pure·systems

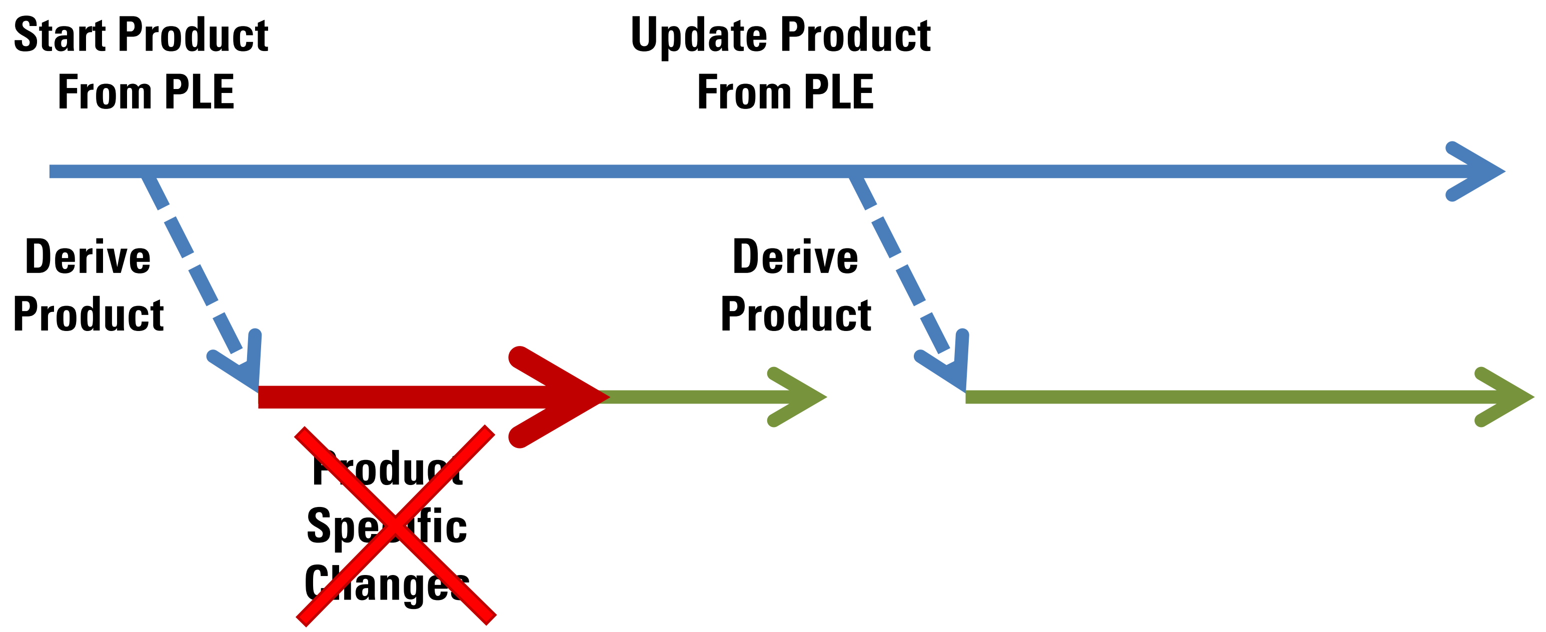
Reverse engineering challenges of the feedback scenario in co-evolving product lines

