Automated Software Engineering: Introduction

Oszkár Semeráth, Kristóf Marussy, Attila Ficsor, Ármin Zavada

MŰEGYETEM 1782

BME FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS

ftsrg Critical Systems Research Group

Welcome everyone to the specialization!

Administration

Teachers and communication channels

Teachers:

- Department of Artificial Intelligence and Systems Engineering
- Critical Systems Research Group - https://ftsrg.mit.bme.hu/en/

Communication:

- Teams
- Moodle + Github
- Email (semerath@mit.bme.hu)

Oszkár Semeráth
(lead lecturer)

Attila Ficsor

Kristóf Marussy

Ármin Zavada

Schedule

- Each week:
  - one **lecture**
  - one **practice / laboratory** session (lab is skipped in first week)
- Laboratory is **mandatory**, there will be an attendance check. Max. 3 lab sessions can be missed
- **Homework**
  - Assignment: 2 stages, published on week #5 and #8
  - Deadline: Week #13 end of Friday (midnight)
  - Homework retake: Retake week end of Friday (midnight)
- **Exam** during the exam period

Homework

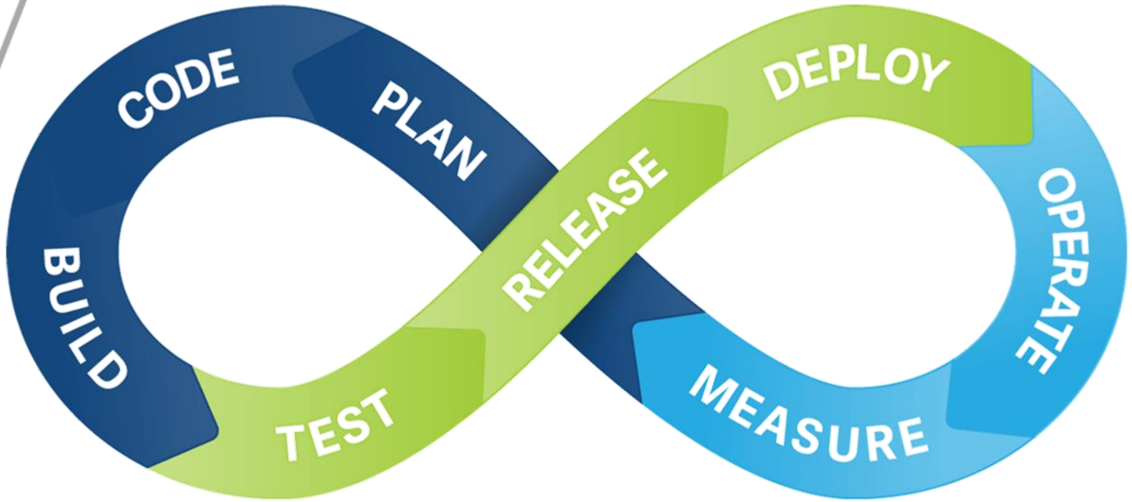
- 3-person teams formed using a Moodle survey
- We ask everyone to create a Github account in advance!
- Topics: development / automation tasks connected to the laboratory materials
- Some degree of documentation is also required *(code comments, few paragraphs of markdown)*

- Different task are deliberately not interdependent
→ Laboratory materials will have a more coherent ‘story’
- You may improve your solution during the retake period if you have submitted until the regular deadline and received feedback

Requirements

- Lab (practice) session attendance (3 can be missed)
- Accepted homework
  *(must obtain 40% of the possible score in both stages,
  constitutes 30% of the total score)*
- Exam
  *(must obtain 40% of the possible score,
  constitutes 70% of the total score)*
- Final mark is as usual (85%, 70%, 55%, 40%)

Course overview



Goals

The aim of the course is to introduce

1. **state-of-the-art**
2. **automated**
3. **technologies** in the field of software engineering.

Outcomes:

Improved software **quality** ⇢ guarantees (formal methods)

Higher **productivity** ⇢ automations (code generation)

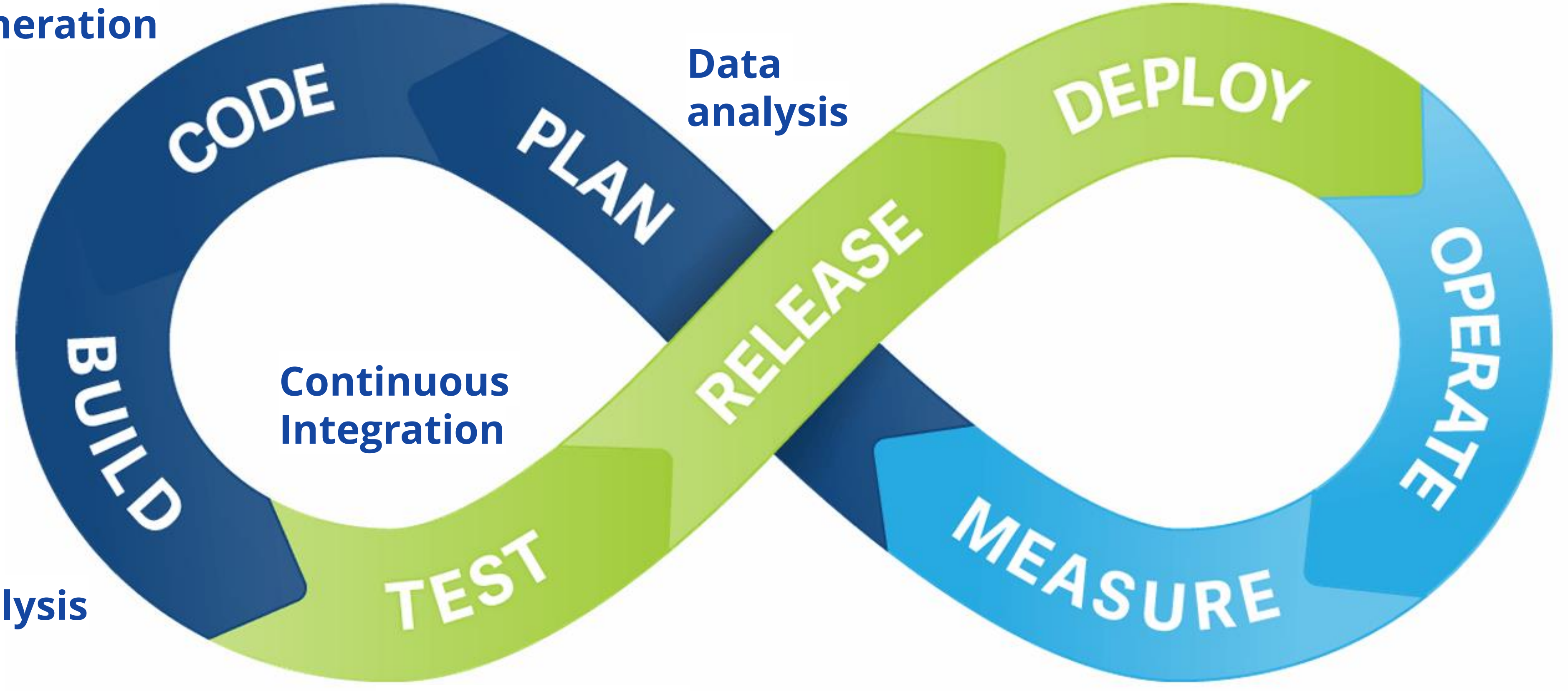
Areas of automation

Automated code generation

Software modeling
Language engineering

Continuous Delivery

Data analysis



Continuous Integration

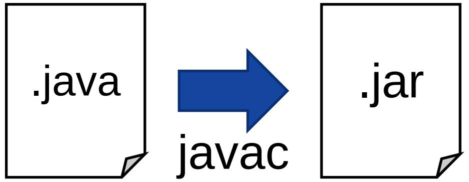
Static analysis

Simulation and Debugging

Benchmarking

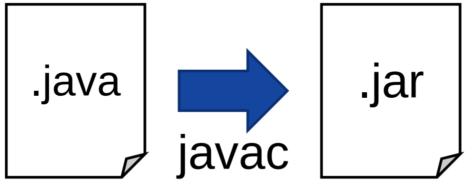
Development processes

- What are the typical elements of software development?
  1. Source files
  2. Compilers
  3. Derived artifacts



Development processes

- What are the typical elements of software development?
    1. Source files
    2. Compilers
    3. Derived artifacts



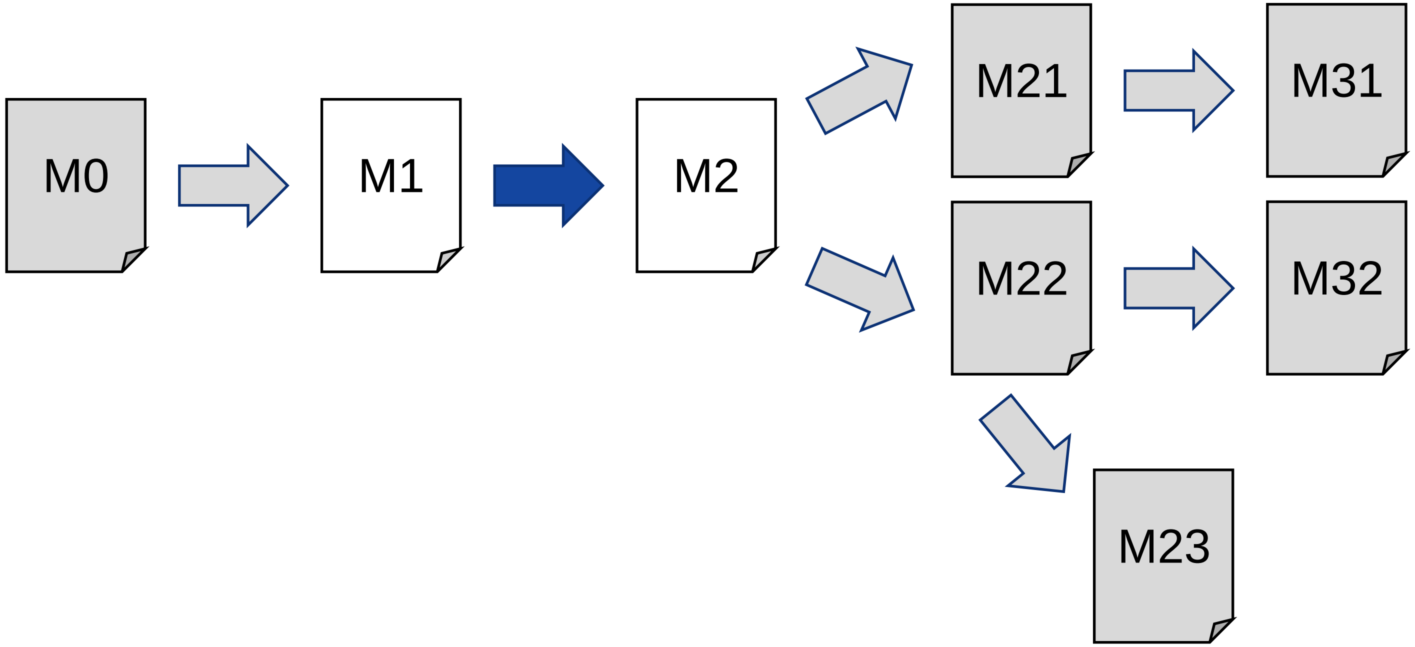
Domain-specific models:
Models and modelling languages created for solving a specific problem in a specific field

- More generally:
    - Models = { source files, documentation, configuration, domain-specific models }
    - Transformations : Models → Models

Development = **Models** + **Transformations**

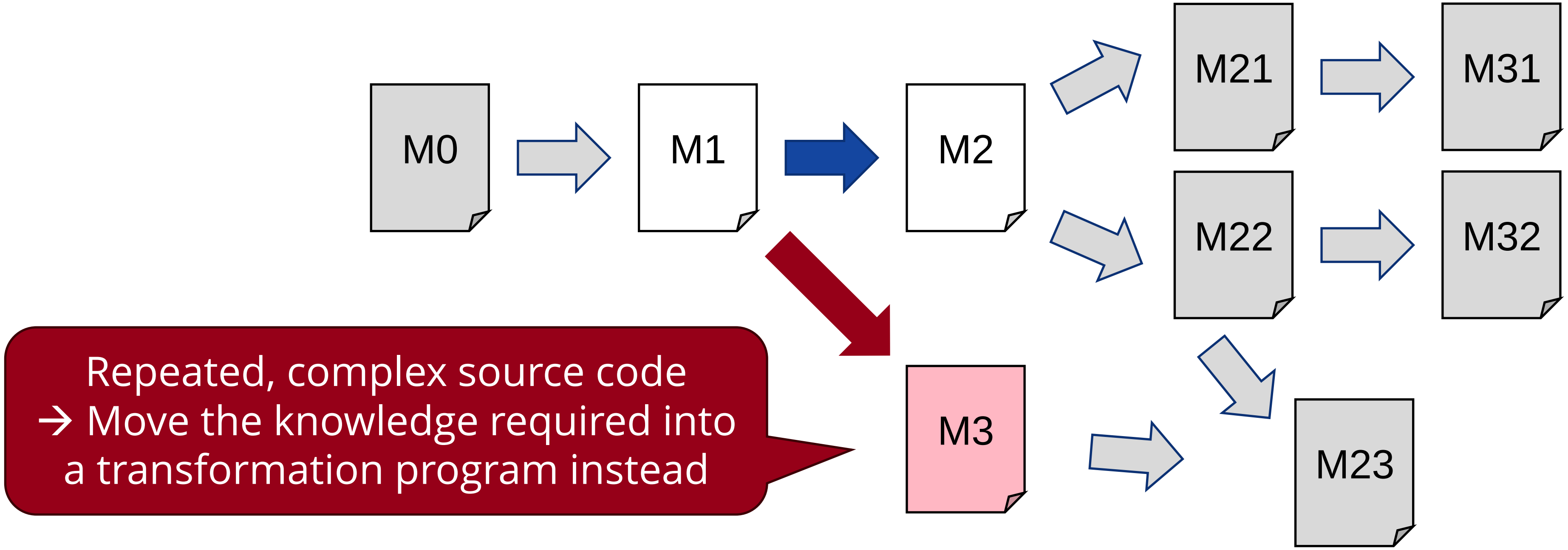
Development processes

- We create models and automate their processing
- Software is constructed by a chain of transformations



Development processes

- We create models and automate their processing
- Software is constructed by a chain of transformations

