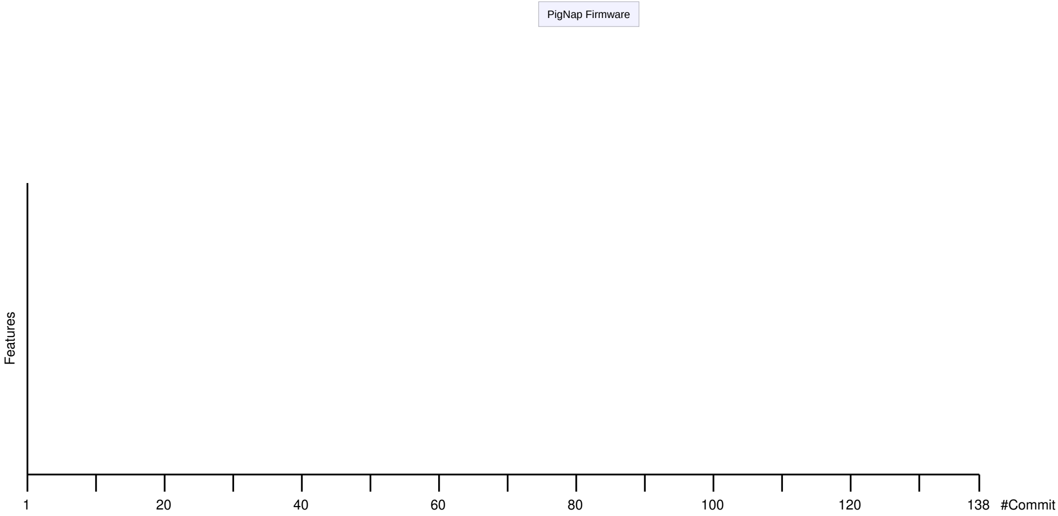
Interpretation of enactment?

Development Timeline

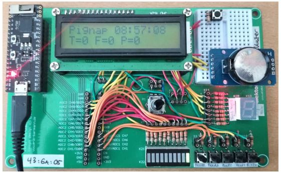
Project duration	June 2019 – June 2020, 138 commits
Platform	ESP32 microcontroller + SDK (C)
Variability mechanism	C preprocessor and GNU Make
Final version	7081 LOC over 59 files