REVE Workshop at SPLC 2017

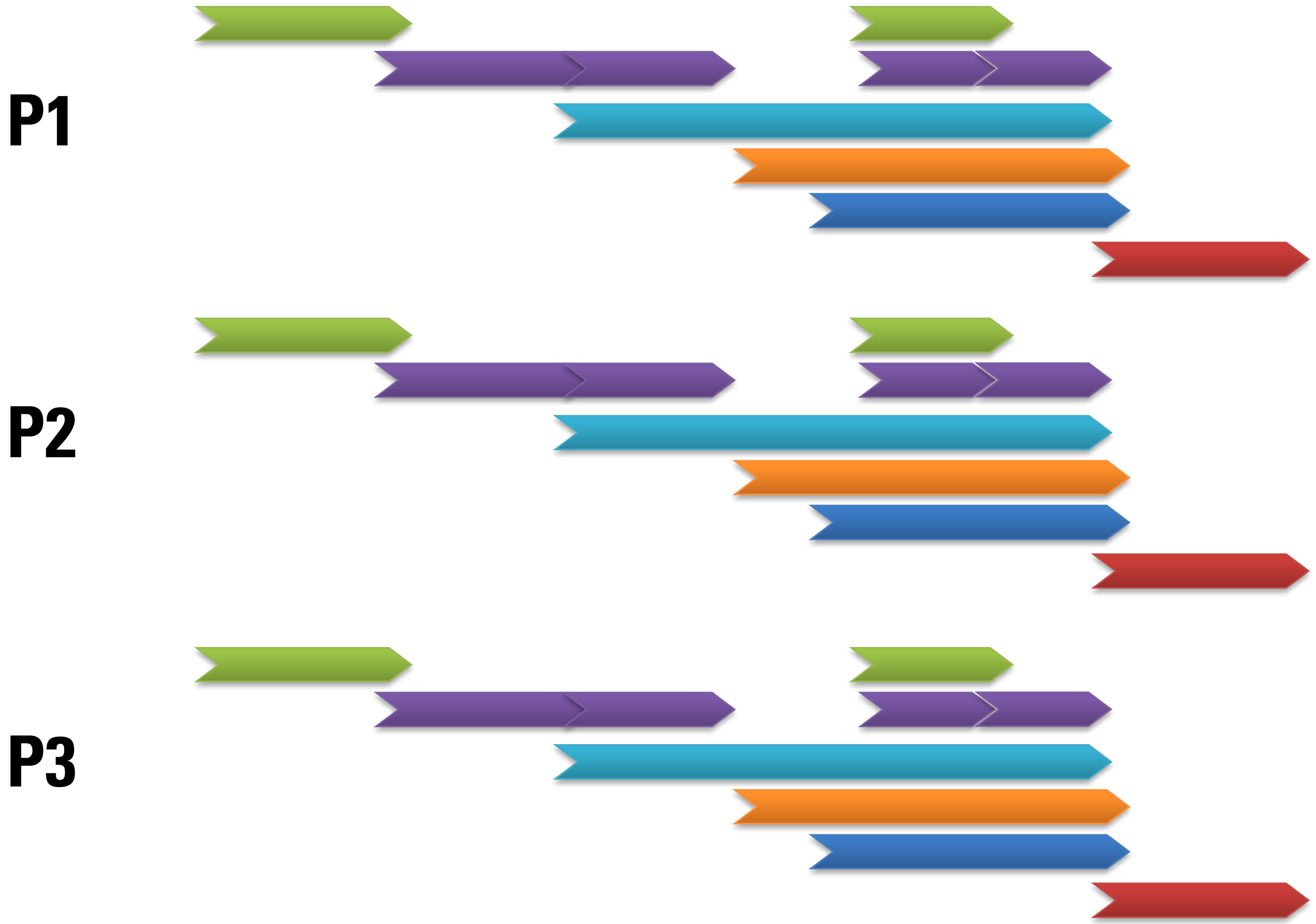
Robert.Hellebrand@pure-systems.com
Michael.Schulze@pure-systems.com
Uwe.Ryssel@pure-systems.com

Agenda

Motivation


Start Product From PLE
Update Product From PLE
Derive Product
Derive Product
~~Product Specific Changes~~

