Clone Detection

John Businge

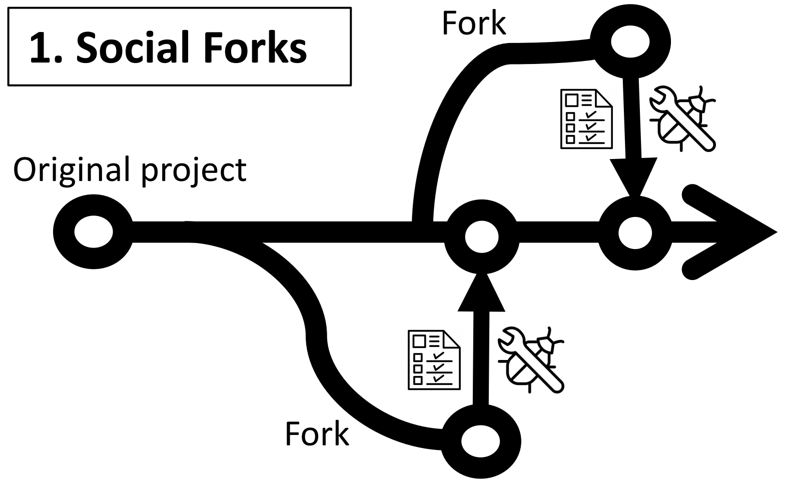
(john.businge@uantwerpen.be)

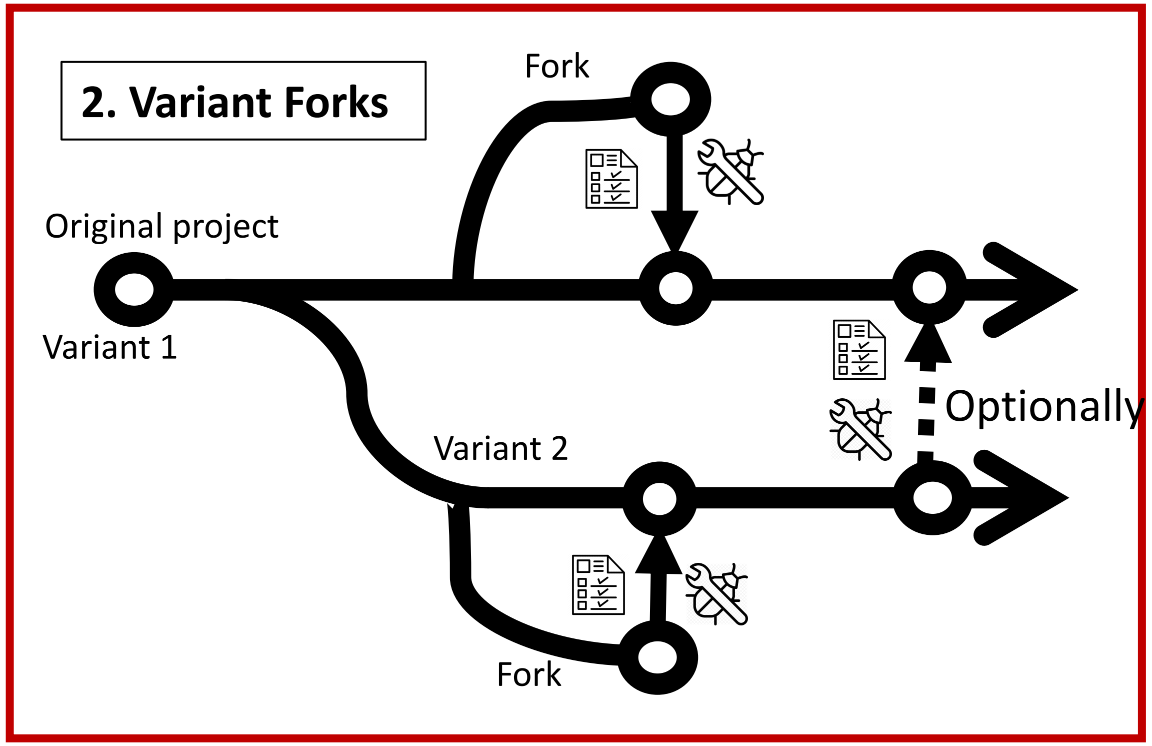
Universiteit
Antwerpen

When is code duplication okay?

- If duplicating something hits a deadline and not duplicating doesn’t, then I would rather deliver today and fix the tech debt tomorrow.
- There are two development paradigms
  - Clone-and-own (forking) - a new variant of a software system is created by copying and adapting an existing variant
  - Software product line - which consists of a set of similar software products with well-defined commonalities and variabilities.