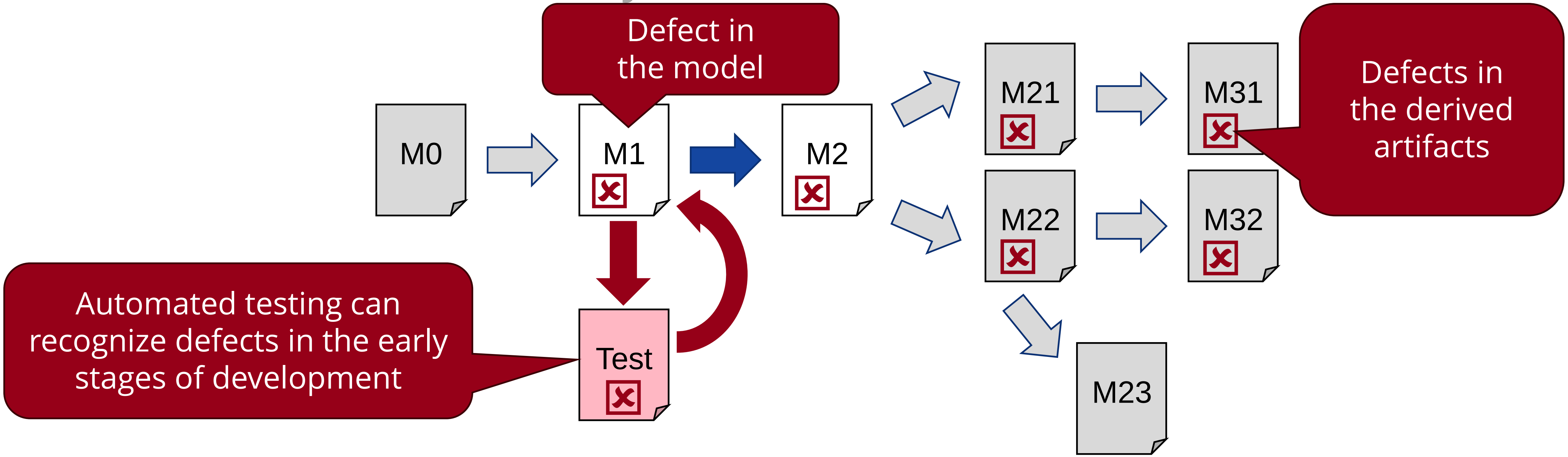


Repeated, complex source code
→ Move the knowledge required into
a transformation program instead

- **Goal 1:** Automation of development tasks with transformations

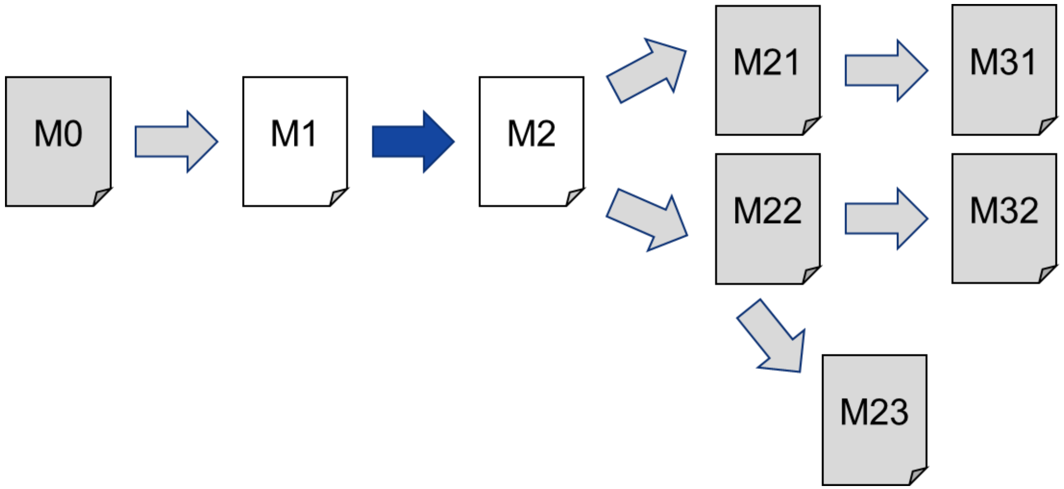
Development processes

- We create models and automate their processing
- Software is constructed by a chain of transformations


Defect in the model
M0
M1
M2
M21
M22
M23
M31
M32
Test
Defects in the derived artifacts
Automated testing can recognize defects in the early stages of development

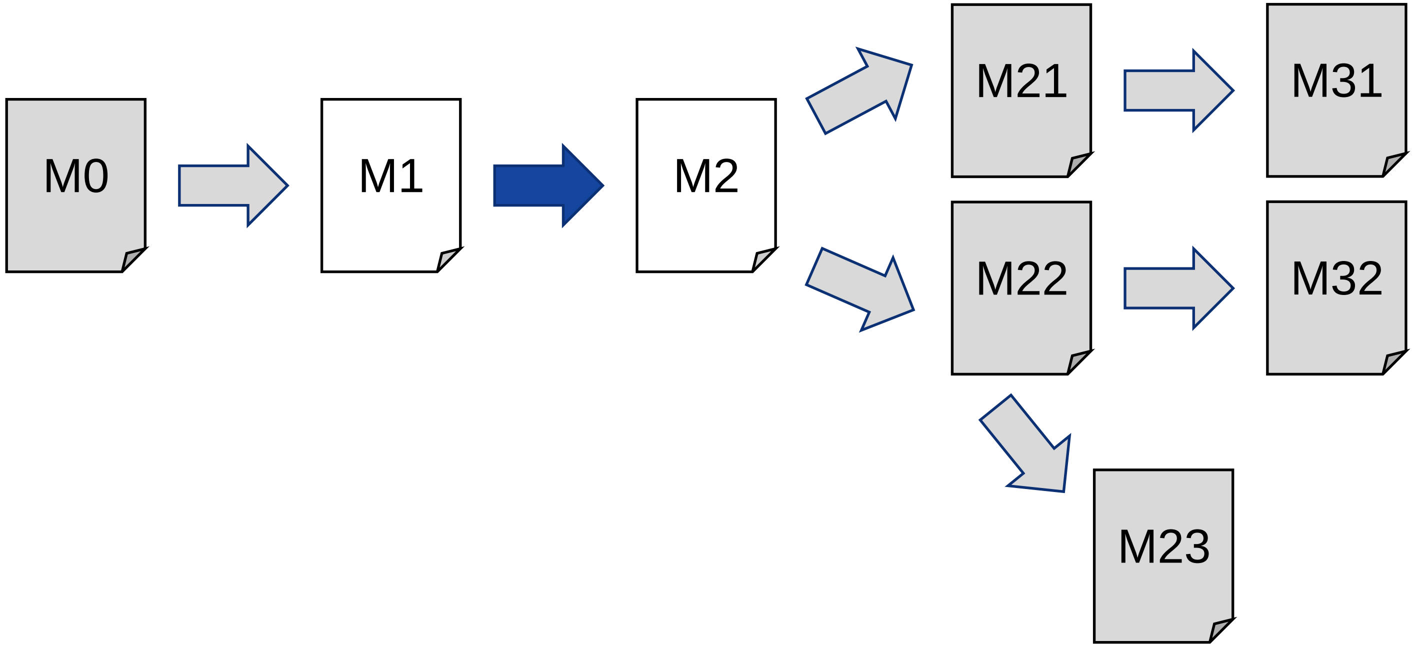
- **Goal 2:** Continuous correctness / performance evaluation

Classification of transformations

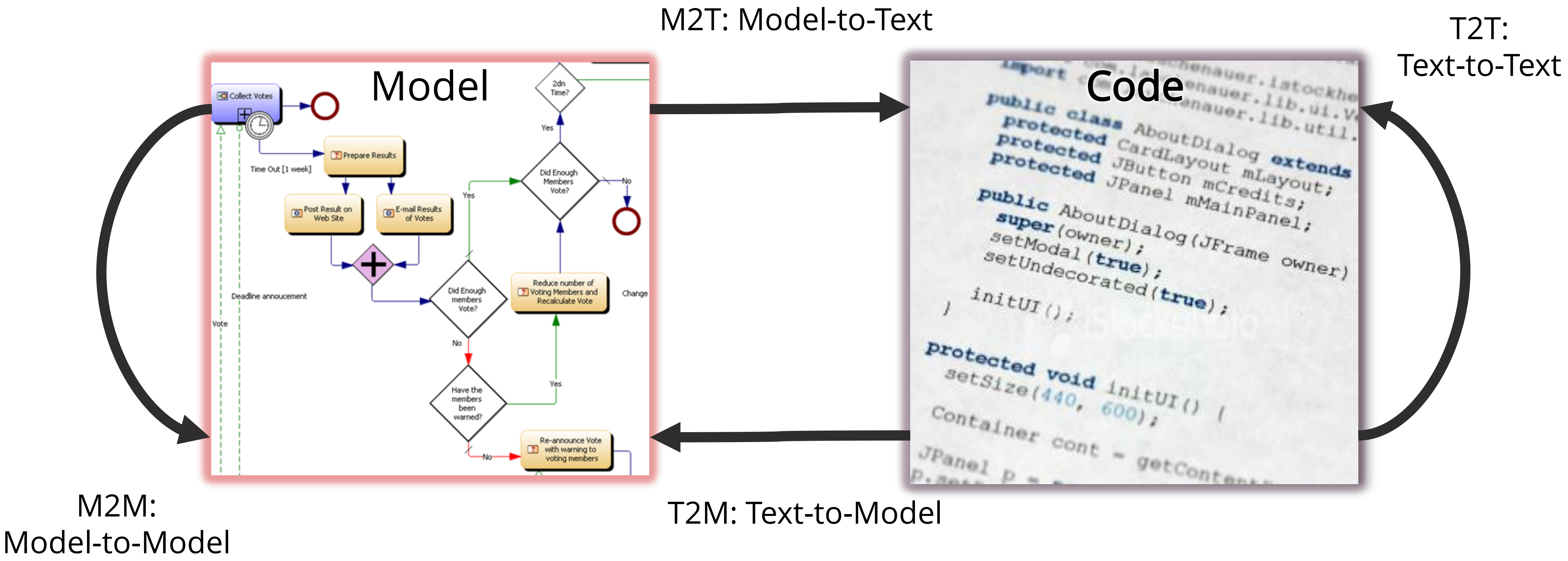


Development processes

- We create models and automate their processing
- Software is constructed by a chain of transformations



Classification of transformations



M2T: Model-to-Text

T2T: Text-to-Text

M2M: Model-to-Model

T2M: Text-to-Model

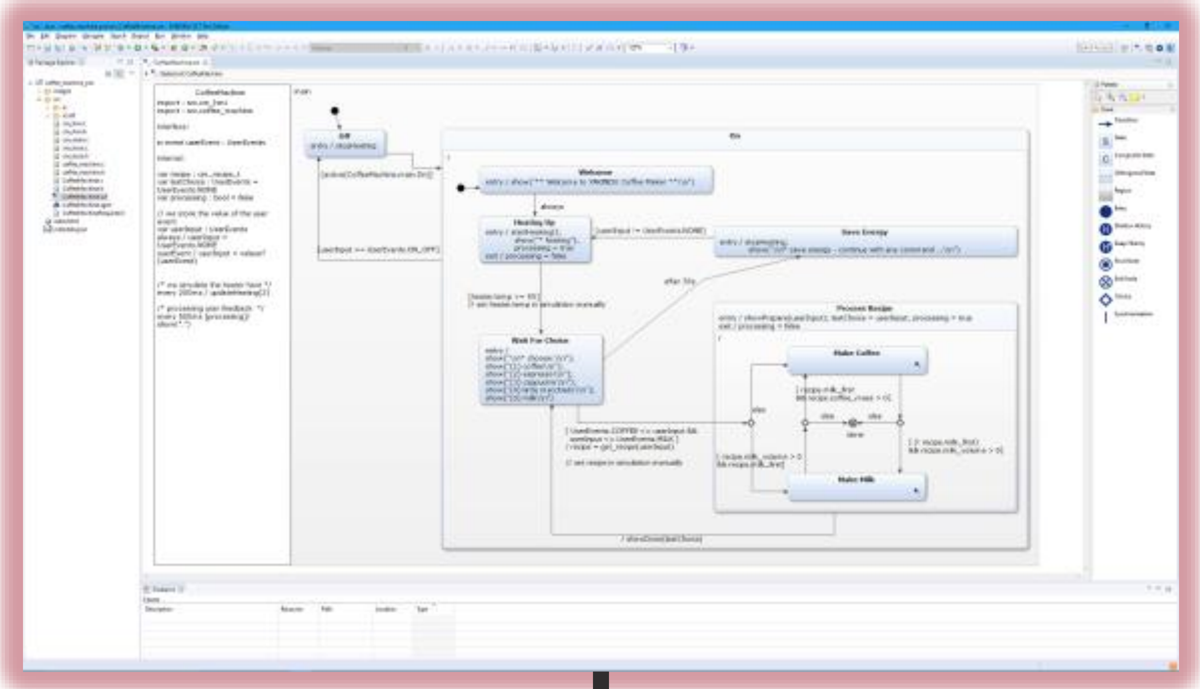
Model

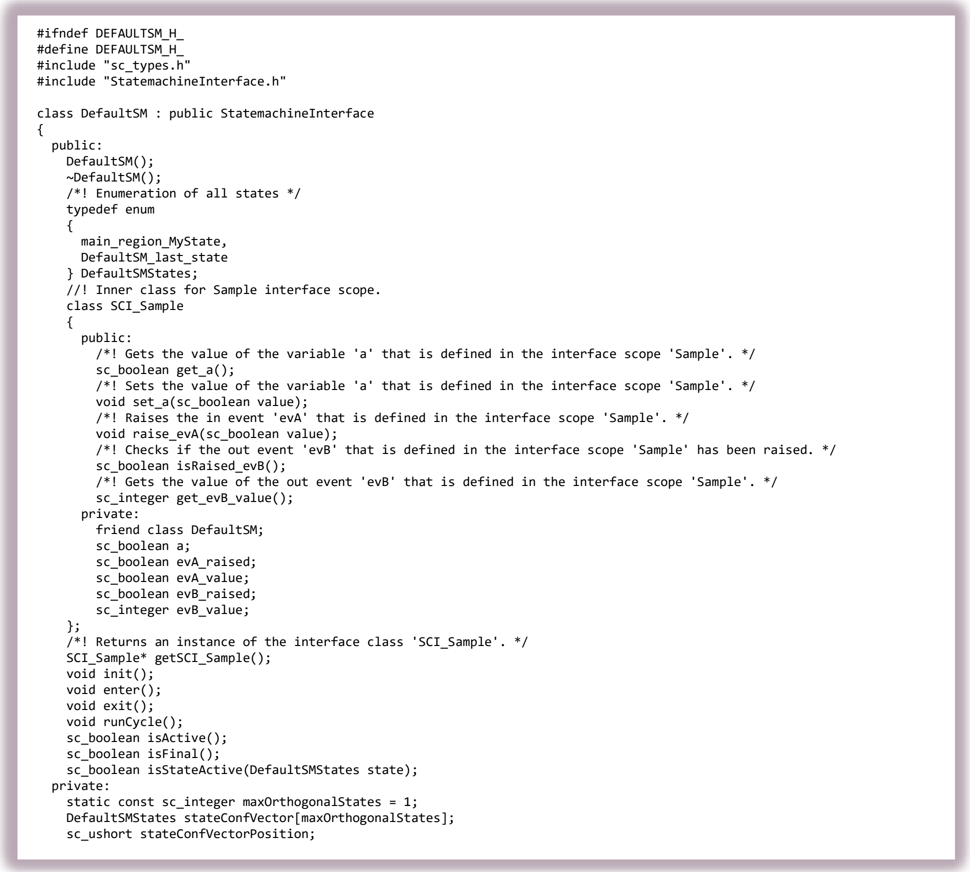
Code

```
import com.lackenauer.latockhe
import com.lackenauer.lib.ui.V
import com.lackenauer.lib.util.
public class AboutDialog extends
  protected CardLayout mLayout;
  protected JButton mCredits;
  protected JPanel mMainPanel;
  public AboutDialog(JFrame owner)
    super(owner);
    setModal(true);
    setUndecorated(true);
    initUI();
  }
  protected void initUI() {
    setSize(440, 600);
    Container cont = getContent
    JPanel p =
```