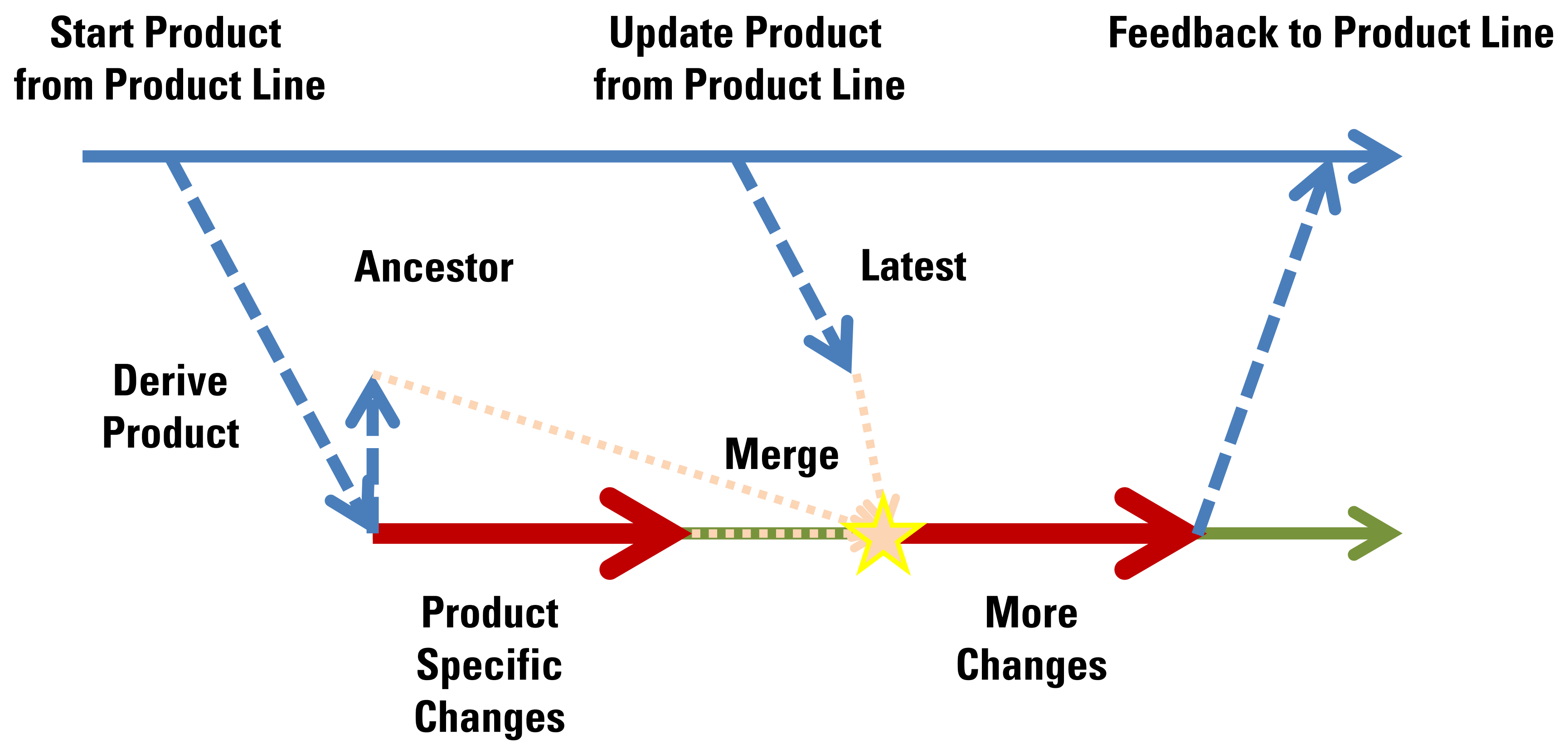
Staggered Development Phases



Tradeoff

Effectivity $_{\text{Reuse}}$ vs. Flexibility $_{\text{Reuse}}$ & Understandability