Clone-and-own





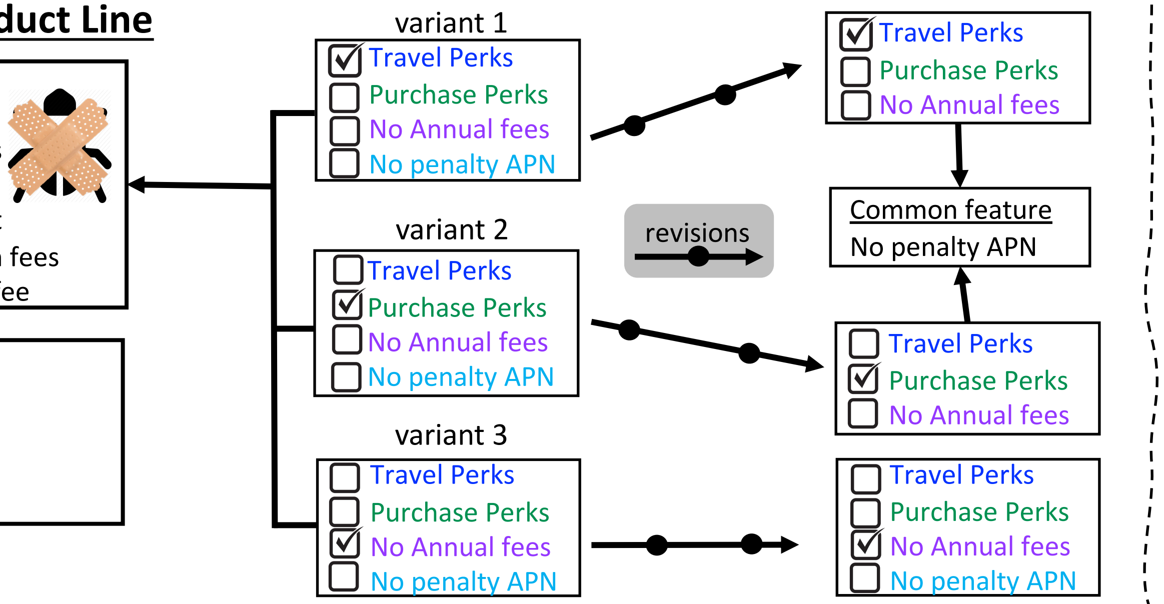
Interest

Engineering of multi-variant software systems (credit card variants)