M2T example

- Code generation



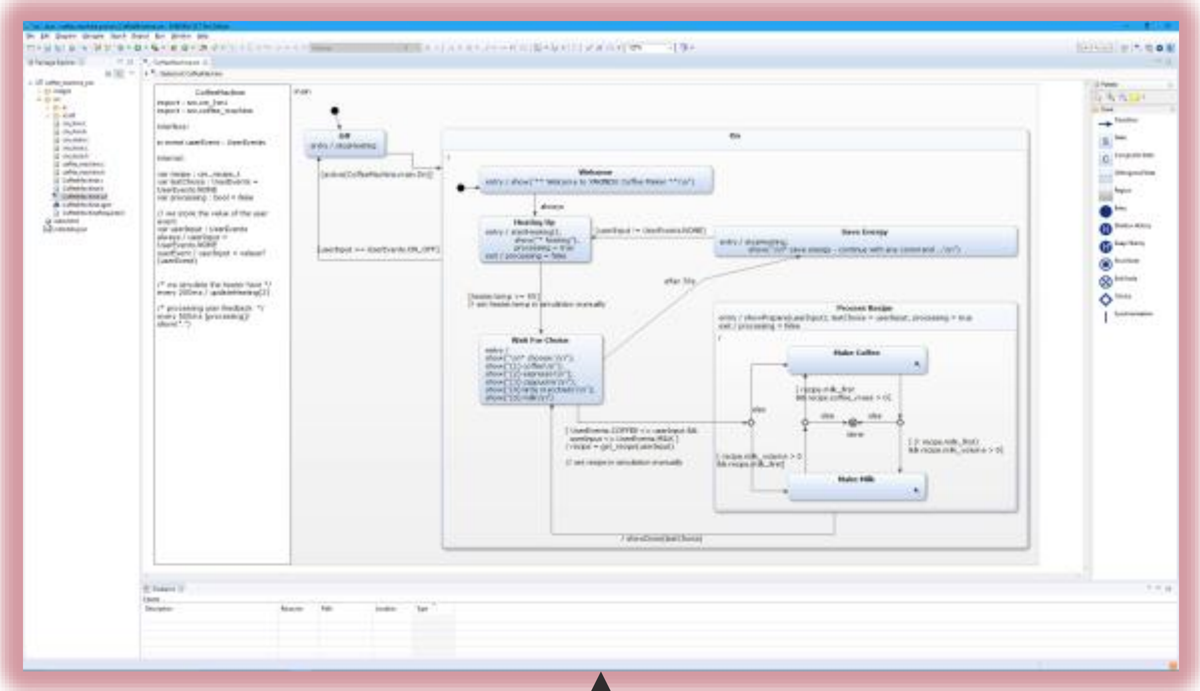


```
#ifndef DEFAULTSM_H_
#define DEFAULTSM_H_
#include "sc_types.h"
#include "StatemachineInterface.h"

class DefaultSM : public StatemachineInterface
{
  public:
    DefaultSM();
    ~DefaultSM();
    /*! Enumeration of all states */
    typedef enum
    {
      main_region_MyState,
      DefaultSM_last_state
    } DefaultSMStates;
    //! Inner class for Sample interface scope.
    class SCI_Sample
    {
      public:
        /*! Gets the value of the variable 'a' that is defined in the interface scope 'Sample'. */
        sc_boolean get_a();
        /*! Sets the value of the variable 'a' that is defined in the interface scope 'Sample'. */
        void set_a(sc_boolean value);
        /*! Raises the in event 'evA' that is defined in the interface scope 'Sample'. */
        void raise_evA(sc_boolean value);
        /*! Checks if the out event 'evB' that is defined in the interface scope 'Sample' has been raised. */
        sc_boolean isRaised_evB();
        /*! Gets the value of the out event 'evB' that is defined in the interface scope 'Sample'. */
        sc_integer get_evB_value();
      private:
        friend class DefaultSM;
        sc_boolean a;
        sc_boolean evA_raised;
        sc_boolean evA_value;
        sc_boolean evB_raised;
        sc_integer evB_value;
    };
    /*! Returns an instance of the interface class 'SCI_Sample'. */
    SCI_Sample* getSCI_Sample();
    void init();
    void enter();
    void exit();
    void runCycle();
    sc_boolean isActive();
    sc_boolean isFinal();
    sc_boolean isStateActive(DefaultSMStates state);
  private:
    static const sc_integer maxOrthogonalStates = 1;
    DefaultSMStates stateConfVector[maxOrthogonalStates];
    sc_ushort stateConfVectorPosition;
```

YAKINDU STATECHART TOOLS

T2M example

- Import source code into a model

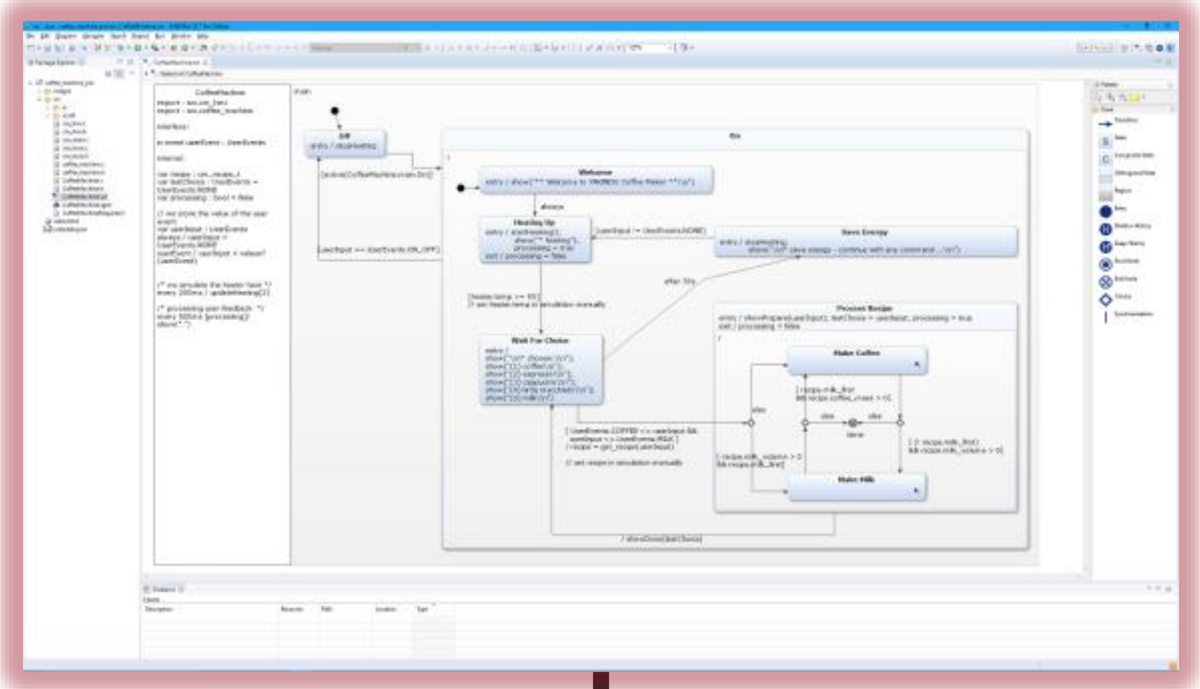


```
class Point
{
    public:
        int32_t get_x();
        void set_x(int32_t x);
        int32_t get_y();
        void set_y(int32_t y);
    private:
        int32_t x;
        int32_t y;
};
```

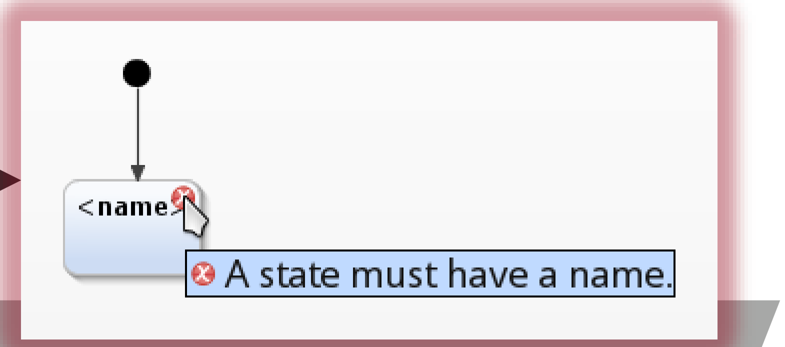
YAKINDU STATECHART TOOLS

M2M example 1

- M2M: Model validation: error pattern → error messages



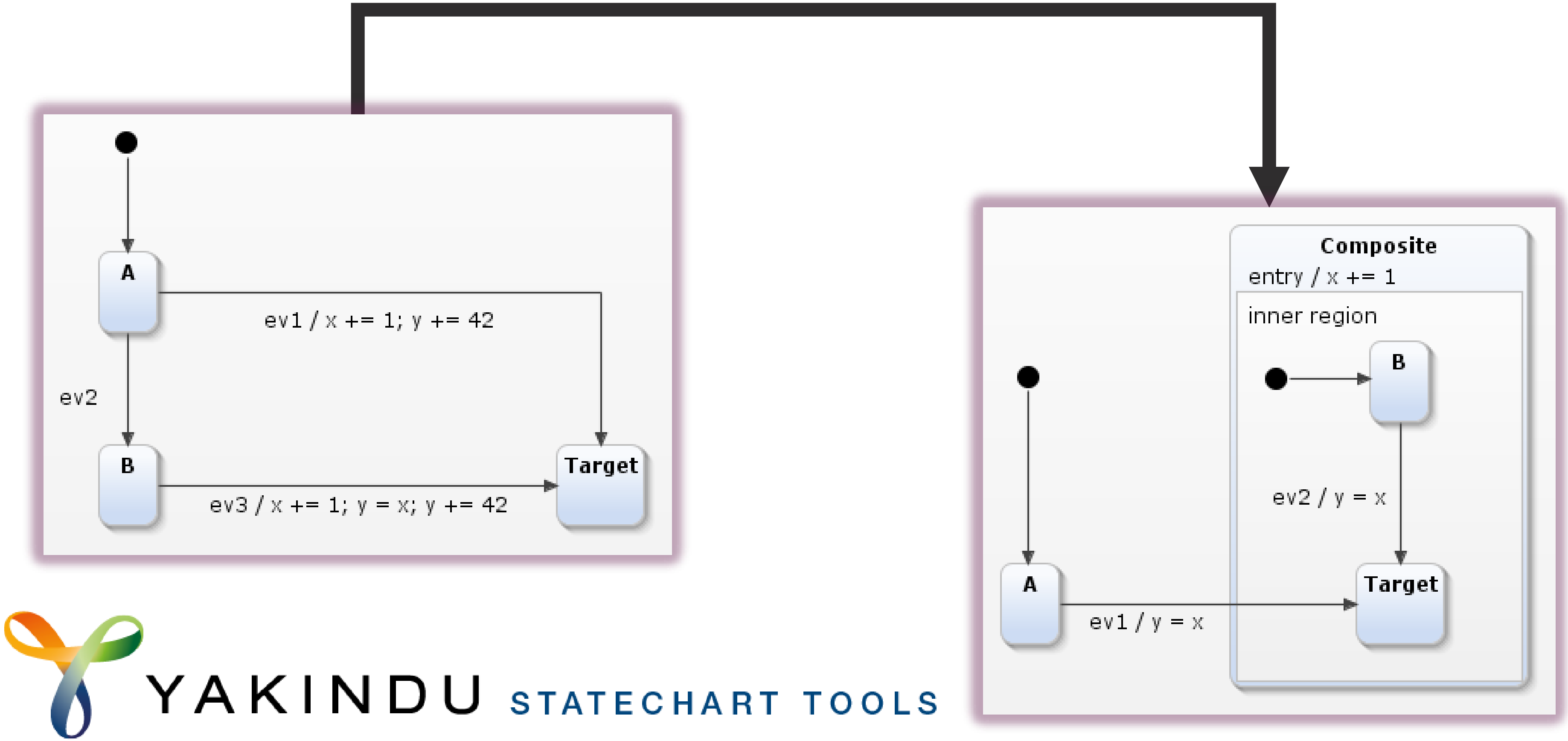
YAKINDU STATECHART TOOLS



A state must have a name.

