Eric Verhulst
Open License Society
ALTREONIC

An Interacting Entities Modeling Methodology For Robust Systems Design

VALID 2010,

Nice 24.08.2010

An Interacting Entities Modelling Methodology for Robust Systems Design

OpenCookbook is a web-based requirements and specifications capturing tool supporting a coherent and unified system development methodology based on the Interacting Entities paradigm

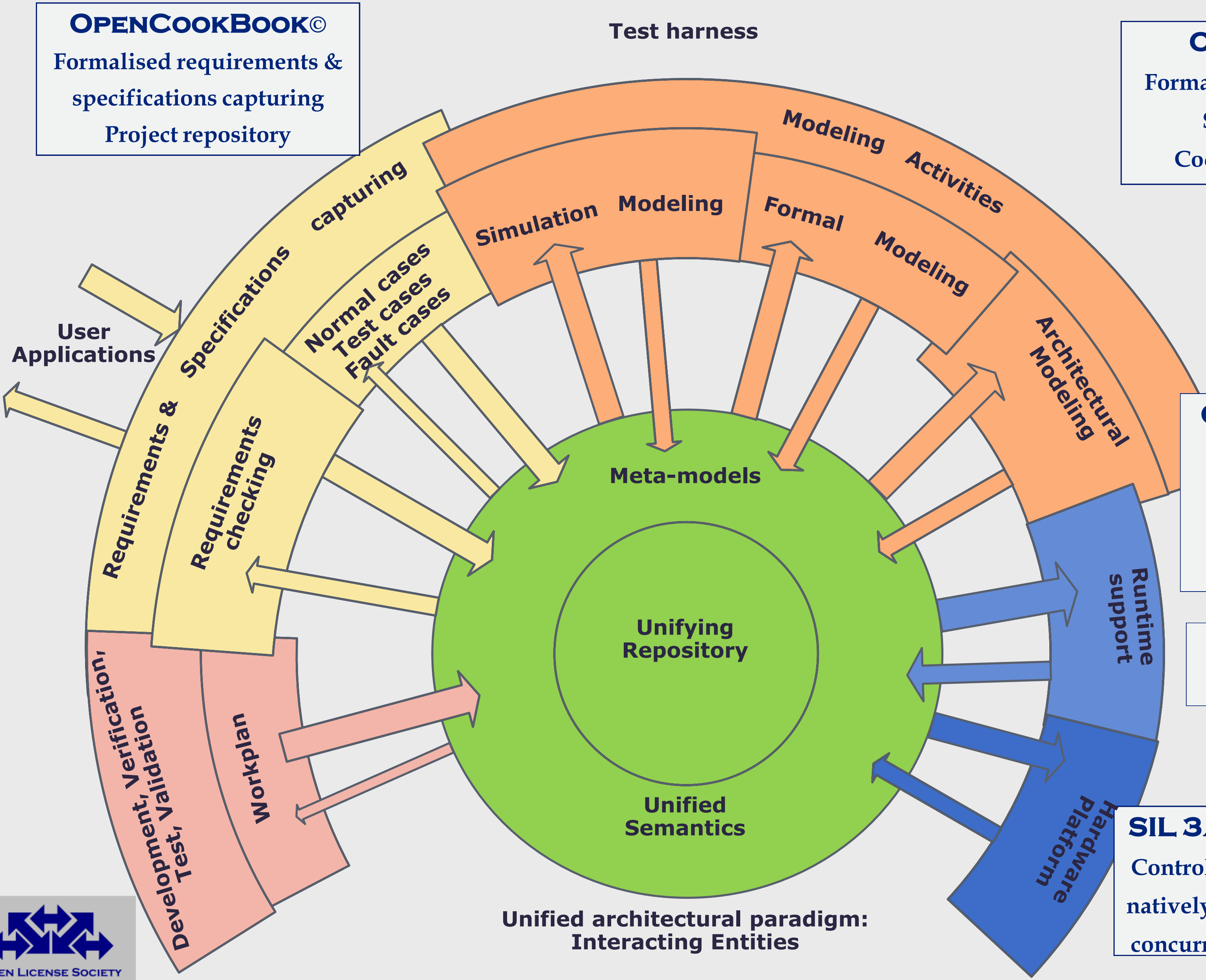
History

- Original R&D project of Open License Society:
  - Metamodel for systems engineering
    - “systems grammar”
  - **OpenSpecs** implemented as web portal
- **EVOLVE** ITEA project
  - **Evol**utionary **V**alidation, **V**erification and C**e**rtification
- **ASIL**: Flanders Drive project on developing a common safety engineering methodology
  - Why are engineering and safety standards so heuristic?
- Currently commercialised and further productised by Altreonic under **OpenCookBook**
  - part of **Concurrent Systems Composer** development framework

Unified Systems/Software engineering

OpenCookBook©
Formalised requirements & specifications capturing
Project repository

OpenVE ©
Formalized modelling
Simulation
Code generation



OpenComRTOS ©
Formally developed
Runtime support for
concurrency and
communication

OpenTracer ©
Visual Event Tracer

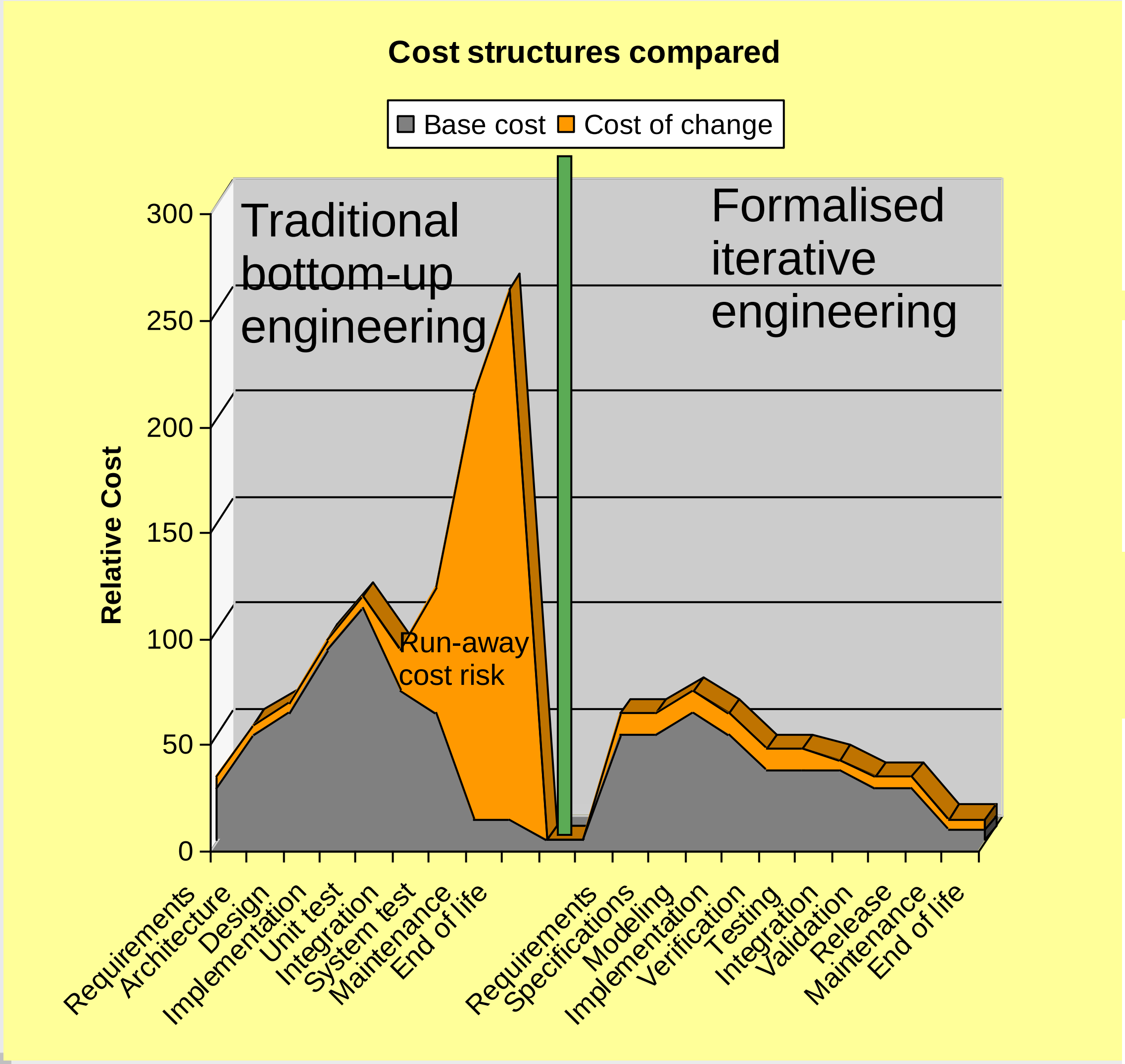
SIL 3/4 Controller ©
Control & processing platform
natively supporting distributed
concurrency & communication

Unified architectural paradigm:
Interacting Entities

Open License Society

Altreonic

Why FORMAL (ISED) ?



First time right

= Less residual errors

= Higher reliability

= Less costs

Incremental changes gives Requirements on process and architecture

Testing will only demonstrate absence of certain errors.

Formal verification can prove absence of any errors.

OpenCookBook design goals

- Universality:
  - modelling any type of system, i.e. physical, software, hardware etc. (possibly with heterogeneous parts)
- Scalability:
  - support the development from small to very large and complex systems
- Extensibility:
  - possibility to change and to modify the meta-model (based on system grammar structure of database)