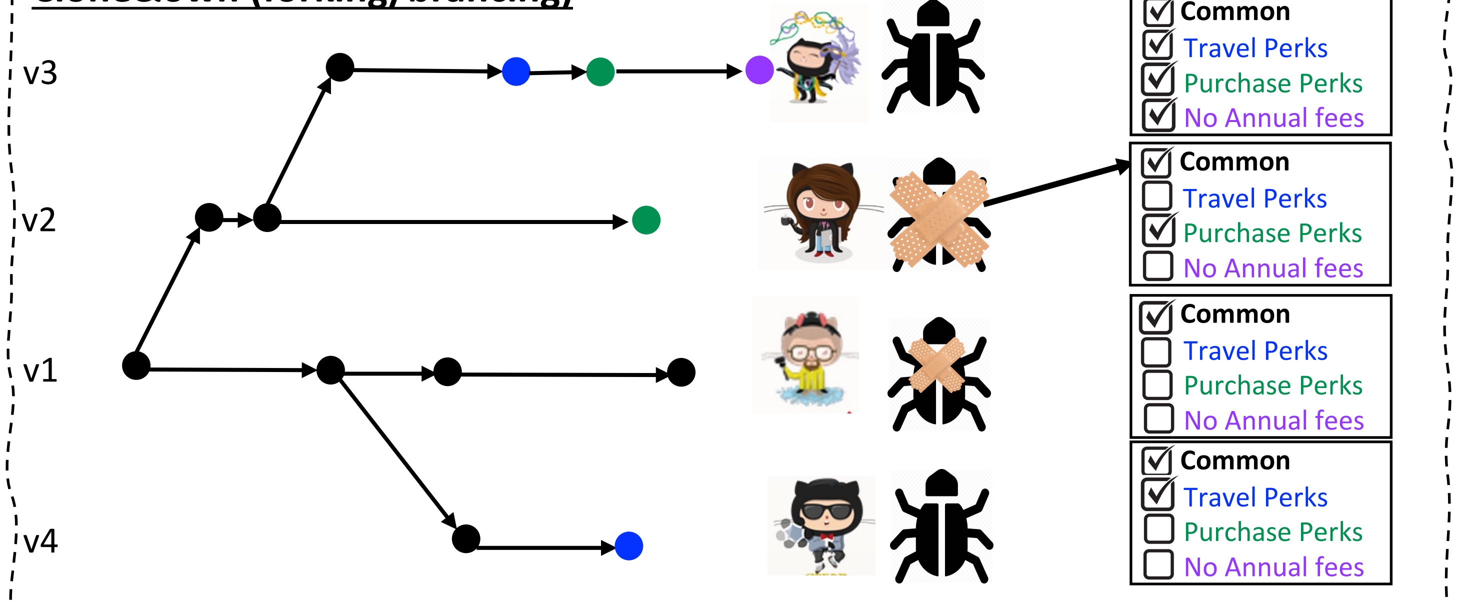
Software Product Line

Common features
- Investiment Benefits
- Credit Boosting Tools
- Big sign-up bonuses
- Zero percent interest
- No foreign transction fees
- No balance transfer fee

Variable Features
- Travel Perks
- Purchase Perks
- No Annual fees
- No penalty APN



Clone&own (forking/brancing)



Benefites
- Enforces systemmatic reuse
- Easy to fix bugs
- Scales easily (variants)
- Common developers (easy to coodinate)
- Saves long-term costs, time and effort (easy to manage)

Limitations
- Upfront heavy investiment
- Limited developer independency
- Not worth with few vatiants

Industry setting (closed source)

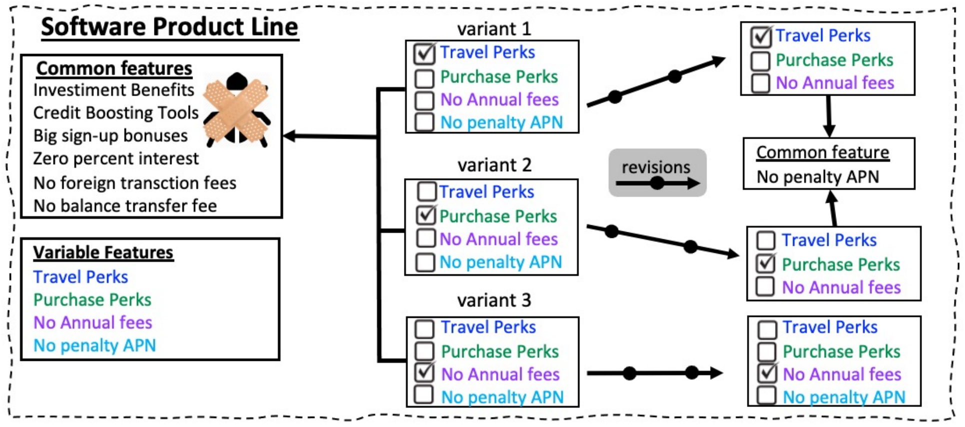
Benefits
- Flexibility in variant creation
- Limited initial costs
- Developer independency

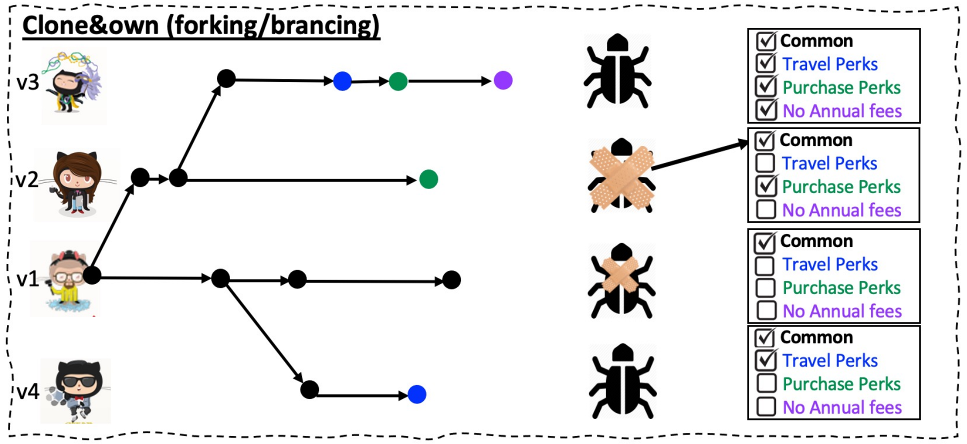
Limitations
- Redudancy/effort duplication
- Does not scale
- Misses opportunity (missed patches)
- Diverse developers (difficult to coodinate)
- Expensive to engineer (SPL)

Social coding platforms (open source)