M2M example 2

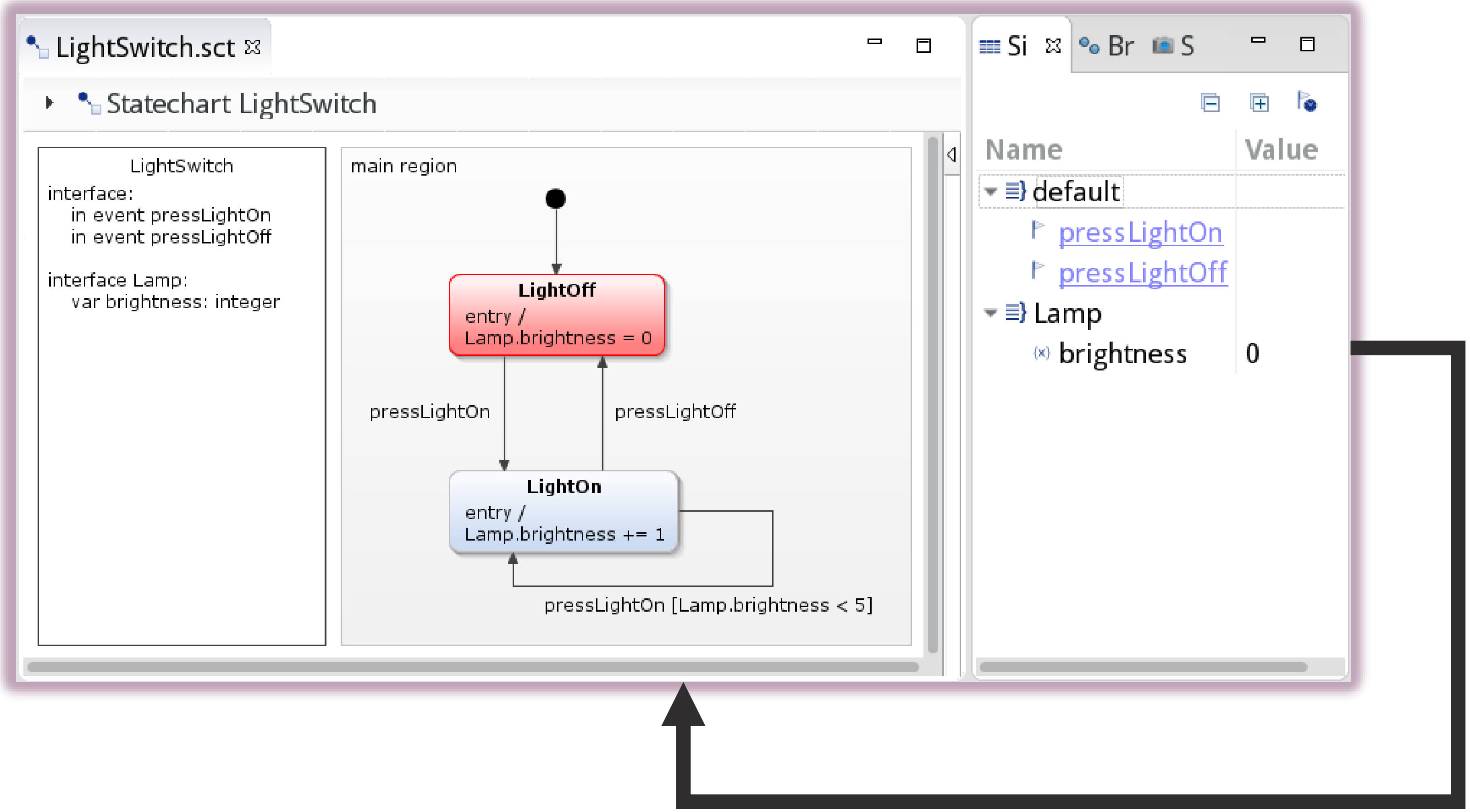
- Model refactoring



YAKINDU STATECHART TOOLS

M2M example 3

- Simulation
- Semantics



LightSwitch.sct

Statechart LightSwitch

```
LightSwitch
interface:
    in event pressLightOn
    in event pressLightOff

interface Lamp:
    var brightness: integer
```

Name	Value
default	
pressLightOn	
pressLightOff	
Lamp	
brightness	0

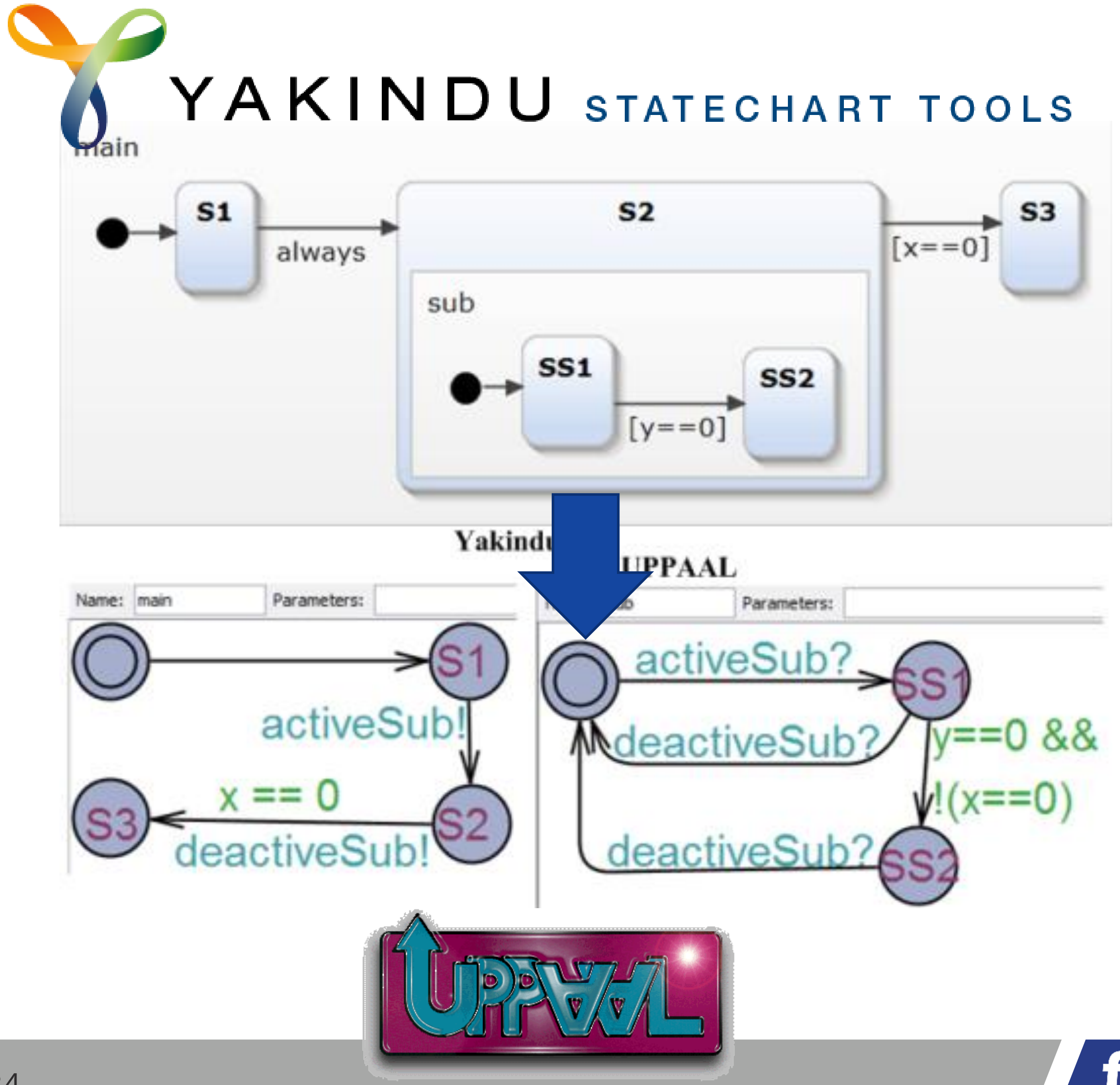
YAKINDU STATECHART TOOLS

M2M example 4

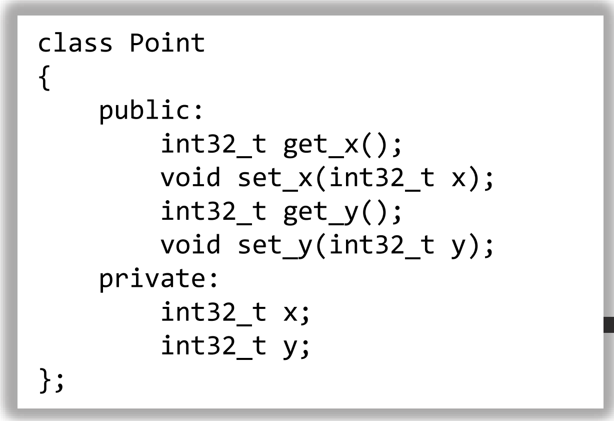
- Formal methods
- Hidden formal methods:

Algorithms than can be applied without verification expertise

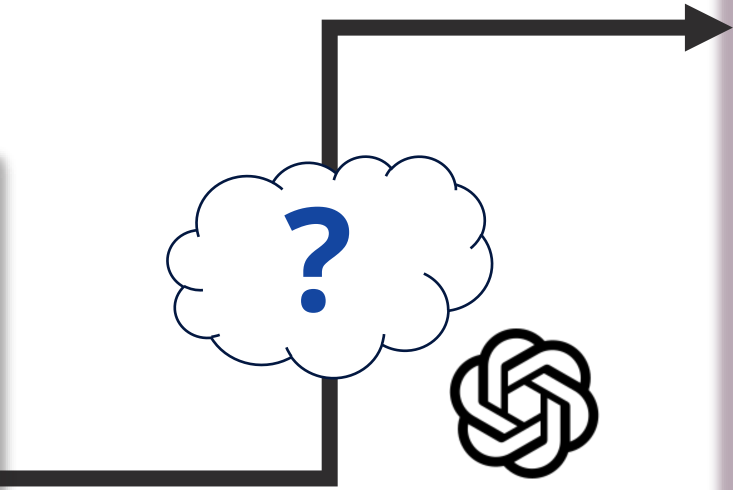
- – Tool support
- – Back-annotation of results

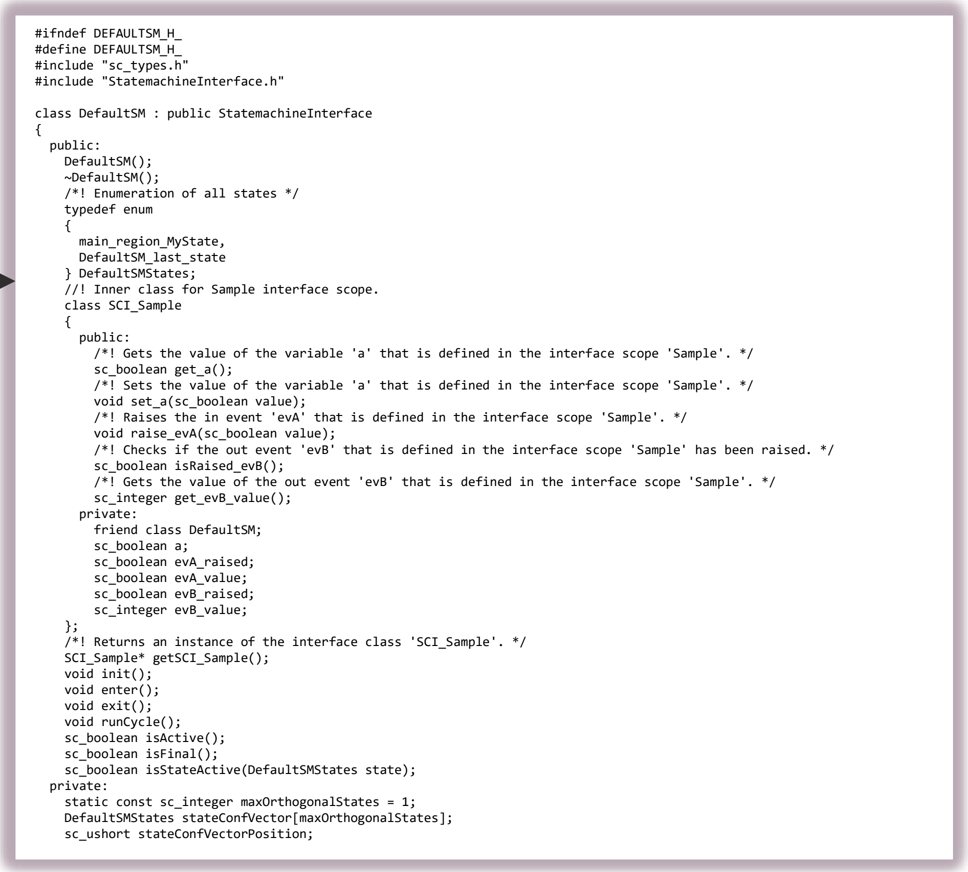


T2T example?



```
class Point
{
    public:
        int32_t get_x();
        void set_x(int32_t x);
        int32_t get_y();
        void set_y(int32_t y);
    private:
        int32_t x;
        int32_t y;
};
```





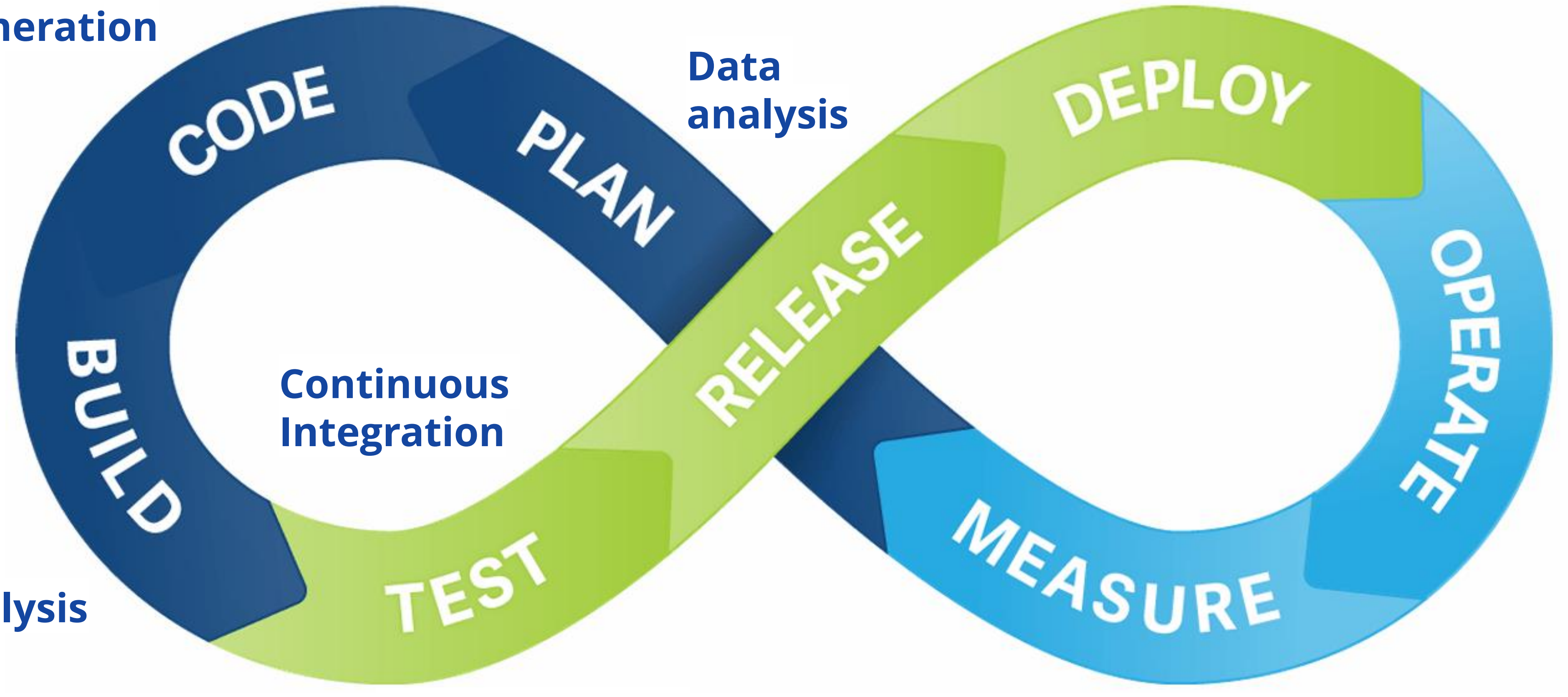
```
#ifndef DEFAULTSM_H_
#define DEFAULTSM_H_
#include "sc_types.h"
#include "StatemachineInterface.h"

class DefaultSM : public StatemachineInterface
{
  public:
    DefaultSM();
    ~DefaultSM();
    /*! Enumeration of all states */
    typedef enum
    {
      main_region_MyState,
      DefaultSM_last_state
    } DefaultSMStates;
    //! Inner class for Sample interface scope.
    class SCI_Sample
    {
      public:
        /*! Gets the value of the variable 'a' that is defined in the interface scope 'Sample'. */
        sc_boolean get_a();
        /*! Sets the value of the variable 'a' that is defined in the interface scope 'Sample'. */
        void set_a(sc_boolean value);
        /*! Raises the in event 'evA' that is defined in the interface scope 'Sample'. */
        void raise_evA(sc_boolean value);
        /*! Checks if the out event 'evB' that is defined in the interface scope 'Sample' has been raised. */
        sc_boolean isRaised_evB();
        /*! Gets the value of the out event 'evB' that is defined in the interface scope 'Sample'. */
        sc_integer get_evB_value();
      private:
        friend class DefaultSM;
        sc_boolean a;
        sc_boolean evA_raised;
        sc_boolean evA_value;
        sc_boolean evB_raised;
        sc_integer evB_value;
    };
    /*! Returns an instance of the interface class 'SCI_Sample'. */
    SCI_Sample* getSCI_Sample();
    void init();
    void enter();
    void exit();
    void runCycle();
    sc_boolean isActive();
    sc_boolean isFinal();
    sc_boolean isStateActive(DefaultSMStates state);
  private:
    static const sc_integer maxOrthogonalStates = 1;
    DefaultSMStates stateConfVector[maxOrthogonalStates];
    sc_ushort stateConfVectorPosition;
```