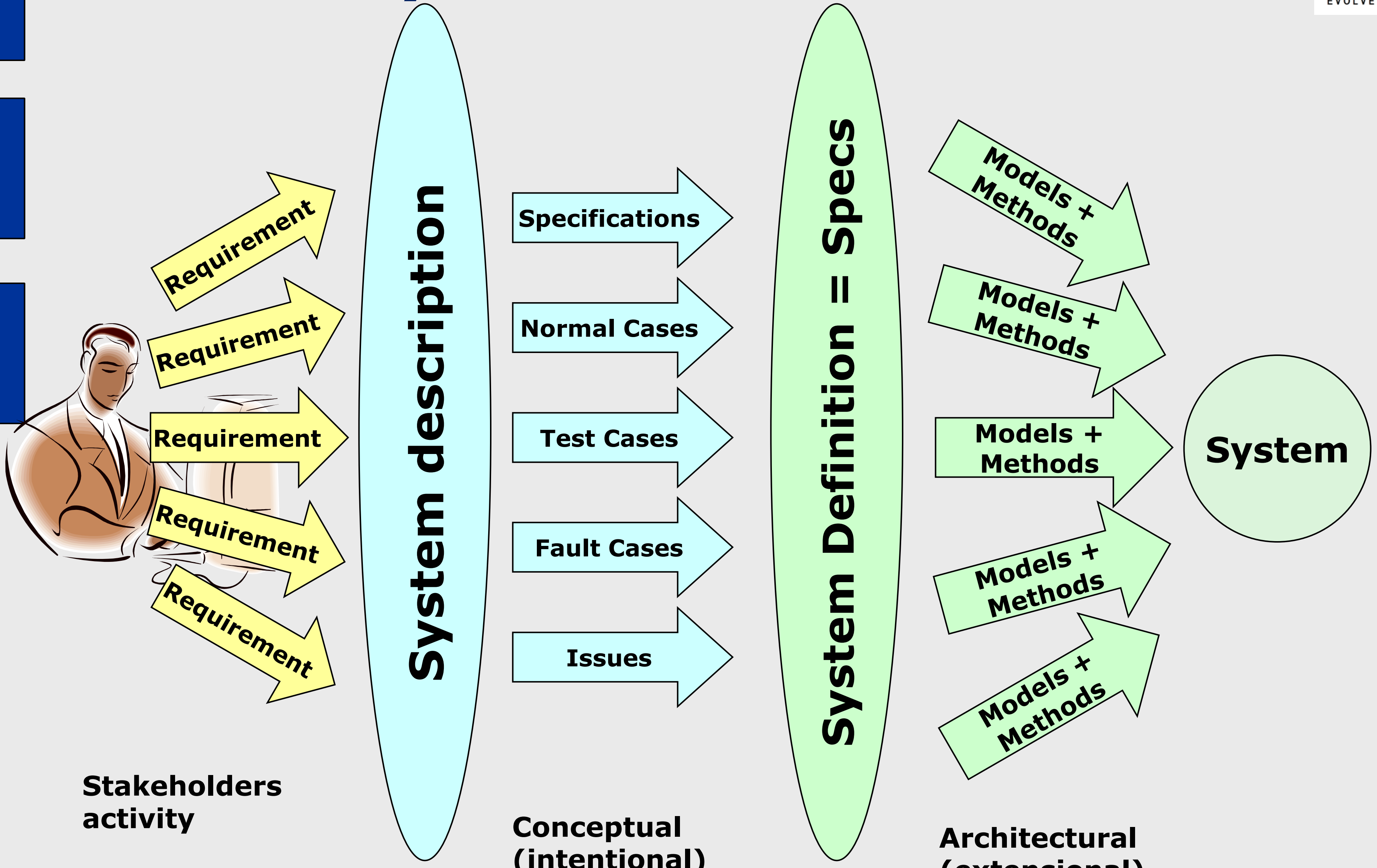
Support for Systems Engineering Process Activities

- Domain can be diverse:
  - technical engineering, organsiation, engineering or business process, …
  - Engineering process will always combine engineering activities with process flow
- Requirements and specifications capturing
- Defining models and methodologies
- Defining architecture of a system in terms of interacting entities
- Defining workplan as set of work packages containing development, verification, test, and validation tasks

OpenCookbook Principles

- Using natural language for requirements and specifications capturing and architecture definitions
- Separation of concerns, concepts hierarchically decomposed and structured
- Unified repository (database) based on the Systems Grammar
- Using unified workflow for whole system engineering process

General System Definition Process

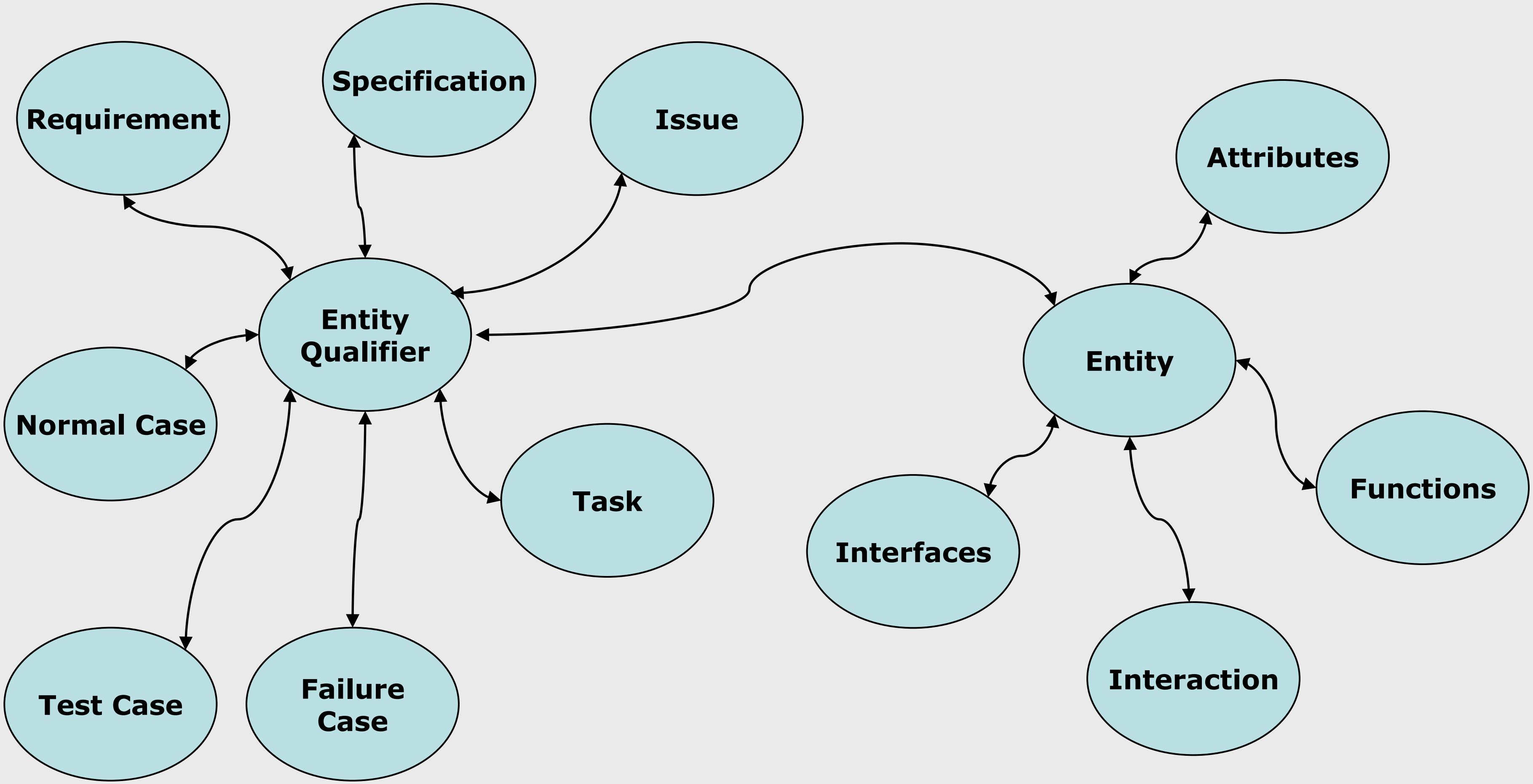


Stakeholders activity

Conceptual (intentional) level

Architectural (extensional) level

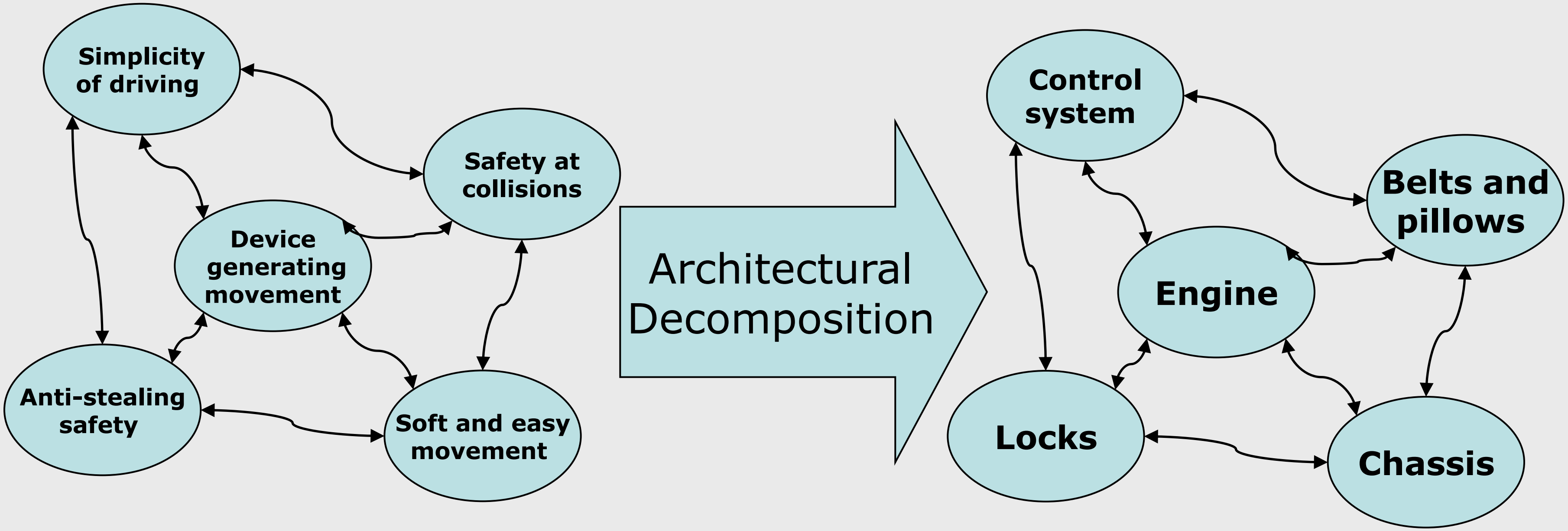
Relationships between conceptual and architectural levels of a system under development (1)



Relationships between conceptual and architectural levels of a system under development (2)