Engineering of multi-variant software systems

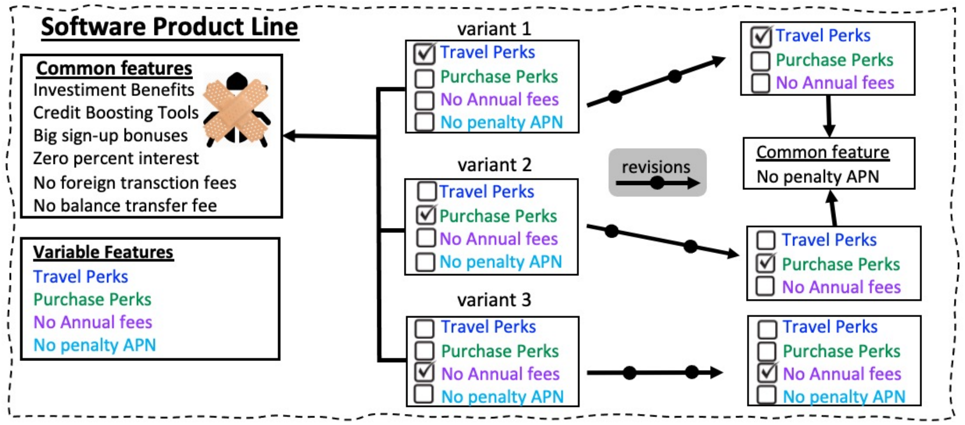
State-of-practice





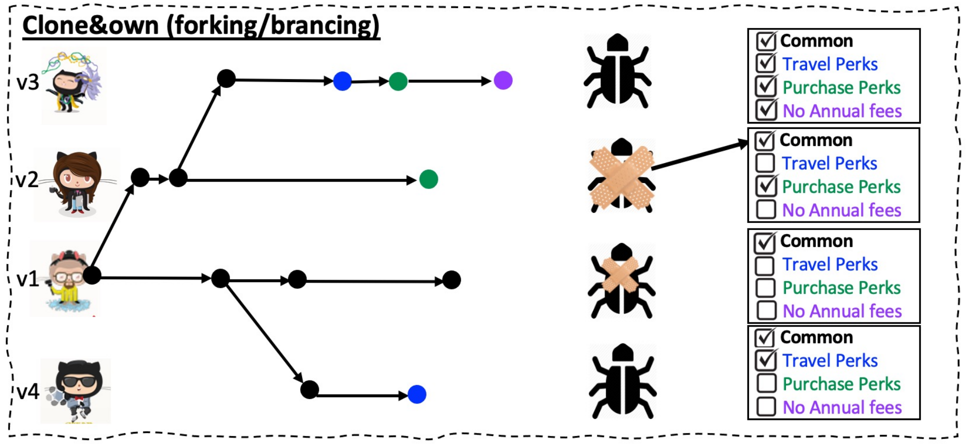
Engineering of multi-variant software systems