- Usually not *directly* applicable: the text must be transformed into a **structured** model (T2M, parser) before programmatic processing
- But recently **Large Language Models** (LLM) became available, which can process text without and intermediate step
  - Challenges: **finite context window**, **hallucinations**, ensuring **correctness**

Even **natural language** input /output

Commonly automated tasks

Areas of automation



Automated code generation

Software modeling
Language engineering

Data analysis

Continuous Delivery

Continuous Integration

Static analysis

Simulation and Debugging

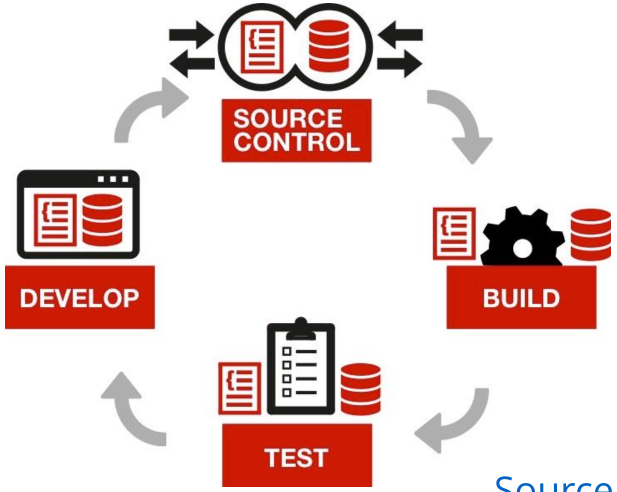
Benchmarking

Continous Integration (CI)

Martin Fowler

https://martinfowler.com/articles/continuousIntegration.html

- *„a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily”*
- *„Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. "*



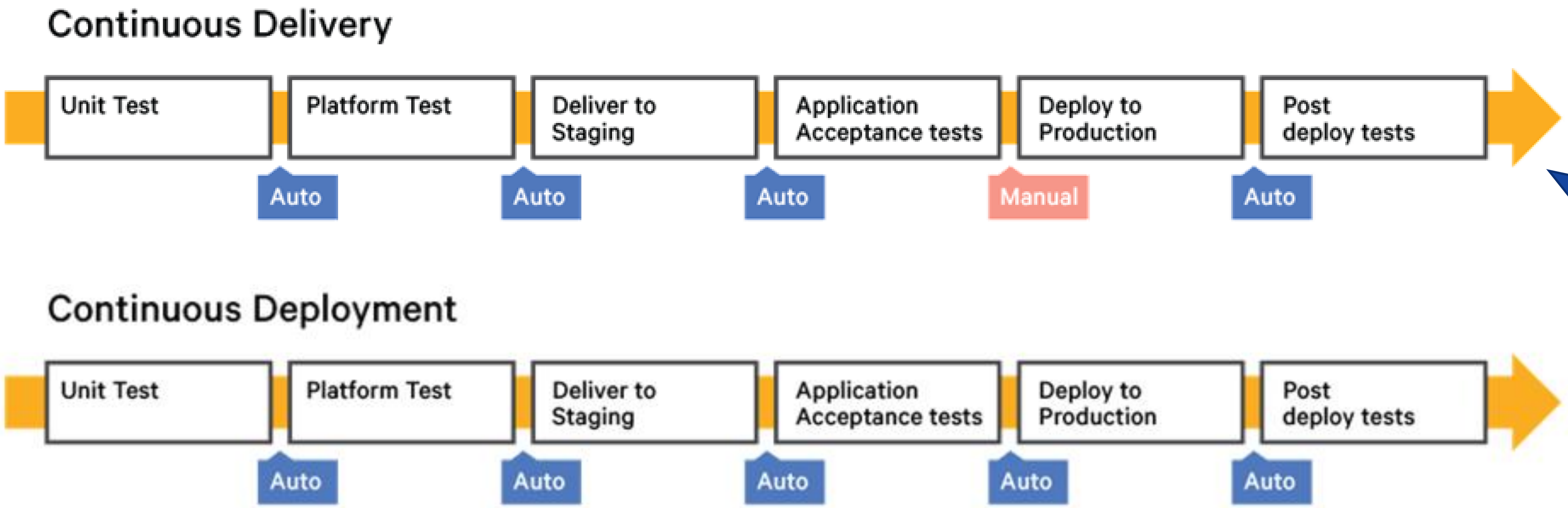
Source

Maven™ Gradle JUnit 5

Continous Delivery (CD)

„build software so that it is always in a state where it could be put into production”

Source: https://martinfowler.com/bliki/ContinuousDelivery.html



Continuous Delivery

Unit Test → Auto → Platform Test → Auto → Deliver to Staging → Auto → Application Acceptance tests → Manual → Deploy to Production → Auto → Post deploy tests

Continuous Deployment

Unit Test → Auto → Platform Test → Auto → Deliver to Staging → Auto → Application Acceptance tests → Auto → Deploy to Production → Auto → Post deploy tests

delivery pipeline

Forrás

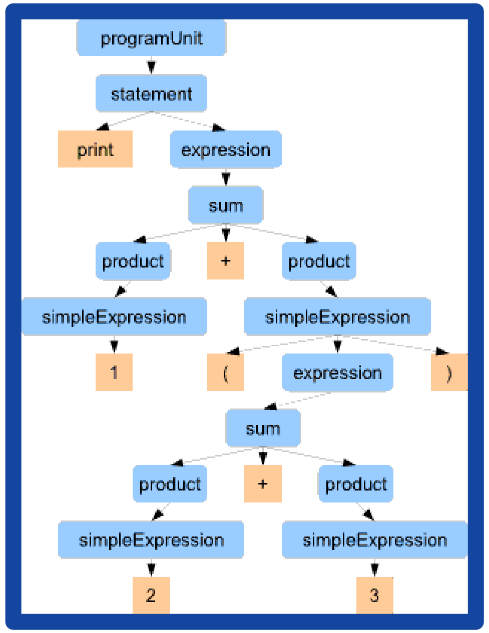
GitHub Actions

docker

Languages and compilers

- How to create our own (programming) langauges?
- How to express grammatical rules?

```
import com.lauchenauer.l
import com.lauchenauer.l
public class AboutDialog e
    protected CardLayout mLa
    protected JButton mCredi
    protected JPanel mMainPan
    public AboutDialog(JFrame
        super(owner);
        setModal(true);
        setUndecorated(true);

        initUI();
    }

    protected void initUI() {
        setSize(440, 600);
        Container cont = getConten
        JPanel p
```



Xtext

langium

Refinery

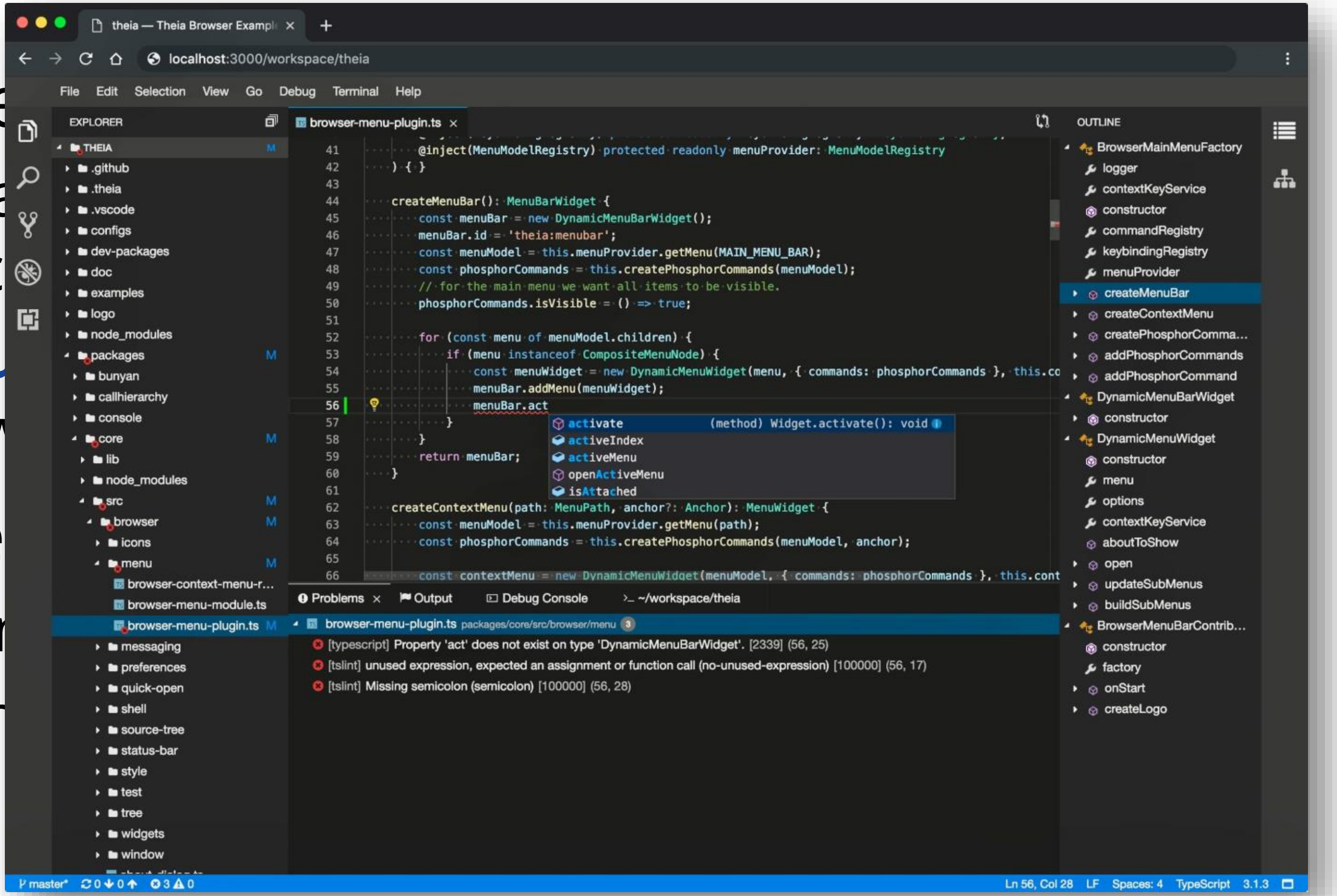
Modern development environments

- **Intelligent functions** in development environments
  - Common helper functions: auto-completion (content assist), just-in-time validation, refactoring
  - More recently: **Lange Language Models** (LLM) supporting coding and software modeling
- Browser-based development tools
  - No need for local installation
  - Easily reproducible environment

langium GitHub Copilot

Modern development environments

- **Intelligent functions** in de
  - Common helper functions: a
    just-in-time validation, refac
  - More recently: **Lange Langu**
    supporting coding and softw
- Browser-based developme
  - No need for local installatior
  - Easily reproducible environn



```
        @inject(MenuModelRegistry) protected readonly menuProvider: MenuModelRegistry
    ) { }

    createMenuBar(): MenuBarWidget {
        const menuBar = new DynamicMenuBarWidget();
        menuBar.id = 'theia:menubar';
        const menuModel = this.menuProvider.getMenu(MAIN_MENU_BAR);
        const phosphorCommands = this.createPhosphorCommands(menuModel);
        // for the main menu we want all items to be visible.
        phosphorCommands.isVisible = () => true;

        for (const menu of menuModel.children) {
            if (menu instanceof CompositeMenuNode) {
                const menuWidget = new DynamicMenuWidget(menu, { commands: phosphorCommands }, this.co
                menuBar.addMenu(menuWidget);
                menuBar.act
            }
        }
        return menuBar;
    }

    createContextMenu(path: MenuPath, anchor?: Anchor): MenuWidget {
        const menuModel = this.menuProvider.getMenu(path);
        const phosphorCommands = this.createPhosphorCommands(menuModel, anchor);

        const contextMenu = new DynamicMenuWidget(menuModel, { commands: phosphorCommands }, this.cont
```

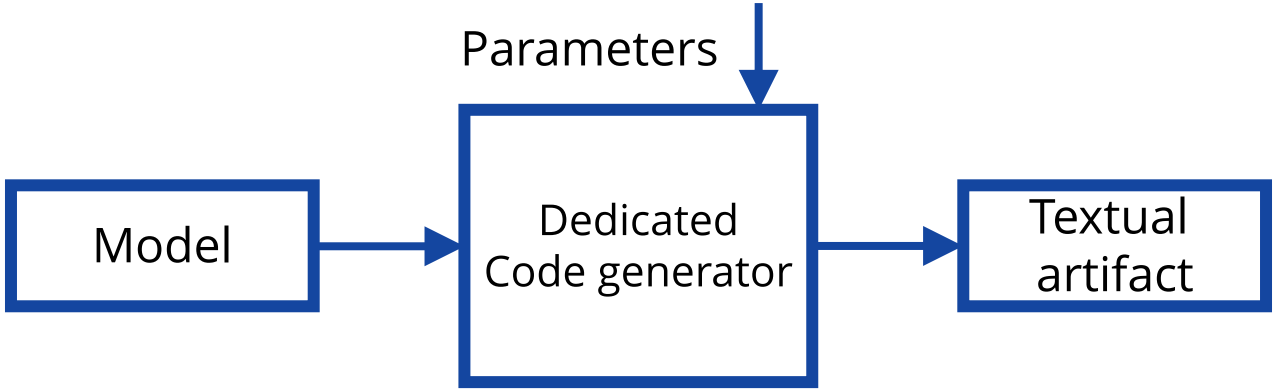
```
browser-menu-plugin.ts packages/core/src/browser/menu 3
[typescript] Property 'act' does not exist on type 'DynamicMenuBarWidget'. [2339] (56, 25)
[tslint] unused expression, expected an assignment or function call (no-unused-expression) [100000] (56, 17)
[tslint] Missing semicolon (semicolon) [100000] (56, 28)
```

langium

GitHub Copilot

Code generators

- How to transform our models into source code?
- How to tackle programming tasks automatically?