Co-evolution

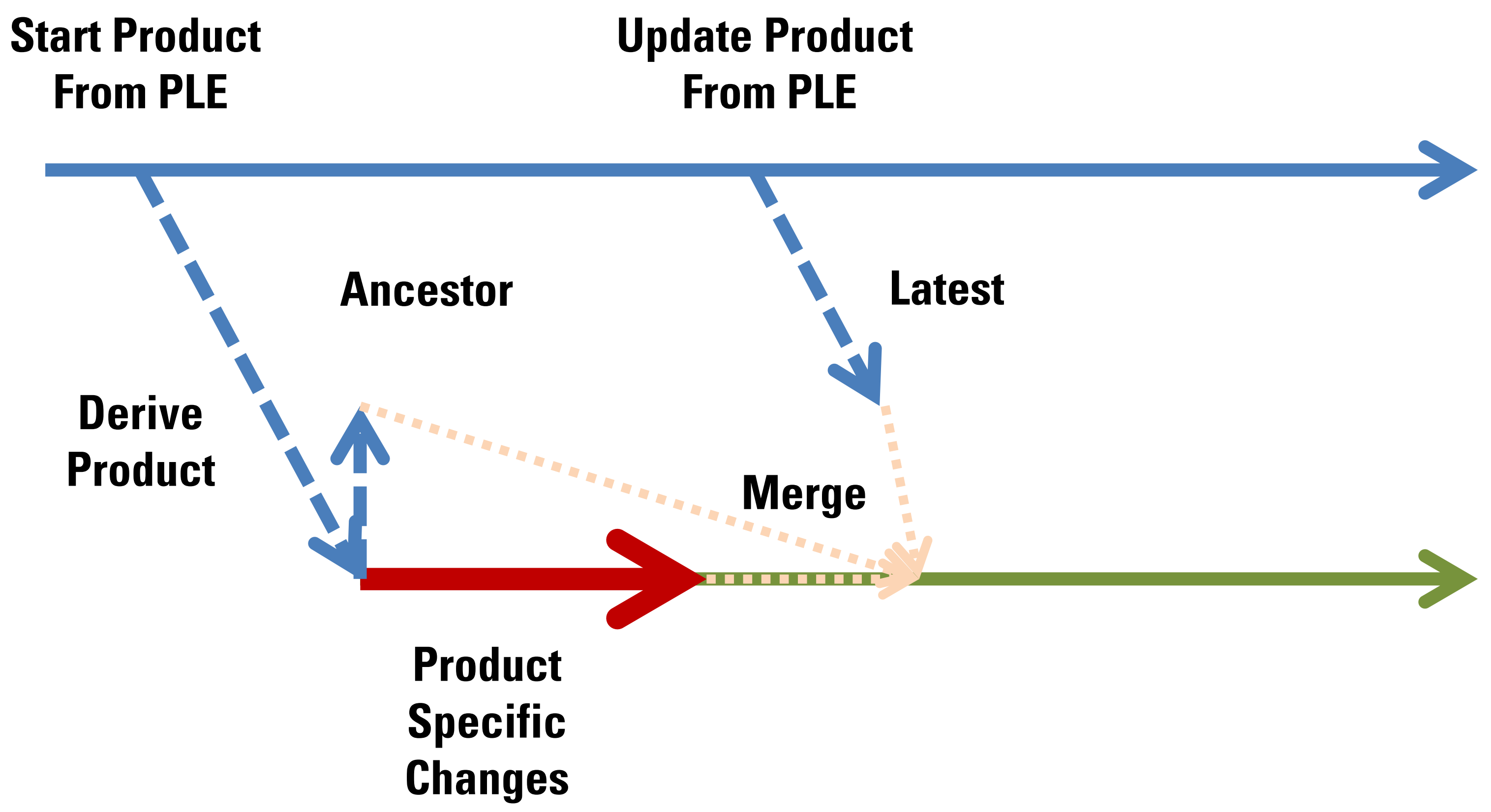


Prior Work

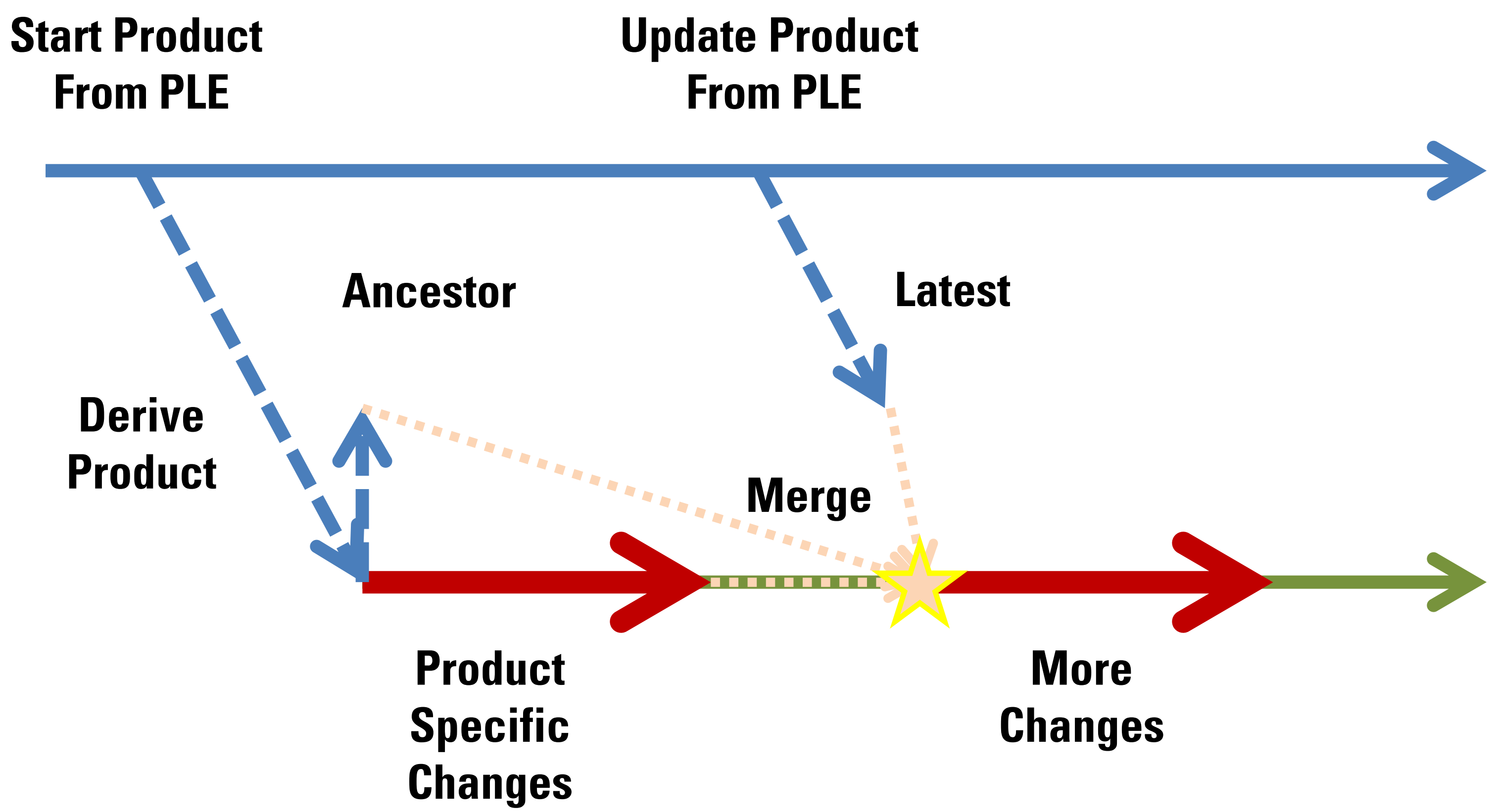
Update Scenario

Sandro Schulze, Michael Schulze, Uwe Ryssel, and Christoph Seidl. 2016. Aligning Coevolving Artifacts Between Software Product Lines and Products. In Proceedings of the Tenth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS ’16). ACM, New York, NY, USA, 9–16. DOI: https://doi.org/10.1145/2866614.2866616