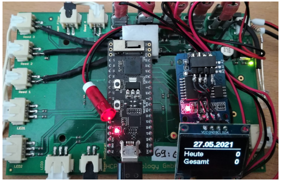
Development Timeline



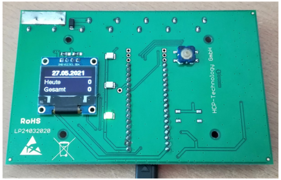
Development Timeline



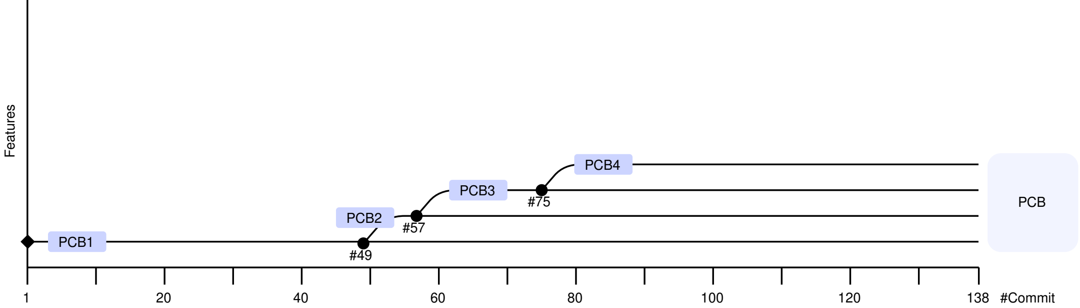
Printed Circuit Board 1



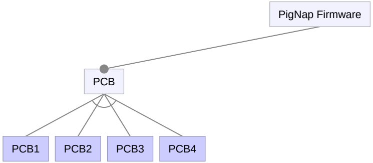
PCB2

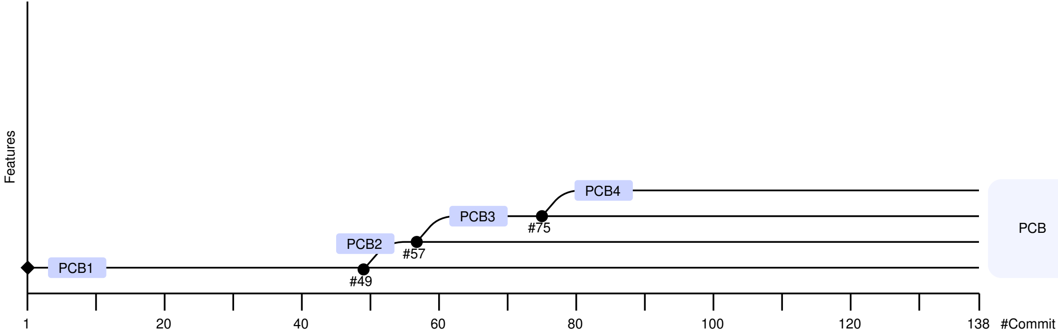


PCB3

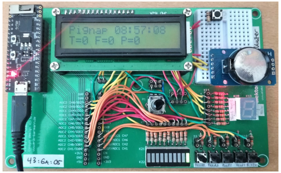


Development Timeline

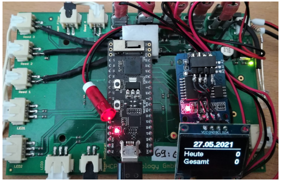




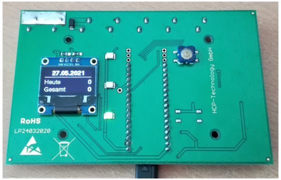
Development Timeline



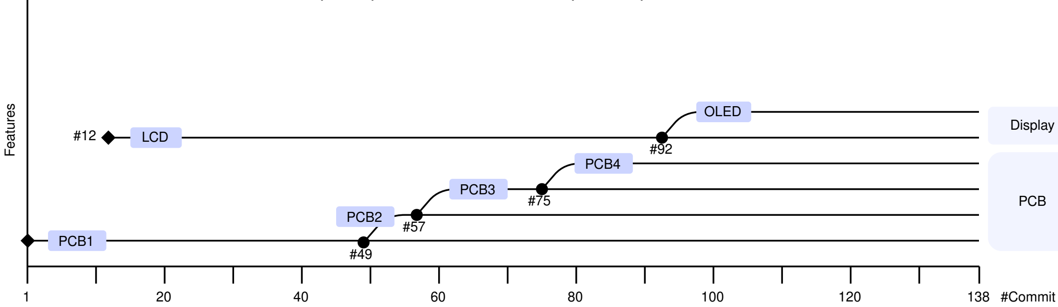
Liquid Crystal Display (LCD)



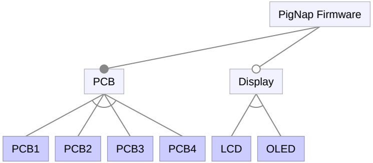
Organic LED (OLED)

