



Stichting NIOC

Stichting NIOC en de NIOC kennisbank

Stichting NIOC (www.nioc.nl) stelt zich conform zijn statuten tot doel: het realiseren van congressen over informatica onderwijs en voorts al hetgeen met een en ander rechtstreeks of zijdelings verband houdt of daartoe bevorderlijk kan zijn, alles in de ruimste zin des woords.

De stichting NIOC neemt de archivering van de resultaten van de congressen voor zijn rekening. De website www.nioc.nl ontsluit onder "Eerdere congressen" de gearchiveerde websites van eerdere congressen. De vele afzonderlijke congresbijdragen zijn opgenomen in een kennisbank die via dezelfde website onder "NIOC kennisbank" ontsloten wordt.

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2025, gehouden op donderdag 27 maart 2025 jl. en georganiseerd door Hogeschool Windesheim). Bij elkaar zo'n 1500 bijdragen!

We roepen je op, na het lezen van het document dat door jou is gedownload, de auteur(s) feedback te geven. Dit kan door je te registreren als gebruiker van de NIOC kennisbank. Na registratie krijg je bericht hoe in te loggen op de NIOC kennisbank.

Het eerstvolgende NIOC vindt plaats op 18 maart 2027 in Arnhem en wordt georganiseerd door HAN University of Applied Sciences.

Reacties over de NIOC kennisbank en de inhoud daarvan kun je richten aan de beheerder:

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.



Software Architecture Education

A Practical Approach with HUSACCT

Leo Pruijt, HU University of Applied Science, Utrecht

Agenda



- Software Architecture
- SA Education
- Architecture Reconstruction with HUSACCT
- Architecture Compliance Checking with HUSACCT
- Questions

© HU

NIOC 2015

2

© HU

NIOC 2015

3

Software Architecture (SA)

■ Software architecture is of major importance to achieve

- the business goals
- functional and non-functional requirements

■ However ...

- Architecture models are often obsolete or incomplete
- Architectural models tend to be of a high-level of abstraction
 - Hard to understand by students and practitioners
 - Deviations of the software architecture arise easily during the development and evolution of a system



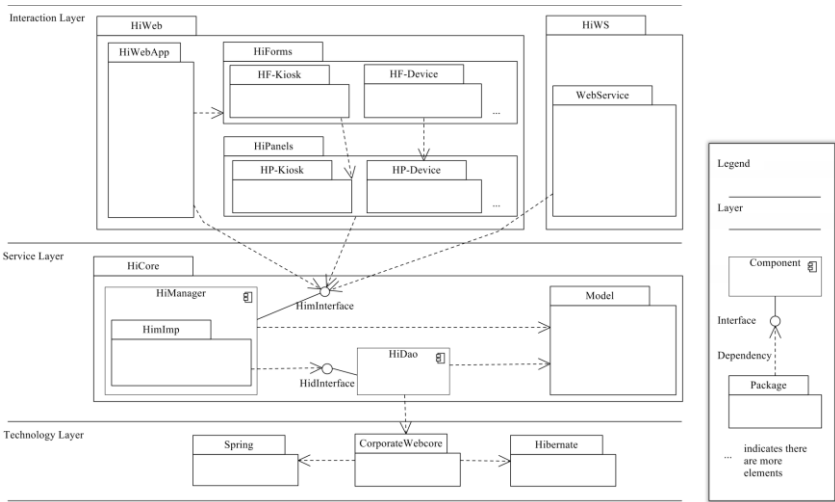
© HU

NIOC 2015

4

SA Example (Schiphol Group)





NIOC 2015

2

Module View



Five types of modules are visible in the example

- Subsystems, Layers, Components, Interfaces, External Systems

Rules of different types are used in the example

- Some of them specific for a certain type of module
- E.g., 'Back call ban' and 'Skip call ban' are specific for Layers

Type of Rule	Example (E)
Is not allowed to use	HiPanels is not allowed to use HiWS.
Back call ban	Service Layer is not allowed to use the Interaction Layer .
Facade convention	Component HiManager may be accessed only via Him Interface .
Is only allowed to use	HiForms is only allowed to use HiPanels.
Is the only module allowed to use	CorporateWebcore is the only module allowed to use Hibernate.

