Exploring the non-functional properties of model transformation techniques used in industry

Ronan Barrett, Ericsson AB

Some General Background

abcdefgh

ÛÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ

ĢģĪīĮįİĶ
ŪūŮůŰű

ΆΈΉΊΰα

СТУФХ
УФХЦЧ
ҐәǽẀẁ

Ericsson in Numbers

40% of all Traffic

2.5 billion

180 Countries

110,000
Employees

5th Largest

50 billion

Modelling In Ericsson

O&M Modelling

Network Modelling

System Modelling

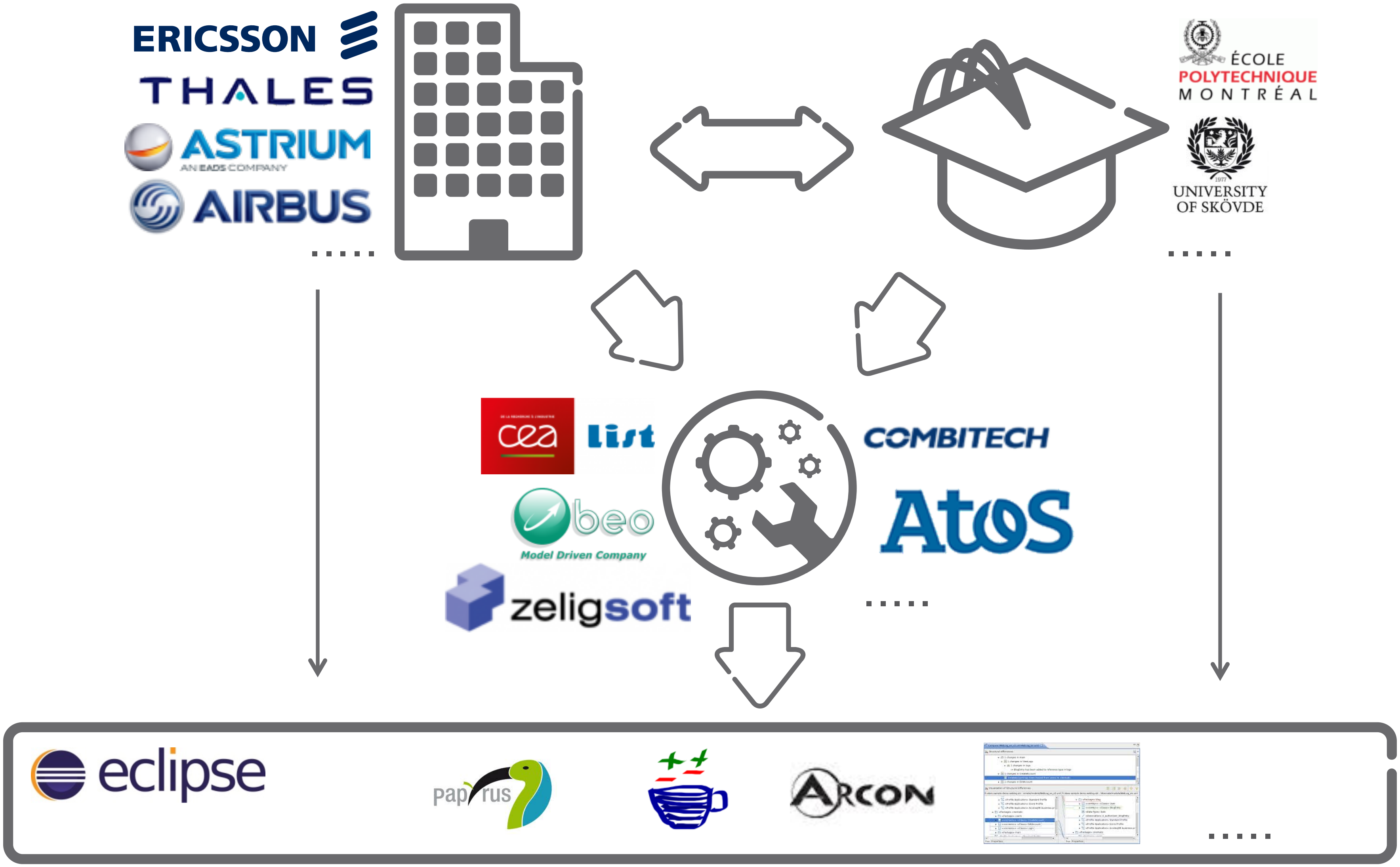
Business Process Modelling

Real-Time Modelling

Executable Models

PolarSys

Open Source Tools for Embedded Systems



Some Specific Background

abcdefgh

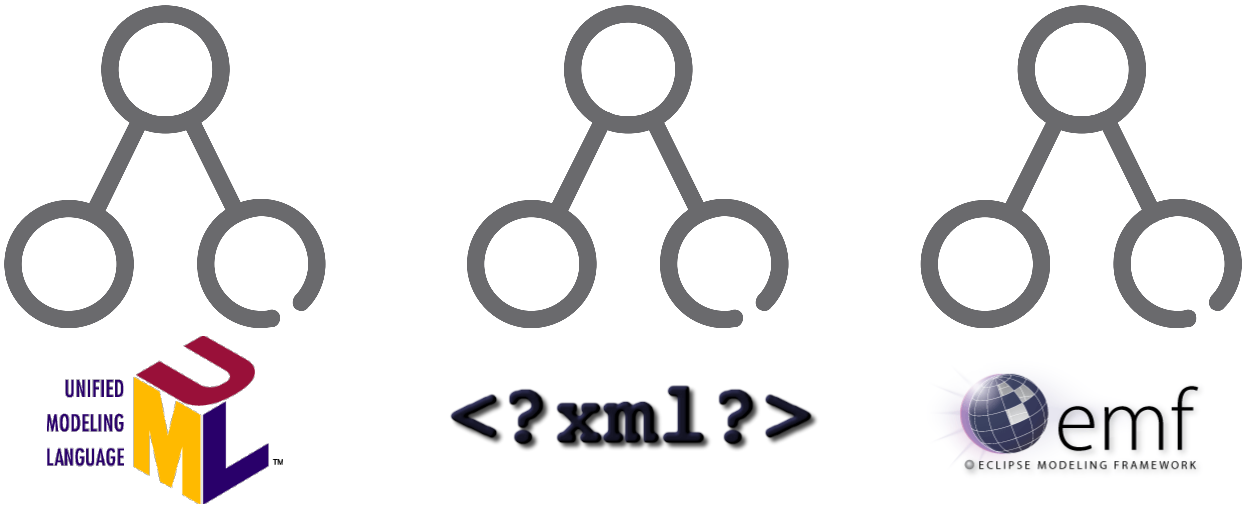
ÛÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ

