



# AbsInt Webinar Release 16.10

AbsInt GmbH  
2016

# Agenda

- New features of Astrée (30min)
- New features of a<sup>3</sup> (30min)
  - StackAnalyzer
  - aiT
  - TimingProfiler

# Handling Absolute Addresses

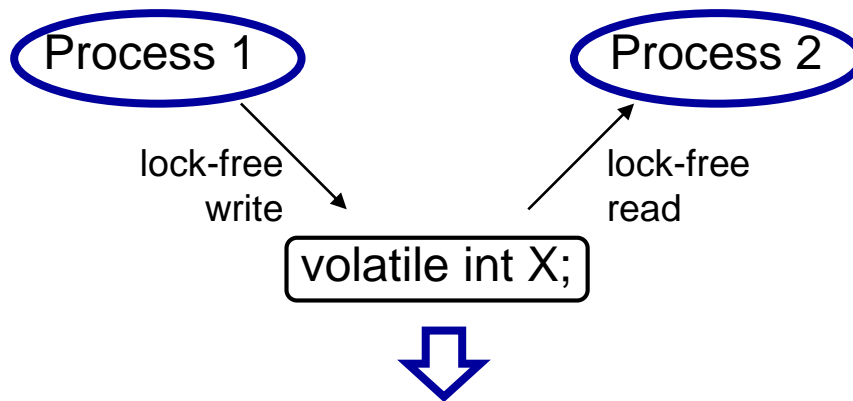
- Analyses no longer stop upon encountering undeclared absolute addresses.
  - Type-A alarm is raised, but not classified as definite runtime error.
  - Suggestions for `__ASTREE_absolute_address` directives are provided.
  - All values written to any undeclared absolute address are collected and used for any read from an undeclared absolute address.
- `@<address>` declarations are automatically translated to `__ASTREE_absolute_address` directives.
  - Example: `int i @0x1234;`

# Analysis of Concurrent Software

- Astrée now also reports all potential **deadlocks**.
  - New alarm category.
  - Deadlock cycles are printed in the analyzer log.
- New domain for precise handling of **process priorities** including dynamic priorities, e.g., according to the **Priority Ceiling Protocol**.
- A new **data flow view** enables users to efficiently explore data races.
- The **call graph view** can now show all possible call paths for **accesses to global variables**.
- The **wrapper generator** now generates entry code using `__astree_create/start_process()` and takes **priorities** into account.


# Analysis of Concurrent Software

- Alarm messages about data races now indicate whether variable access is **volatile** and **atomic**.
  - **Atomicity** of basic data types can be specified in the ABI configuration.
  - New option to **suppress** data race alarms on **volatile atomic variables**.



read/write data race is harmless if access is atomic

# Astrée Client (GUI)

- Analyzed code lines with **no errors nor type A/B/C alarms** are marked in **green**.
- Exclamation marks  denote code lines with alarms or notifications. Clicking on the exclamation mark opens all findings from this line in the findings view.
- Watch window functionality (**invariants**) available for lines with **blue line numbers**.
- Redesigned **Reports view**
  - Directories and file names of report files can be freely chosen
- Simplified configuration of **rule checks**
  - Easy enabling/disabling rules according to MISRA categories



# Reporting

- Jenkins Continuous Integration plugin now available
- New customizable HTML reports available
- Percentage of 'proven statements' shown:
  - Statement is considered proven if it is reached by the analyzer and causes no error nor a type A/B/C alarm.
  - Lines with only proven statements are colored green (see above).
- Astrée XML result file format now has an XML schema definition at [absint.com/dtd/astree-dax-report-configuration-16.10.xsd](http://absint.com/dtd/astree-dax-report-configuration-16.10.xsd)

# Rule Checking

- MISRA C:2012 Amendment 1 is now supported.
- ISO/IEC TS 17961 C secure coding rules supported.
- SEI Cert C supported (preliminary).
- Many semantic MISRA rules now can be checked without running the sound run-time error analysis (`skip-analysis=yes`).
  - 5 MISRA C:2004 rules and 4 MISRA C:2012 rules partially checked. Example: "There shall be no unreachable code".
  - 2 MISRA C:2004 rules and 2 MISRA C:2012 rules fully checked (lower precision). Example: uninitialized variable access.
  - 5 MISRA C:2004 rules and 6 MISRA C:2012 rules only checked with activated analyzer (`skip-analysis=no`).
- Eclipse plugin for Astrée rulechecker (KERCI) available from Konzept Informationssysteme GmbH.




