

SECRET NINJA FORMAL METHODS

Joseph R. Kiniry and Daniel M. Zimmerman

Emanuele Rudel

MOTIVATION

High quality software

Correct

Reusable

Robust

Extendible

Using **formal methods**

Extensively

Seamlessly

FORMAL METHODS

“Formal methods refer to mathematically rigorous techniques and tools for the specification, design and verification of software and hardware systems.”

– Ricky W. Butler, senior research engineer at NASA

BON

Assertions

Design by Contract

JML

TOOLS

Common JML tool suite

Typechecker

Compiler

Runtime Checking Environment

Documentation tool

JUnit with JML

Ninja Advice

Hide tools in the IDE

STATIC CHECKING

Runtime checking and testing are not enough

Extended **S**tatic **C**hecking / Java2

Null Pointer Exception

Class Cast Exception

Out of Bounds Exception

Integrated in the IDE

ADDITIONAL TOOLS

Moodle

GForge, Trac

Eclipse Plugins

SVN, Git

DESIGN & ANALYSIS FIRST

1. Define the system thoroughly
2. Build the system using formal methods

Use BON process together with co-analysis and co-design

CLASS	CITIZEN	Part: 1/1
TYPE OF OBJECT Person born or living in a country	INDEXING cluster: <i>CIVIL_STATUS</i> created: 1993-03-15 jmn revised: 1993-05-12 kw	
Queries	Name, Sex, Age, Single, Spouse, Children, Parents, Impediment to marriage	
Commands	Marry. Divorce.	
Constraints	Each citizen has two parents. At most one spouse allowed. May not marry children or parents or person of same sex. Spouse's spouse must be this person. All children, if any, must have this person among their parents.	

DESIGN & ANALYSIS FIRST

CLASS	CITIZEN	Part: 1/1
TYPE OF OBJECT Person born or living in a country	INDEXING cluster: CIVIL_STATUS created: 1993-03-15 jmn revised: 1993-05-12 kw	
Queries	Name, Sex, Age, Single, Spouse, Children, Parents, Impediment to marriage	
Commands	Marry, Divorce.	
Constraints	Each citizen has two parents. At most one spouse allowed. May not marry children or parents or person of same sex. Spouse's spouse must be this person. All children, if any, must have this person among their parents.	



```
public class Citizen {
    private String name;
    private int age;
    private boolean single;
    private boolean married;
    private boolean divorced;
    private boolean deceased;
    private boolean spouse;
    private boolean child;
    private boolean parent;
    private boolean impediment;

    public Citizen(String name, int age, boolean single, boolean married, boolean divorced, boolean deceased, boolean spouse, boolean child, boolean parent, boolean impediment) {
        this.name = name;
        this.age = age;
        this.single = single;
        this.married = married;
        this.divorced = divorced;
        this.deceased = deceased;
        this.spouse = spouse;
        this.child = child;
        this.parent = parent;
        this.impediment = impediment;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public boolean isSingle() {
        return single;
    }

    public boolean isMarried() {
        return married;
    }

    public boolean isDivorced() {
        return divorced;
    }

    public boolean isDeceased() {
        return deceased;
    }

    public boolean isSpouse() {
        return spouse;
    }

    public boolean isChild() {
        return child;
    }

    public boolean isParent() {
        return parent;
    }

    public boolean hasImpediment() {
        return impediment;
    }
}
```

Abstraction

Specification

Implementation

Think twice, build once - process

CONCLUSION

Proper tools setup

Deep integration with the language

Accompanied with thought-through process

High quality software using formal methods