tNi Europe

“State of the art” Software Modeling

Tony Elliston

SIGADA 2004
Atlanta

tNi Europe

TNI Europe Limited

- Market our own software modelling tools:
  - CP-Hood and Stood.
- Distributor for TNI Software range of products.

TNI Europe

2000 — Created near Manchester (UK)

2001 — Acquisition of CP-HOOD from Critical Path

2004 — Acquisition of Stood from TNI-Valiosys Office in Brest (F)

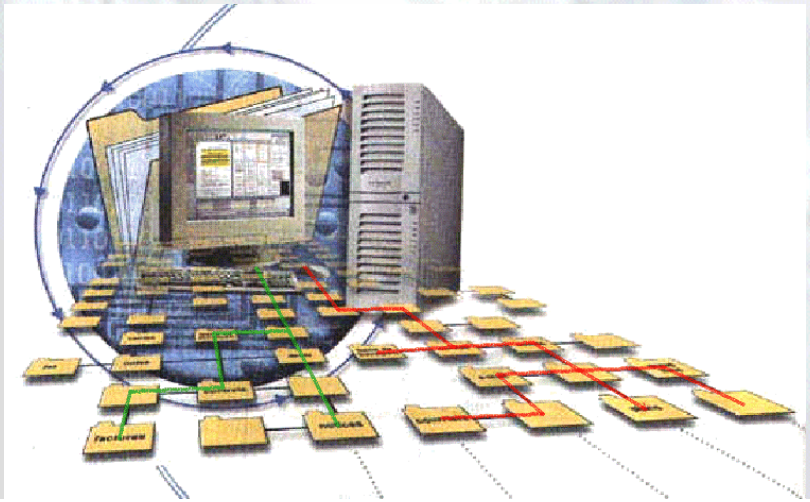
Release of Stood 5.0

tNi Europe

tNi SOFTWARE

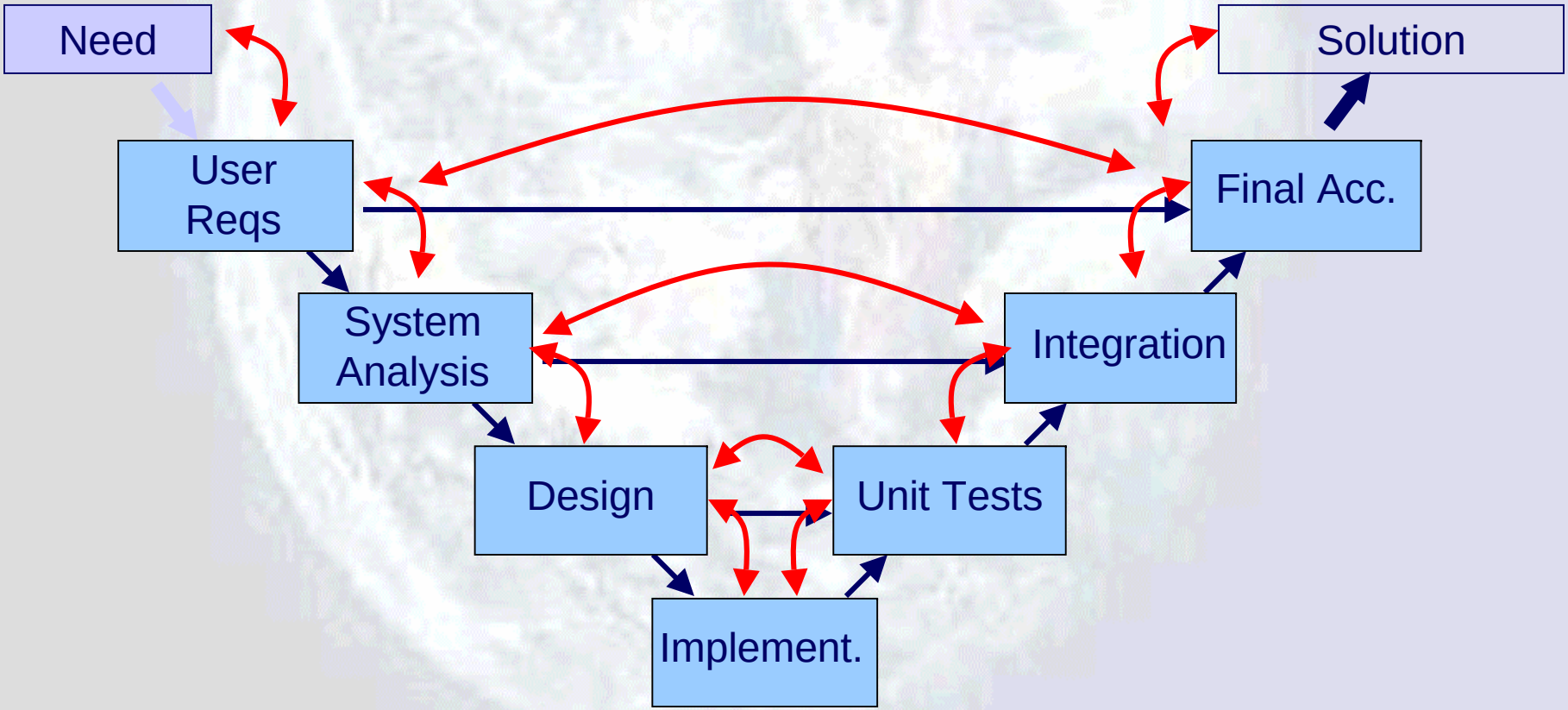
Reqtify

A light and powerful solution for requirements traceability



Requirements Traceability

For a given process, evolutions and modifications can be necessary at each step, and the impact must be analysed before decision :



Easy to integrate

A non-intrusive approach :

No modification of your development and configuration management process.

Traceability during the whole process
(text tools, analysis and modelling tools, code,…)

Qualified DO178-B as a verification tool for A380, complies with D0254 and other standards.

Simple user interface allowing powerful navigation in the traceability graph

Reqtify can even be used on projects already started !

Immediate ROI

A minimal investment :

Easy to handle, very short training course,

No need for database administration,

A Floating licence

Windows/UNIX interoperability

A small investment in Reqtify and training can provide a truly extraordinary payback even on the first project.