Conceptual Level

Architectural Level



Simplicity of driving

Safety at collisions

Device generating movement

Anti-stealing safety

Soft and easy movement

Architectural Decomposition

Control system

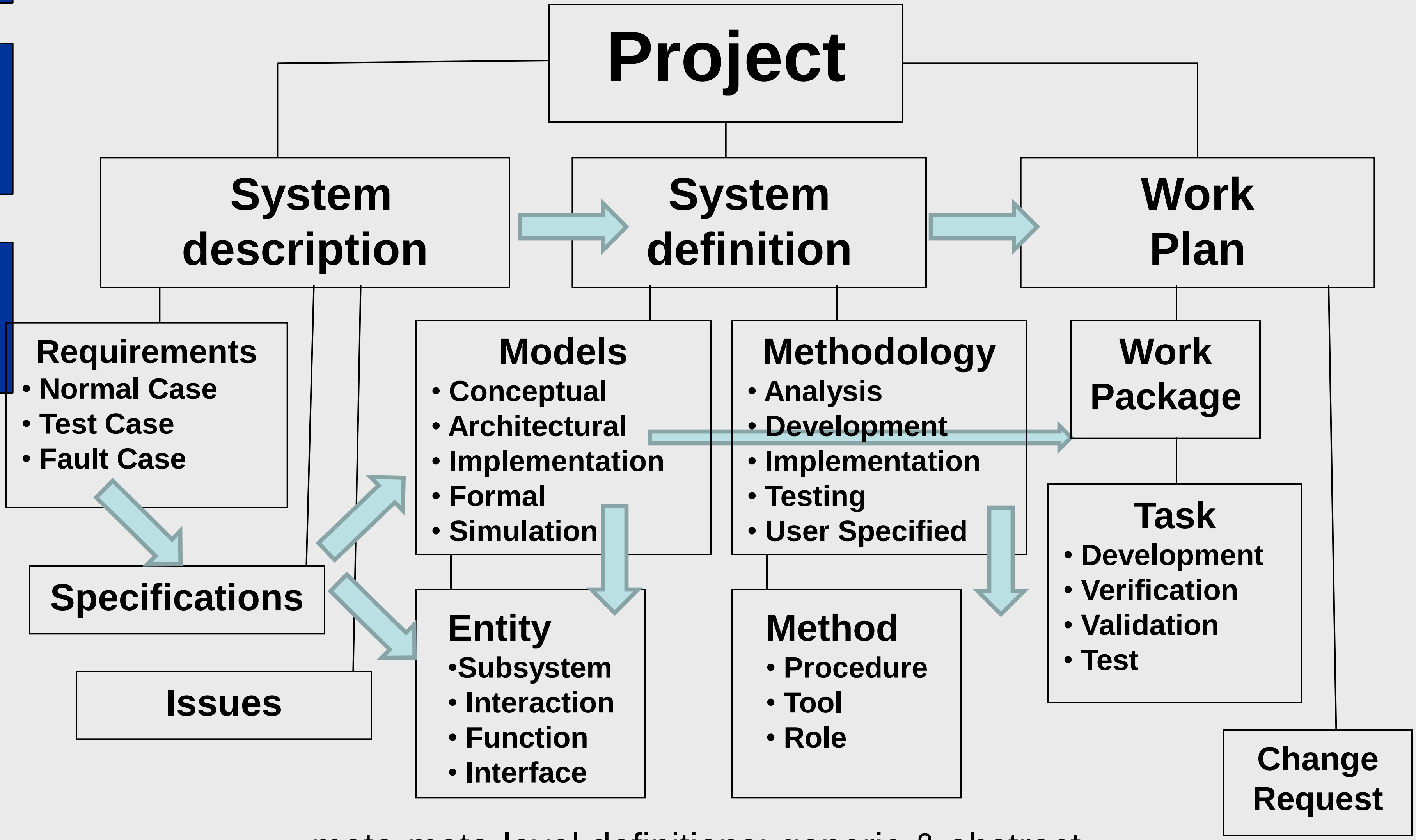
Belts and pillows

Engine

Locks

Chassis

OpenCookBook conceptual schema = project's state space



Project

System description

System definition

Work Plan

Requirements
- Normal Case
- Test Case
- Fault Case

Specifications

Issues

Models
- Conceptual
- Architectural
- Implementation
- Formal
- Simulation

Entity
- Subsystem
- Interaction
- Function
- Interface

Methodology
- Analysis
- Development
- Implementation
- Testing
- User Specified

Method
- Procedure
- Tool
- Role

Work Package

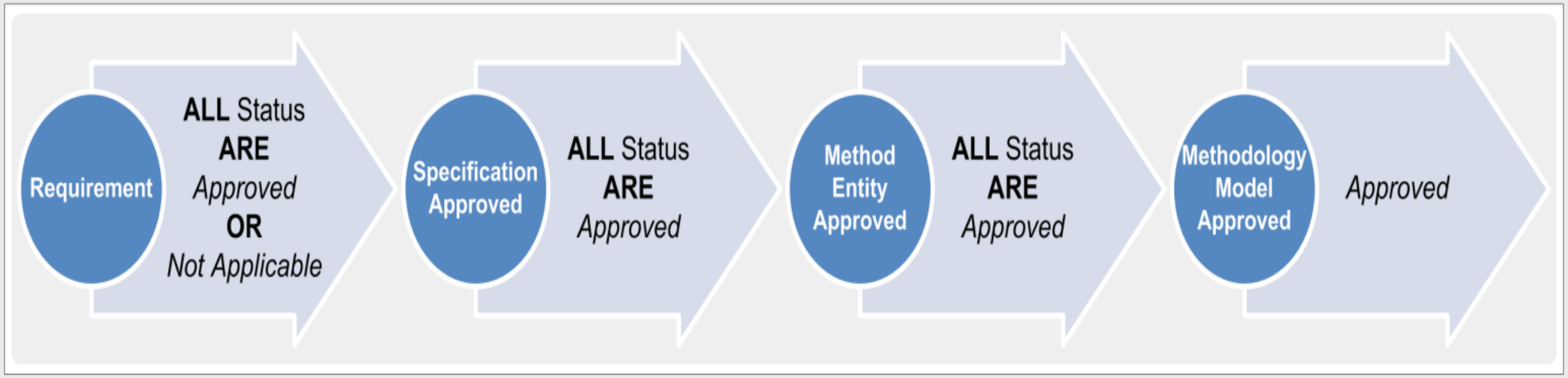
Task
- Development
- Verification
- Validation
- Test

Change Request

meta-meta-level definitions: generic & abstract

Meta-level: domain specific

The state transitions during system definition



Requirement

ALL Status **ARE** *Approved* **OR** *Not Applicable*

Specification Approved

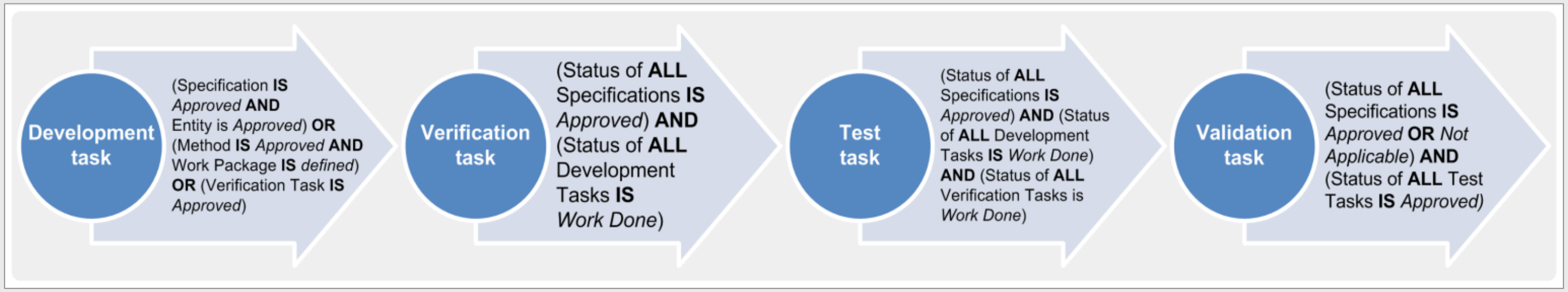
ALL Status **ARE** *Approved*

Method Entity Approved

ALL Status **ARE** *Approved*

Methodology Model Approved

Approved



Development task

(Specification **IS** *Approved* **AND** Entity is *Approved*) **OR** (Method **IS** *Approved* **AND** Work Package **IS** *defined*) **OR** (Verification Task **IS** *Approved*)

Verification task

(Status of **ALL** Specifications **IS** *Approved*) **AND** (Status of **ALL** Development Tasks **IS** *Work Done*)

Test task

(Status of **ALL** Specifications **IS** *Approved*) **AND** (Status of **ALL** Development Tasks **IS** *Work Done*) **AND** (Status of **ALL** Verification Tasks is *Work Done*)

Validation task

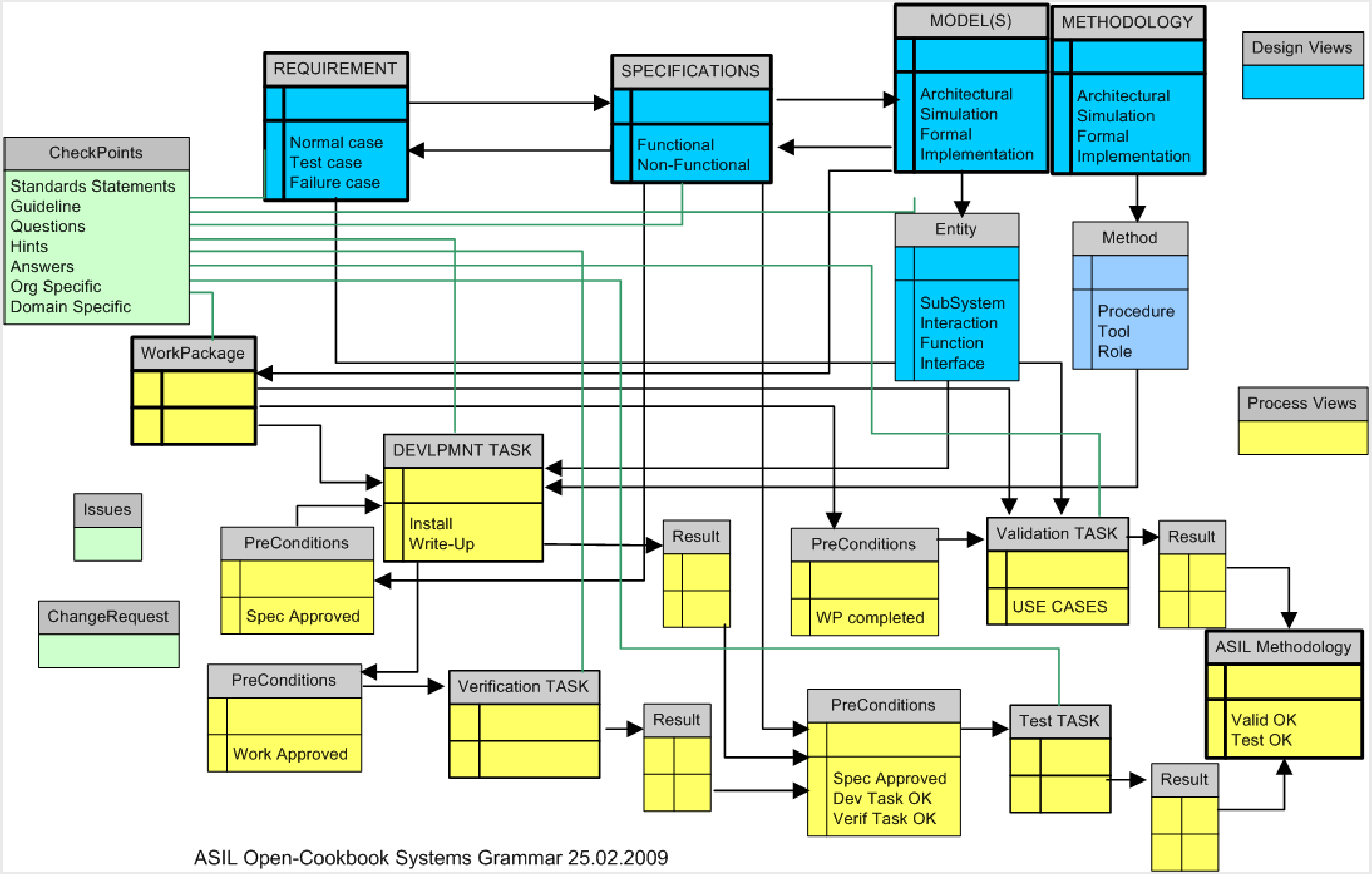
(Status of **ALL** Specifications **IS** *Approved* **OR** *Not Applicable*) **AND** (Status of **ALL** Test Tasks **IS** *Approved)*

$$
\begin{aligned}
&\forall((\mathrm{Re}\,quirement.Status = Approved)\\
&\vee(\mathrm{Re}\,quirement.Status = Not\,applicable))\\
&\rightarrow Specification.Status = Approved
\end{aligned}
$$

Requirements for evolutionary/incremental verification/validation/certification

- Product/system development process builds several dependent **“state-spaces”**
  - Top level is “mission” (top-requirement) for requirements/specifications view
  - Top level is system under development in its environment for architectural view
  - Validation/certification is top level for workplan view
- **Consequences**:
  - Orthogonality requirement to reduce dependencies and localise state-spaces
  - Strict version management
  - Tracebility