ĢĢĪīĮįİĶ
ŪūŮůŰ

ΆΈΉΊΰα

СТУФХ
УФХЦЧ
ƬəǽẀẁ

Modeling Context



abcdefgh

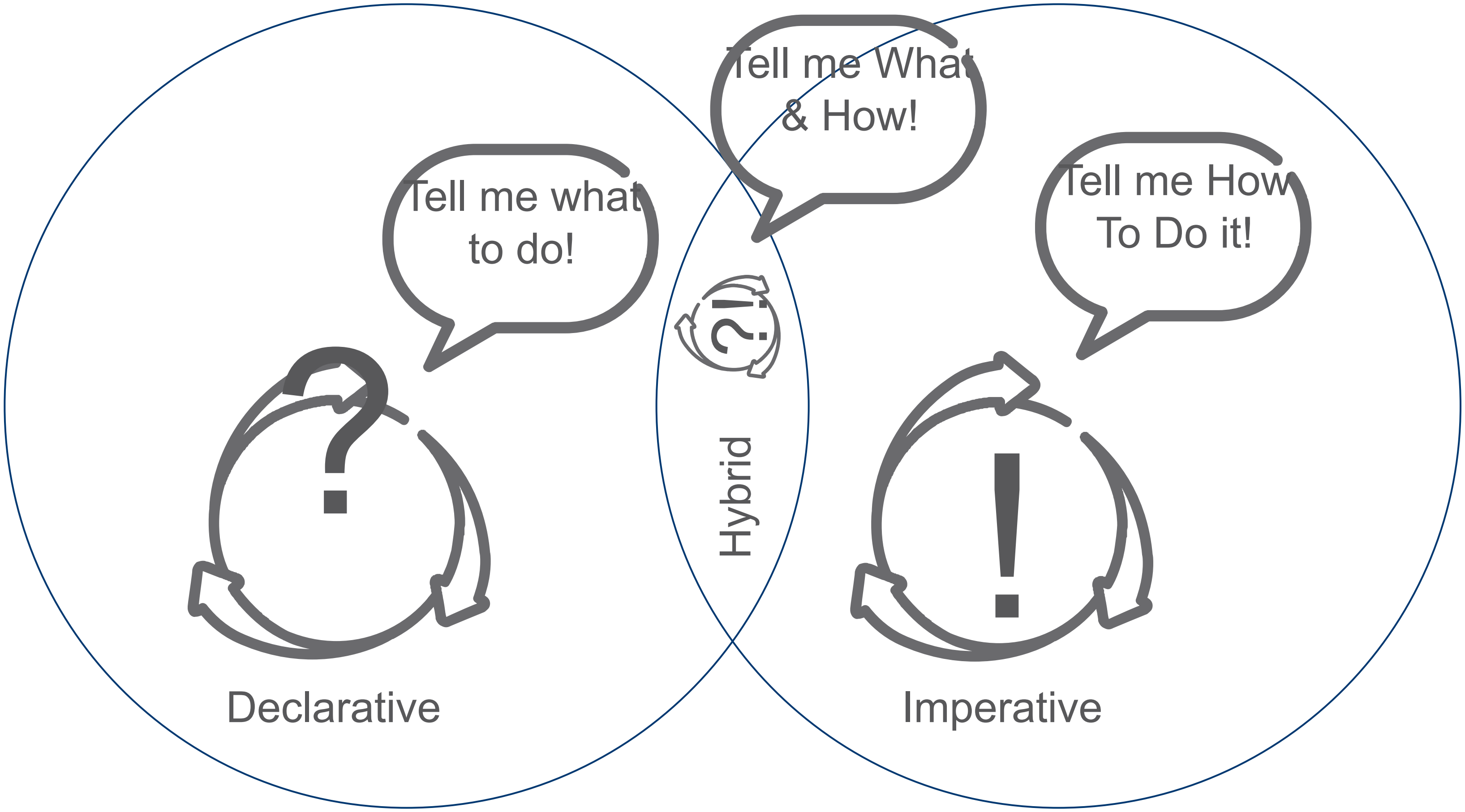
ÛÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ

ĠĢģĪīĮįİĶ
ŪūŮůŰű

ΆΈΉΊΰα

СТУФХ
УФХЦЧ
ҐəǽẀẁ

Transformation Styles



abcdefgh

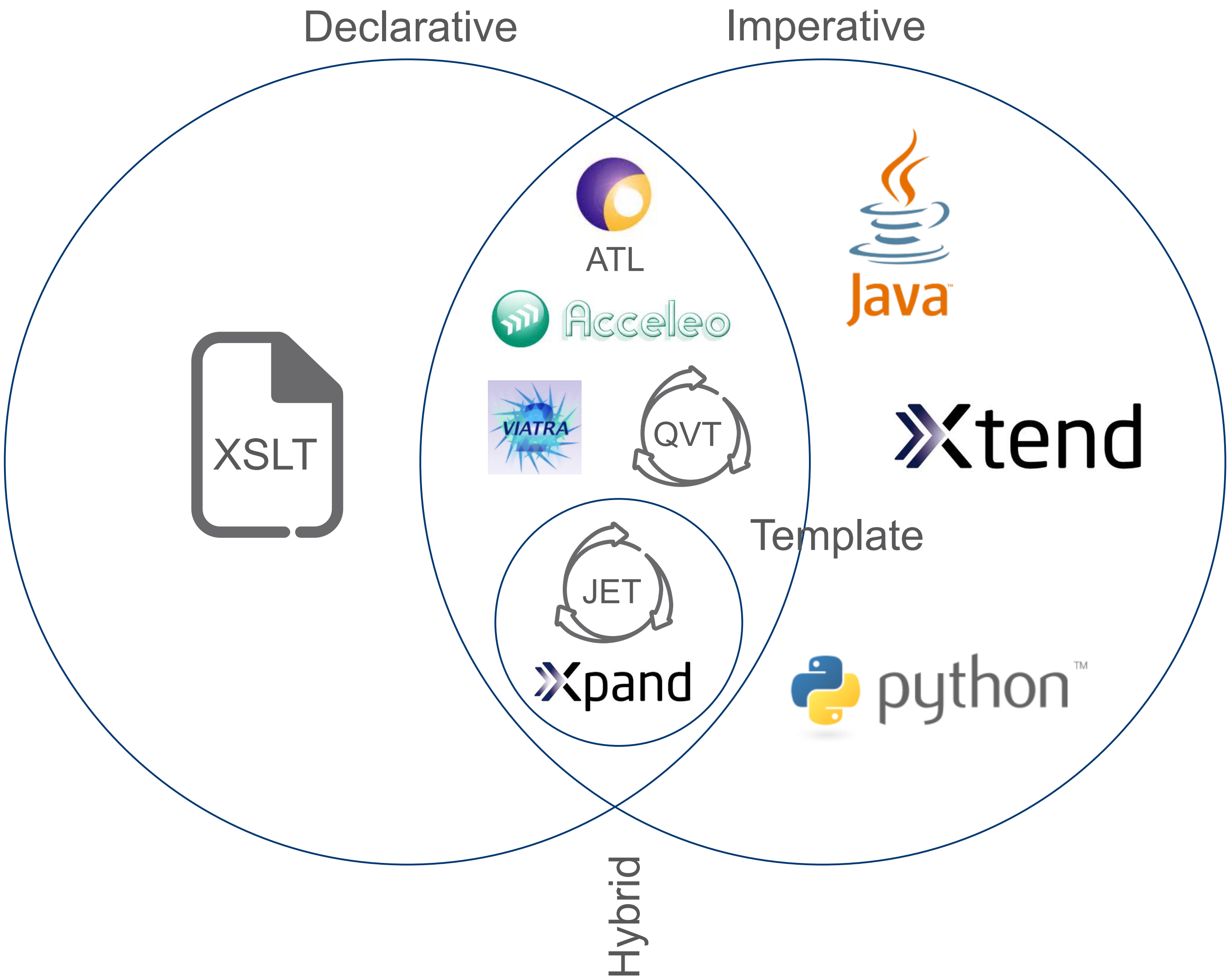
ÜÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ

ĢĢĪĮįİĶ
ŪŪŮůŰ

ΆΈΉΊΰα

СТУФХ
УФХЦЧ
ҐəǽẀẁ

Transformation Languages



abcdefgh
ÜÜÝÞßà
cĆĈĎďĐ
ŒœŔŕŘŗ
¨Ww
GĠĪĮįĶ
ÚÛÜÜ
ΆΈΉΰα
СТУФХ
УФХЦЧ
ҐәæWw

[Let's Explore Some NFP(s)]

abcdefgh

ÜÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ

ĢĢĪĮįİĶ
ŪŪŮůŰ

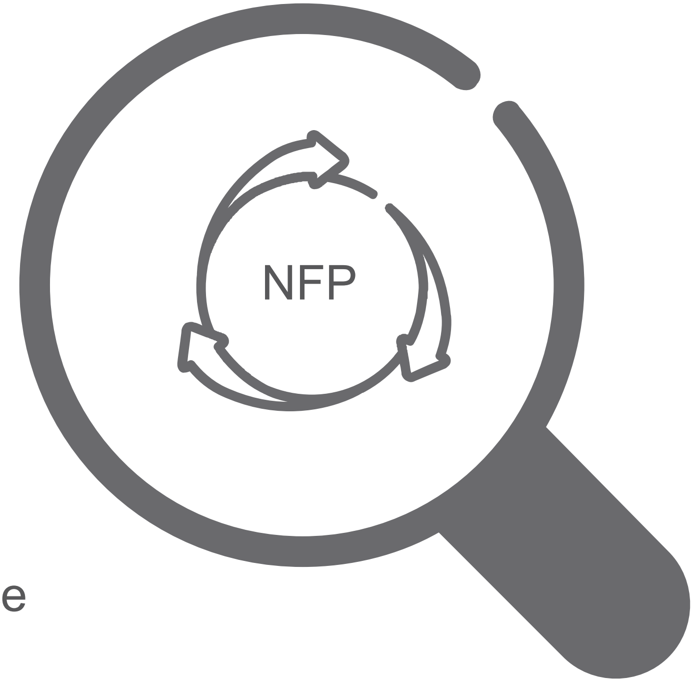
ΆΈΉΊΰα

СТУФХ
УФХЦЧ
ŤəǽẀẁ

Table of Contents

- Identify NFP(s)
- Wanted Position
- Enumerate NFP(s)
- Current Position
- Improvement Areas