Documentation generation

Documents generated :

- Traceability matrix,
- Upstream and downstream impact analysis,
- Project description,
- Synthesis of added information,
- User defined templates...

Generated formats :

- RTF (Word)
- PDF
- HTML
- LaTeX
- TPS (InterLeaf)
- MIF (FrameMaker)
- ASCII
- Text only

Reqtify coupling capabilities

- **Office tools**
  Word, Excel, Access, Powerpoint, PDF,Text, Framemaker(Win & UNIX), Interleaf, Quicksilver,
  MS Project
- **UML tools**
  Rhapsody, Rose, Objecteering.
- **Modeling tools**
  Stood, CP-Hood, Simulink, Statemate, Scade, RTBuilder, System Architect, Matlab.
- **Code files**
  C, Ada, SDL, VHDL, Verilog, Matlab (.m) files, Test Script, Test log, all ASCII files.
- **Configuration Management**
  Clearcase, CVS, PVCS.
- **Hardware design tools**
  VisualElite, VNCover.
- **Requirements Management Tools**
  Doors, Requisite Pro.

New tools are easy to integrate

Who uses Reqtify ?

- **AIRBUS** for A340 and A380 software and avionics Corporate agreement
- **THALES** across a number of divisions and projects, both in France and the UK Corporate agreement
- **MBDA** for missile software developments
- **ALCATEL Space** for Satellite ground projects
- **EUROCOPTER** for the Australian TIGER helicopter
- **CNES** (French space agency) for Satellite projects
- **Siemens VDO :** Automotive computers
- **ALSTOM :** Singapore & Lausanne metros,**…**