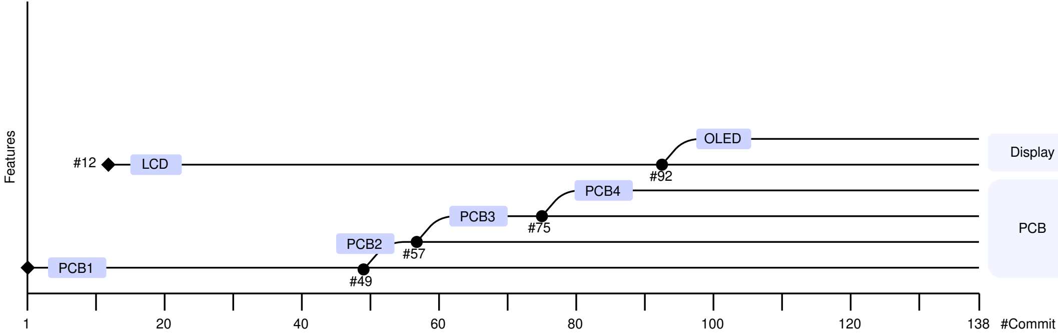


OLED

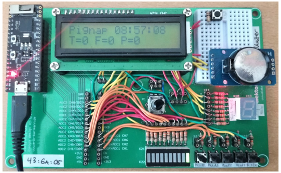


Development Timeline

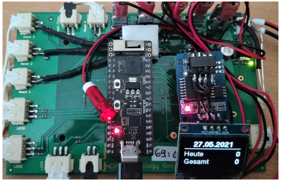




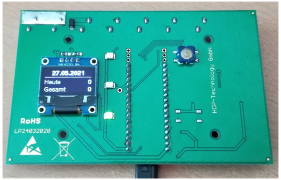
Development Timeline



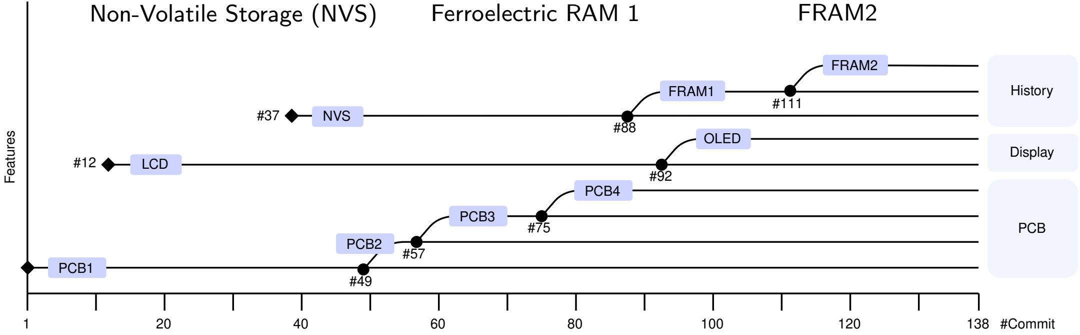
Non-Volatile Storage (NVS)