Systems grammar = information model



REQUIREMENT
Normal case
Test case
Failure case

SPECIFICATIONS
Functional
Non-Functional

MODEL(S)
Architectural
Simulation
Formal
Implementation

METHODOLOGY
Architectural
Simulation
Formal
Implementation

Design Views

CheckPoints
Standards Statements
Guideline
Questions
Hints
Answers
Org Specific
Domain Specific

Entity
SubSystem
Interaction
Function
Interface

Method
Procedure
Tool
Role

WorkPackage

Process Views

DEVLPMNT TASK
Install
Write-Up

Issues

PreConditions
Spec Approved

Result

PreConditions
WP completed

Validation TASK
USE CASES

Result

ChangeRequest

ASIL Methodology
Valid OK
Test OK

PreConditions
Work Approved

Verification TASK

Result

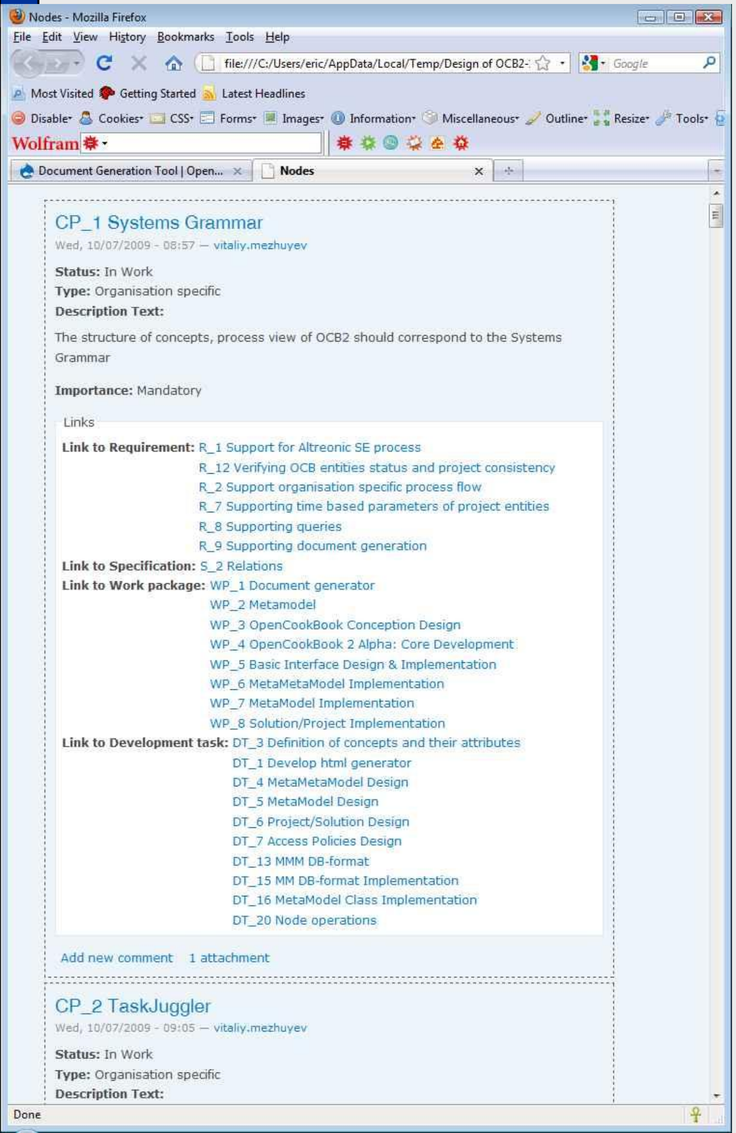
PreConditions
Spec Approved
Dev Task OK
Verif Task OK

Test TASK

Result

ASIL Open-Cookbook Systems Grammar 25.02.2009

OpenCookBook developed as a multi-user web portal



Nodes - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Users/eric/AppData/Local/Temp/Design of OCB2-
Google
Most Visited Getting Started Latest Headlines
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools
Wolfram
Document Generation Tool | Open...
Nodes
Done

CP_1 Systems Grammar

Wed, 10/07/2009 - 08:57 — vitaliy.mezhuyev

Status: In Work
Type: Organisation specific
Description Text:

The structure of concepts, process view of OCB2 should correspond to the Systems Grammar

Importance: Mandatory

Links

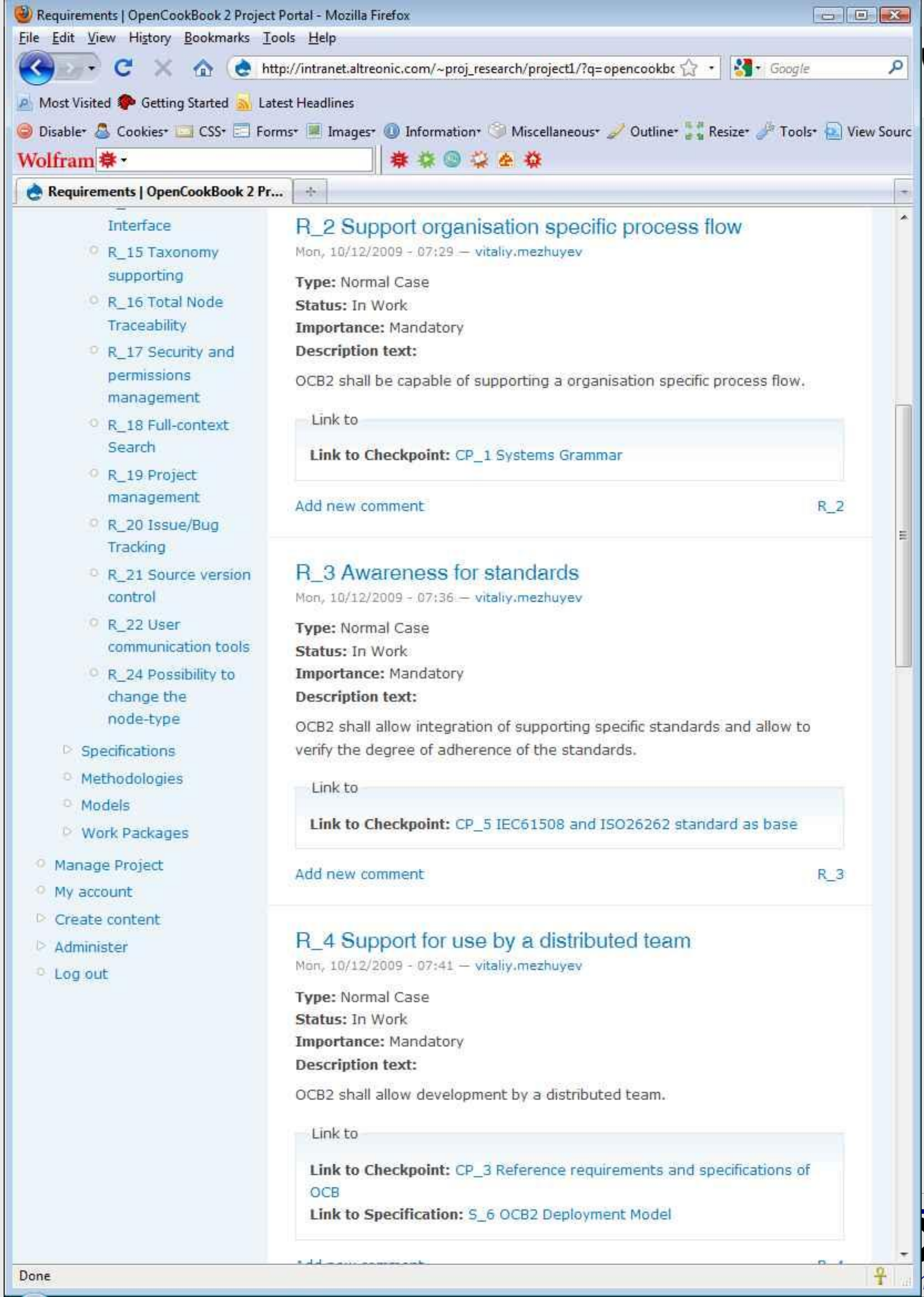
Link to Requirement: R_1 Support for Altreonic SE process
R_12 Verifying OCB entities status and project consistency
R_2 Support organisation specific process flow
R_7 Supporting time based parameters of project entities
R_8 Supporting queries
R_9 Supporting document generation
Link to Specification: S_2 Relations
Link to Work package: WP_1 Document generator
WP_2 Metamodel
WP_3 OpenCookBook Conception Design
WP_4 OpenCookBook 2 Alpha: Core Development
WP_5 Basic Interface Design & Implementation
WP_6 MetaMetaModel Implementation
WP_7 MetaModel Implementation
WP_8 Solution/Project Implementation
Link to Development task: DT_3 Definition of concepts and their attributes
DT_1 Develop html generator
DT_4 MetaMetaModel Design
DT_5 MetaModel Design
DT_6 Project/Solution Design
DT_7 Access Policies Design
DT_13 MMM DB-format
DT_15 MM DB-format Implementation
DT_16 MetaModel Class Implementation
DT_20 Node operations

Add new comment 1 attachment

CP_2 TaskJuggler

Wed, 10/07/2009 - 09:05 — vitaliy.mezhuyev

Status: In Work
Type: Organisation specific
Description Text:



Requirements | OpenCookBook 2 Project Portal - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://intranet.altreonic.com/~proj_research/project1/?q=opencookbc
Google
Most Visited Getting Started Latest Headlines
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Sourc
Wolfram
Requirements | OpenCookBook 2 Pr...
Done

- Interface
- R_15 Taxonomy supporting
- R_16 Total Node Traceability
- R_17 Security and permissions management
- R_18 Full-context Search
- R_19 Project management
- R_20 Issue/Bug Tracking
- R_21 Source version control
- R_22 User communication tools
- R_24 Possibility to change the node-type
- Specifications
- Methodologies
- Models
- Work Packages
- Manage Project
- My account
- Create content
- Administer
- Log out

R_2 Support organisation specific process flow

Mon, 10/12/2009 - 07:29 — vitaliy.mezhuyev

Type: Normal Case
Status: In Work
Importance: Mandatory
Description text:

OCB2 shall be capable of supporting a organisation specific process flow.

Link to

Link to Checkpoint: CP_1 Systems Grammar

Add new comment R_2

R_3 Awareness for standards

Mon, 10/12/2009 - 07:36 — vitaliy.mezhuyev

Type: Normal Case
Status: In Work
Importance: Mandatory
Description text:

OCB2 shall allow integration of supporting specific standards and allow to verify the degree of adherence of the standards.

Link to

Link to Checkpoint: CP_5 IEC61508 and ISO26262 standard as base

Add new comment R_3

R_4 Support for use by a distributed team

Mon, 10/12/2009 - 07:41 — vitaliy.mezhuyev

Type: Normal Case
Status: In Work
Importance: Mandatory
Description text:

OCB2 shall allow development by a distributed team.