# Further Improvements

- TargetLink Coupling
  - TargetLink **importer** settings are now **shared** between all clients.
  - **Missing sources files** no longer added to the analysis configuration.
  - **Data directory** file can now be specified using a **relative path** to the global base directory.
- The analyzer now supports **pointers of different sizes**, e.g., near/far/huge pointers.


# Agenda

- ✓ New features of Astrée (30min)
  
- New features of a<sup>3</sup> (30min)
  - StackAnalyzer
  - aiT
  - TimingProfiler


# New a<sup>3</sup> Features in 16.10

- a<sup>3</sup> Workspaces
  - Save complete analysis state & results (graph, statistics,...) for later review
  - Workspace export in alauncher (Option: `--export-workspace ws.apx` )
- Generating AIS annotations from
  - Graph/GUI
  - Message View (as annotation hints)
  - AIS Editor
- Loop Bound View in Statistics
  - AIS annotation generation
- Project File Generator from built-in Editor and Information Views
- Stack area specification in GUI
- a<sup>3</sup> Jenkins Plugin 
- Extended AIS2 annotations for
  - Initialization Value Analysis
  - Handling Tail Calls
  - New end/offset functor


# New a<sup>3</sup> Features in 16.10

- a<sup>3</sup> Workspaces
  - Save complete analysis state & results (graph, statistics,...) for later review
  - Workspace export in alauncher (Option: `--export-workspace ws.apx` )
- Generating AIS annotations from
  - Graph/GUI
  - Message View (as annotation hints)
  - AIS Editor
- Loop Bound View in Statistics
  - AIS annotation generation
- Project File Generator from built-in Editor and Information Views
- Stack area specification in GUI
- a<sup>3</sup> Jenkins Plugin 
- Extended AIS2 annotations for
  - Initialization Value Analysis
  - Handling Tail Calls
  - New end/offset functor

# New a<sup>3</sup> Features in 16.10


- a<sup>3</sup> Workspaces
  - Save complete analysis state & results (graph, statistics,...) for later review
  - Workspace export in alauncher (Option: `--export-workspace ws.apx` )
- Generating AIS annotations from
  - Graph/GUI
  - Message View (as annotation hints)
  - AIS Editor
- Loop Bound View in Statistics
  - AIS annotation generation
- Project File Generator from built-in Editor and Information Views
- Stack area specification in GUI
- a<sup>3</sup> Jenkins Plugin 
- Extended AIS2 annotations for
  - Initialization Value Analysis
  - Handling Tail Calls
  - New end/offset functor

# New a<sup>3</sup> Features in 16.10


- a<sup>3</sup> Workspaces
  - Save complete analysis state & results (graph, statistics,...) for later review
  - Workspace export in alauncher (Option: `--export-workspace ws.apx` )
- Generating AIS annotations from
  - Graph/GUI
  - Message View (as annotation hints)
  - AIS Editor
- Loop Bound View in Statistics
  - AIS annotation generation
- Project File Generator from built-in Editor and Information Views
- Stack area specification in GUI
- a<sup>3</sup> Jenkins Plugin 
- Extended AIS2 annotations for
  - Initialization Value Analysis
  - Handling Tail Calls
  - New end/offset functor



# New a<sup>3</sup> Features in 16.10

- a<sup>3</sup> Workspaces
  - Save complete analysis state & results (graph, statistics,...) for later review
  - Workspace export in alauncher (Option: `--export-workspace ws.apx` )
- Generating AIS annotations from
  - Graph/GUI
  - Message View (as annotation hints)
  - AIS Editor
- Loop Bound View in Statistics
  - AIS annotation generation
- Project File Generator from built-in Editor and Information Views
- Stack area specification in GUI
- a<sup>3</sup> Jenkins Plugin 
- Extended AIS2 annotations for
  - Initialization Value Analysis
  - Handling Tail Calls
  - New end/offset functor

# New a<sup>3</sup> Features in 16.10


- a<sup>3</sup> Workspaces
  - Save complete analysis state & results (graph, statistics,...) for later review
  - Workspace export in alauncher (Option: `--export-workspace ws.apx` )
- Generating AIS annotations from
  - Graph/GUI
  - Message View (as annotation hints)
  - AIS Editor
- Loop Bound View in Statistics
  - AIS annotation generation
- Project File Generator from built-in Editor and Information Views
- Stack area specification in GUI
- a<sup>3</sup> Jenkins Plugin 
- Extended AIS2 annotations for
  - Initialization Value Analysis
  - Handling Tail Calls
  - New end/offset functor