Identified NFP(s)



NFP

Readability

Maintainability

Scalability

Performance

Usability

Deploy-ability

Testability

NFP Wanted Position

Code readable/auditable by developers/systems guys

Transformation code should be 100% verifiable

Changes in input model shouldn't require full re-transformation

Bugs fixed quickly and without side-effect

Parallelizable transformation parts should be executed in parallel

Code base can grow without constant refactoring

Implementer should *not* have to learn a new obscure language

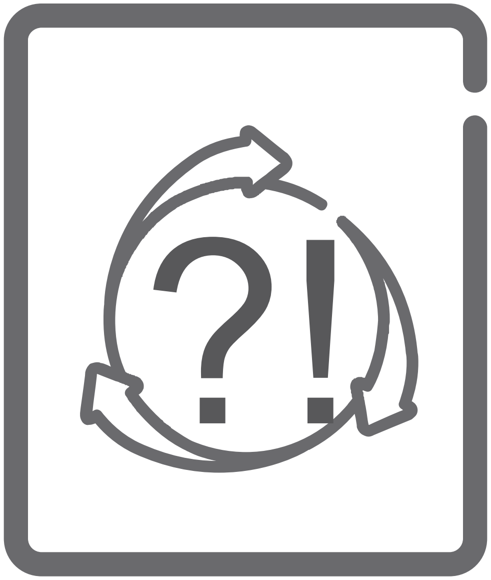
Execution doesn't consume all resorces as model size gets "large"

Transformation code should be runnable "anywhere" with GUI or without

No developer bottleneck for transformation code

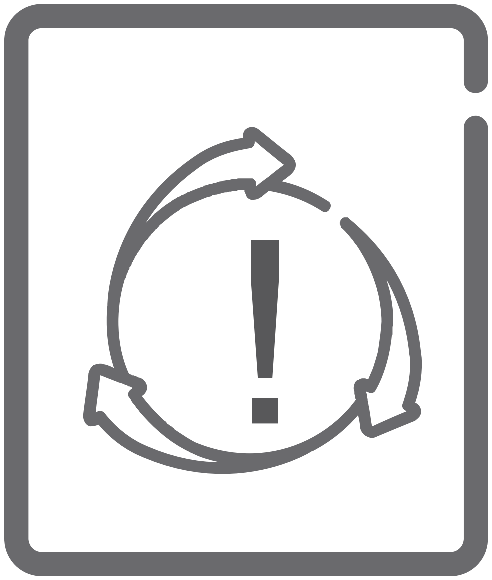
Readability

Declarative/Hybrid



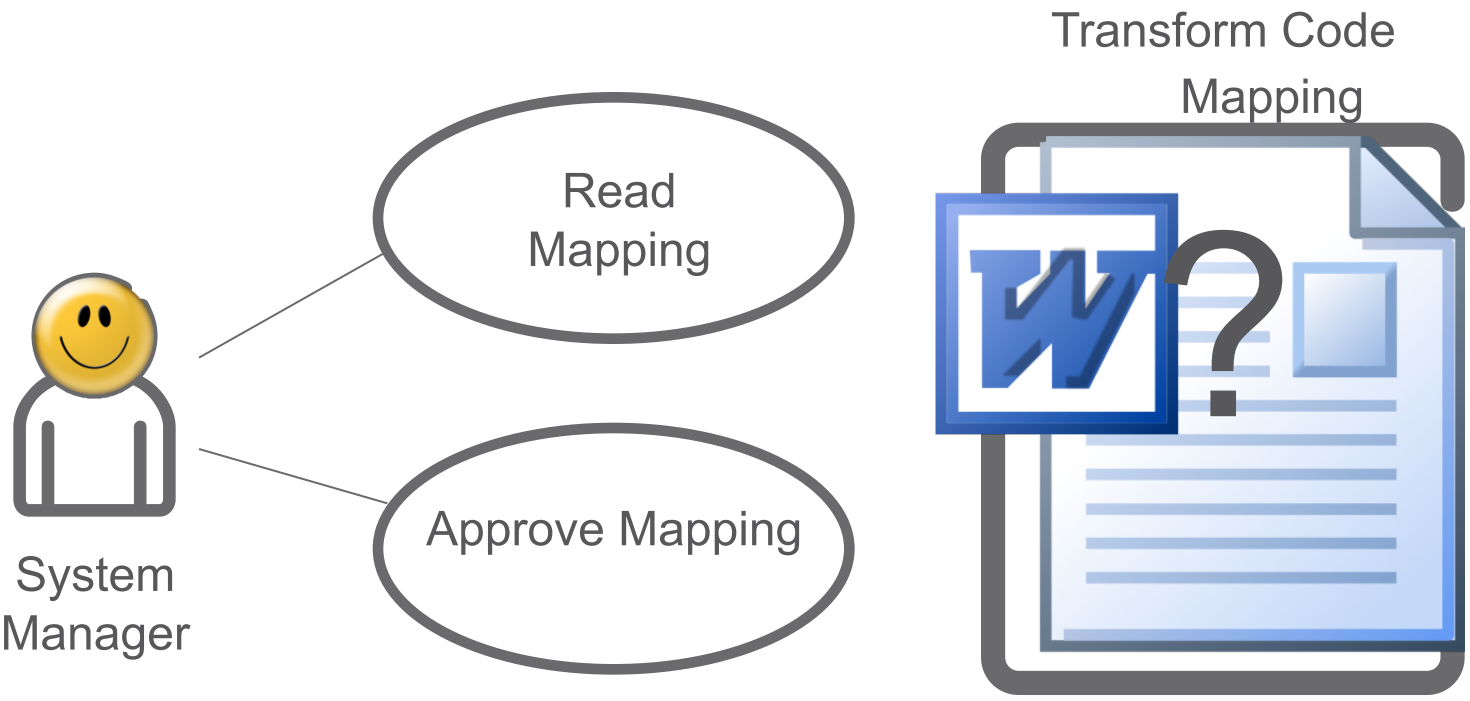
Often succinct
Often easy to read
Must know language