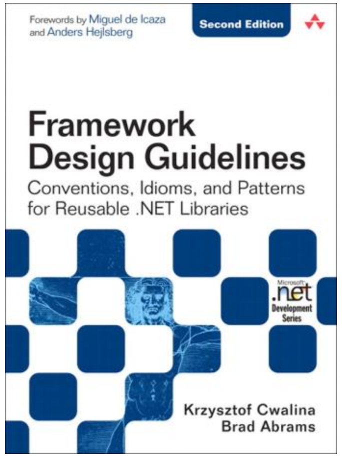
- By automating common coding tasks we can
  - incorporate **mathematical algorithms** into the development process
  - make tasks requiring **boring/long/complex** code feasible

Static analysis

- Verification of an artifact (documentation, model, code) **without executing** it
- Approaches for evaluating code quality
  - Syntactic correctness: checked by compiler

Static analysis

- Verification of an artifact (documentation, model, code) **without executing** it
- Approaches for evaluating code quality
  - Syntactic correctness: checked by compiler
  - Coding conventions: 'linter'



Forewords by Miguel de Icaza and Anders Hejlsberg

Second Edition

Framework Design Guidelines

Conventions, Idioms, and Patterns for Reusable .NET Libraries

Krzysztof Cwalina
Brad Abrams

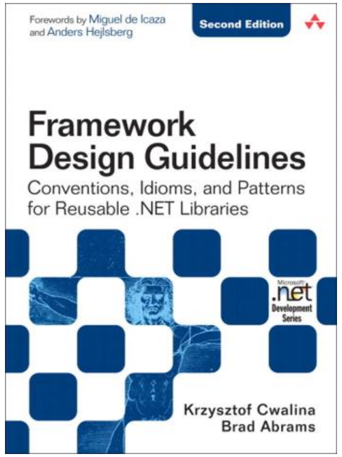
MISRA

Google

https://google.github.io/styleguide/javaguide.html

Static analysis

- Verification of an artifact (documentation, model, code) **without executing** it
- Approaches for evaluating code quality
  - Syntactic correctness: checked by compiler
  - Coding conventions: 'linter'



Forewords by Miguel de Icaza and Anders Hejlsberg

Second Edition

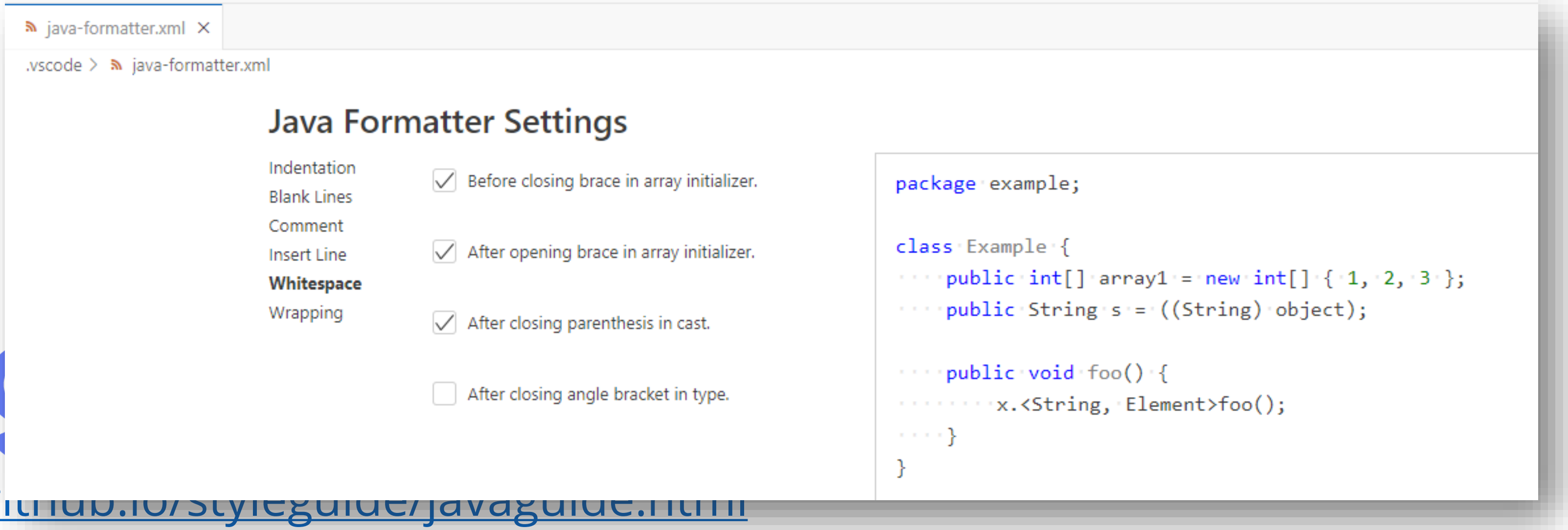
Framework Design Guidelines

Conventions, Idioms, and Patterns for Reusable .NET Libraries

.net Development Series

Krzysztof Cwalina
Brad Abrams

https://google.github.io/styleguide/javaguide.html



Java Formatter Settings

☑ Before closing brace in array initializer.

☑ After opening brace in array initializer.

☑ After closing parenthesis in cast.

□ After closing angle bracket in type.

```
package example;

class Example {
    public int[] array1 = new int[] { 1, 2, 3 };
    public String s = ((String) object);

    public void foo() {
        x.<String, Element>foo();
    }
}
```

Statikus analízis

- Verification of an artifact (documentation, model, code) **without executing** it
- Approaches for evaluating code quality
  - Syntactic correctness: checked by compiler
  - Coding conventions: ‘linter’
  - Correctness: static analysis (e.g. null pointer analysis)

SpotBugs

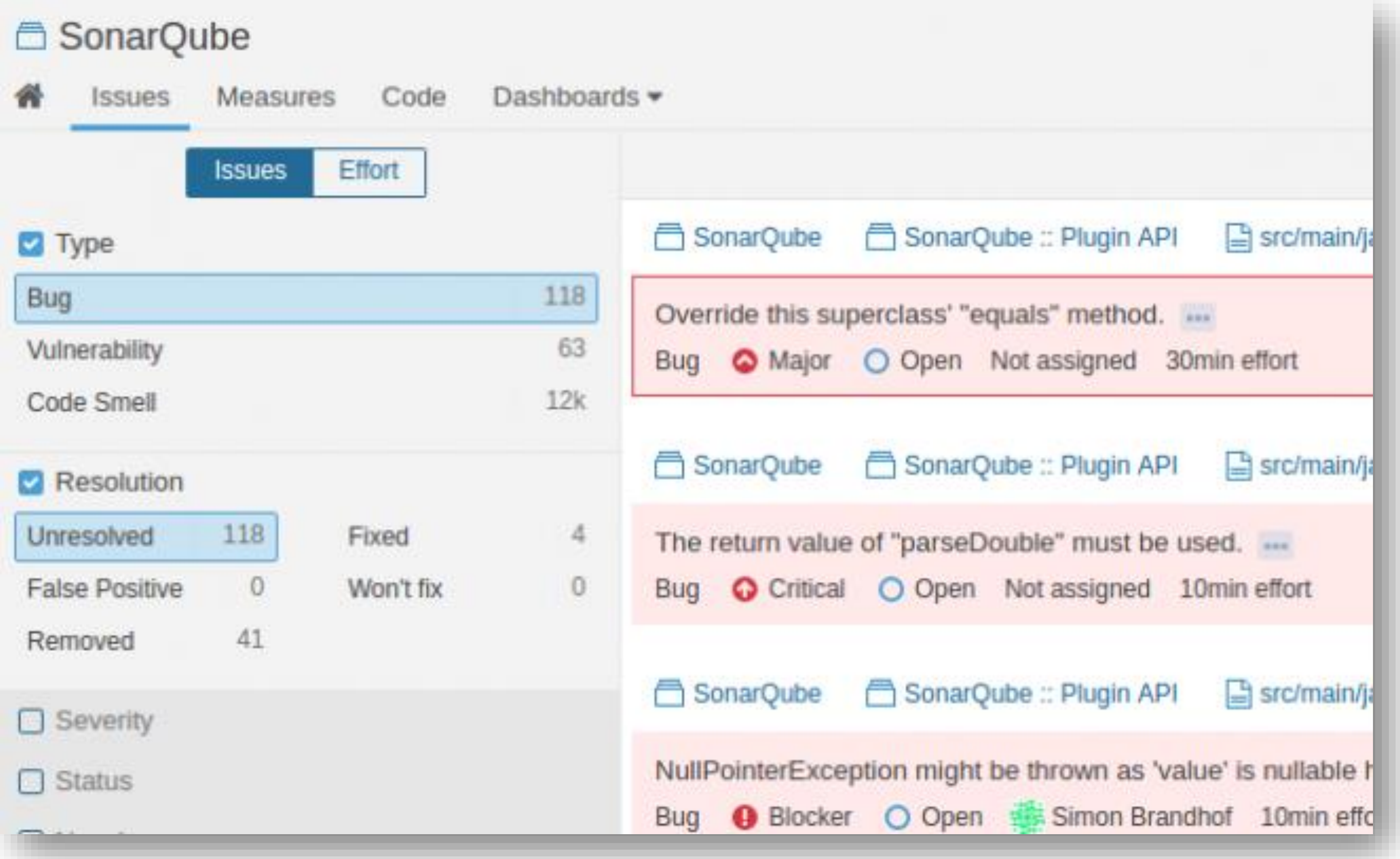
sonarcloud

Statikus analízis

- Verification of an artifact (documentation, model, code) **without executing** it
- Approaches for evalu
  - Syntactic correctness:
  - Coding conventions: '
  - Correctness: static an

SpotBugs

sonarcloud



SonarQube

☑ Type

Bug	118
Vulnerability	63
Code Smell	12k

☑ Resolution

Unresolved	118	Fixed	4
False Positive	0	Won't fix	0
Removed	41		

□ Severity

□ Status

SonarQube SonarQube :: Plugin API src/main/j

Override this superclass' "equals" method.

Bug Major ○ Open Not assigned 30min effort

SonarQube SonarQube :: Plugin API src/main/j

The return value of "parseDouble" must be used.

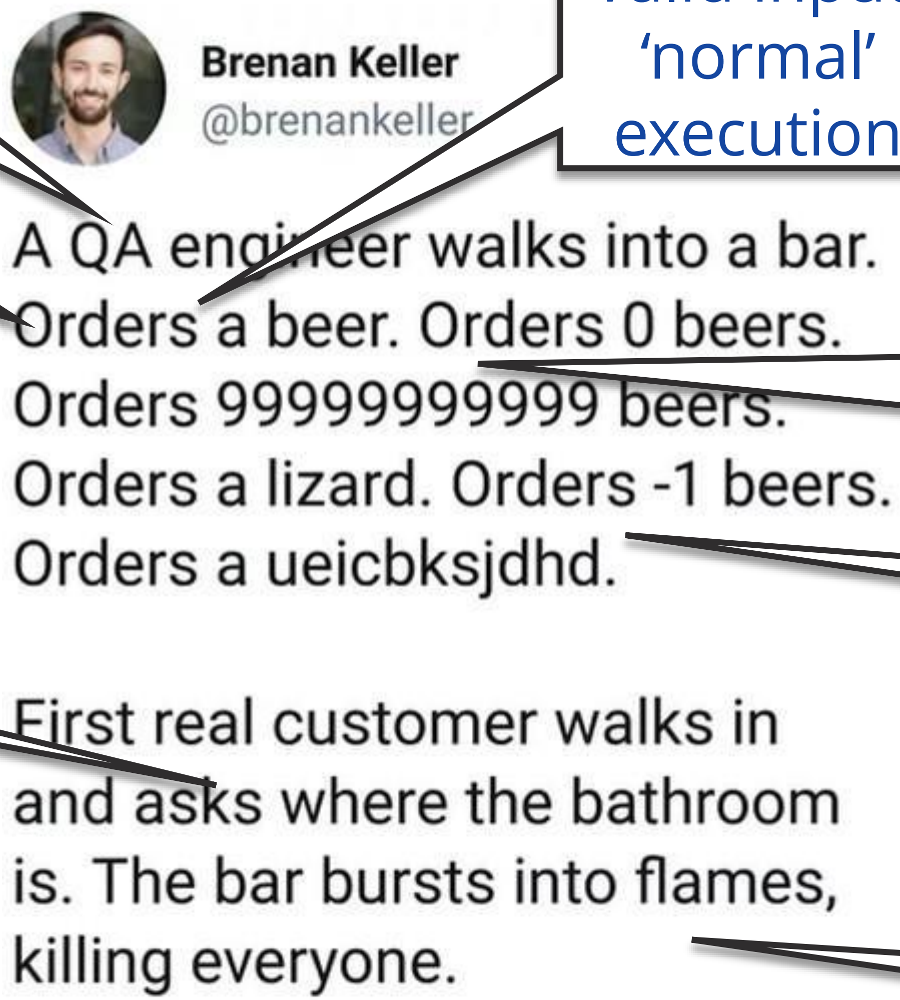
Bug Critical ○ Open Not assigned 10min effort

SonarQube SonarQube :: Plugin API src/main/j

NullPointerException might be thrown as 'value' is nullable h

Bug Blocker ○ Open Simon Brandhof 10min effo

Test design



Brenan Keller
@brenankeller

A QA engineer walks into a bar. Orders a beer. Orders 0 beers. Orders 99999999999 beers. Orders a lizard. Orders -1 beers. Orders a ueicbksjdhd.