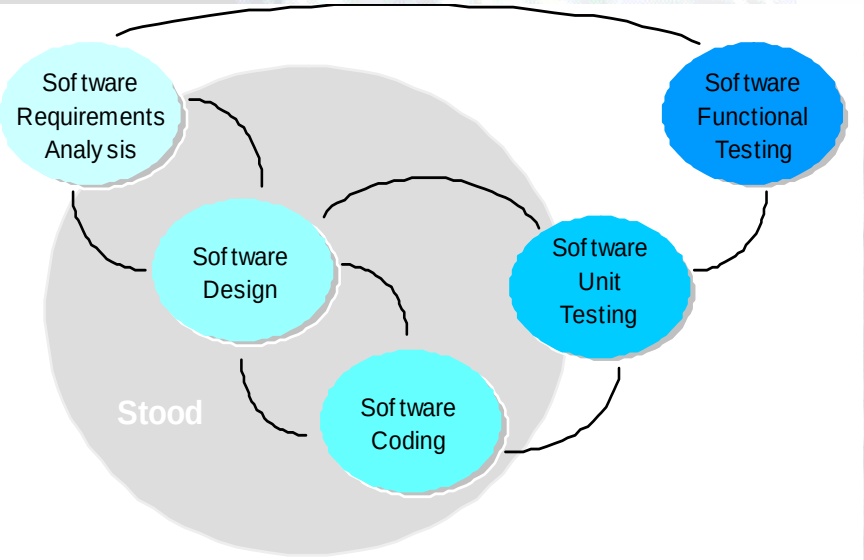
tNi Europe

Stood 5

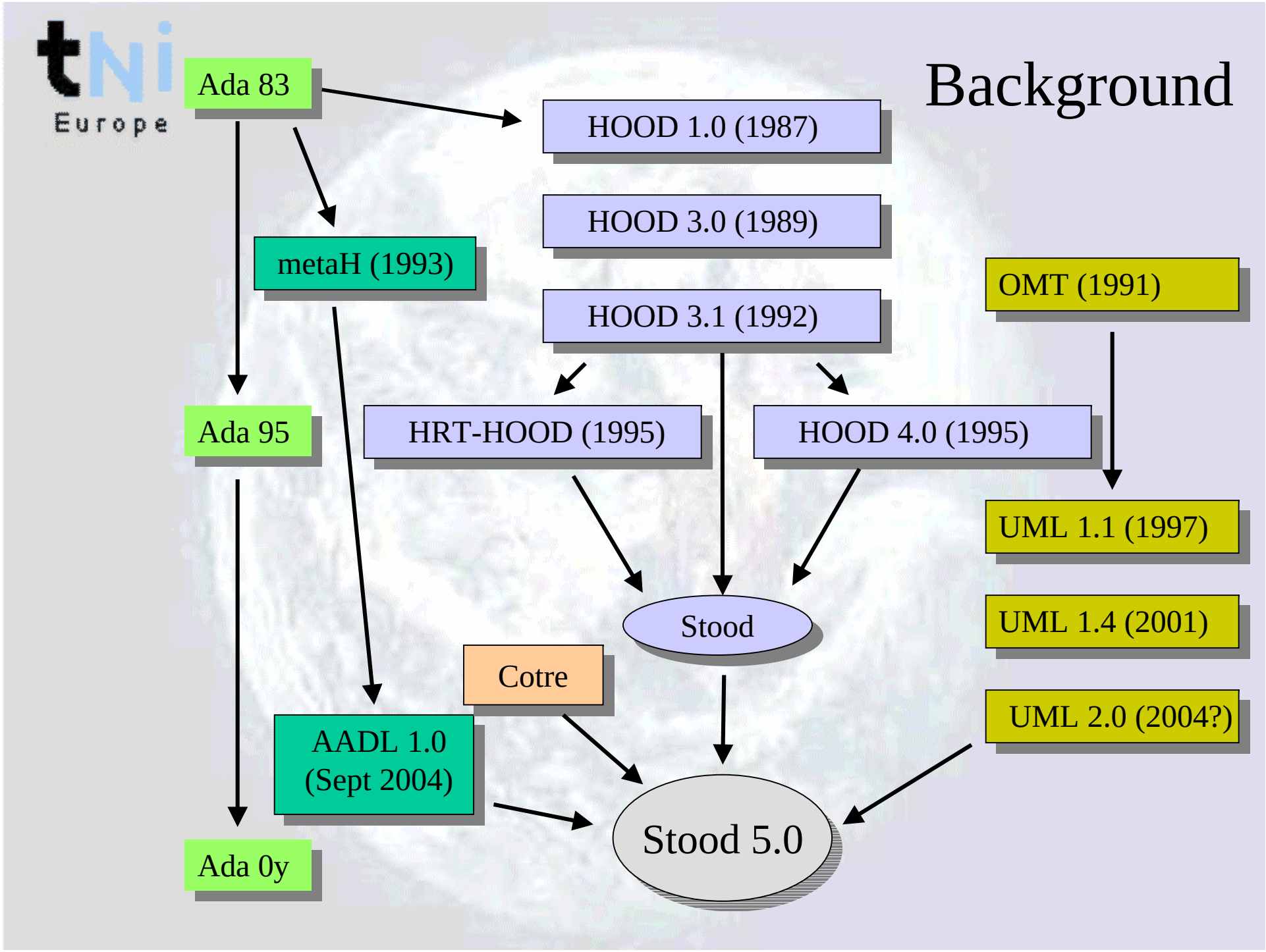
www.tni-world.com

Stood

- An industrial software design tool
- Already deployed & supported on many critical projects (DO-178B, ECSS-E40, MIL-STD-498)
- UML 2.0 front end & AADL plug-in



Background


tNi Europe
Ada 83
HOOD 1.0 (1987)
HOOD 3.0 (1989)
metaH (1993)
HOOD 3.1 (1992)