Link to

Link to Checkpoint: CP_3 Reference requirements and specifications of OCB
Link to Specification: S_6 OCB2 Deployment Model

OpenCookbook functionality

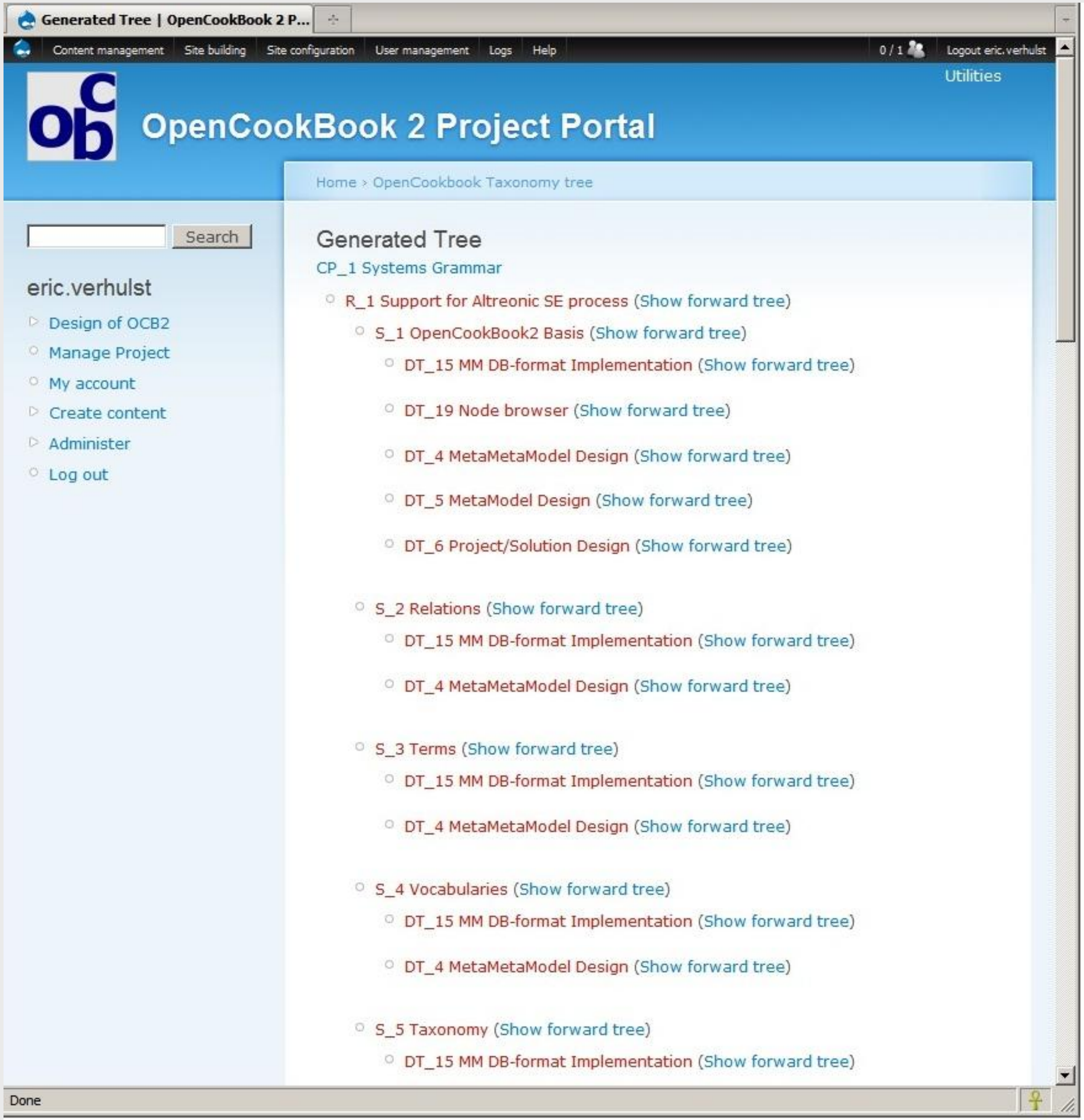
- System definition through the web
- Possibility of work in local mode on PC
- Organisation of discussion on system requirements, specifications, architecture and work plan
- Queries to project database
- Intuitive interface and easy navigation, using WYSIWYG web-based editors

OpenCookbook functionality

- Generation of project documentation (in html)
- Generation of Task Juggler reports
- Import/export project database
- Implementation of mapping between project levels by hyperlinks.

Dependency tree

- From checkpoint to release, dependency tree can be displayed and navigated
- => first step towards "delta-management" for incremental verification/ validation/ certification



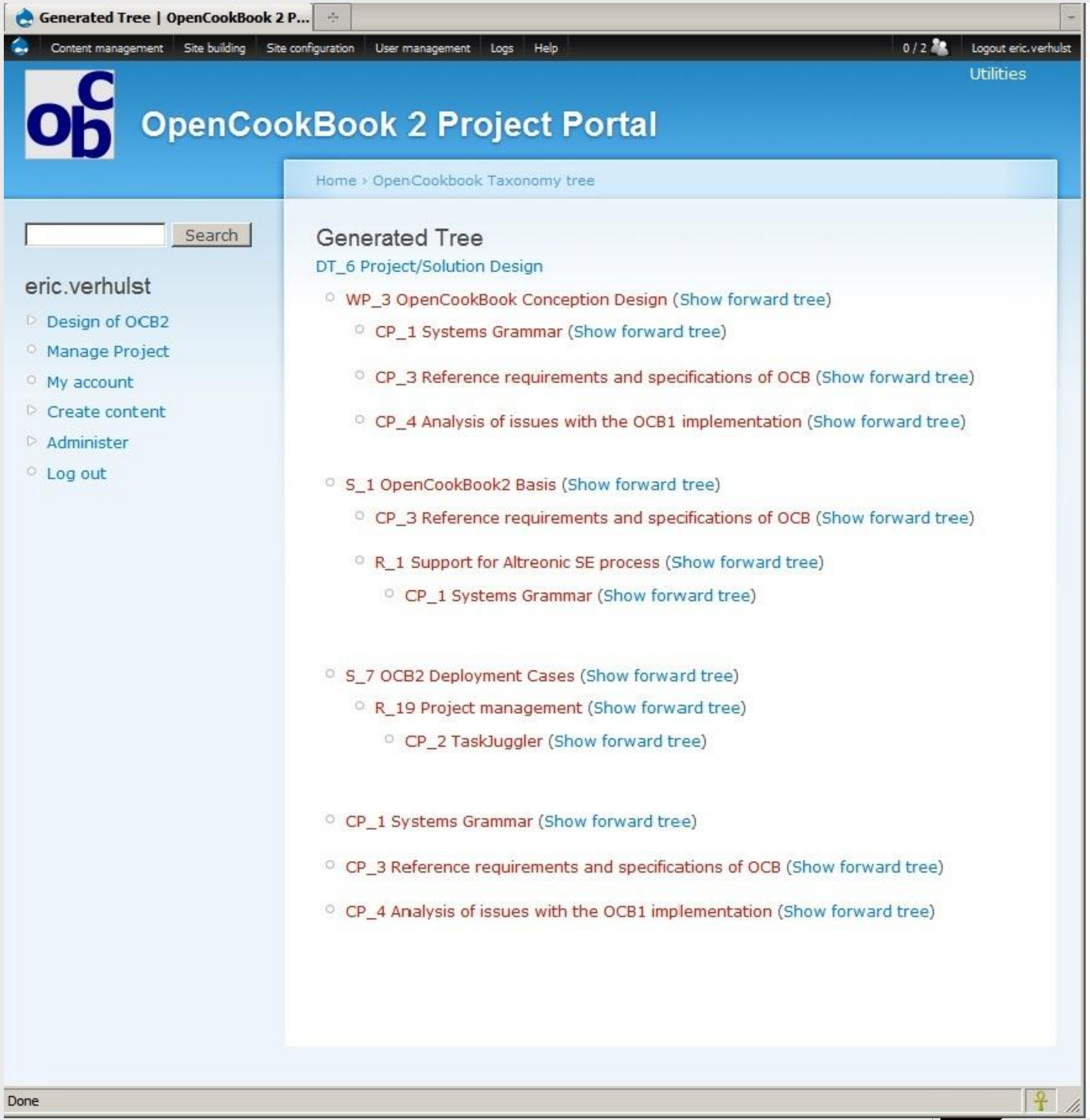
Generated Tree

CP_1 Systems Grammar

- R_1 Support for Altreonic SE process (Show forward tree)
  - S_1 OpenCookBook2 Basis (Show forward tree)
    - DT_15 MM DB-format Implementation (Show forward tree)
    - DT_19 Node browser (Show forward tree)
    - DT_4 MetaMetaModel Design (Show forward tree)
    - DT_5 MetaModel Design (Show forward tree)
    - DT_6 Project/Solution Design (Show forward tree)
  - S_2 Relations (Show forward tree)
    - DT_15 MM DB-format Implementation (Show forward tree)
    - DT_4 MetaMetaModel Design (Show forward tree)
  - S_3 Terms (Show forward tree)
    - DT_15 MM DB-format Implementation (Show forward tree)
    - DT_4 MetaMetaModel Design (Show forward tree)
  - S_4 Vocabularies (Show forward tree)
    - DT_15 MM DB-format Implementation (Show forward tree)
    - DT_4 MetaMetaModel Design (Show forward tree)
  - S_5 Taxonomy (Show forward tree)
    - DT_15 MM DB-format Implementation (Show forward tree)

Precedence tree

- From release or validation task to requirement, precedence tree can be displayed and navigated
- => first step towards “delta-management” for incremental verifcation/validation/ certification