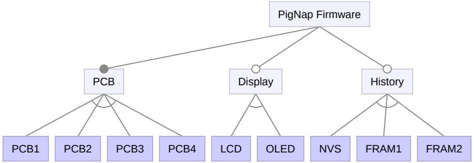
Ferroelectric RAM 1

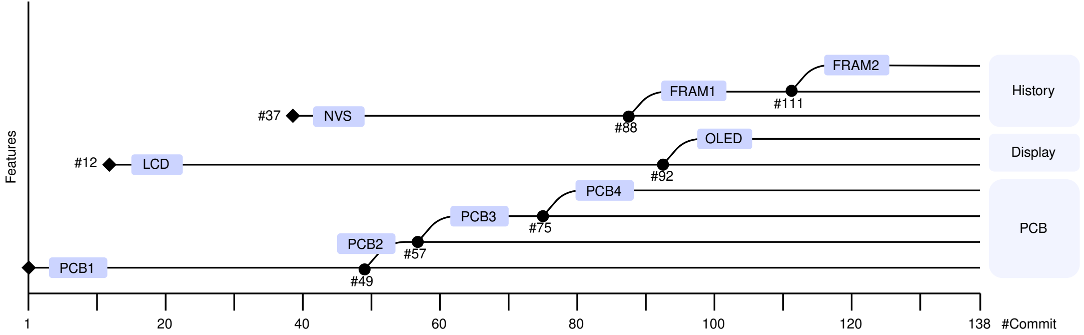


FRAM2

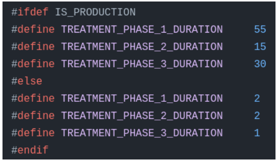


Development Timeline

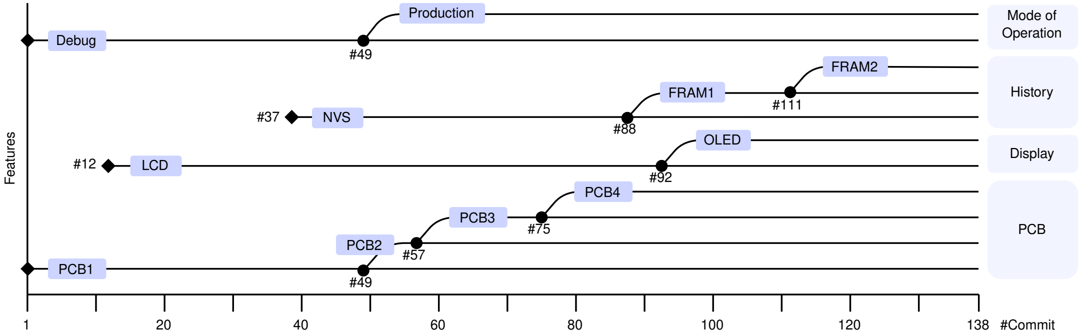




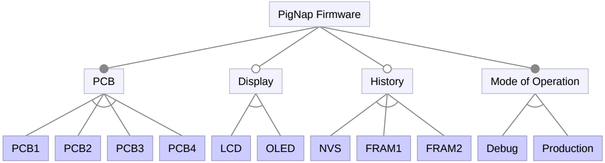
Development Timeline

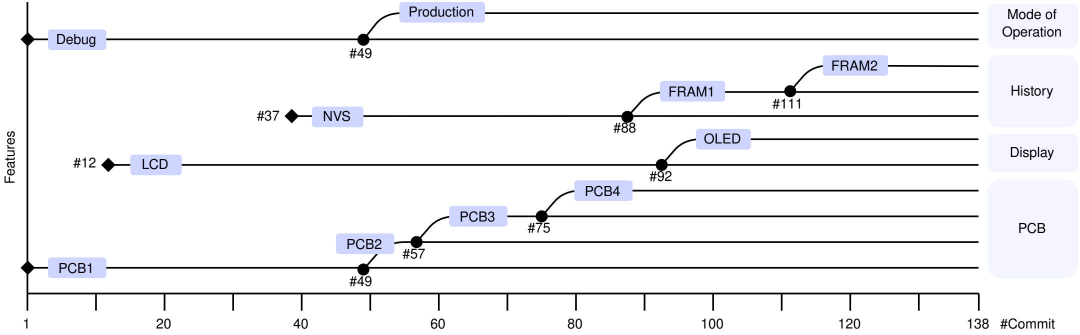


```
#ifdef IS_PRODUCTION
#define TREATMENT_PHASE_1_DURATION      55
#define TREATMENT_PHASE_2_DURATION      15
#define TREATMENT_PHASE_3_DURATION      30
#else
#define TREATMENT_PHASE_1_DURATION      2
#define TREATMENT_PHASE_2_DURATION      2
#define TREATMENT_PHASE_3_DURATION      1
#endif
```



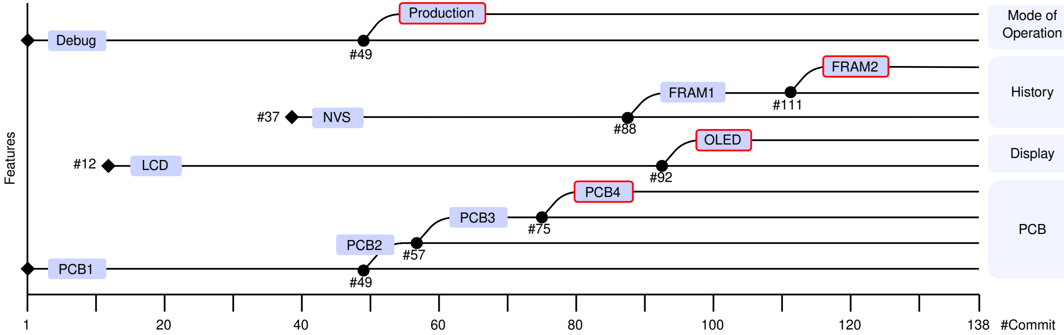
Development Timeline



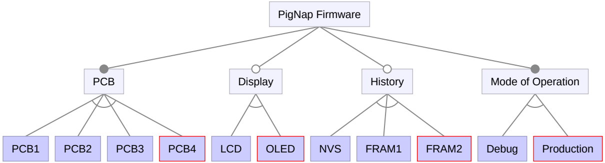


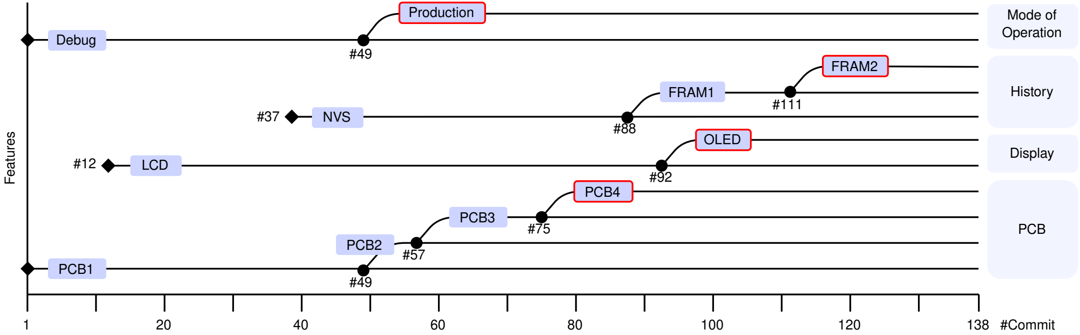
Development Timeline

“Canonical” configuration sold to customers



Development Timeline





Drivers of Variability

An 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"