OMT (1991)
Ada 95
HRT-HOOD (1995)
HOOD 4.0 (1995)
UML 1.1 (1997)
Stood
UML 1.4 (2001)
Cotre
AADL 1.0
(Sept 2004)
UML 2.0 (2004?)
Stood 5.0
Ada 0y

In line with current trends

- promotes Model Driven Engineering: « designing before coding »

- advanced modeling solution
- model transformations

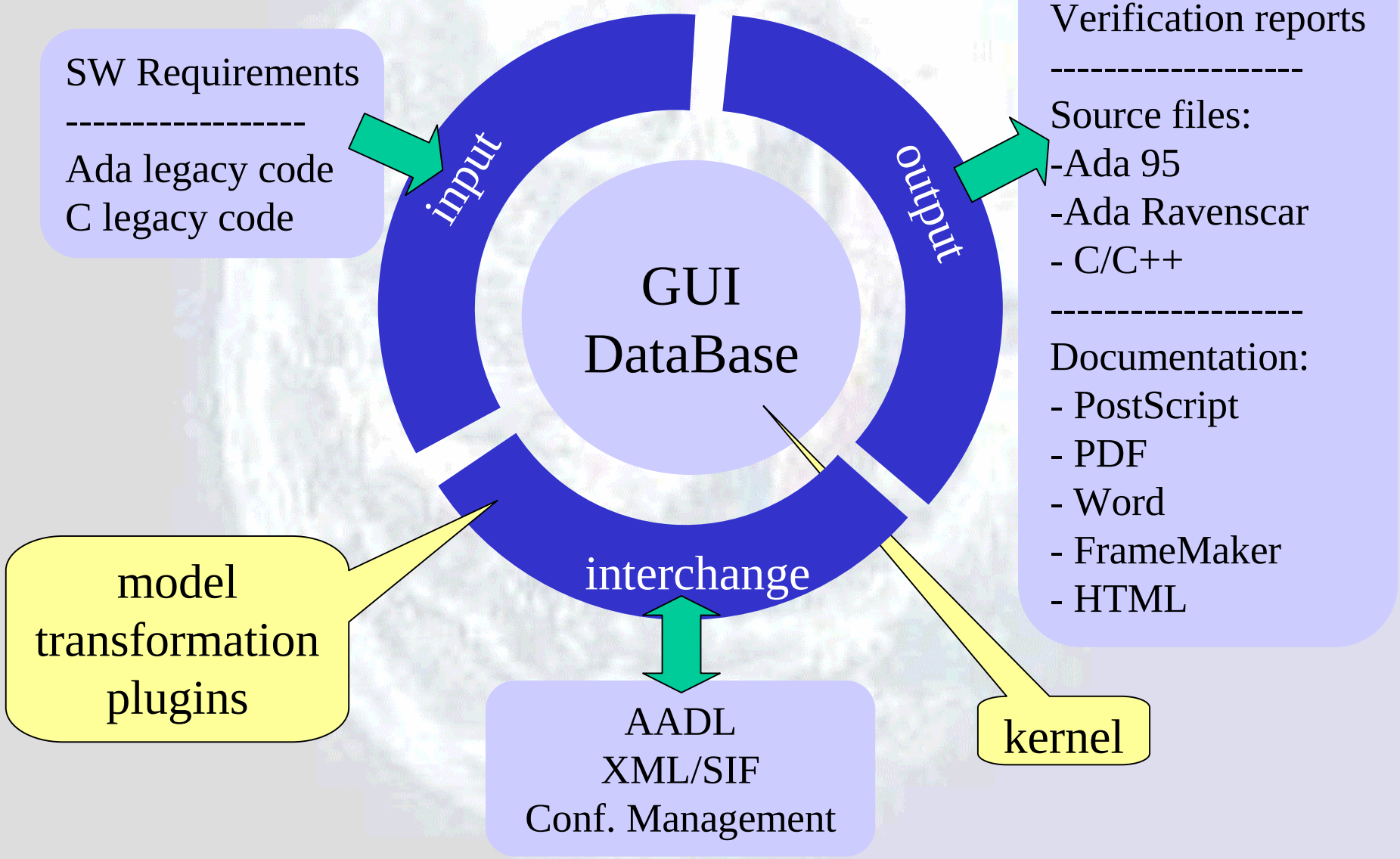
- promotes Component Based Architectures to ease:

- team development
- reuse
- testing
- maintenance

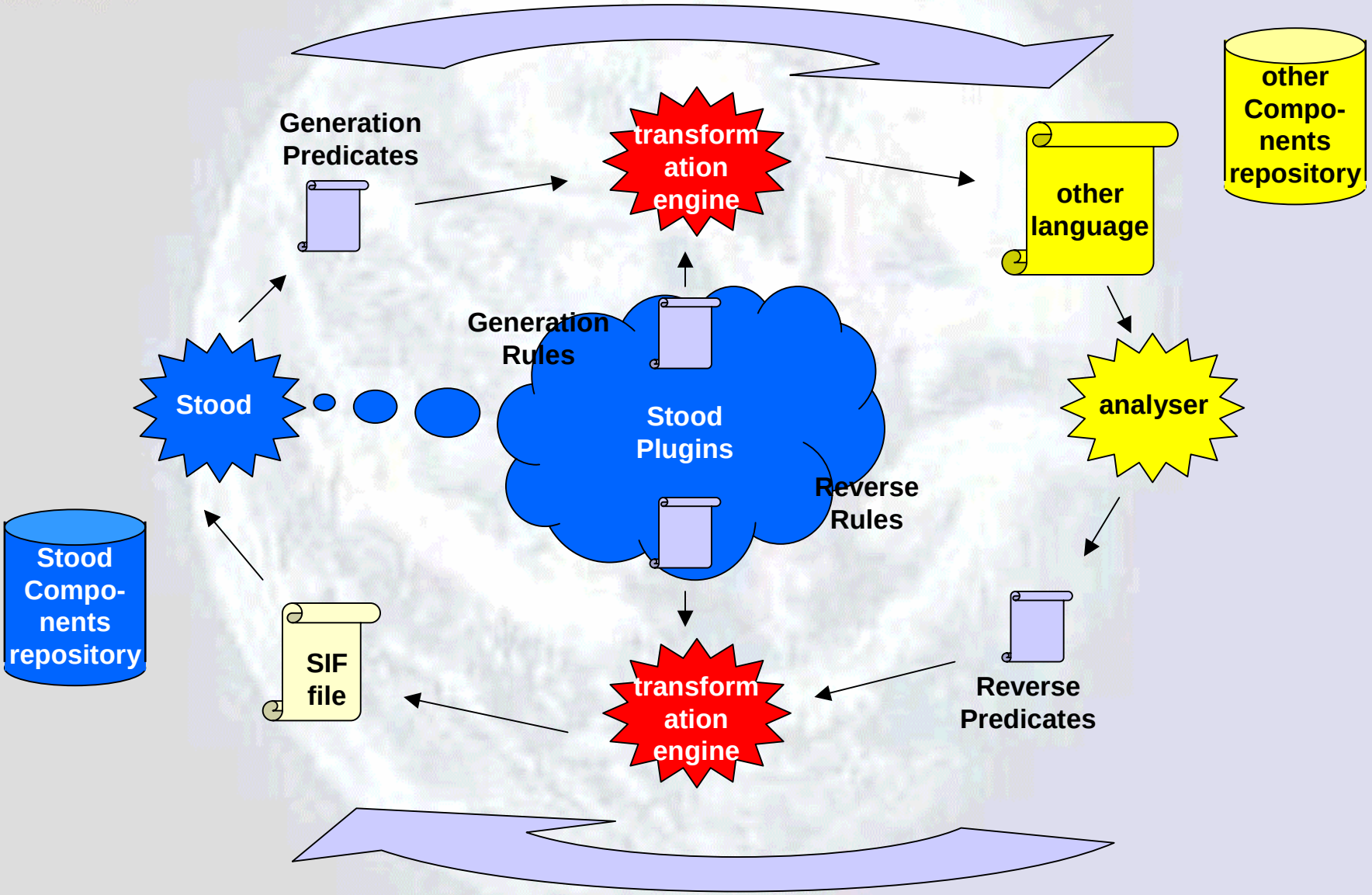
- promotes flexible Software Design practices:

- incremental documentation
- incremental coding and round-trip engineering
- incremental requirements traceability
- extensive tool customization capabilities

Tool overview



Model transformations



Formal transformation rules

example: AADL generator

- AADL definition:

```
component_type_extension ::=
component_category defining_component_type_identifier
extends unique_component_type_identifier
[ features ( { feature | feature_refinement }+ | none_statement ) ]
[ flows ( { flow_spec | flow_spec_refinement }+ | none_statement ) ]
[ properties ( { component_type_property_association }+ | none_statement ) ]
{ annex_subclause }*
end defining_component_type_identifier ;
```

- Corresponding code generation rule in prolog:

```
genComponentType(X,C,I,P) :-
  indent(I), write(C), sp, write(X),
  opt_EXTENSION(X,C), nl,
  opt_FEATURES(X,I,P),
  opt_FLOWSPEC(X,I),
  opt_TYPPROPERTIES(X,I),
  opt_ANNEXES(X,I),
  indent(I), write('END '), write(X), sc, nl, nl.
```

What is a Component ?

- UML 2.0 *(final adopted specification)*

  « A component can always be considered an autonomous unit within a system or subsystem. It has one or more **provided** and **required** interfaces (...), and its **internals** are hidden and inaccessible other than as provided by its interfaces. Although it may be dependent on other elements in terms of interfaces that are required, a component is **encapsulated** and its **dependencies** are designed such that it can be treated as independently as possible. »

- AADL 1.0 *(AS5506)*

  « A *component* represents some hardware or software entity that is part of a system being modeled in AADL. A component has a *component type*, which defines a **functional interface**. The component type acts as the specification of a component that other components can operate against. (...) A component has zero or more *component implementations.* A component implementation specifies an **internal structure** for a component as an assembly of subcomponents. »