First real customer walks in and asks where the bathroom is. The bar bursts into flames, killing everyone.

Quality Assurance

Use case

Valid input, 'normal' execution

Sytem under test

Valid input, alternative execution

Invalid input

Use case

Error

Automated test design

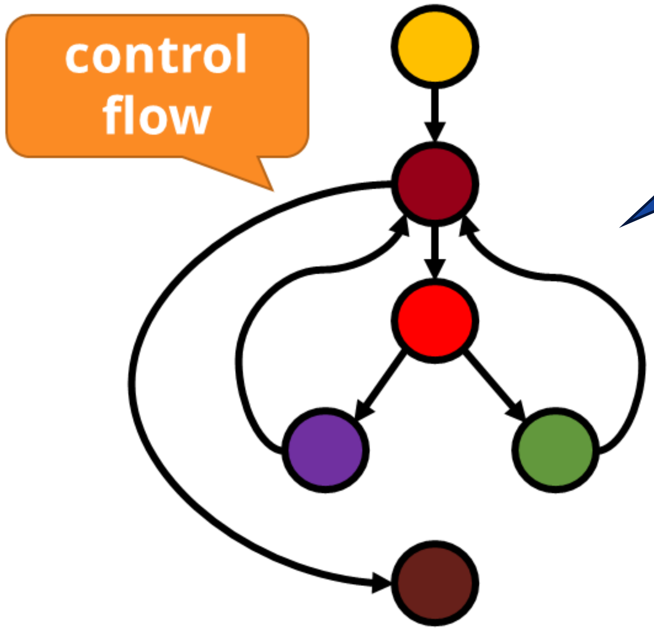
- **Model-based** testing, test generation (M2T, M2M)
- **Strucutre-based** techniques:

Representing the **structure** of source code

Source code:

```
int a = read();
while(a < 16) {
  if(a < 10) {
      a += 2;
  } else {
      a++;
  }
}
a = a * 2;
```

Control-flow graph (CFG):



Code coverage evaluation

Automated test design

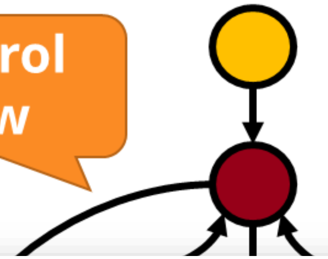
- **Model-based** testing, test generation (M2T, M2M)
- **Strucutre-based** techniques:

Source code:

```
int a = read();
while(a < 16) {
  if(a < 10) {
      a += 2;
  } else {
      a++;
  }
}
a = a * 2;
```

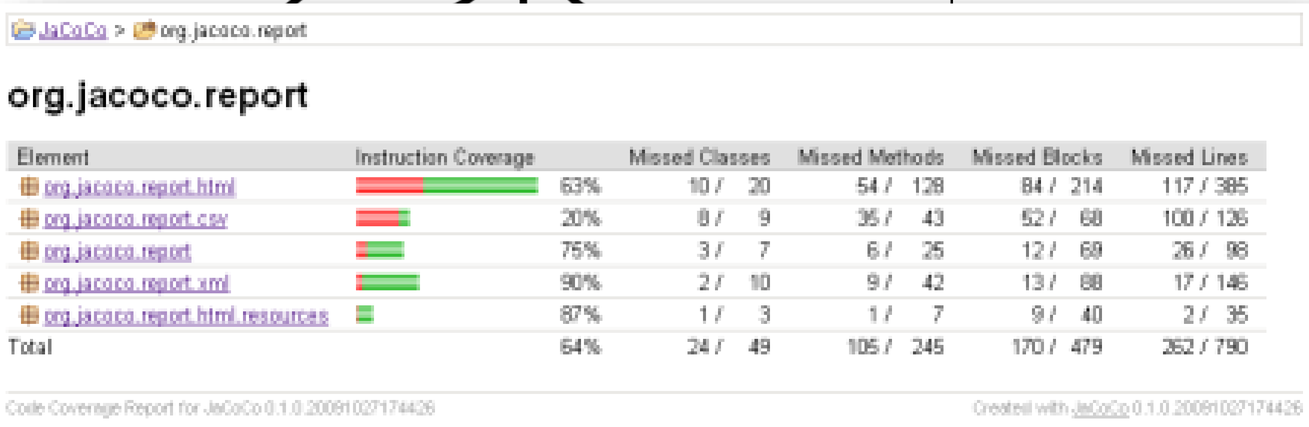
Representing the **structure** of source code

Control-flow graph (CFG):



control flow

Code coverage evaluation



JaCoCo > org.jacoco.report

org.jacoco.report

Element	Instruction Coverage		Missed Classes	Missed Methods	Missed Blocks	Missed Lines
org.jacoco.report.html	🟥🟩	63%	10 / 20	54 / 128	84 / 214	117 / 385
org.jacoco.report.csv	🟥🟩	20%	8 / 9	35 / 43	52 / 68	100 / 126
org.jacoco.report	🟥🟩	75%	3 / 7	6 / 25	12 / 69	26 / 98
org.jacoco.report.xml	🟥🟩	90%	2 / 10	9 / 42	13 / 88	17 / 146
org.jacoco.report.html.resources	🟩	87%	1 / 3	1 / 7	9 / 40	2 / 35
Total		64%	24 / 49	105 / 245	170 / 479	262 / 790

Code Coverage Report for JaCoCo 0.1.0.20081027174426

Created with JaCoCo 0.1.0.20081027174426

Benchmarking, data analysis

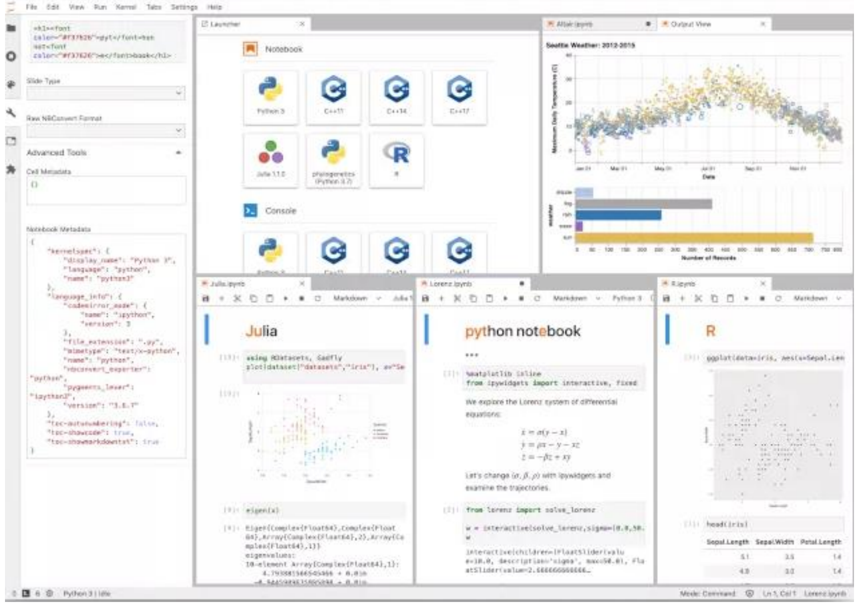
- How to evaluate software performance?
  - Measurement setup
  - Software benchmarking best practices
  - Tools: JMH, VisualVM
- How to evaluate measurement results?
  - Data processing
  - Data visualizations
  - Hypothesis testing

VisualVM

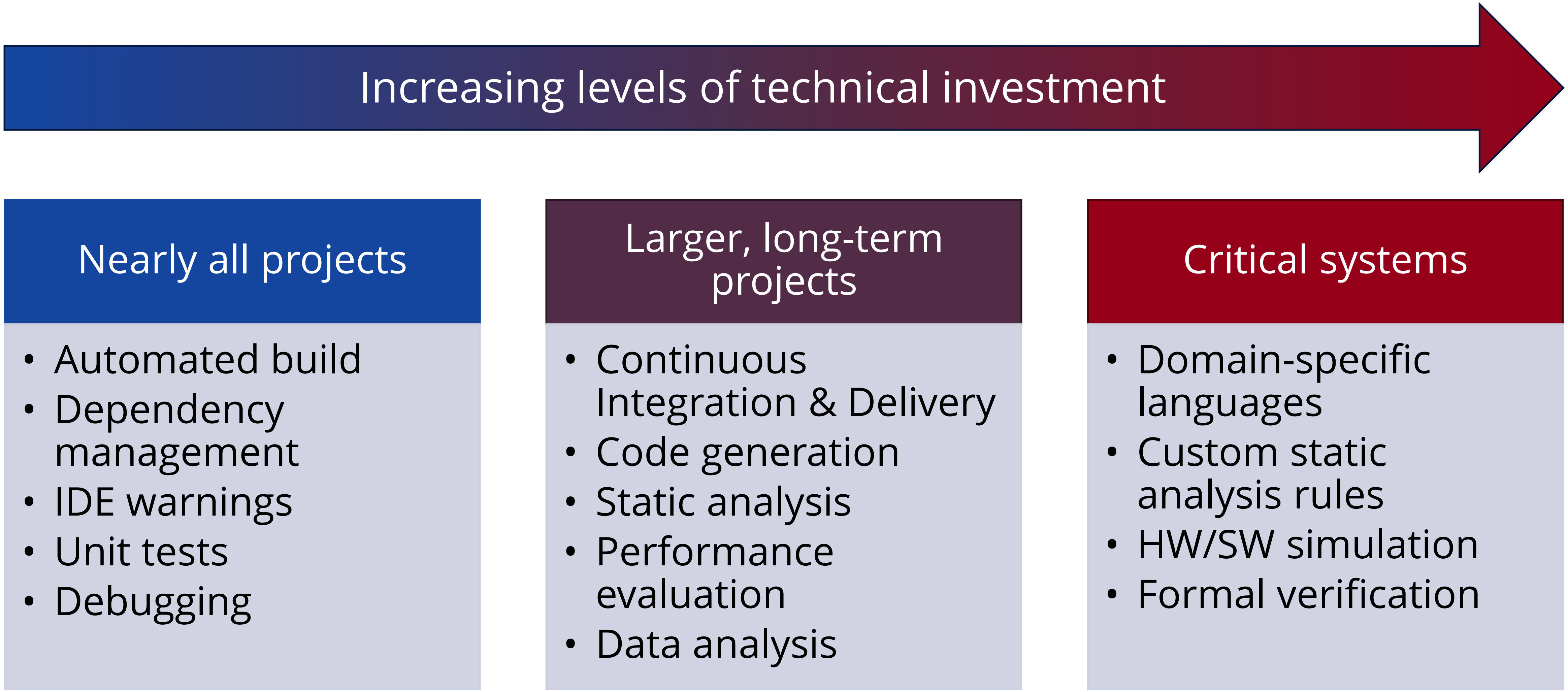
openjdk/jmh

https://openjdk.org/projects/code-tools/jmh

jupyter



Where should we use these methods?



Increasing levels of technical investment

Nearly all projects

- Automated build
- Dependency management
- IDE warnings
- Unit tests
- Debugging

Larger, long-term projects

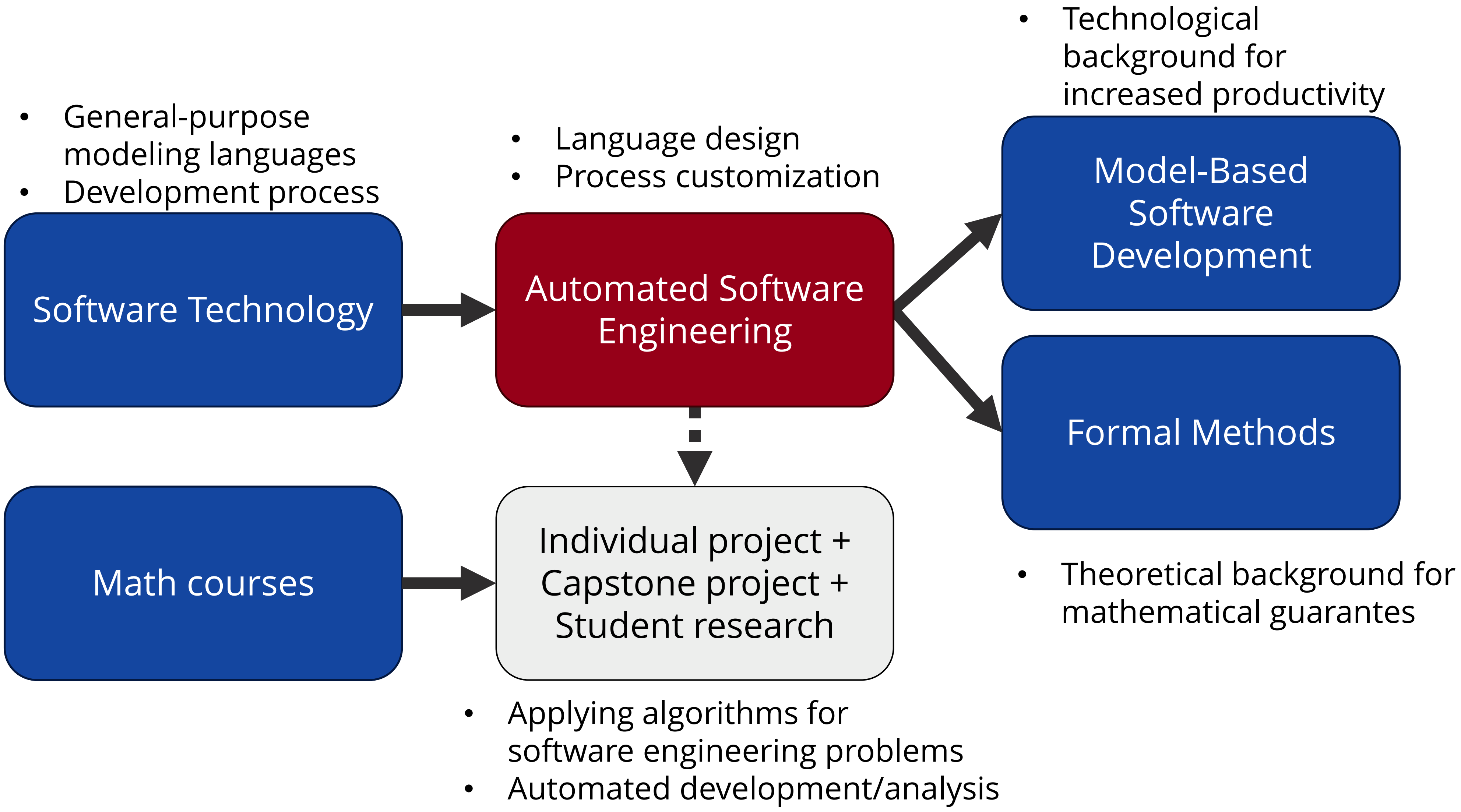
- Continuous Integration & Delivery
- Code generation
- Static analysis
- Performance evaluation
- Data analysis

Critical systems

- Domain-specific languages
- Custom static analysis rules
- HW/SW simulation
- Formal verification

Outlook

Related courses



- General-purpose modeling languages
- Development process

Software Technology

- Language design
- Process customization

Automated Software Engineering

- Technological background for increased productivity

Model-Based Software Development

Formal Methods

- Theoretical background for mathematical guarantes

Math courses

Individual project + Capstone project + Student research

- Applying algorithms for software engineering problems
- Automated development/analysis

Industry relations

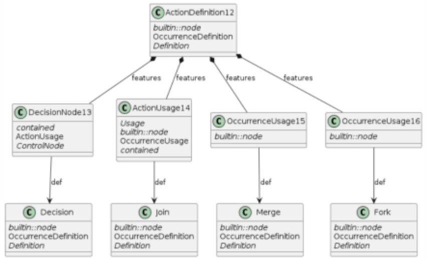
- Automated software development techniques enable us to create novel **development environments**
- These may comprise
  - Mathematical **algorithms** / physical **simulators**
  - **Artificial Intelligence** models
  - Evaluation of **safety rules**
- This lets us work with **domain experts**

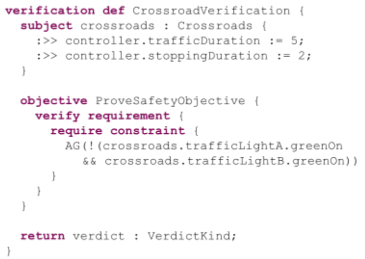
In the following, we will show some industrial collaborations and recent results of our research group. Everyone is welcome to participate if they wish to do their capstone project at our group!