State-of-practice

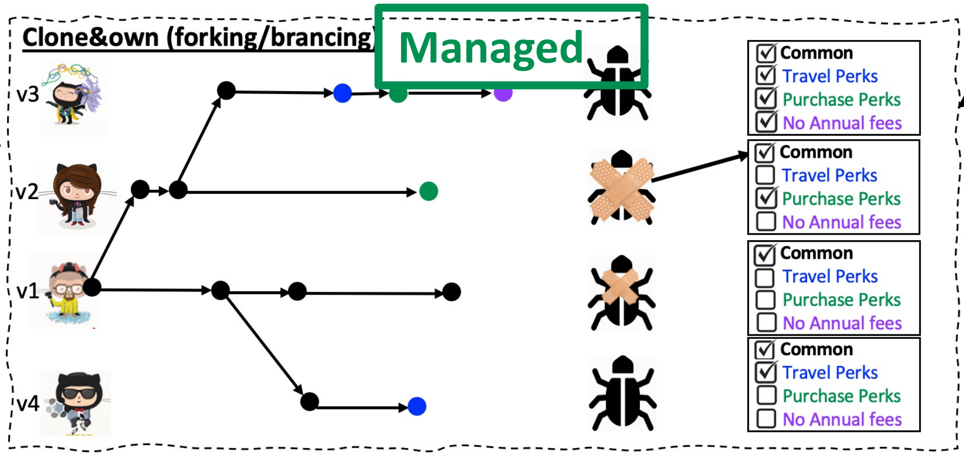


Researchers

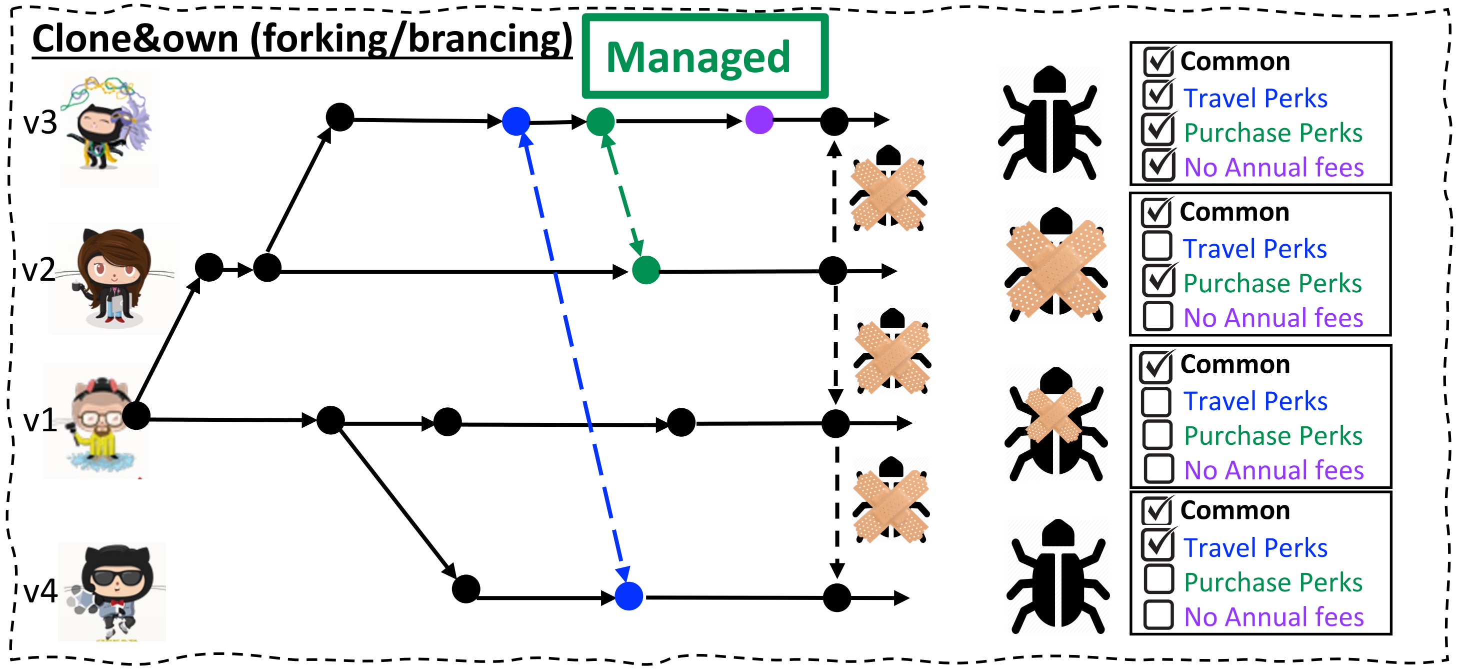
Exploring middle-grounds



Limitations



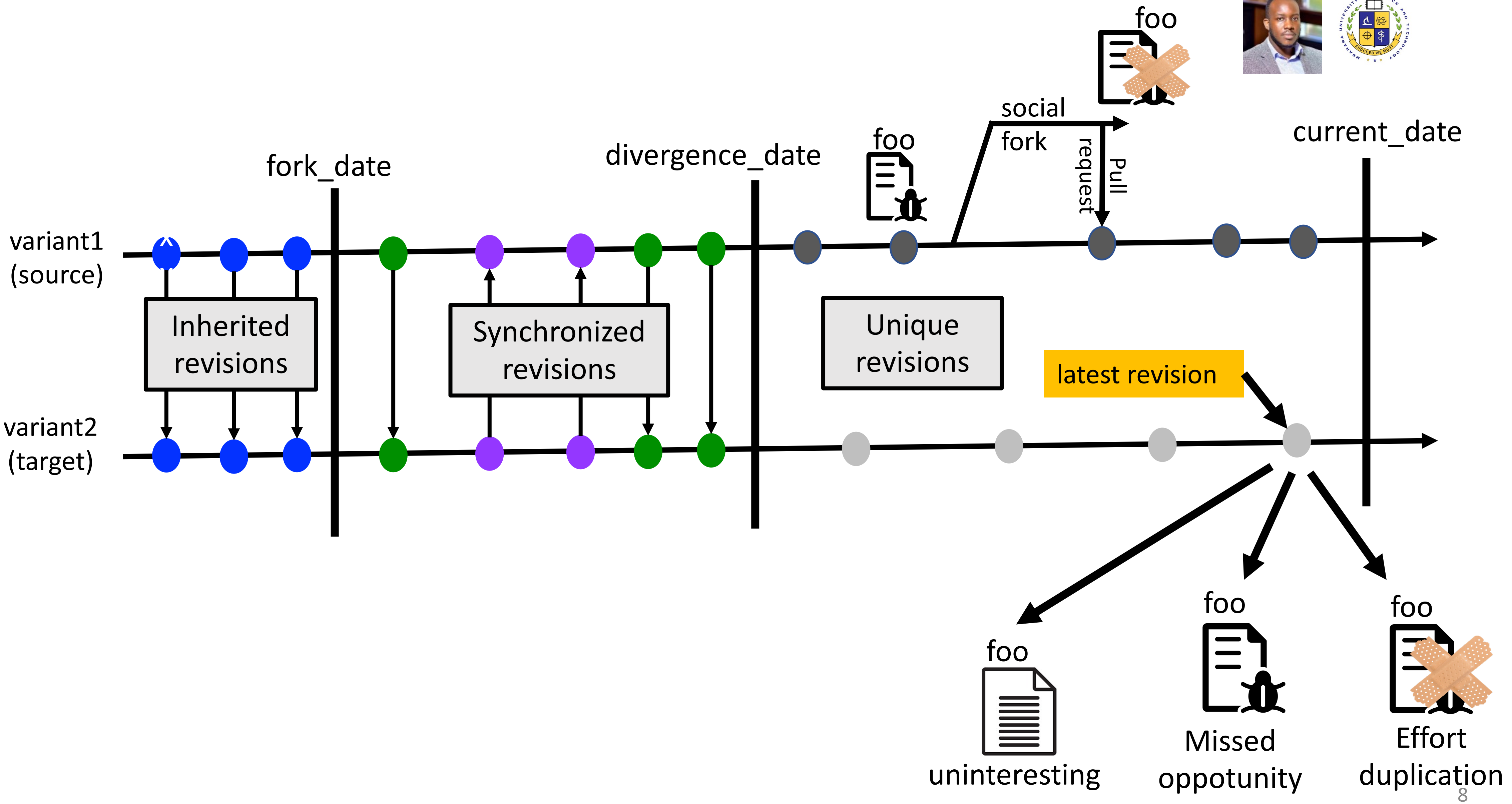
Engineering of multi-variant software systems



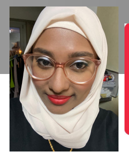
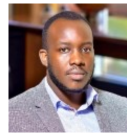
<-- - - - - - - -> synchronization

- - - - - - - - -> integration (cherry-picking)