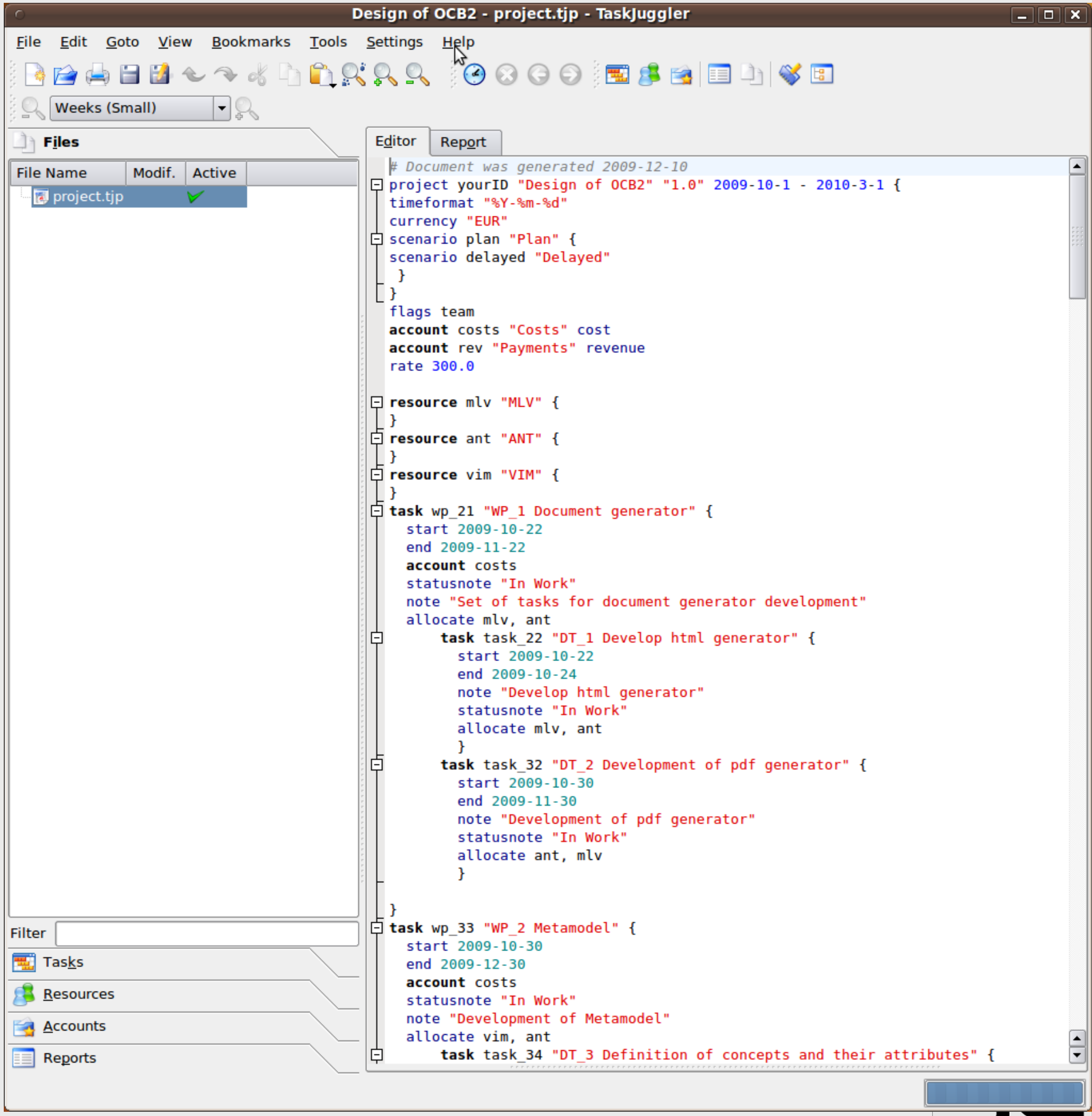
Generated Tree

DT_6 Project/Solution Design

- WP_3 OpenCookBook Conception Design (Show forward tree)
  - CP_1 Systems Grammar (Show forward tree)
  - CP_3 Reference requirements and specifications of OCB (Show forward tree)
  - CP_4 Analysis of issues with the OCB1 implementation (Show forward tree)
- S_1 OpenCookBook2 Basis (Show forward tree)
  - CP_3 Reference requirements and specifications of OCB (Show forward tree)
  - R_1 Support for Altreonic SE process (Show forward tree)
    - CP_1 Systems Grammar (Show forward tree)
- S_7 OCB2 Deployment Cases (Show forward tree)
  - R_19 Project management (Show forward tree)
    - CP_2 TaskJuggler (Show forward tree)
- CP_1 Systems Grammar (Show forward tree)
- CP_3 Reference requirements and specifications of OCB (Show forward tree)
- CP_4 Analysis of issues with the OCB1 implementation (Show forward tree)

Export to Task Juggler

- For all tasks in WPs, task project management parameters are exported to Project Management tool (Task Juggler)

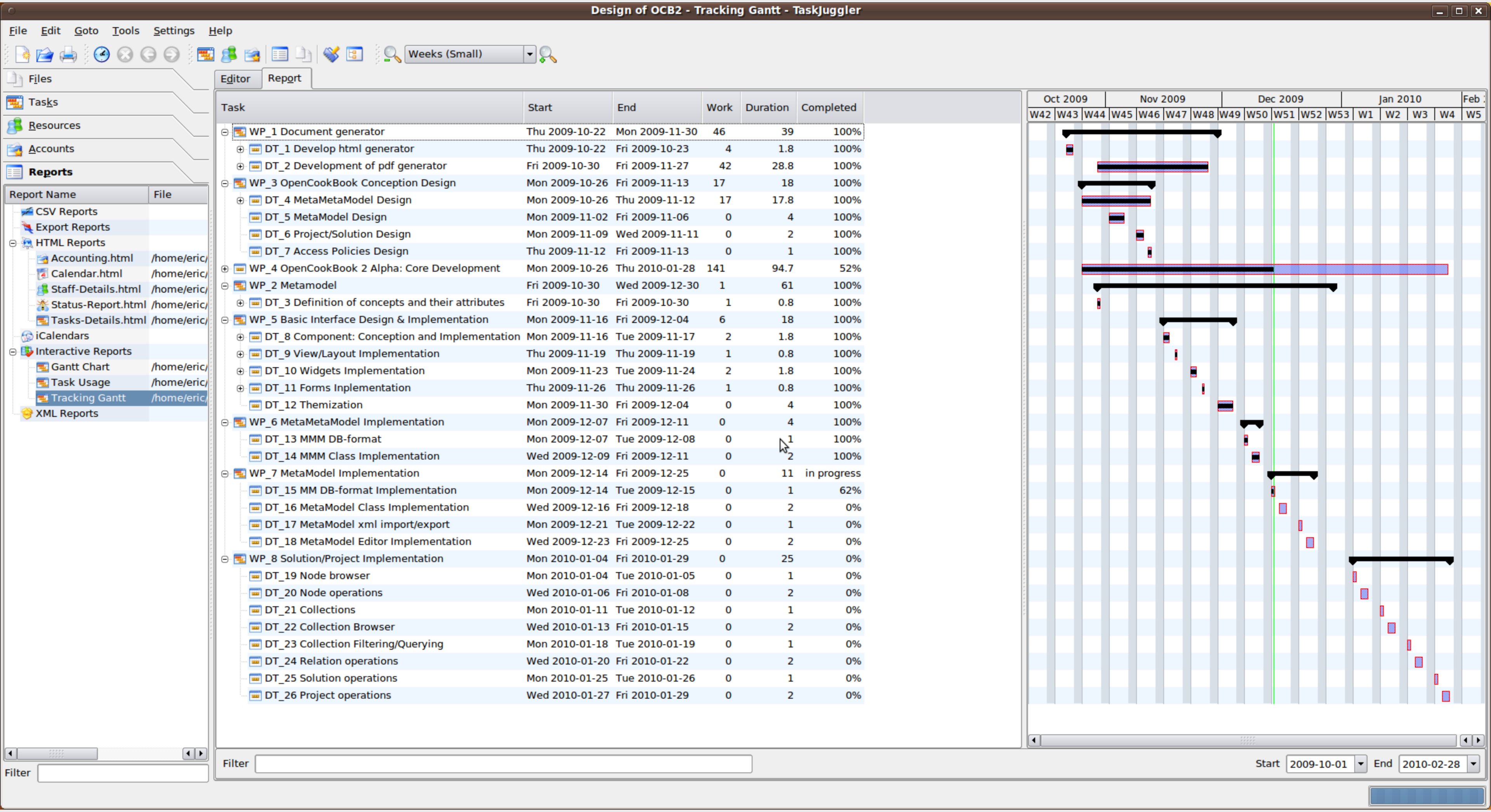


```
# Document was generated 2009-12-10
project yourID "Design of OCB2" "1.0" 2009-10-1 - 2010-3-1 {
timeformat "%Y-%m-%d"
currency "EUR"
scenario plan "Plan" {
scenario delayed "Delayed"
 }
}
flags team
account costs "Costs" cost
account rev "Payments" revenue
rate 300.0

resource mlv "MLV" {
}
resource ant "ANT" {
}
resource vim "VIM" {
}
task wp_21 "WP_1 Document generator" {
  start 2009-10-22
  end 2009-11-22
  account costs
  statusnote "In Work"
  note "Set of tasks for document generator development"
  allocate mlv, ant
      task task_22 "DT_1 Develop html generator" {
        start 2009-10-22
        end 2009-10-24
        note "Develop html generator"
        statusnote "In Work"
        allocate mlv, ant
        }
      task task_32 "DT_2 Development of pdf generator" {
        start 2009-10-30
        end 2009-11-30
        note "Development of pdf generator"
        statusnote "In Work"
        allocate ant, mlv
        }

}
task wp_33 "WP_2 Metamodel" {
  start 2009-10-30
  end 2009-12-30
  account costs
  statusnote "In Work"
  note "Development of Metamodel"
  allocate vim, ant
      task task_34 "DT_3 Definition of concepts and their attributes" {
```

Gant chart, generated from Task entries



Report Name	File
CSV Reports	
Export Reports	
HTML Reports	
Accounting.html	/home/eric/
Calendar.html	/home/eric/
Staff-Details.html	/home/eric/
Status-Report.html	/home/eric/
Tasks-Details.html	/home/eric/
iCalendars	
Interactive Reports	
Gantt Chart	/home/eric/
Task Usage	/home/eric/
Tracking Gantt	/home/eric/
XML Reports	