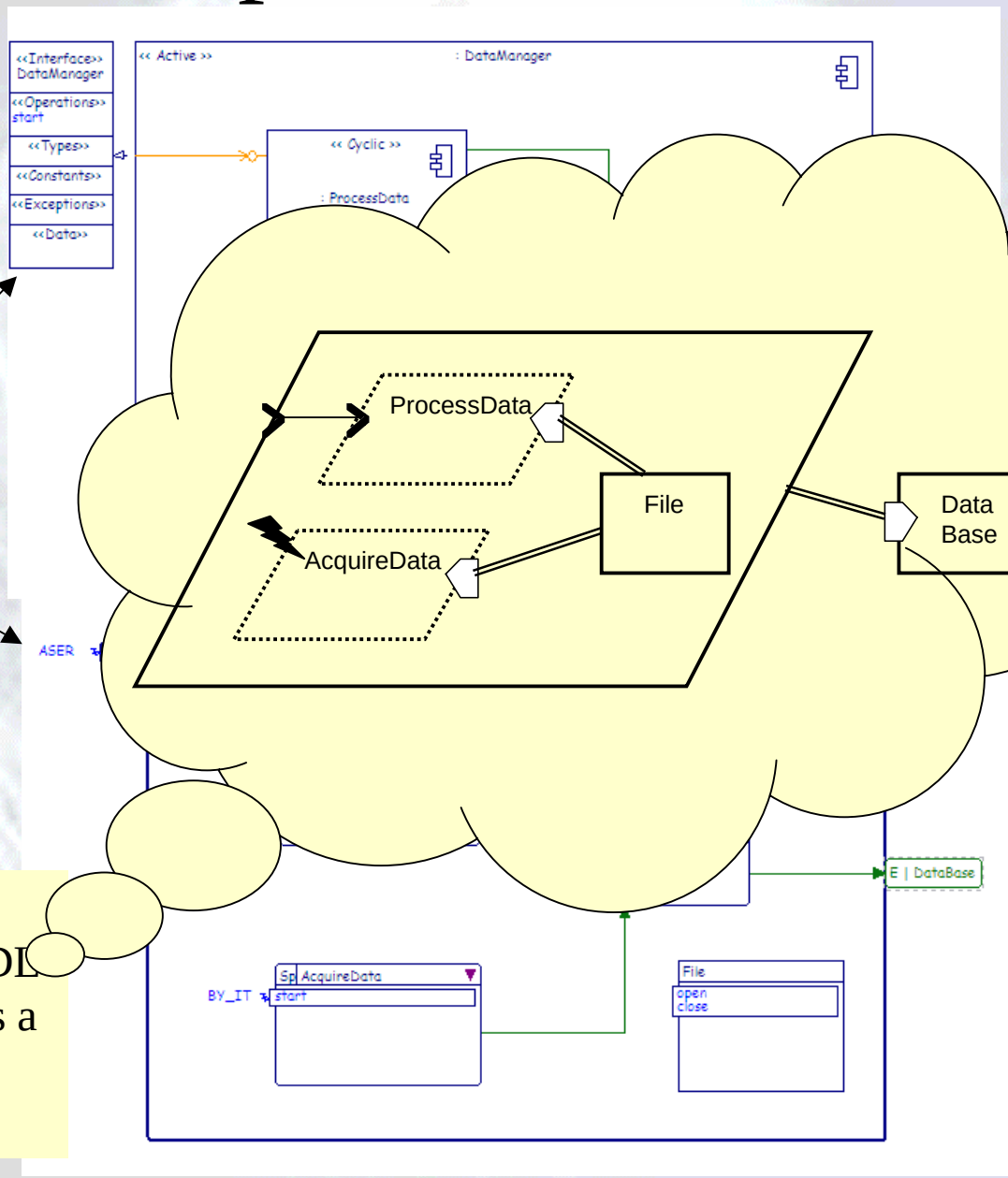
- HOOD *(HRM 4)*

  « A HOOD object is thus a software module specification, being primarily an encapsulation of services provided to other client software. (...) An object has a visible part (the **interface**) , and a hidden part (the **internals**) which cannot be accessed directly by external objects. (...) The interface part defines the services (...) **provided** by the object, as well as the services **required** from other objects. »

Why AADL ?

- AADL is System oriented and can be used in the early phases of a project.
  - It complements and easily interacts with the UML 2.0 / HOOD Software modeling approach
  - It may become an efficient communication media all along the project lifecycle.
- It brings a default predefined behavioural semantics to real-time components.
  - It can be used at System level for simulation
  - It can be used at Software level for advanced real-time code generation
- It offers wide extension mechanisms
  - Property_sets and Annexes
  - Already used by the COTRE (ending) and ASSERT (starting) projects
- It is already supported by the industry of critical systems in the USA and in Europe.

Graphical notations



Note:
an annex of the AADL standard also defines a specific graphical notation

STOOD 5 Summary

UML gives the general background:

What is a component ?

+

AADL brings precise semantics for real-time components:

What is the behaviour of a periodic thread ?

+

HOOD offers a well structured process to build the system:

How do I define and assemble my components ?

=

Stood provides the appropriate framework to support all that in the context of real industrial projects:

- **productivity**: distributed development, reuse of legacy data, code generation
- **quality**: verifications, documentation, certification issues

Features summary 1/2

Support of the Software Design activities

Architectural Design

- components based approach
with black-box and white-box views
- UML 2.0 graphical notation
- AADL import/export
- support of HOOD and HRT HOOD methodology
- built-in real-time model

Verifications

- cross references table
- automatic calculation of the required interfaces
- automatic generation of call trees and dataflow graphs
- real-time schedulability analysis
- requirements traceability matrix
- design rules checker
- design metrics

Detailed Design & Coding

- customizable structured detailed design framework
- incremental documentation
- incremental coding and round-trip engineering
- incremental requirements coverage
- legacy Ada and C code reverse engineering

Features summary 2/2

Workflow Integration

Project management

- full Windows-Unix interoperability
- network distributed project bases
- integrated interface to remote Configuration Management Systems
- multi user management at system and subsystem level
- SIF and XML design model interchange

Compliancy to Standards

- DO-178B for embedded avionics
- ECSS-E40 for space systems
- EN-50128 for railways
- MIL-STD-498 for military