© HU

NIOC 2015

5

SA Education at the HU



- Base SA on foundation of Analysis & Design
 - Preceding courses (in year 1 and 2) on
 - Business processes, Use cases
 - Domain model, Object interaction, Database design
- Relate architecture models to code
 - Architecture Reconstruction with HUSACCT
 - Architecture Compliance Checking with HUSACCT
 - In year 2 (basic) and 3 (advanced)
- Apply in student projects
- Apply on practical cases (open source, visit companies)

© HU

NIOC 2015

6

HUSACCT: Hogeschool Utrecht
Software Architecture Compliance Checking Tool

Menu

- Analyse Application
- Explore Implemented Modular Architecture
- Define Intended Modular Architecture
- Check Conformance
- Explore Results Conformance Check

Analyse implemented architecture

- Browse Modules & Dependencies
- Visualize Modules & Dependencies
- Report Modules & Dependencies

Define intended architecture

- Browse Violations
- Visualize Violations
- Report Violations
- Export Violations

© HU NIOO 2015 7

Outstanding Characteristics



- HUSACCT is free-to-use & open source
 - Download, video & manual at <http://husacct.github.io/HUSACCT/>
- Support of rich sets of Module and Rule Types
 - 5 common Module Types with different semantics
 - Subsystem, Layer, Component, Interface, External system
 - 11 common Rule Types
- Extensive Semantic Support of the Module and Rule Types, e.g.:
 - Automatic creation of default rules, according to the Module Type
 - Type of Module determines which Rule Types are selectable
- Configurable support
 - Enable/Disable rules, Exception rules, Default rule configuration

© HU

NIOC 2015

8

Architecture Reconstruction

Architecture reconstruction:
comprehending and documenting an architecture
based on the source code
(and other information)

The diagram illustrates the process of architecture reconstruction. It starts with 'Source code' (represented by a stack of papers) and 'Extraction Techniques' (represented by a box with 'B extends A' and '* (a)/s --> ...'). An arrow labeled '1' points from these inputs to an 'Extracted architecture' box. This box contains a UML class diagram with classes A, B, and C, where B inherits from A and A is associated with C. A curved arrow labeled '2' points from the 'Extracted architecture' box back to itself, labeled 'Refinement'.

Source: Ducasse & Polle (2009)

© HU

NIOC 2015

9

Analyse Implemented Architecture

Decomposition View: How is the software structured?

The screenshot shows the 'Analysed Application Overview' window. The 'Decomposition View' tab is active, showing a tree structure of the application's components. The tree starts with 'Application' and branches into 'husact', 'Main', 'ServiceProvider', 'analyse', 'bootstrap', 'common', 'control', 'define', 'DefineServiceImpl', 'DomainParser', 'IDefineService', 'domain', 'persistence', 'presentation', 'task', 'graphics', 'validate', 'libraries', 'com', 'common', 'Logger', 'java', 'javax', 'j', and 'org'. The 'Statistics' panel on the right shows the following data:

Complete Application	
Packages:	104
Classes:	957
Lines of Code:	135923
Dependencies:	62626

Selection	
Packages:	19
Classes:	92
Lines of Code:	9841

Buttons at the bottom: Export Dependencies, Cancel

© HU

NIOC 2015

10

Analysis Application Overview

Decomposition View Usage View

From Module

- Application
 - husacct
 - Main
 - ServiceProvider
 - analyse
 - bootstrap
 - common
 - control
 - define
 - graphics
 - validate
 - xLibraries

To Module

- Application
 - husacct
 - Main
 - ServiceProvider
 - analyse
 - bootstrap
 - common
 - control
 - define
 - graphics
 - validate
 - xLibraries