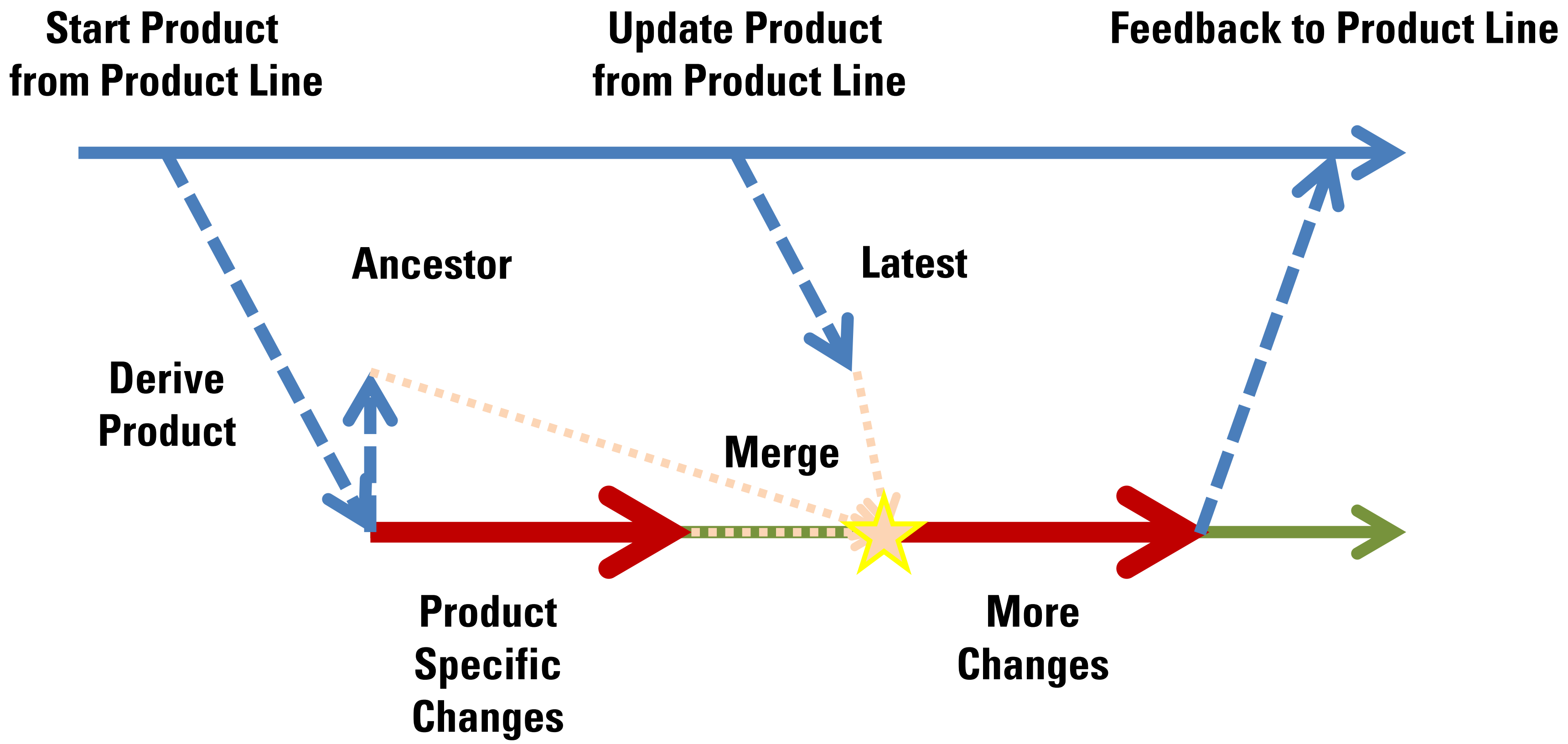
Update with Compare & Merge



Update with Compare & Merge



Feedback with Compare & Merge



Challenges of the feedback scenario

In co-evolving product lines

General Challenges

#	Challenge Description
1	Identify whether feedback parts are common or variable.
2	
3	
4	

Context Specific Challenges

Variation Point Mechanism

- direct annotation of super set assets with feature rules
- indirect annotation of super set assets with feature rules
- substitutional expressions

Transformation Type

- Preserving
- Negative
- Substitutive
- Translating

Variant-Specific Implementation

- Ad-hoc
- managed

Context Specific Challenges

- Variation Point Mechanism
- Transformation Type
- Variant-Specific Implementation

#	Challenge Description
5	Identify those new parts of the variant assets that shall be part of the feedback.
6	
7	
8	
9	

Context Specific Challenges

#	Variation Point Mechanism	Transformation Type	Variant-Specific Implementation
5	*	*	Ad-hoc
6	Direct annotation	Negative	*
7	Direct annotation	Negative	*
8	*	Translating	*
9	multiple	Two different types	Ad-hoc