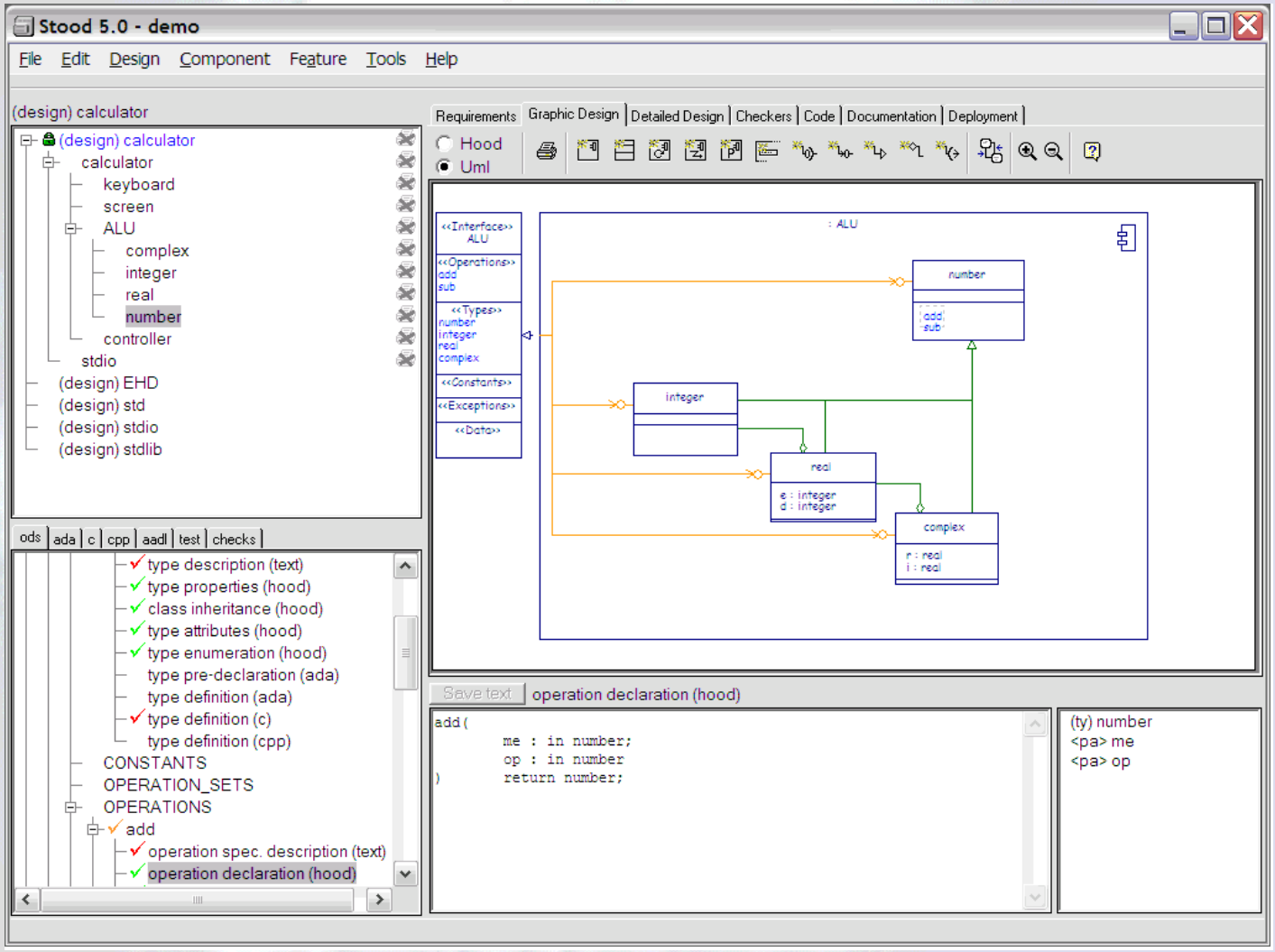
Requirements traceability

- import of high level requirements
- incremental requirements coverage
- management of the derived requirements
- bidirectional interface with Reqtify™

Code & Doc generators

- Ada95
- C/C++
- HTML
- PostScript/PDF
- RTF (Word™)
- MIF (FrameMaker™)

Try it...



stood@tni-world.com