Number of Dependencies: 39

Number Of Dependencies between the Selected Modules: 39

Dependency filter

☒ Include direct dependencies ☒ Include indirect dependencies

Dependencies between the Selected Modules

| From | To | Type | Line | Direct |
|--|---|-------------|------|----------|
| husacct.define.presentation.ApplicationInternalFrame | husacct.control.LocalChangeListener | Import | 3 | Direct |
| husacct.define.presentation.ApplicationInternalFrame | husacct.control.LocalChangeListener | Inheritance | 15 | Direct |
| husacct.define.presentation.dialog.AddModuleValuesDialog | husacct.control.ControlServiceImpl | Import | 5 | Direct |
| husacct.define.presentation.dialog.AddModuleValuesDialog | husacct.control.LocalChangeListener | Import | 6 | Direct |
| husacct.define.presentation.dialog.AddModuleValuesDialog | husacct.control.LocalChangeListener | Inheritance | 28 | Direct |
| husacct.define.presentation.dialog.AddModuleValuesDialog | husacct.control.ControlServiceImpl | Declaration | 44 | Direct |
| husacct.define.presentation.dialog.AddModuleValuesDialog | husacct.control.IControlService | Call | 44 | Indirect |
| husacct.define.presentation.dialog.ApplyRuleDialog | husacct.control.ControlServiceImpl | Import | 5 | Direct |
| husacct.define.presentation.dialog.ApplyRuleDialog | husacct.control.presentation.util.DialogUtils | Import | 6 | Direct |
| husacct.define.presentation.idtlan.AnnieModRuleDialog | husacct.control.ControlServiceImpl | Declaration | 50 | Direct |

Export Dependencies Cancel

implemented architecture diagram

Root > hasacc > define

```

graph TD
    IDefineService["«interface» IDefineService"]
    DefineServiceImpl["DefineServiceImpl"]
    presentation["presentation"]
    task["task"]
    domain["domain"]
    persistency["persistency"]

    IDefineService -.->|1| DefineServiceImpl
    IDefineService -.->|1| presentation
    IDefineService -.->|19| task
    IDefineService -.->|18| domain
    IDefineService -.->|192| persistency
    DefineServiceImpl -.->|1| presentation
    DefineServiceImpl -.->|68| task
    DefineServiceImpl -.->|17| domain
    DefineServiceImpl -.->|10| persistency
    presentation -.->|388| task
    presentation -.->|112| domain
    task -.->|454| domain
    domain -.->|192| persistency
  
```

Architecture Compliance Checking (ACC)

ACC verifies the conformance of

- Implemented program code to
- High-level models of architectural design

Currently ...

- Adoption of ACC in practice is limited
- Tool support of the common sets of module & rule types is limited

Research Goals ...

- Improve tool support
- Promote ACC in practice & education

Intended Architecture

Layers: L1, L2, L3

Rules

L3 is not allowed to use L2 (Back call), ...

Mapping

L1 → P1, ...

Violations

Architect

Conformance Check

Static Analysis

Program Code

Package P1

Class C1, Class C2 ...

Package P2

Class C3 ...

© HU

NIOC 2015

13

ACC Example: Modular Architecture HUSACCT_1.0

General GUI & Control

Graphics

General GUI Interface

GraphicsInterface

AnalyseInterface

DefineInterface

ValidateInterface

Analyse

Define

Validate

Top-level Components

Rules

| From-Module | Constraint | To-Module |
|-----------------------|-----------------------------------|-----------|
| Analyse | is not allowed to use | Define |
| Analyse | is not allowed to use | Validate |
| General GUI & Control | Is the only module allowed to use | Graphics |
| All five components | Facade convention | |

© HU

NIOC 2015

14

Define Intended Architecture



Define intended architecture

Module Hierarchy

SoftwareArchitecture

General GUI & Control

Analyse

Facade<Analyse>

Define

Validate

Graphics

Common

ExternalSystems

New module

Move up

Remove module

Move down

Warnings

View in Browser

<

>

Module Properties

Module name

Analyse

Description

Module Type

Component

Assigned Software Units

| Software unit name | Type |
|--------------------|---------|
| husacct.analyse | PACKAGE |