Drivers of Variability

An 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"
$\Longrightarrow$ not an SPL because there is no external variability
$\Longrightarrow$ all optional features are "accidental" complexity (byproducts of our development choices)

Drivers of Variability

An 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"
$\Longrightarrow$ not an SPL because there is no external variability
$\Longrightarrow$ all optional features are "accidental" complexity (byproducts of our development choices)

Driver \ Feature subtree	**PCB**	**Display**	**History**	**Mode**

Drivers of Variability

An 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"

$\Longrightarrow$ not an SPL because there is no external variability

$\Longrightarrow$ all optional features are "accidental" complexity (byproducts of our development choices)

Driver \ Feature subtree	**PCB**	**Display**	**History**	**Mode**
Iterative development of hardware	●	●	●	◐

Drivers of Variability

An 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"
$\Longrightarrow$ not an SPL because there is no external variability
$\Longrightarrow$ all optional features are "accidental" complexity (byproducts of our development choices)

Driver \ Feature subtree	**PCB**	**Display**	**History**	**Mode**
Iterative development				
of hardware	●	●	●	◐
of firmware	○	○	○	●

Drivers of Variability

An 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”
$\Longrightarrow$ not an SPL because there is no external variability
$\Longrightarrow$ all optional features are “accidental” complexity (byproducts of our development choices)

Driver \ Feature subtree	**PCB**	**Display**	**History**	**Mode**
Iterative development				
of hardware	●	●	●	◐
of firmware	○	○	○	●
Changing requirements				
from customers	◐	●	○	○

Drivers of Variability

An 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"
$\Longrightarrow$ not an SPL because there is no external variability
$\Longrightarrow$ all optional features are "accidental" complexity (byproducts of our development choices)

Driver \ Feature subtree	**PCB**	**Display**	**History**	**Mode**
Iterative development				
of hardware	●	●	●	◐
of firmware	○	○	○	●
Changing requirements				
from customers	◐	●	○	○
from BMEL and DLG	◐	○	●	◐

HW Evolution Scenarios

Could/should the variability have been avoided?

HW Evolution Scenarios

Could/should the variability have been avoided?

$\Longrightarrow$ It **could**, by avoiding all drivers (e.g., waterfall model)

But: Impossible due to feedback loop, unknown requirements, HW/FW interaction

$\Longrightarrow$ **Alternative**: Handle emerging variability differently

HW Evolution Scenarios

Could/should the variability have been avoided?

$\Longrightarrow$ It **could**, by avoiding all drivers (e.g., waterfall model)

But: Impossible due to feedback loop, unknown requirements, HW/FW interaction

$\Longrightarrow$ **Alternative**: Handle emerging variability differently

4 HW evolution scenarios (t_0: design, t_1: production)

Property \ Scenario	**ES**	**LS**	**TP**	**HR**
Supports seamless shift to new revision				
Supports old revisions during transition				
Supports old revisions after transition				
Avoids variability				

HW Evolution Scenarios

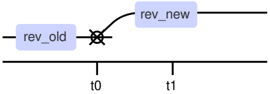
Could/should the variability have been avoided?

$\Longrightarrow$ It **could**, by avoiding all drivers (e.g., waterfall model)

But: Impossible due to feedback loop, unknown requirements, HW/FW interaction

$\Longrightarrow$ **Alternative**: Handle emerging variability differently

4 HW evolution scenarios (t_0: design, t_1: production)



Early Shift

Property \ Scenario	**ES**	**LS**	**TP**	**HR**
Supports seamless shift to new revision	●			
Supports old revisions during transition	○			
Supports old revisions after transition	○			
Avoids variability	●			

HW Evolution Scenarios

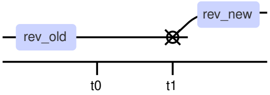
Could/should the variability have been avoided?