Imperative



Can be verbose
Laborious to read
Well known language

Readability Issue



Transform Code Mapping

Read Mapping

Approve Mapping

System Manager

abcdefgh

ÜÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ

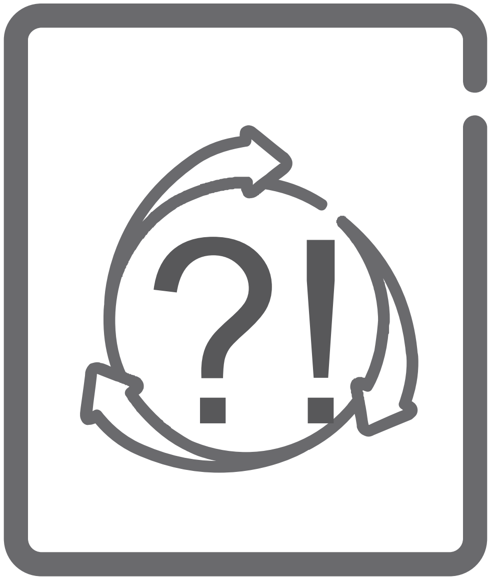
ĢĢĪĮįİĶ
ŪŪŮůŰ

ΆΈΉΊΰα

СТУФХ
УФХЦЧ
ҐəǽẀẁ

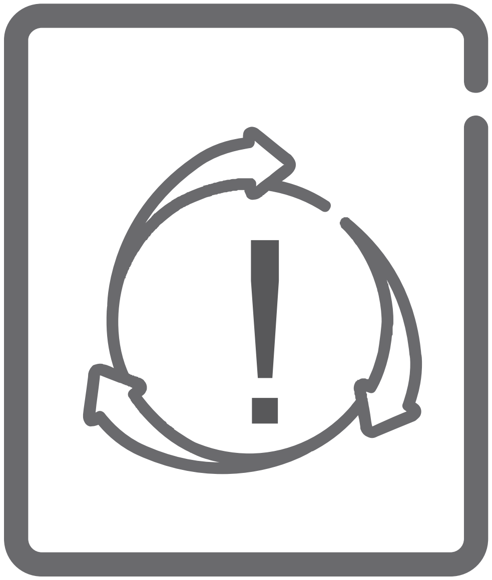
Maintainability

Declarative/Hybrid



Fixes rarely cause refactoring (fast/no side-effect)
Must know language

Imperative



Fixes often require refactoring (slower/often side-effects)
Well known language

abcdefgh

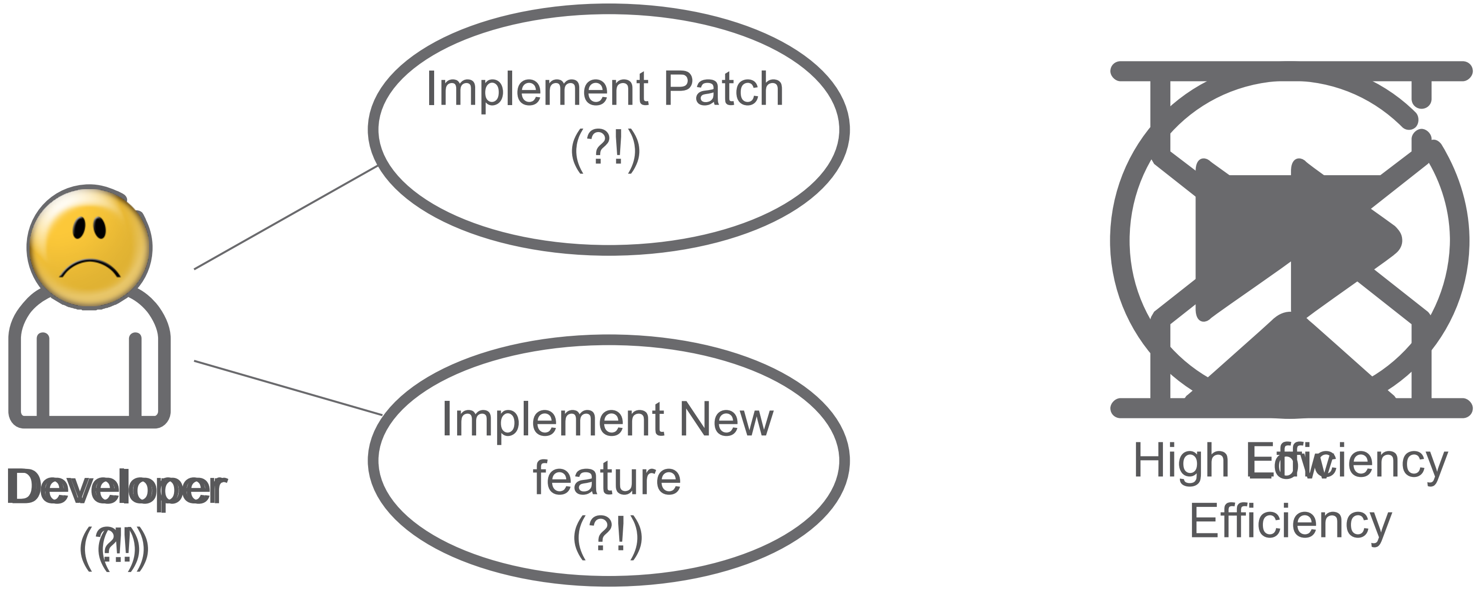
ÛÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ

ĢĢĪīĮįİĶ
ŪūŮůŰ

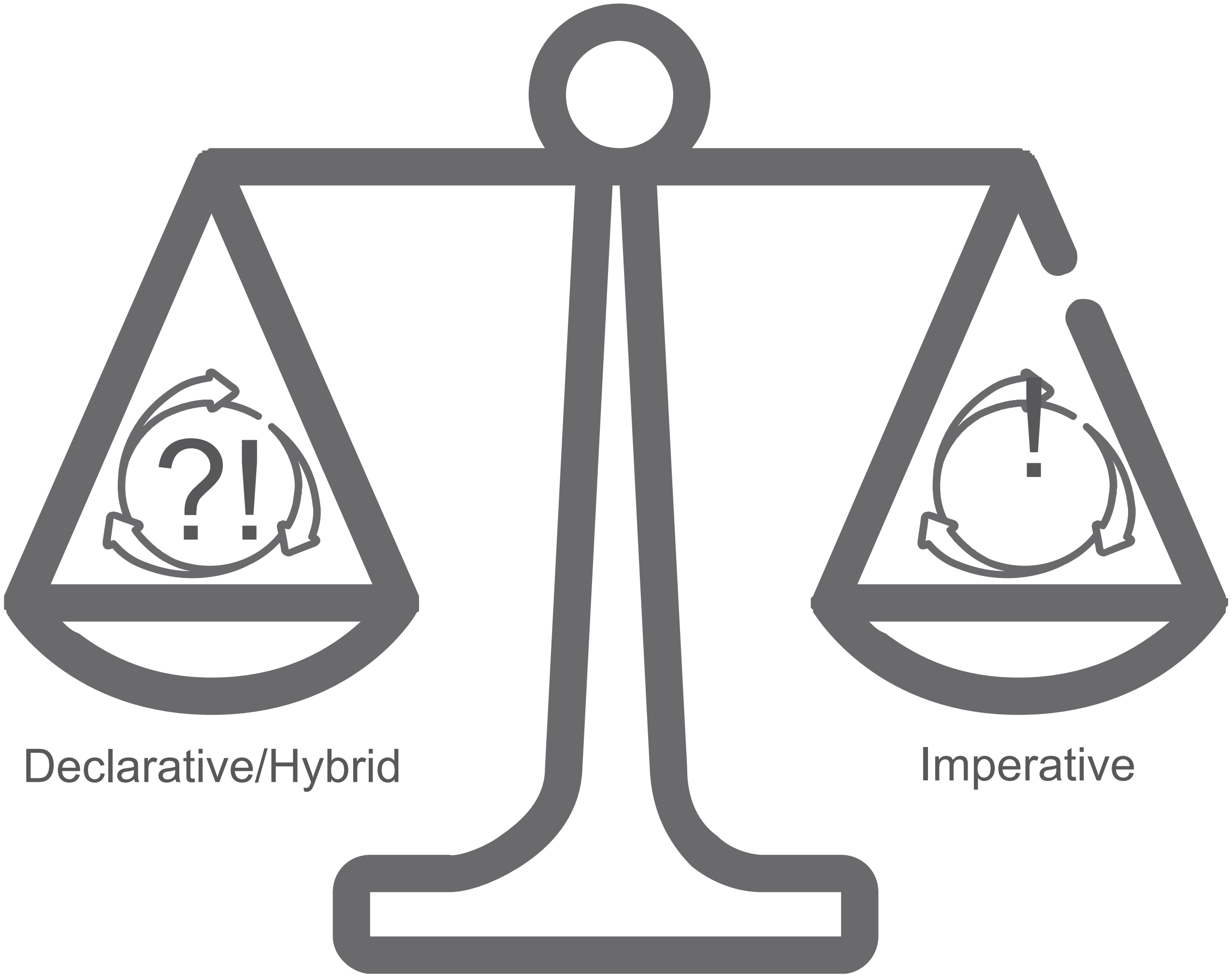
ΆΈΉΊΰα

СТУФХ
УФХЦЧ
ƮəǽẀẁ

Maintainability Issue



Scalability/Performance



Declarative/Hybrid

Imperative

abcdefgh

ÛÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ

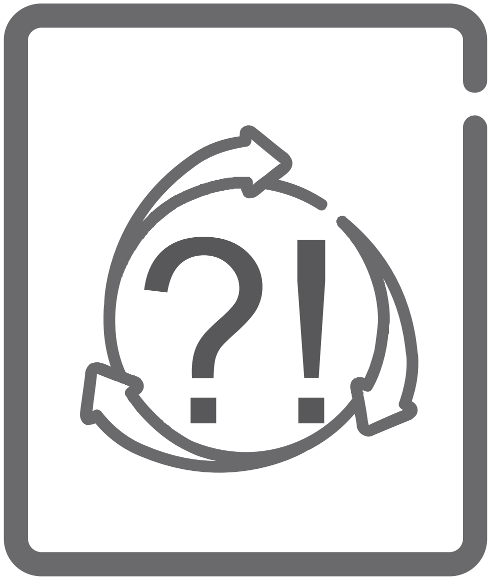
ĜĢĢĪĮįİĶ
ŪŪŮůŰ

ΆΈΉΊΰα

СТУФХ
УФХЦЧ
ґəǽẀẁ

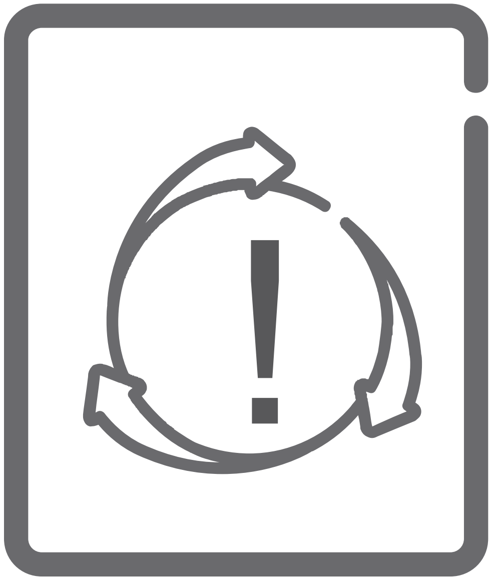
Scalability/Performance Issues

Declarative/Hybrid



Parallel execution
Incremental execution*
Infinite/Event execution

Imperative



Explicit coding required
to achieve similar
execution semantics

* Often only the declarative part can be incrementally executed

Usability

Community

Learning Resources

Development Environment

Debugging

abcdefgh

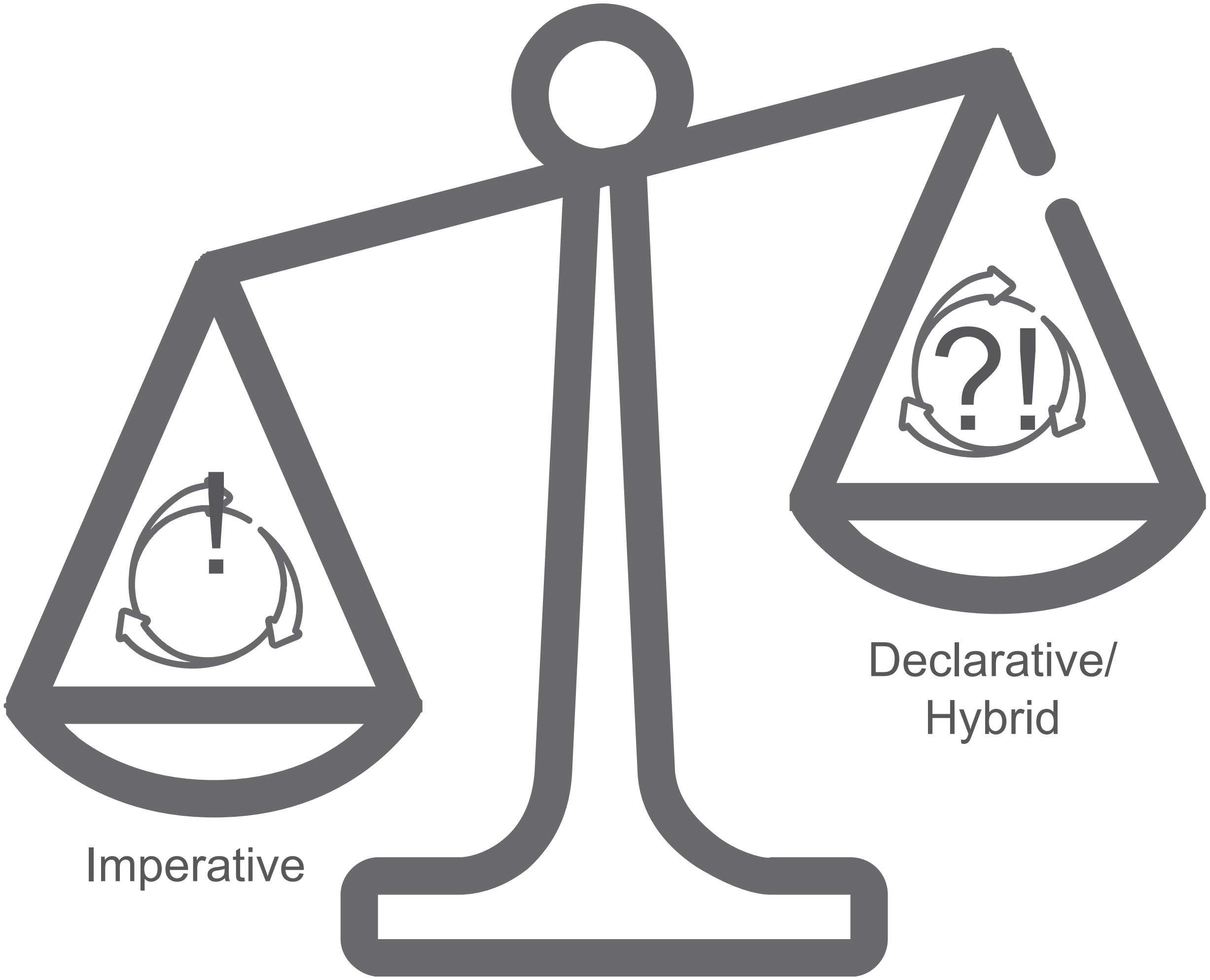
ÜÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ

ĢĢĪĪĮįİĶ
ŪŪŮůŰ

ΆΈΉΊΰα

СТУФХ
УФХЦЧ
ЃǝǽẀẁ

Usage in Industry

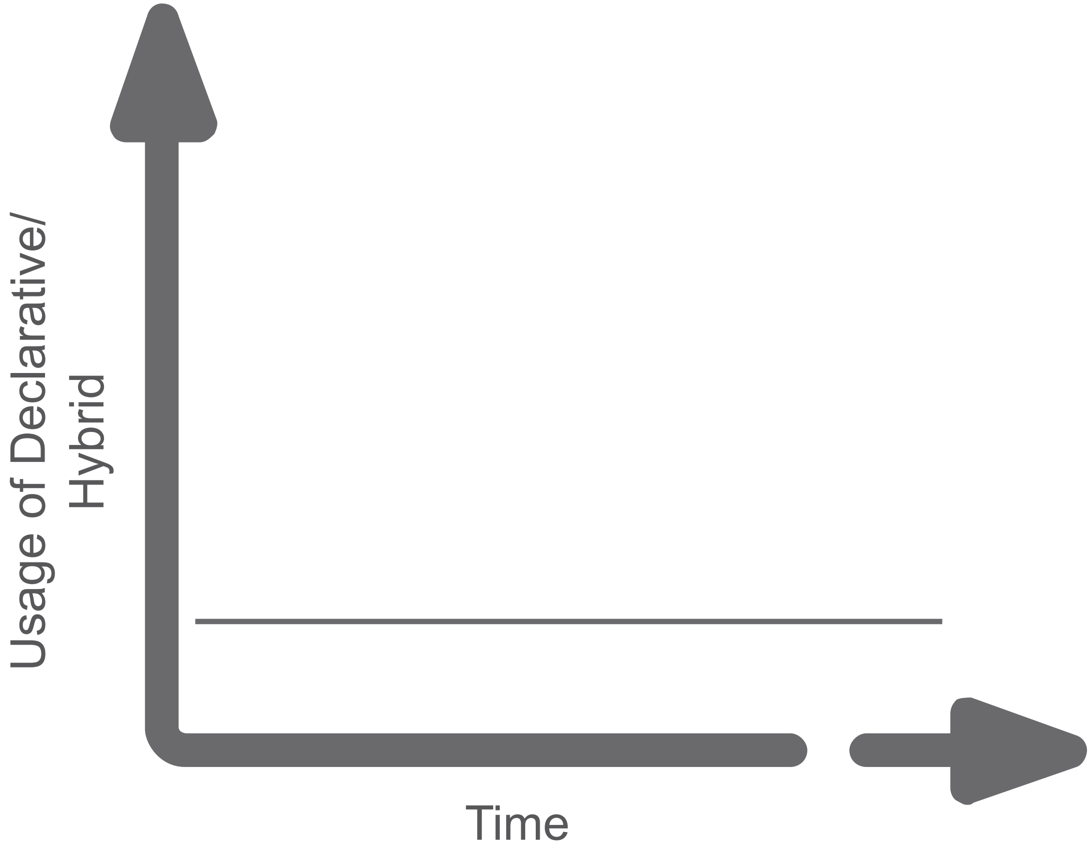


Imperative

Declarative/
Hybrid

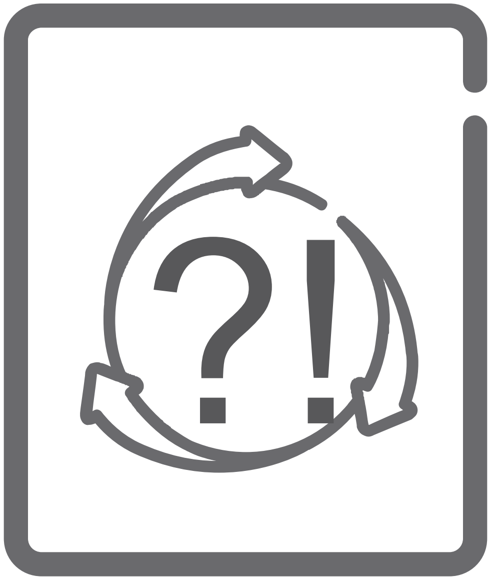
abcdefgh
ÛÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ
ĢĢĪīĮįİĶ
ŪūŮůŰ
ΆΈΉΊΰα
СТУФХ
УФХЦЧ
ЃəǽẀẁ

Usage Trend



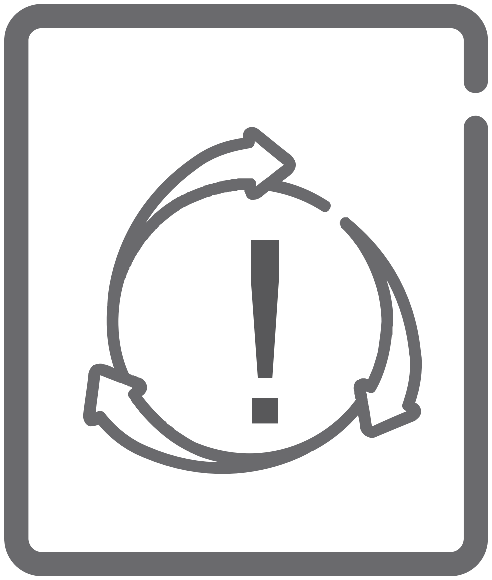
Usability Issues

Declarative/Hybrid



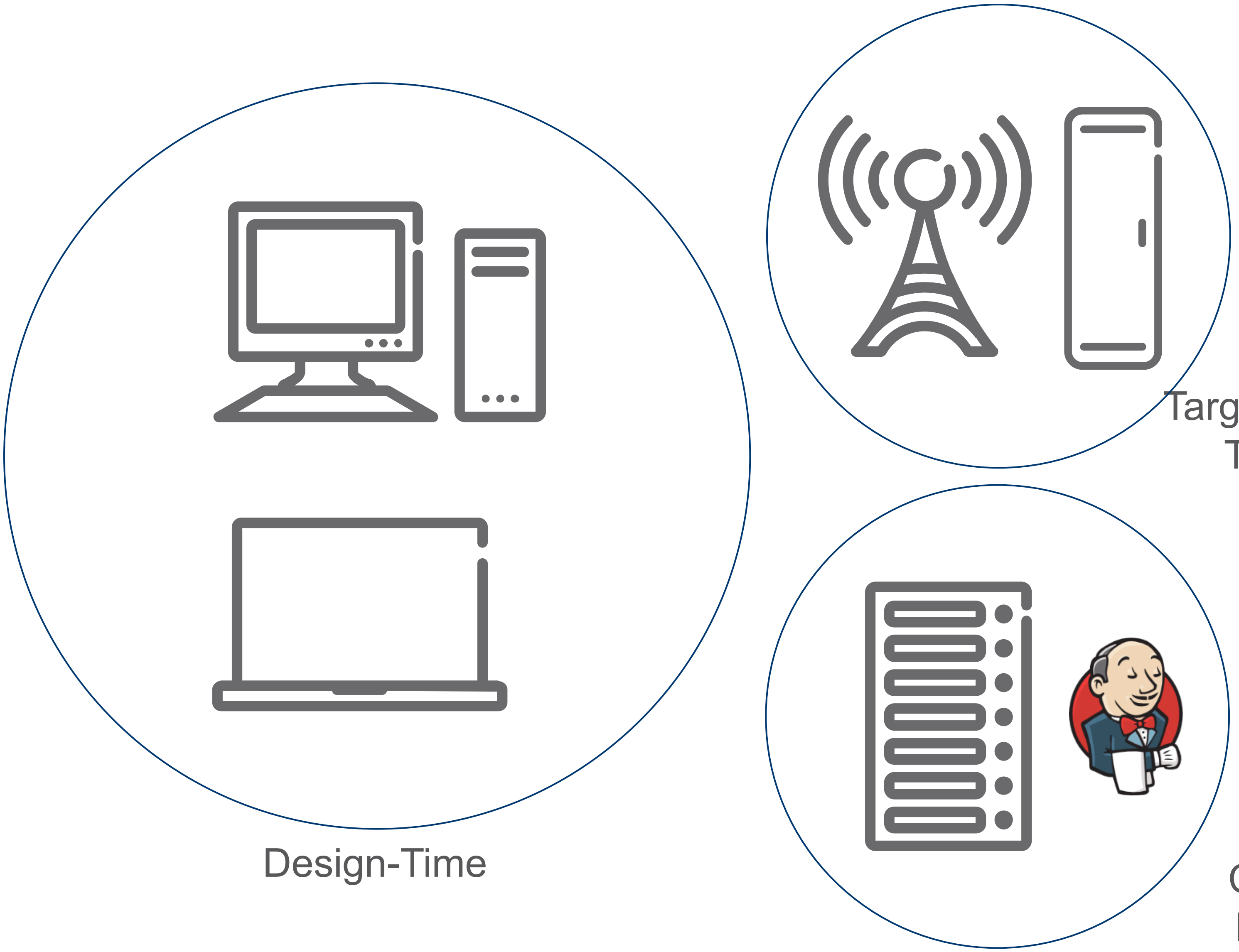
Small community
Development bottleneck
Less robust tooling
Debugging very difficult

Imperative



Massive community
Easy to outsource
Excellent tooling
Debugging not so hard

Deploy-ability



Design-Time

Target (Run-Time)

Continuous Integration

Deploy-ability Issues

Declarative/Hybrid

?!