Illustration of diverged variants

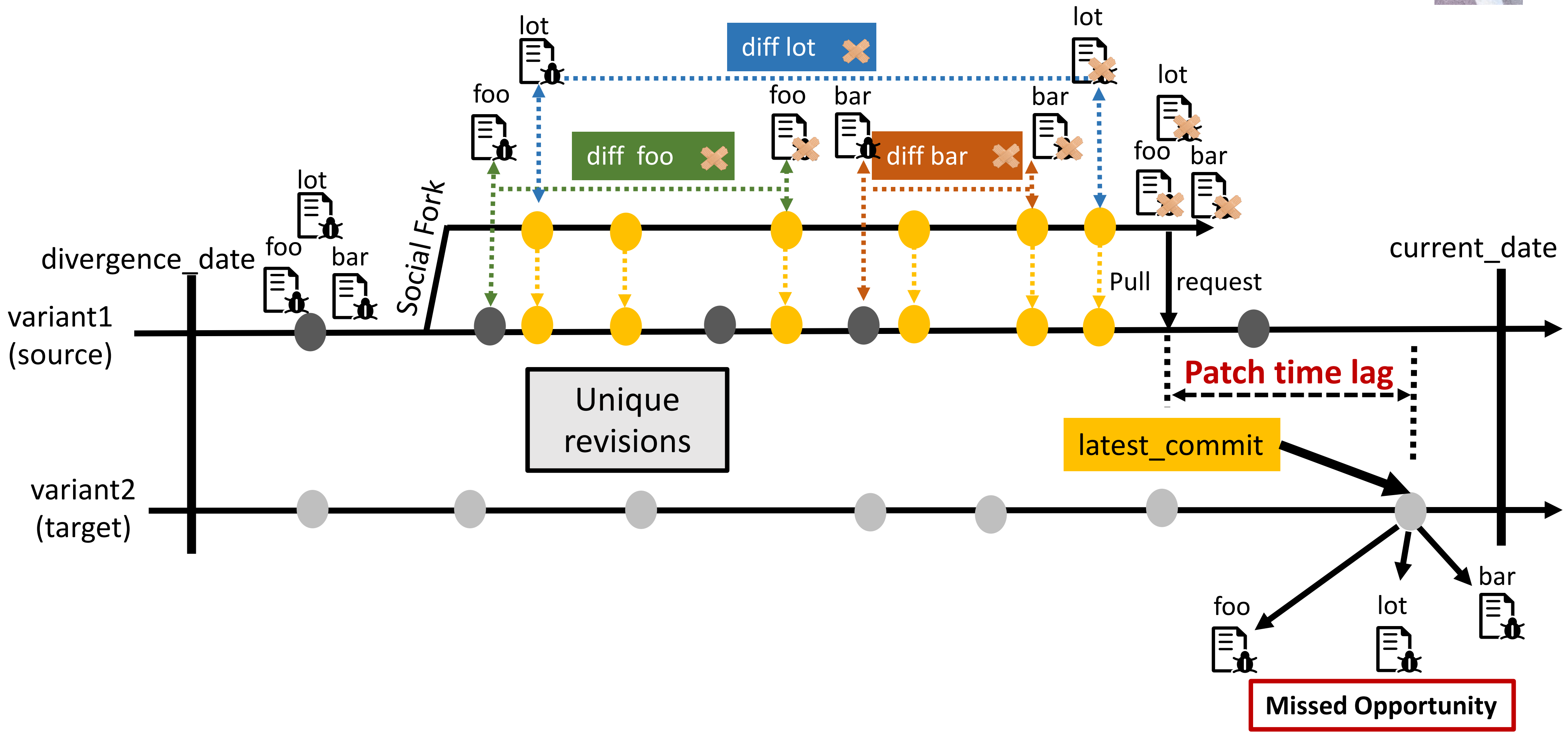


Engineering of multi-variant software systems

University of Antwerp

Illustration of diverged variants – Zoomed patch



Concrete Example – Missed Opportunity

Buggy snippet from upstream

```
                return;
            }
        } while (p < (uint16_t *)SYMVAL(__eeprom_workarea_end__));
        flashend = (uint32_t)((uint16_t *)SYMVAL(__eeprom_workarea_end__) - 1);
    }
```

Buggy line

Patched snippet from upstream

```
                return;
            }
        } while (p < (uint16_t *)SYMVAL(__eeprom_workarea_end__));
        flashend = (uint32_t)(p - 1);
    }
```

Patched line

Diff for patch in upstream

```
@@ -363,7 +363,7 @@

        } while (p < (uint16_t *)SYMVAL(__eeprom_workarea_end__));
-       flashend = (uint32_t)((uint16_t *)SYMVAL(__eeprom_workarea_end__) - 1);
+       flashend = (uint32_t)(p - 1);
```

Hunk

`Latest_commit` snippet from divergent fork

```
                return;
            }
        } while (p < (uint16_t *)SYMVAL(__eeprom_workarea_end__));
        flashend = (uint32_t)((uint16_t *)SYMVAL(__eeprom_workarea_end__) - 1);
    }
```

Buggy line

Concrete Example – Effort Duplication

Buggy snippet from upstream

```
# http://ss64.com/nt/syntax-esc.html
_escape_re = re.compile(r'(?<!\^)[&<>]|(?<!\^)\^(?![&<>\^])')
_escaper = partial(_escape_re.sub, lambda m: '^' + m.group(0))
```

Buggy line

Patched snippet from upstream

```
# http://ss64.com/nt/syntax-esc.html
_escape_re = re.compile(r'(?<!\^)[&<>]|(?<!\^)\^(?![&<>\^])|(\|)')
_escaper = partial(_escape_re.sub, lambda m: '^' + m.group(0))
```

Patched line

Diff for patch in upstream

```
    @@ -24,7 +24,7 @@

     # http://ss64.com/nt/syntax-esc.html
-    _escape_re = re.compile(r'(?<!\^)[&<>]|(?<!\^)\^(?![&<>\^])')
+    _escape_re = re.compile(r'(?<!\^)[&<>]|(?<!\^)\^(?![&<>\^])|(\|)')
     _escaper = partial(_escape_re.sub, lambda m: '^' + m.group(0))
```