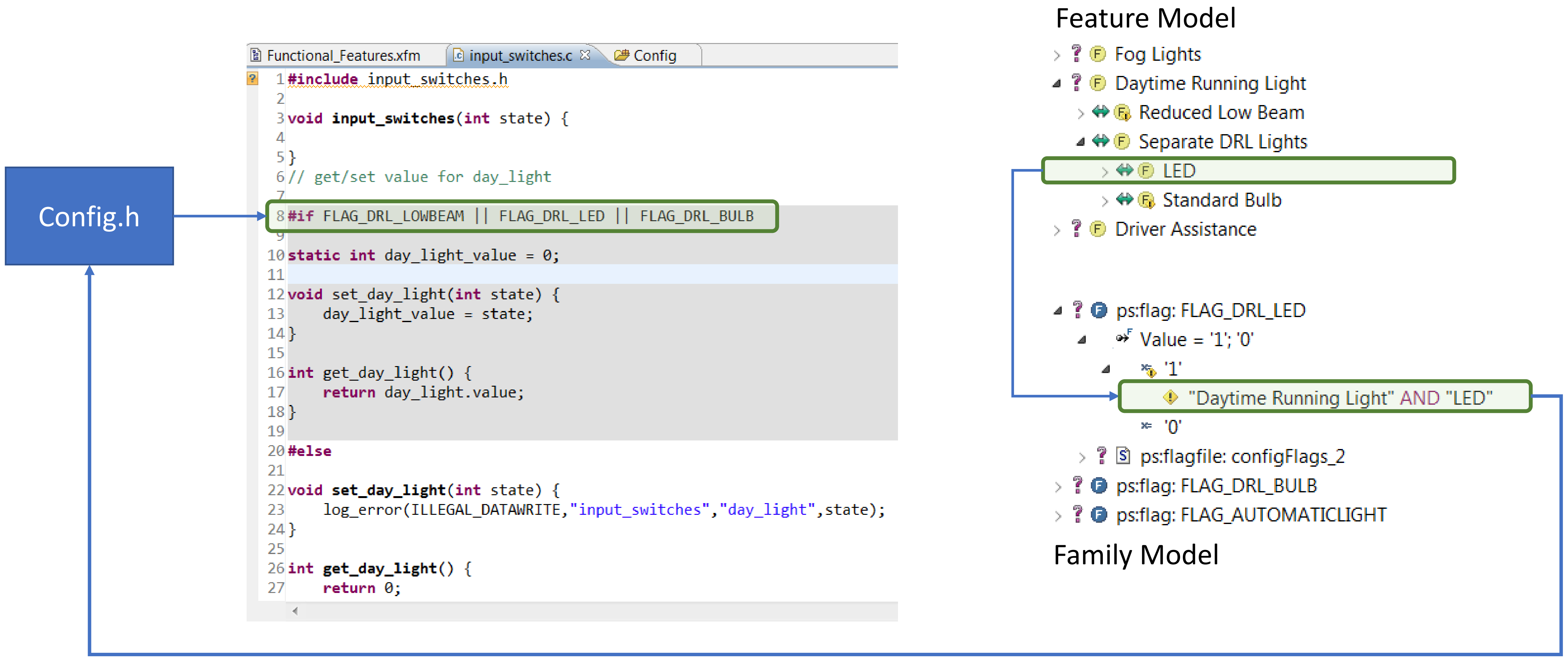
#	Challenge Description
5	Identify those new parts of the variant assets that shall be part of the feedback.
6	Identify spots in super set assets where variant specific changes shall be merged in.
7	Identify if a group of variant specific parts has to be split up and merged around a variation point.
8	Feedback of variant specific changes if assets have been translated into another language during transformation.
9	Feedback of variant specific changes if variant assets have been generated using a two-level transformation process with heterogeneous transformation types.

Discussion

Direct and indirect annotation of code switches

& identification of changes that are part of feedback

Feature-Controlled Code Switches (indirect, preserving, ad-hoc)