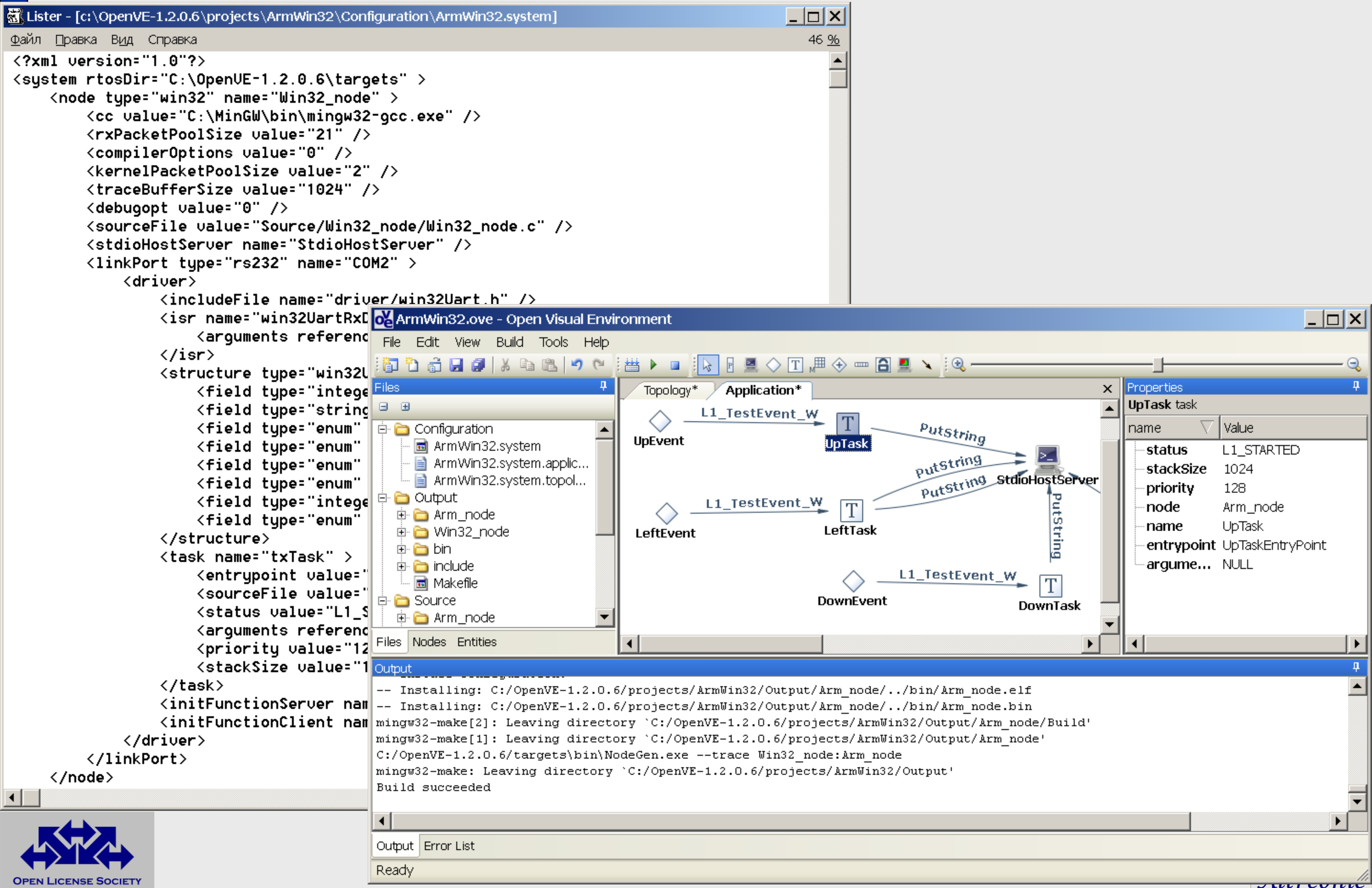
Task	Start	End	Work	Duration	Completed
WP_1 Document generator	Thu 2009-10-22	Mon 2009-11-30	46	39	100%
DT_1 Develop html generator	Thu 2009-10-22	Fri 2009-10-23	4	1.8	100%
DT_2 Development of pdf generator	Fri 2009-10-30	Fri 2009-11-27	42	28.8	100%
WP_3 OpenCookBook Conception Design	Mon 2009-10-26	Fri 2009-11-13	17	18	100%
DT_4 MetaMetaModel Design	Mon 2009-10-26	Thu 2009-11-12	17	17.8	100%
DT_5 MetaModel Design	Mon 2009-11-02	Fri 2009-11-06	0	4	100%
DT_6 Project/Solution Design	Mon 2009-11-09	Wed 2009-11-11	0	2	100%
DT_7 Access Policies Design	Thu 2009-11-12	Fri 2009-11-13	0	1	100%
WP_4 OpenCookBook 2 Alpha: Core Development	Mon 2009-10-26	Thu 2010-01-28	141	94.7	52%
WP_2 Metamodel	Fri 2009-10-30	Wed 2009-12-30	1	61	100%
DT_3 Definition of concepts and their attributes	Fri 2009-10-30	Fri 2009-10-30	1	0.8	100%
WP_5 Basic Interface Design & Implementation	Mon 2009-11-16	Fri 2009-12-04	6	18	100%
DT_8 Component: Conception and Implementation	Mon 2009-11-16	Tue 2009-11-17	2	1.8	100%
DT_9 View/Layout Implementation	Thu 2009-11-19	Thu 2009-11-19	1	0.8	100%
DT_10 Widgets Implementation	Mon 2009-11-23	Tue 2009-11-24	2	1.8	100%
DT_11 Forms Inplementation	Thu 2009-11-26	Thu 2009-11-26	1	0.8	100%
DT_12 Themization	Mon 2009-11-30	Fri 2009-12-04	0	4	100%
WP_6 MetaMetaModel Implementation	Mon 2009-12-07	Fri 2009-12-11	0	4	100%
DT_13 MMM DB-format	Mon 2009-12-07	Tue 2009-12-08	0	1	100%
DT_14 MMM Class Implementation	Wed 2009-12-09	Fri 2009-12-11	0	2	100%
WP_7 MetaModel Implementation	Mon 2009-12-14	Fri 2009-12-25	0	11	in progress
DT_15 MM DB-format Implementation	Mon 2009-12-14	Tue 2009-12-15	0	1	62%
DT_16 MetaModel Class Implementation	Wed 2009-12-16	Fri 2009-12-18	0	2	0%
DT_17 MetaModel xml import/export	Mon 2009-12-21	Tue 2009-12-22	0	1	0%
DT_18 MetaModel Editor Implementation	Wed 2009-12-23	Fri 2009-12-25	0	2	0%
WP_8 Solution/Project Implementation	Mon 2010-01-04	Fri 2010-01-29	0	25	0%
DT_19 Node browser	Mon 2010-01-04	Tue 2010-01-05	0	1	0%
DT_20 Node operations	Wed 2010-01-06	Fri 2010-01-08	0	2	0%
DT_21 Collections	Mon 2010-01-11	Tue 2010-01-12	0	1	0%
DT_22 Collection Browser	Wed 2010-01-13	Fri 2010-01-15	0	2	0%
DT_23 Collection Filtering/Querying	Mon 2010-01-18	Tue 2010-01-19	0	1	0%
DT_24 Relation operations	Wed 2010-01-20	Fri 2010-01-22	0	2	0%
DT_25 Solution operations	Mon 2010-01-25	Tue 2010-01-26	0	1	0%
DT_26 Project operations	Wed 2010-01-27	Fri 2010-01-29	0	2	0%

Start 2009-10-01 End 2010-02-28

Integration with real-time embedded frameworks

- Integration with OpenVE RT-modelling environment
  - – Software entities => OpenComRTOS tasks and SW functions
  - – Interactions=> OpenComRTOS hubs and comm protocols
  - – RT attributes (e.g. UML Marte, SMART, …)
- Attributes and state transition conditions for project scheduling and management
- Attributes and transition conditions to support certification processes
- Metamodel supporting organisation specific flows

Integration with OpenVE using metamodel (xml)


Lister - [c:\OpenVE-1.2.0.6\projects\ArmWin32\Configuration\ArmWin32.system]
Файл Правка Вид Справка
46 %

```
<?xml version="1.0"?>
<system rtosDir="C:\OpenVE-1.2.0.6\targets" >
    <node type="win32" name="Win32_node" >
        <cc value="C:\MinGW\bin\mingw32-gcc.exe" />
        <rxPacketPoolSize value="21" />
        <compilerOptions value="0" />
        <kernelPacketPoolSize value="2" />
        <traceBufferSize value="1024" />
        <debugopt value="0" />
        <sourceFile value="Source/Win32_node/Win32_node.c" />
        <stdioHostServer name="StdioHostServer" />
        <linkPort type="rs232" name="COM2" >
            <driver>
                <includeFile name="driver/win32Uart.h" />
                <isr name="win32UartRxD
                    <arguments referenc
                </isr>
                <structure type="win32U
                    <field type="intege
                    <field type="string
                    <field type="enum"
                    <field type="enum"
                    <field type="enum"
                    <field type="enum"
                    <field type="intege
                    <field type="enum"
                </structure>
                <task name="txTask" >
                    <entrypoint value="
                    <sourceFile value="
                    <status value="L1_S
                    <arguments referenc
                    <priority value="12
                    <stackSize value="1
                </task>
                <initFunctionServer nam
                <initFunctionClient nam
            </driver>
        </linkPort>
    </node>
```

name	Value
status	L1_STARTED
stackSize	1024
priority	128
node	Arm_node
name	UpTask
entrypoint	UpTaskEntryPoint
argume...	NULL

```
-- Installing: C:/OpenVE-1.2.0.6/projects/ArmWin32/Output/Arm_node/../bin/Arm_node.elf
-- Installing: C:/OpenVE-1.2.0.6/projects/ArmWin32/Output/Arm_node/../bin/Arm_node.bin
mingw32-make[2]: Leaving directory `C:/OpenVE-1.2.0.6/projects/ArmWin32/Output/Arm_node/Build'
mingw32-make[1]: Leaving directory `C:/OpenVE-1.2.0.6/projects/ArmWin32/Output/Arm_node'
C:/OpenVE-1.2.0.6/targets\bin\NodeGen.exe --trace Win32_node:Arm_node
mingw32-make: Leaving directory `C:/OpenVE-1.2.0.6/projects/ArmWin32/Output'
Build succeeded
```

Technical info

- OpenSpecs
  - Based on Drupal 5.21 Content Management System
  - Web server (tested with Apache v. 2.2)
  - PHP (tested with 5.3)
  - MySQL (tested with 5.0)
  - scalability and maintainability issues
- OpenCookBook v1
  - See OpenSpecs
- OpenCookBook v2
  - Wt: compiled web portal in C++
  - Enhanced metamodel

Conclusion

- Systems engineering process can be formalised using common metamodel
- Challenges
  - – Integration of different domains
    - Process, architectural, certification
    - System Engineering processes (“standards”) are heuristic standards
  - – Human interface design: must be intuitive
  - – Formal(ised) analysis of requirements
- Progress through formalisation
  - – Reduction of design space give reliability

Conclusion

- More info:

www.altreonic.com

Eric.Verhulst@altreonic.com

OpenCookBook1 freely downloadable

Panel

- In search for hardware that executes specifications efficiently
- Correlate:
  - In seach for software that executes requirements efficiently

Panel

- Project is “walking the tree” in project’s statespace
  - – Requirements -> specifications -> model -> implementation in SW and HW
  - – Final model is implementation (model)
  - – The larger the statespace the more error prone, more difficult to verify and validate
  - – Less is also less for power and cost!

Panel

- Changing / increasing requirements
  - Before: only “normal” case: easy (sic)
  - Then: also “test case” (intrusive)
  - Now also: “fault case” => safety & security!
    - Decomposition in entities and interactions
      - (concurrency and communication)
    - Error trapping
    - Fault containment
    - Fault recovery
    - Resource metering (time, memory, bandwidth, power)
    - => additional complexity and system behaviour!

Panel

- But:
  - – We program mostly with sequential programming languages as abstraction layers on top of sequential von Neuman CPUs
  - – Software doesn’t execute hardware!
  - – Software must be efficient in translating requirements in specifications
- Hence:
  - – Hardware must be efficient to execute specifications!

Challenges in Testing and Validating Complex Systems

Keith Stobie

Microsoft

Validating approximately:

Decision Systems & Loose Consistency

Decision Systems

- systems created via Machine learning
  - Rule based
  - Neural Networks
  - Decision Trees
- How to test approximations?

  Heuristic Oracles for what is clearly wrong.

Large Scale Distributed Systems

- Asynchronous, loosely coupled
- NoSQL architectures provide weak consistency guarantees such as eventual consistency
- database terminology, BASE (**B**asically **A**vailable, **S**oft state, **E**ventual consistency)
- PAXOS (consistency) - Liveness(C;L)
  - If value C has been proposed, then ***eventually*** learner L will learn some value (if sufficient processors remain non-faulty).

BASE

- Weak consistency
  - – stale data OK
- Availability first
- Best effort
- Approximate answers OK
- Aggressive (optimistic)
- Simpler!, Faster, Easier evolution

Brewer, Eric. Towards Robust Distributed Systems, PODC Keynote, July 19, 2000

Testing Eventual Consistency?