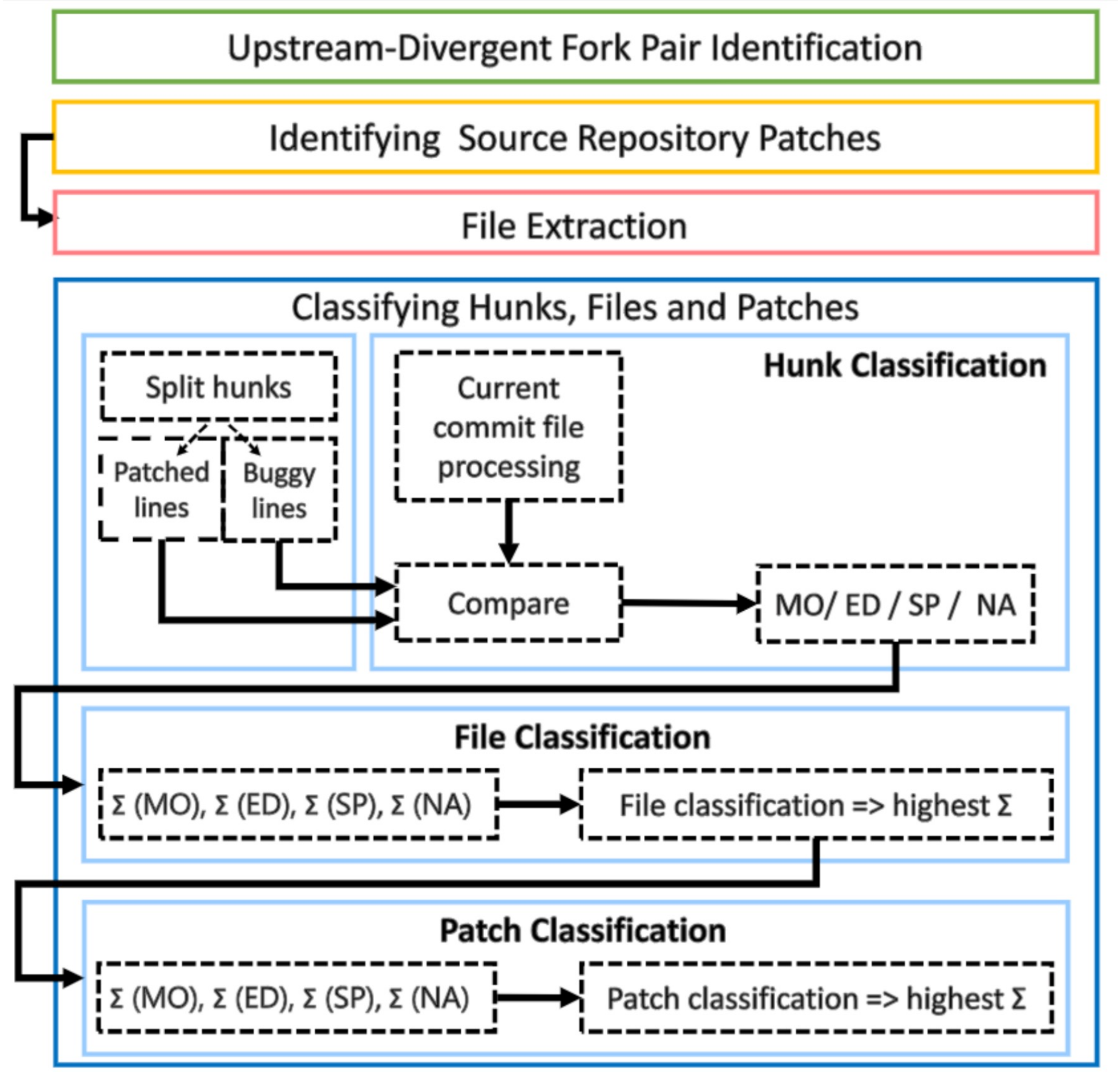
Hunk

`Latest_commit` snippet from divergent fork

```
# http://ss64.com/nt/syntax-esc.html
_escape_re = re.compile(r'(?<!\^)[&<>]|(?<!\^)\^(?![&<>\^])|(\|)')
_escaper = partial(_escape_re.sub, lambda m: '^' + m.group(0))
```

Patched line

Tool - PeReco


Upstream-Divergent Fork Pair Identification
Identifying Source Repository Patches
File Extraction
Classifying Hunks, Files and Patches
Split hunks
Patched lines
Buggy lines
Current commit file processing
Hunk Classification
Compare
MO/ ED / SP / NA
File Classification
Σ (MO), Σ (ED), Σ (SP), Σ (NA)
File classification => highest Σ
Patch Classification
Σ (MO), Σ (ED), Σ (SP), Σ (NA)
Patch classification => highest Σ

Let us play with the tool