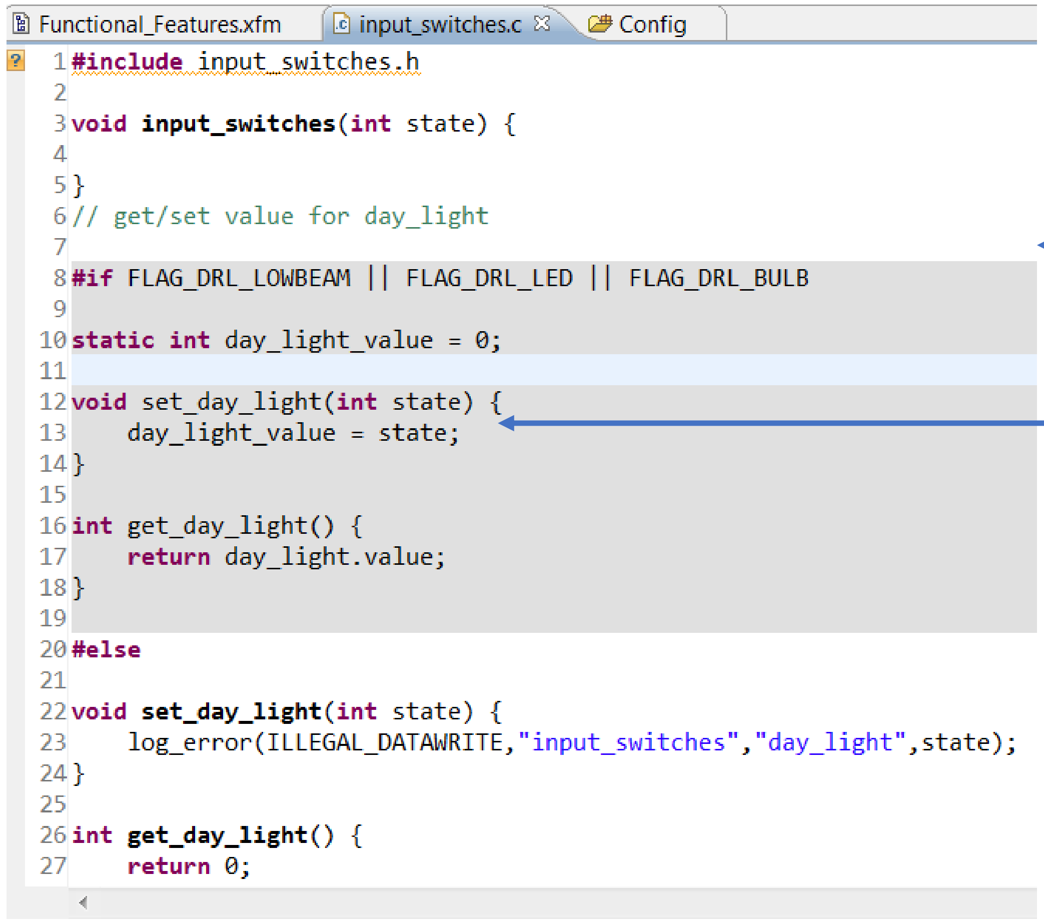


```
#include input_switches.h

void input_switches(int state) {

}
// get/set value for day_light

#if FLAG_DRL_LOWBEAM || FLAG_DRL_LED || FLAG_DRL_BULB

static int day_light_value = 0;

void set_day_light(int state) {
    day_light_value = state;
}

int get_day_light() {
    return day_light.value;
}

#else

void set_day_light(int state) {
    log_error(ILLEGAL_DATAWRITE,"input_switches","day_light",state);
}

int get_day_light() {
    return 0;
```

- Fog Lights
- Daytime Running Light
  - Reduced Low Beam
  - Separate DRL Lights
    - LED
    - Standard Bulb
- Driver Assistance

- ps:flag: FLAG_DRL_LED
  - Value = '1'; '0'
    - '1'
      - "Daytime Running Light" AND "LED"
    - '0'
  - ps:flagfile: configFlags_2
- ps:flag: FLAG_DRL_BULB
- ps:flag: FLAG_AUTOMATICLIGHT