Add

Edit

Remove

Rules

| Rule type | To module | Enabled | Exceptions |
|-----------------------|-----------|---------|------------|
| Facade convention | | On | 2 |
| Is not allowed to use | Validate | On | 0 |
| Is not allowed to use | Define | On | 0 |

Add

Edit

Remove

Idle

Validate Conformance

Results of the Conformance Check



Validate conformance

Violations Per Rule

All Violations

Rules with Number of Violations

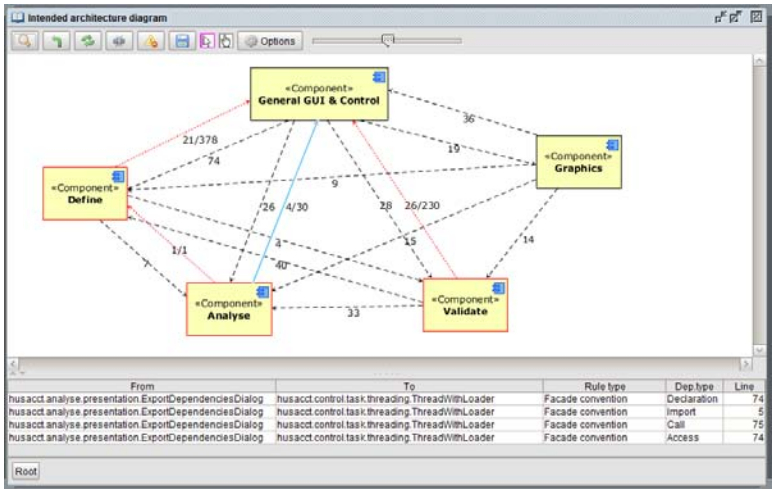
| Id | Logical module from | Rule type | Logical module to | Violations |
|----|-----------------------|-----------------------------|-------------------|------------|
| 1 | Analyse | Is not allowed to use | Define | 1 |
| 2 | Define.Presentation | Is not allowed to skip call | | 10 |
| 3 | Define.Task | Is not allowed to back call | | 113 |
| 4 | Define.Task | Is not allowed to skip call | | 18 |
| 5 | General GUI & Control | Facade convention | | 51 |

Violations

| From | To | Rule type | Dep.type | Direct | Line |
|--|---|-------------------|-------------|----------|------|
| husacct.analyse.presentation.ExportDependency | husacct.control.task.threading.ThreadWithLoader | Facade convention | Import | Direct | 5 |
| husacct.analyse.presentation.ExportDependency | husacct.control.task.threading.ThreadWithLoader | Facade convention | Access | Indirect | 74 |
| husacct.analyse.presentation.ExportDependency | husacct.control.task.threading.ThreadWithLoader | Facade convention | Declaration | Direct | 74 |
| husacct.analyse.presentation.ExportDependency | husacct.control.task.threading.ThreadWithLoader | Facade convention | Call | Direct | 75 |
| husacct.validate.presentation.BrowseViolations | husacct.control.task.threading.ThreadWithLoader | Facade convention | Access | Indirect | 201 |
| husacct.validate.presentation.BrowseViolations | husacct.control.task.threading.ThreadWithLoader | Facade convention | Declaration | Direct | 201 |
| husacct.validate.presentation.BrowseViolations | husacct.control.task.threading.ThreadWithLoader | Facade convention | Call | Direct | 202 |
| husacct.validate.presentation.BrowseViolations | husacct.control.task.threading.ThreadWithLoader | Facade convention | Access | Indirect | 393 |
| husacct.validate.presentation.BrowseViolations | husacct.control.task.threading.ThreadWithLoader | Facade convention | Declaration | Direct | 393 |
| husacct.validate.presentation.BrowseViolations | husacct.control.task.threading.ThreadWithLoader | Facade convention | Call | Direct | 394 |
| husacct.validate.presentation.BrowseViolations | husacct.control.task.threading.ThreadWithLoader | Facade convention | Access | Indirect | 398 |
| husacct.validate.presentation.BrowseViolations | husacct.control.task.threading.ThreadWithLoader | Facade convention | Declaration | Direct | 398 |
| husacct.validate.presentation.BrowseViolations | husacct.control.task.threading.ThreadWithLoader | Facade convention | Call | Direct | 399 |
| husacct.validate.presentation.BrowseViolations | husacct.control.task.threading.ThreadWithLoader | Facade convention | Import | Direct | 399 |