- Heuristic Oracles for answers that are
  - too approximate
  - too inconsistent for too long

Composition of services for an effective measurement process

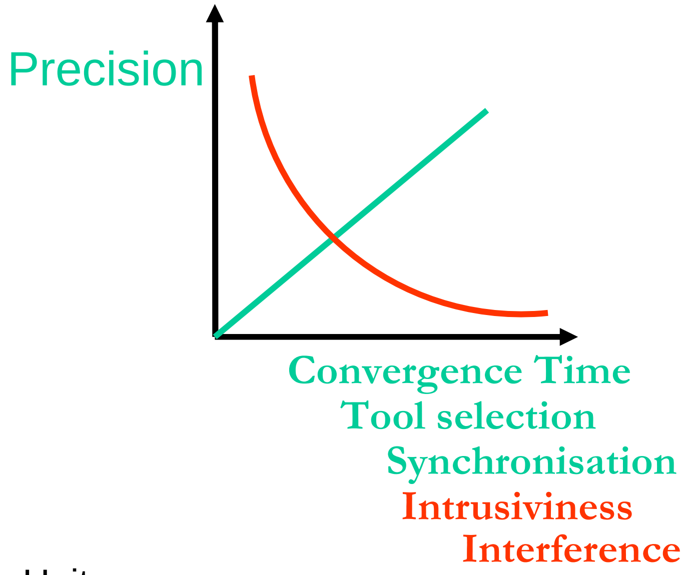
Maurizio D’Arienzo
Dipartimento di Studi Europei e Mediterranei
Seconda Università degli Studi di Napoli

Monitoring of complex systems

- Testing and validation through measurement process is not a straightforward task
- The use of a specific mechanism (familiar tool) may be inaccurate under certain conditions
- Some contraints on precision:

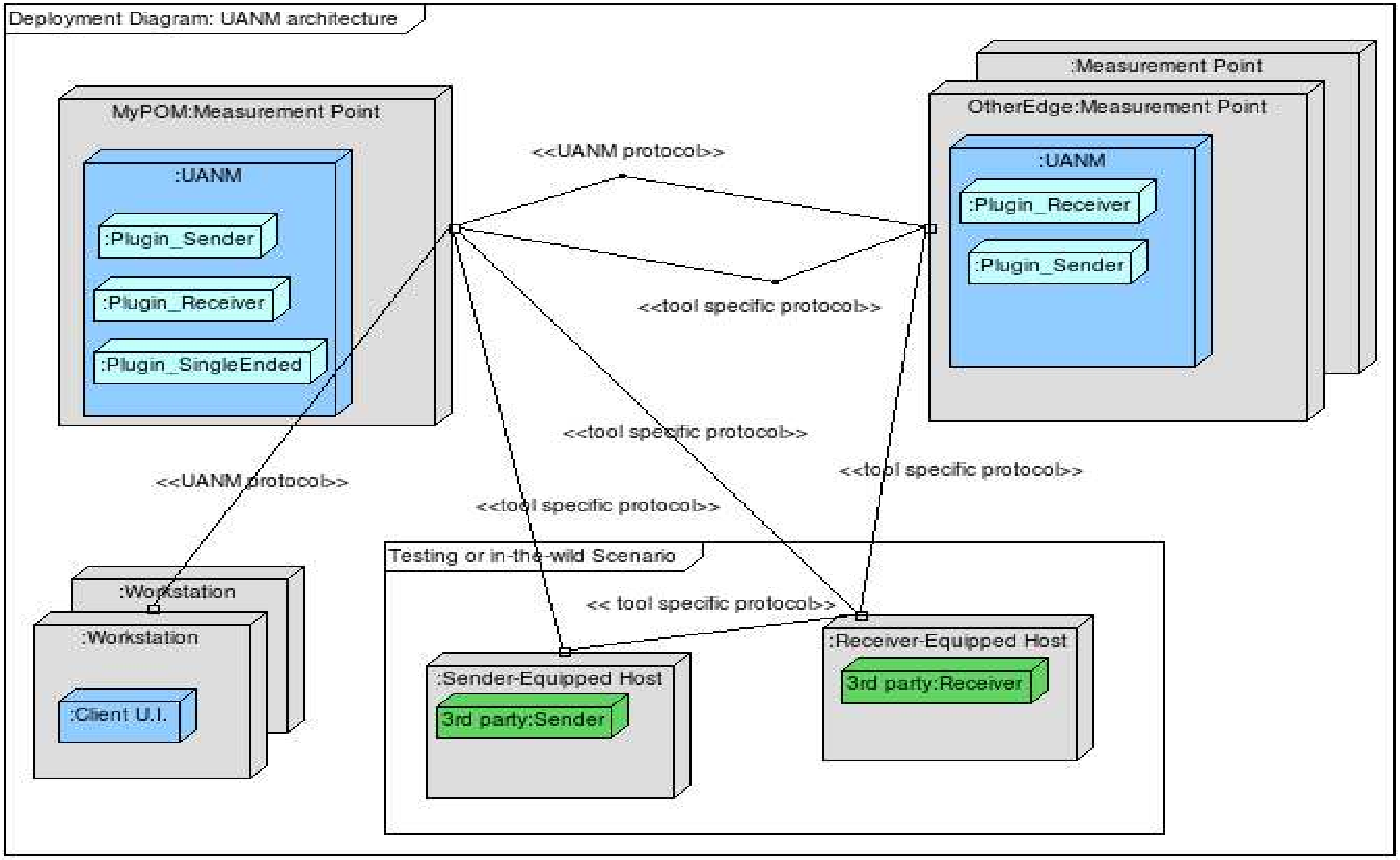
user
  - Convergence time
  - Tool selection

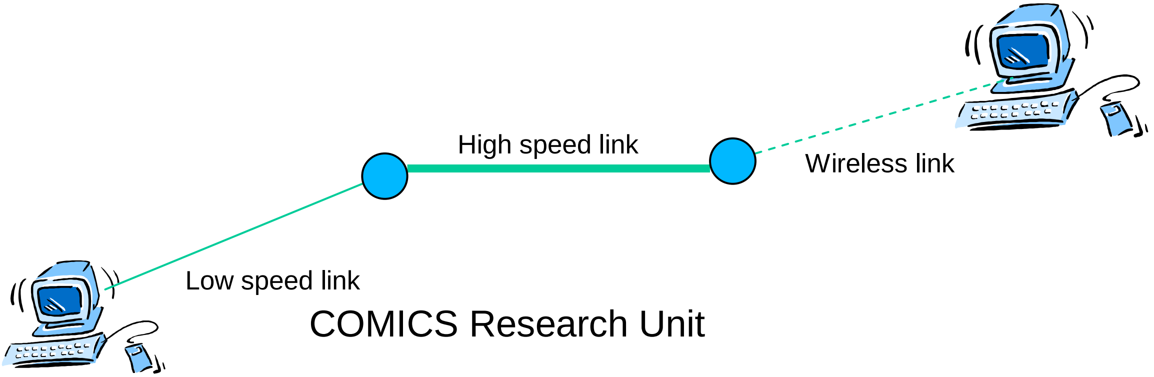
system
  - Synchronisation
  - Intrusiviness
  - Interference



Composition of services

- Static, independent tools are outdated, dynamic, interoperating tools are sound.
- Design of novel monitoring systems:
  - Composition of different measurement techniques in a fair environment
  - full compliance and open interaction with existing tools
  - mutual exclusion of concurrent measurements
  - automatic tool selection and their configuration





COMICS Research Unit

Many Ways to Automate Development of Automated Tests

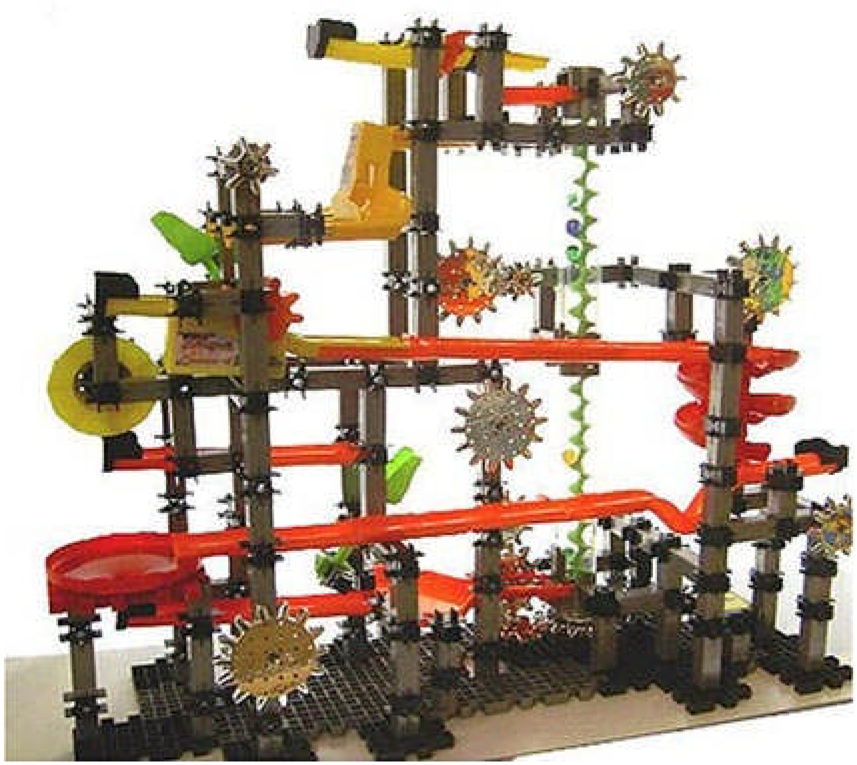
Vladimir Rubanov, Ph.D.

vrub@ispras.ru

Head of Operating Systems Department at the Institute for System Programming of the Russian Academy of Sciences (ISPRAS)

Director of Russian Linux Verification Center (linuxtesting.org)

[Usually] Test Execution Should Be Automated



How To Create Automated SW Tests?

- Manual Development:

Plain programming language

Test development frameworks

- Automatic Generation:

Based on “nothing”

Based on thorough models

Dimensions of Automated Test Suite Quality

1. “**Wideness**”: how many target functions/blocks are tested at all?
   – the scope of testing suite.
2. “**Deepness**”: how many (and how smart) various input combinations in various internal states for particular function/block are iterated?
   – the quality of test actions / sequences.
3. “**Checking Thoroughness**”: how well is the correctness of the SUT responses checked?
   – the quality of test oracles.

There Are Different Technologies Available That Help Automate Development of Automated Tests

- Sometimes you need to use **a combination of the technologies**.
- It is good when you can “adjust” the test suite quality by the mentioned dimensions **independently**.
- Need to take into account:
  - ▫ Resources / cost / time to develop tests
  - ▫ Importance of particular SUT functions/blocks

Example: Testing Linux For Conformance with Linux Standard Base (LSB) Specification

1. LSB defines requirements for more than **30,000 APIs** in more than **50 system libraries**.
2. It is impossible to create good automated tests for all of these in reasonable time/resources.
3. We have classified all APIs by **3 categories of importance**.
4. A combination of **3 different test development technologies** have been used for creating the necessary tests.

Example: LSB Conformance Test Suite (1)

Low Importance APIs:

1. Generate shallow tests fully automatically based on "nothing".
   - High "wideness", low "deepness", low "check thoroughness".
   - Very low cost per API.
2. Further improvements as resources appear:
   - Add additional info for more advanced test generation – "nothing" converts to "little hints" – increasing "deepness" & "check thoroughness".
   - Manual test cases' tweaks using "normal" test development framework.

Example: LSB Conformance Test Suite (2)

Medium Importance APIs:

1. Step 1: Generate test templates automatically.
2. Step 2: Manually develop “normal” unit tests based on the templates using a unit test development framework:
   - Moderate “deepness”
   - Moderate-high “check thoroughness”.
   - Moderate cost per API.

Example: LSB Conformance Test Suite (3)

High Importance APIs:

1. Create a model-driven test suite based on formal specifications of the APIs.
    - High “deepness” & high “check thoroughness”, which can be independently adjusted (including dynamically configured from some minimal to maximum for different test runs).
    - High cost per API.

ISPRAS Linux Verification Center

- Founded in **2005**
- A division of **ISPRAS**
- Working closely with **Linux Foundation** (formerly **FSG**), **Intel**, **Motorola**, local companies.
- ~**30** engineers