Next-Generation MBSE – SysML v2

Participation

- Authorship in the specification
- SMC Leadership
- Formal Methods WG (leading)
- Conformance WG (leading)
- Execution WG (core member)
- Semantics WG (core member)
- Reference Implementation WG
- Certification WG





```
verification def CrossroadVerification {
  subject crossroads : Crossroads {
    :>> controller.trafficDuration := 5;
    :>> controller.stoppingDuration := 2;
  }

  objective ProveSafetyObjective {
    verify requirement {
      require constraint {
        AG(!(crossroads.trafficLightA.greenOn
          && crossroads.trafficLightB.greenOn))
      }
    }
  }

  return verdict : VerdictKind;
}
```

Recent results

Several new **partners** and **cooperations**

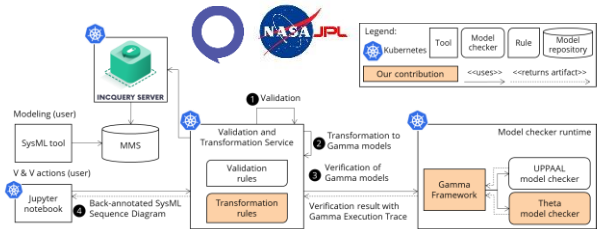
First SysML v2 formal **verification** and **model generator tool**

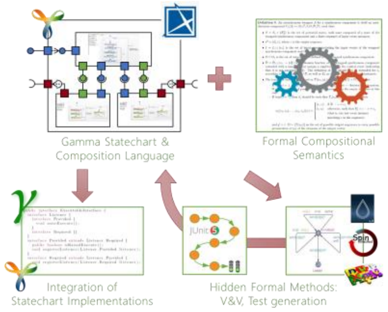
Coordinating first **academic paper** about SysML v2 verification

Automated V&V in Systems Engineering

“Use your models!”

- Semantic analysis
- Early verification of critical requirements
- Code/Test/Monitor generation





Gamma Statechart & Composition Language
Formal Compositional Semantics
Integration of Statechart Implementations
Hidden Formal Methods: V&V, Test generation

Gamma: Bridge MBSE and FM

- Formal verification integrated into SysML, SysML v2 and DSLs

Recent results

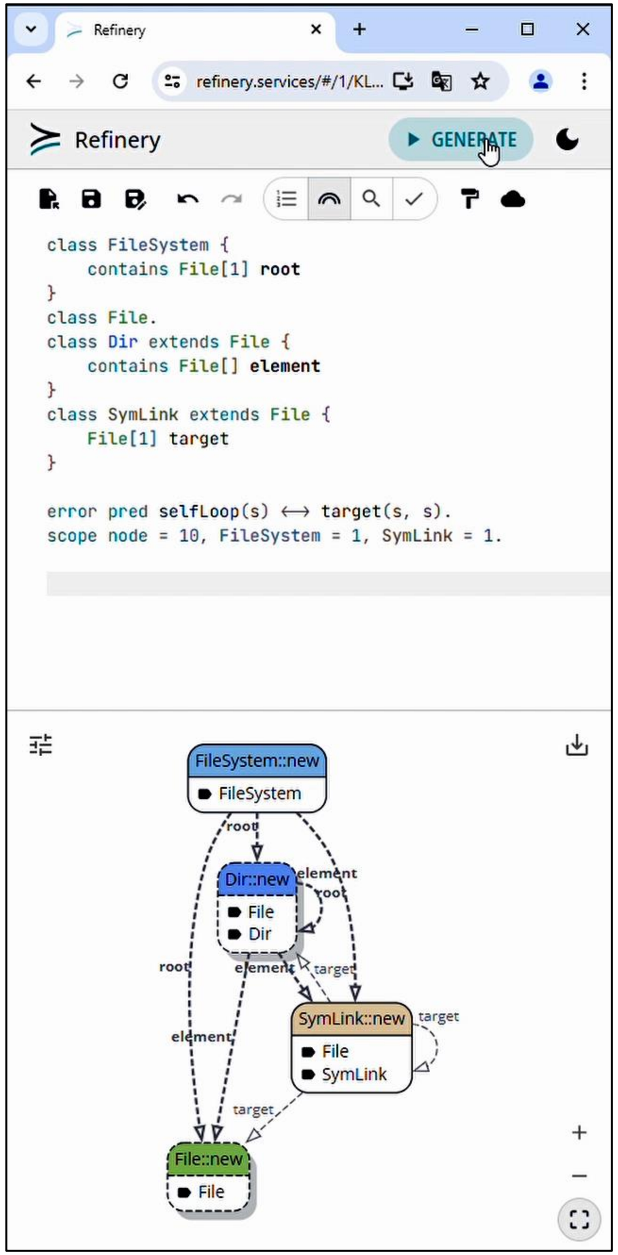
Automated analysis and generation for **railway systems**

Automated formal methods in **engineering workflows** (NASA, IQL)

Modern tools

- **Tool development** with modern technologies
- Interactive state-of-the-art **web-based editors**
- Generation algorithms **running on server**
- Various domains

  blockchain architecture | chemical reaction railway topology | modeling environment, satellite network | video game maps
- **Goal:** support **engineers** and **experts** solving problems



```
class FileSystem {
    contains File[1] root
}
class File.
class Dir extends File {
    contains File[] element
}
class SymLink extends File {
    File[1] target
}

error pred selfLoop(s) ⟷ target(s, s).
scope node = 10, FileSystem = 1, SymLink = 1.
```

Recent Results

- Refinery

  Continuously deployed:

  https://refinery.tools

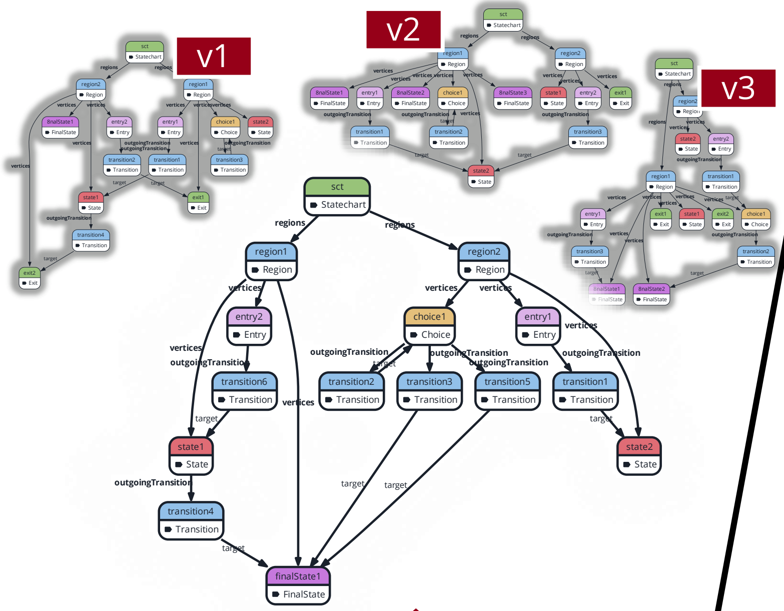
- Amazon Research Award

  First in the region

amazon | science

Graph analysis and synthesis

- Powerful mathematical **analysis techniques** for models
- Novel graph-based logic solver for the **automated synthesis** of design alternatives
- **Precision + Scalability**
- **Goal:** solve problems with complex structure



analysis ⬇ ⬆ synthesis

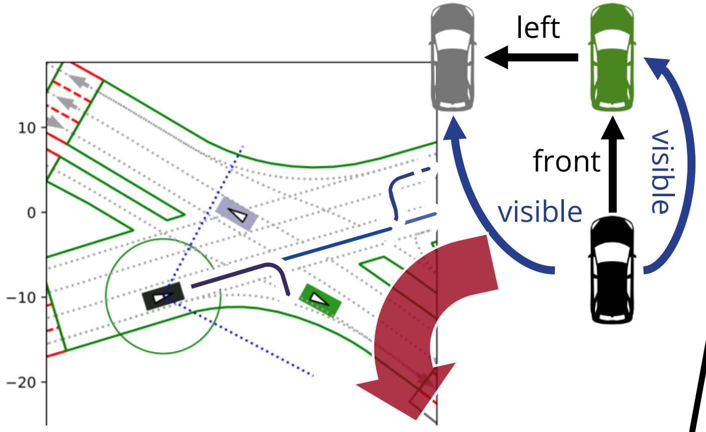
☒ ☑	validation
$$$	cost
◢◢	performance
%%	coverage

Recent Results

- Research project *VERIFIABLE AI/ML TECHNIQUES FOR PNT APPLICATIONS*

Verification/Testing of AI/ML Applications

- AI applications are **data-oriented** systems
- **Complex, dynamic** environment
- Novel **generation** + Advanced **simulators** → Diverse **tests**
- Systematic testing of **AI applications**

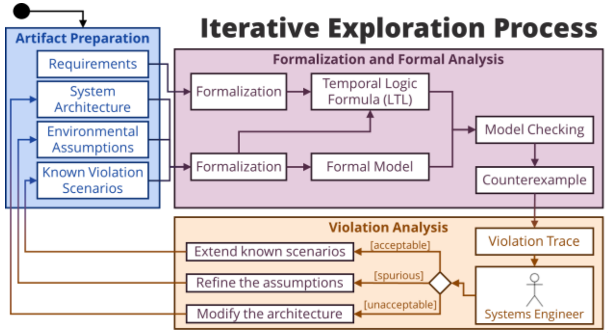


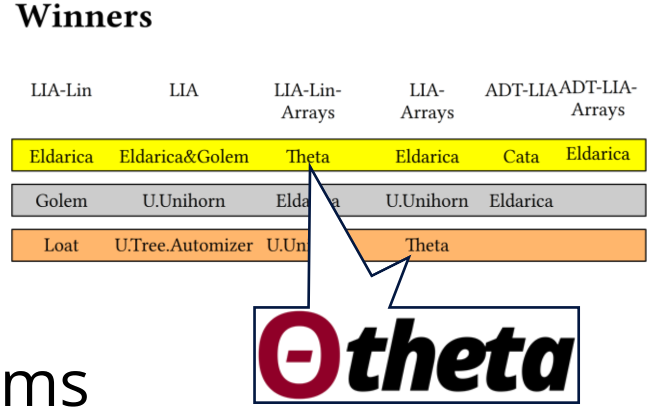
Recent Results

- R&D project with Knorr-Bremse
- Research project with USA Navy *Office of Naval Research Global*

Finding Critical Design Errors (Verification)

- **Methodology** to find **errors** in system design
- **State-of-the-art** verification algorithms
  - (Concurrent) software verification
  - Real-time systems
  - High-level engineering models
  - Component-based systems




Winners

LIA-Lin	LIA	LIA-Lin-Arrays	LIA-Arrays	ADT-LIA	ADT-LIA-Arrays
Eldarica	Eldarica&Golem	Theta	Eldarica	Cata	Eldarica
Golem	U.Unihorn	Elda[illegible]a	U.Unihorn	Eldarica	
Loat	U.Tree.Automizer	U.Un[illegible]	Theta		

Recent results

Found **real problems** in industrial use-cases (automotive, railway)

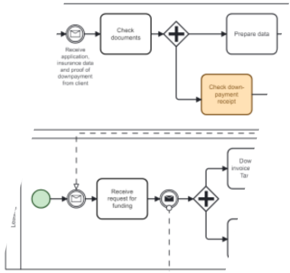
Last 5 years: 17 **first prizes** in university level TDK competitions

Recent theta **awards**:
🏆 Distinguished artifact
🥇 Horn close competition

Advancing novel applications – key activities

BLOCKCHAIN Koalíció

- **Hungarian Blockchain Coalition**
  - Prof. Pataricza – member of the board
  - I. Kocsis: Education WG lead, L. Gönczy: FinTech WG
- **Supporting the EMAP project (PM/NAV)**
  - “Even-based Data-sharing Platform” pilot
  - Employer data provisions: event-based, single-channel
  - Blockchain-based implementation in preparation
- **CBDC research cooperation with MNB**
  - Mapping out: blockchain ↔ Central Bank Digital Currency
  - Payment, car leasing, energy support, industrial cooperation
  - Currently: “ecosystem” research
- **EDGE-Skills: data veracity in EU data spaces**
  - Blockchain-backed Verifiable Credentials


Receive application, insurance data and proof of downpayment from client
Check documents
Prepare data
Check down-payment receipt
Receive request for funding
Dow invoice Ta

Recent results

Energy price support CBDC prototype: BIS Rosalind finalist

Fabric ↔ Ethereum CBDC bridge in Hyperledger Cacti

Smart gas meters and readings – in production

Welcome everyone to the specialization!

We look forward to also working with you in the project courses of our research group!