Normally cannot run on target environment (except XSLT)

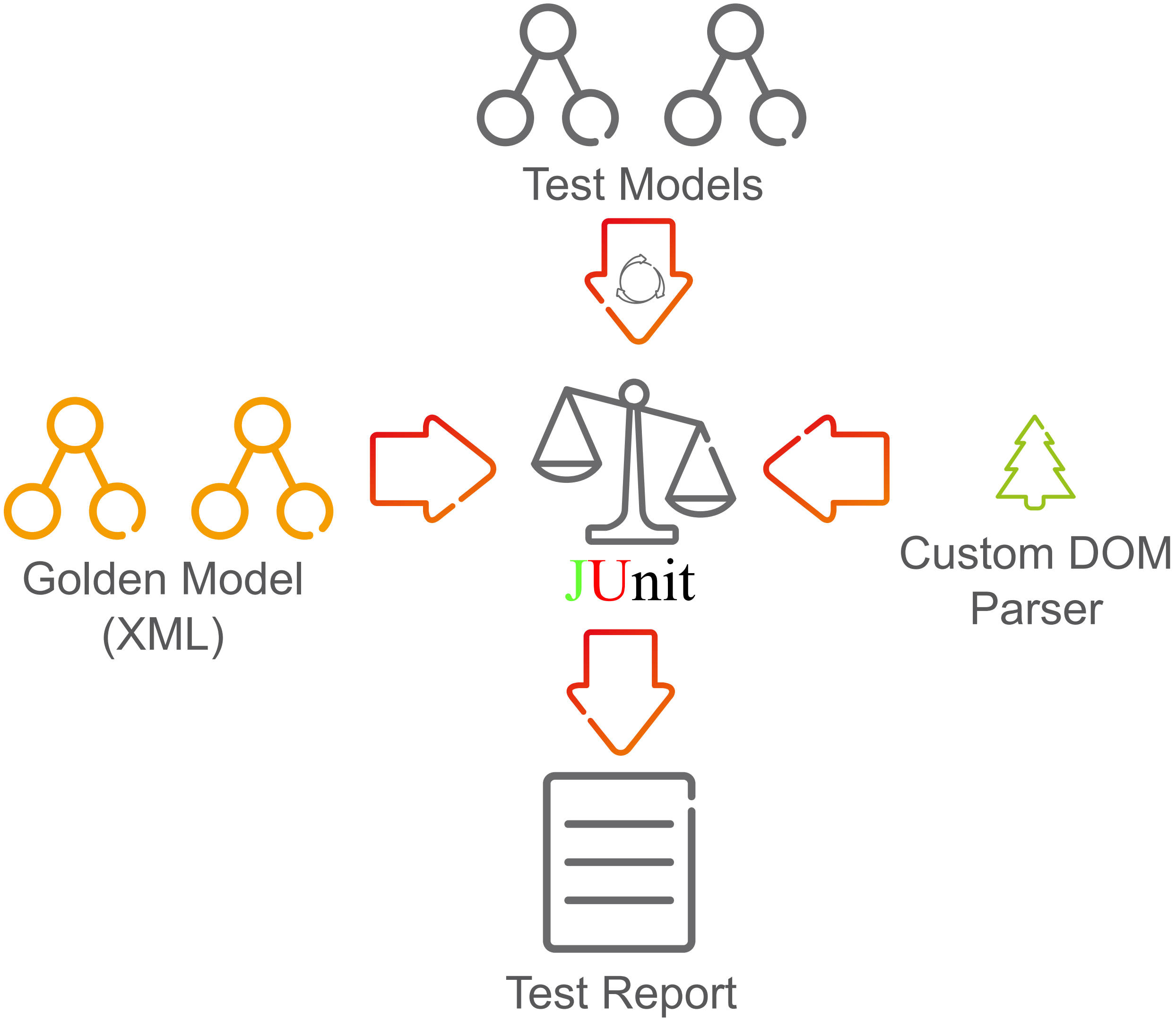
Imperative

!

Can, in theory, run on all deployment environments

Testability

Model Comparison



Test Models

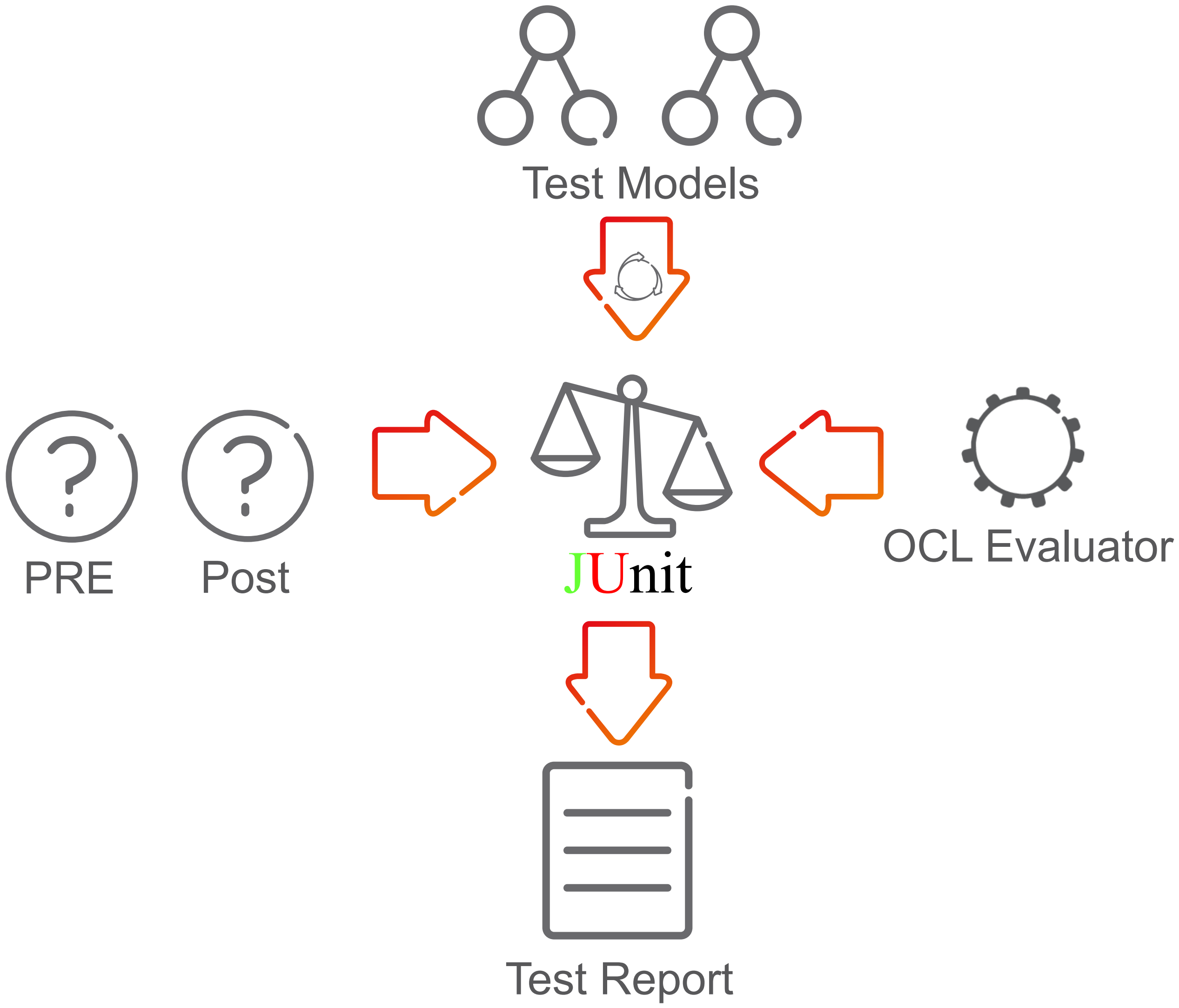
Golden Model (XML)