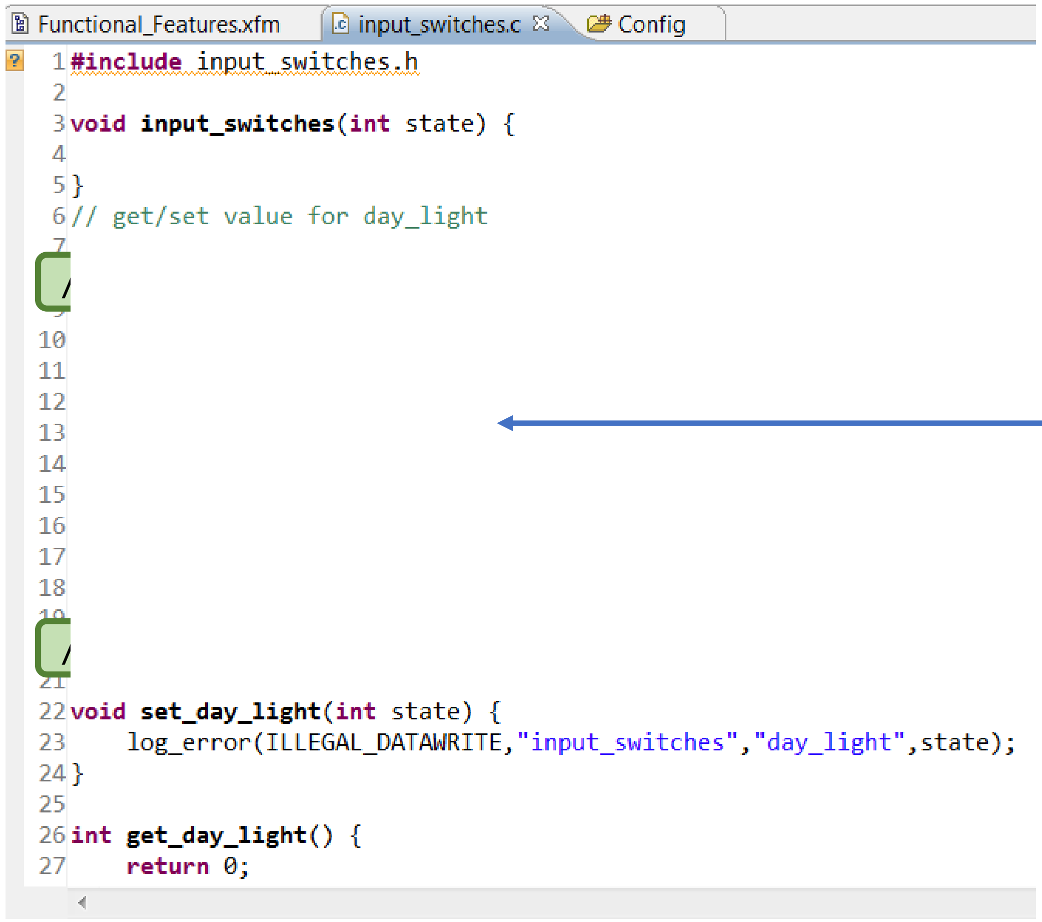
Feature-Controlled Code Switches (indirect, preserving, ad-hoc)



```
#include input_switches.h

void input_switches(int state) {

}
// get/set value for day_light

#if FLAG_DRL_LOWBEAM || FLAG_DRL_LED || FLAG_DRL_BULB

static int day_light_value = 0;

void set_day_light(int state) {
    day_light_value = state;
}

int get_day_light() {
    return day_light.value;
}

#else

void set_day_light(int state) {
    log_error(ILLEGAL_DATAWRITE,"input_switches","day_light",state);
}

int get_day_light() {
    return 0;
```

Add code in variant

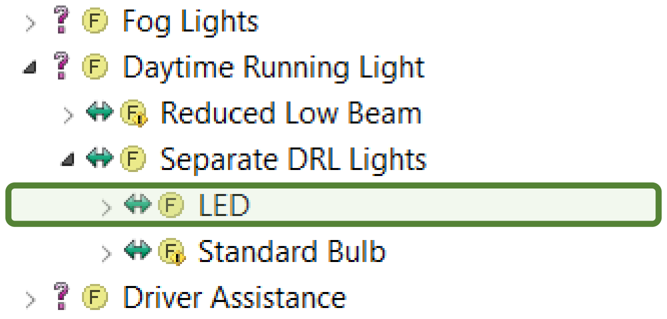
Feature-Controlled Code Switches (direct, negative, ad-hoc)



```
#include input_switches.h

void input_switches(int state) {

}
// get/set value for day_light
```

```
void set_day_light(int state) {
    log_error(ILLEGAL_DATAWRITE,"input_switches","day_light",state);
}

int get_day_light() {
    return 0;
```

Feature Model

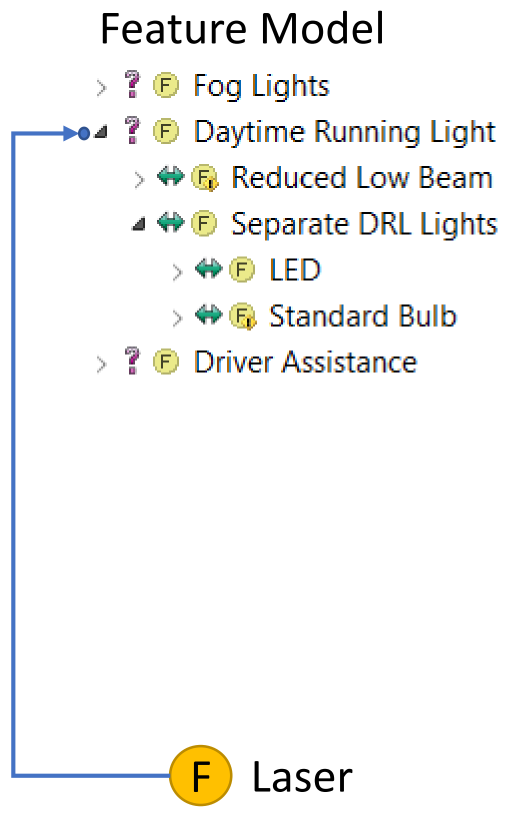


Add code in variant

Suggestion

Challenges:

- common or variable?
- Restrictable with existing features?
- Which changes need to be propagated?
  - Suggestion: named change sets
  - Suggestion: Variant-specific feature model



Thank You

The research leading to these results has received funding from the ITEA project REVaMP2. The work has been funded by the German Ministry for Education and Research (BMBF) under the funding ID 01IS16042G. The responsibility for the content rests with the authors.

pure·systems