Intended Architecture Diagram

Top-level components, with dependencies and violations



© HU

NIOC 2015

17

Code Viewer Double-click on dependency



```
ExportDependenciesDialog.java
52 exportButton.setEnabled(false);
53 pathField.setEnabled(false);
54
55 getRootPane().setDefaultButton(exportButton);
56
57 add(pathLabel);
58 add(pathField);
59 add(browseButton);
60 add(exportButton);
61
62
63 private void setListeners(){
64     browseButton.addActionListener(new ActionListener() {
65         public void actionPerformed(ActionEvent arg0) {
66             showFileDialog();
67         }
68     });
69     exportButton.addActionListener(new ActionListener() {
70         public void actionPerformed(ActionEvent arg0) {
71             if(validateData()) {
72                 dispose();
73                 ThreadedDependency/Export dependencyExport = new ThreadedDe
74 selectedFile.getAbsolutePath());
75                 ThreadWithLoader analyseExportThread =
76 ServiceProvider.getInstance().getControlService().getThreadWithLoader(ServiceProvider.getInstance().getLocalService().getTranslate
77 "ExportingDependencies", dependencyExport);
78                 analyseExportThread.run();
79             }
80         }
81     });
82     private void showFileDialog() {
83         FileNameExtensionFilter filter = new FileNameExtensionFilter("xls", "xls");
```

© HU

NIOC 2015

18

Intended Architecture Diagram

Layers within Define, with dependencies and violations

The diagram illustrates the relationships between software layers and interfaces. It features five components: **«Interface» Define.Facade<Define>**, **«Layer» Define.Presentation**, **«Layer» Define.Task**, **«Layer» Define.Domain**, and **«Layer» Define.Infrastructure**. Dependencies are shown as dashed arrows with counts: Define.Facade depends on Define.Presentation (1), Define.Task (1), and Define.Infrastructure (19); Define.Presentation depends on Define.Task (113/113) and Define.Domain (10/10); Define.Task depends on Define.Domain (576) and Define.Infrastructure (18/18); Define.Domain depends on Define.Infrastructure (192). Violations are indicated by red dotted arrows: from Define.Presentation to Define.Task (388), from Define.Task to Define.Domain (18/18), and from Define.Domain to Define.Infrastructure (18/18).

© HU

NIOC 2015

19

Finally

■ More information:

■ Ask me: Leo Pruijt, leo.pruijt@hu.nl

■ Watch the video at <http://husacct.github.io/HUSACCT/>

■ Read the published papers

■ Contact me for course materials

■ Thank you for your attention!

■ Questions?

1) HUSACCT: Architecture Compliance Checking with Rich Sets of Module and Rule Types. 2014 IEEE/ACM Int. Conf. on Automated Software Engineering, ASE 2014-09, Vasteras, Sweden

2) Architecture Compliance Checking of Semantically Rich Modular Architectures: A Comparative Study of Tool Support. Int. Conf. on Software Maintenance, ICSM 2013-09, Eindhoven, NL

3) A Metamodel for the Support of Semantically Rich Modular Architectures in the Context of Architecture Compliance Checking. SAeroCon workshop, WICSA 2014-04, Sydney, Australia

4) On the Accuracy of Architecture Compliance Checking Support: Accuracy of Dependency Analysis and Violation Reporting. Int. Conf. on Program Comprehension, ICPC 2013-05, San Francisco, USA

© HU

NIOC 2015

20

NIOC 2015

10