JUnit

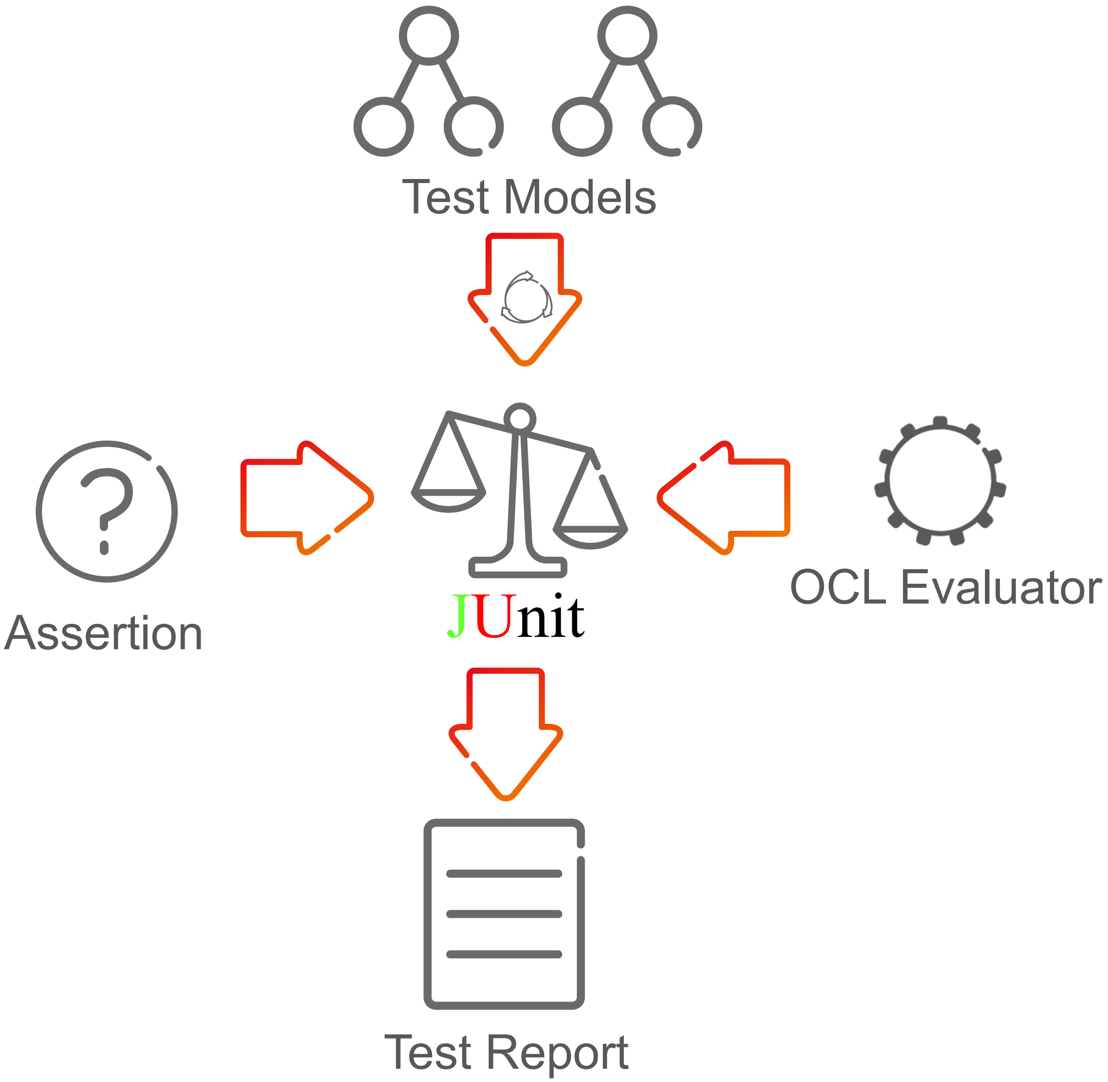
Custom DOM Parser

Test Report

Model Contracts



Pattern Matching (Assertions)



Test Models

Assertion

JUnit

OCL Evaluator

Test Report

abcdefgh

ÛÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ

ĢĢĪĮįİĶ
ŪŪŮŮŰ

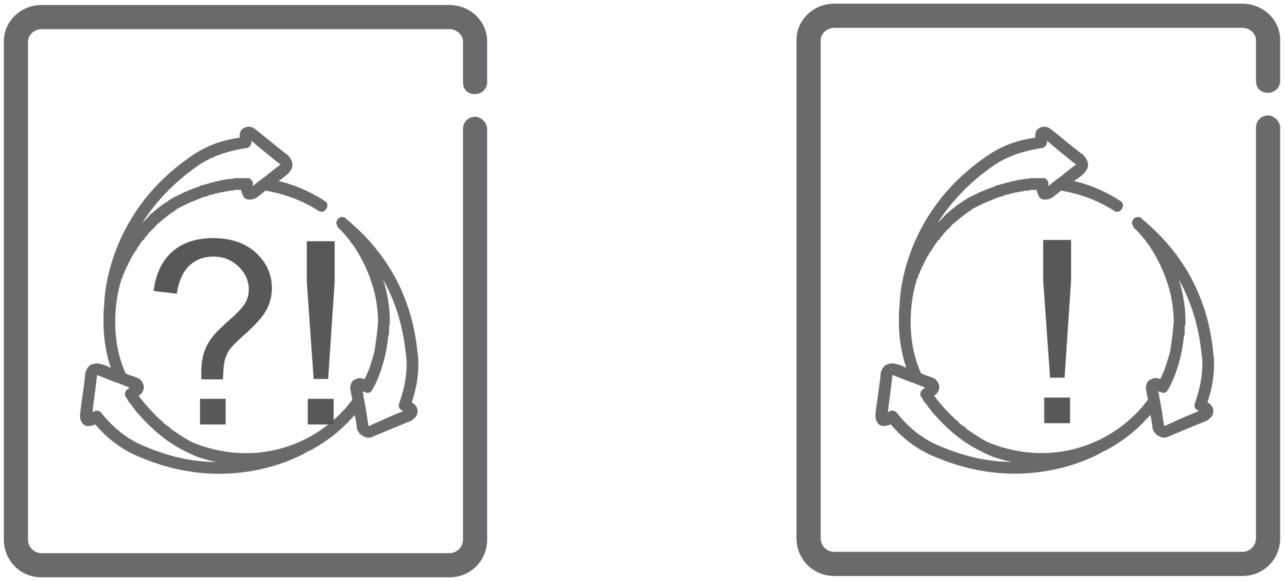
ΆΈΉΊΰα

СТУФХ
УФХЦЧ
ƮəǽẀẁ

Testability Issues

Declarative/Hybrid

Imperative



Many test models needed for 100% coverage
Verification is only as good as the Oracles

No code coverage tools

Many code coverage tools

Summary/Conclusions

abcdefgh

ÛÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ

ĢĢĪĮįİĶ
ŪŪŮŮŰ

ΆΈΉΊΰα

СТУФХ
УФХЦЧ
ʼҐəǽẀẁ

NFP Current Position

- 🟥 Code not readable/ auditable by many developers/all systems guys
- 🟩 Transformation code is 100% verifiable if we write enough test models
- 🟨 Changes in input model require full re-transformation
- 🟨 Bugs are fixed quickly and without side-effect but debugging is often difficult

- 🟨 Some parallelizable transformation parts can be executed in parallel
- 🟩 Code base grows without constant refactoring
- 🟥 Resistance from developers to learn a new obscure language
- 🟩 Execution doesn’t consume all resorces as model size gets ”large”
- 🟨 Transformation code cannot be runnable ”anywhere” but can run in GUI or without
- 🟥 Significant developer bottleneck for transformation code

🟩 Achieved 🟨 Partial 🟥 Not Achieved

Potential NFP Improvements

Transform code that non techie folks can read

Debuggable transformation code with proper execution handling

Transformation code that runs ”anywhere”

Incremental/Parallel transformation execution for imperative code

Close the gap betwen imperative and declarative languages (Java8/Xtend?)

Develop a declarative language that imperative coders will adopt

Questions

abcdefgh

ÛÜÝÞßà
ĊċČĎďĐ
ŒœŔŕŖŗ
˙˚˛˜˝Ẁẁ

ĞĢģĪīĮįİĶ
ŪūŮůŰ

ΆΈΉΊΰα

СТУФХ
УФХЦЧ
ƮəǽẀẁ

ERICSSON