$\Longrightarrow$ It **could**, by avoiding all drivers (e.g., waterfall model)

But: Impossible due to feedback loop, unknown requirements, HW/FW interaction

$\Longrightarrow$ **Alternative**: Handle emerging variability differently

4 HW evolution scenarios (t_0: design, t_1: production)



Late Shift

Property \ Scenario	**ES**	**LS**	**TP**	**HR**
Supports seamless shift to new revision	●	○		
Supports old revisions during transition	○	●		
Supports old revisions after transition	○	○		
Avoids variability	●	●		

HW Evolution Scenarios

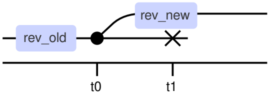
Could/should the variability have been avoided?

⟹ It **could**, by avoiding all drivers (e.g., waterfall model)

But: Impossible due to feedback loop, unknown requirements, HW/FW interaction

⟹ **Alternative**: Handle emerging variability differently

4 HW evolution scenarios (t_0: design, t_1: production)



Transition Phase

Property \ Scenario	**ES**	**LS**	**TP**	**HR**
Supports seamless shift to new revision	●	○	●	
Supports old revisions during transition	○	●	●	
Supports old revisions after transition	○	○	○	
Avoids variability	●	●	◐	

HW Evolution Scenarios

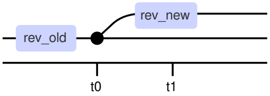
Could/should the variability have been avoided?

⟹ It **could**, by avoiding all drivers (e.g., waterfall model)

But: Impossible due to feedback loop, unknown requirements, HW/FW interaction

⟹ **Alternative**: Handle emerging variability differently

4 HW evolution scenarios (t_0: design, t_1: production)



Hardware Reuse

Property \ Scenario	**ES**	**LS**	**TP**	**HR**
Supports seamless shift to new revision	●	○	●	●
Supports old revisions during transition	○	●	●	●
Supports old revisions after transition	○	○	○	●
Avoids variability	●	●	◐	○

HW Evolution Scenarios

Which scenario is appropriate depends (more research needed):

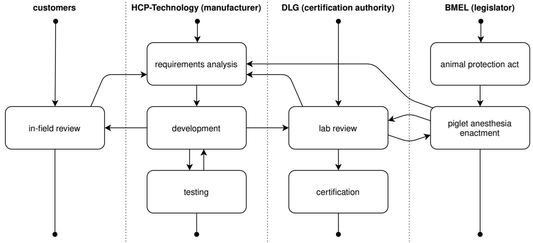
- Is maintaining the variability costly? Is there a high risk for variability bugs? Do developers have sufficient SPL expertise? here: no, no, yes
- Is HW development more costly than FW development? Is fast time-to-market valued more than quick and dirty FW development? here: yes, yes

Property \ Scenario	**ES**	**LS**	**TP**	**HR**
Supports seamless shift to new revision	●	○	●	●
Supports old revisions during transition	○	●	●	●
Supports old revisions after transition	○	○	○	●
Avoids variability	●	●	◐	○

$\Longrightarrow$ Applicable on all real-world embedded projects with parallel and iterative HW/FW development (which is natural to reduce risks in validation/verification)

Conclusion

Case Study



customers

HCP-Technology (manufacturer)

DLG (certification authority)

BMEL (legislator)

requirements analysis

animal protection act

in-field review

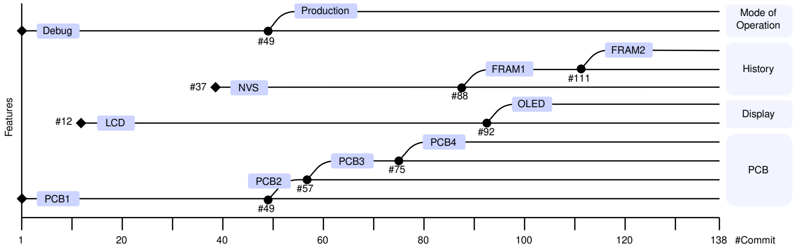
development

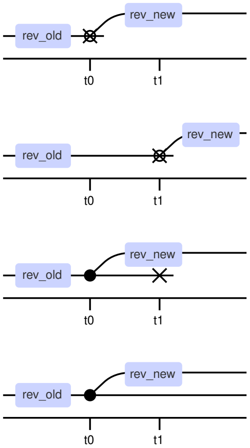
lab review

piglet anesthesia enactment

testing

certification





HW Evolution Scenarios

Attribution

- Wilgengebroed on Flickr, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons
- Matti Blume, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- interCaribbean Airways, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- U.S. Air Force photo by Staff Sergeant Jonathan Steffen, Public domain, via Wikimedia Commons
- Ubahnverleih, CC0, via Wikimedia Commons