# The a<sup>3</sup> Jenkins Plug-In



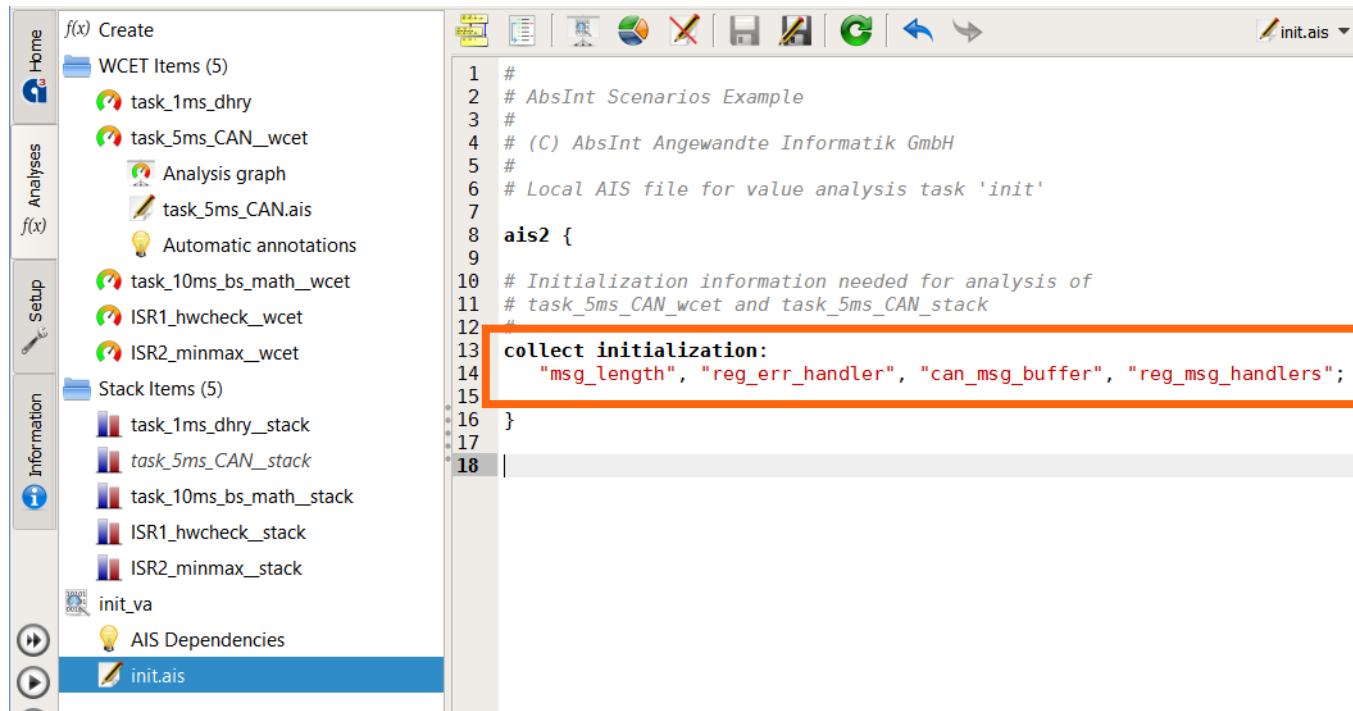
- Provides automatic integration of TimingProfiler, StackAnalyzer and aiT in the **Continuous Integration Framework Jenkins**
- One-click installation
- Easy to configure
- Fully integrated
- Features:
  - Configure an analyzer run as a **Jenkins build step**
  - Launch an a<sup>3</sup> analysis project
  - **Automatically** mark a build step as **failed** depending on
    - **Violated expectations** in the analysis results
    - **Warnings/Errors** in the analysis results (pedantic level)
  - Prints a **compact result table** and **lists failed analysis items** in the Jenkins Build output
  - Archives **analysis reports** directly in your **Jenkins workspace**
  - Access **analysis results** via the **Jenkins web interface**

# New a<sup>3</sup> Features in 16.10

- a<sup>3</sup> Workspaces
  - Save complete analysis state & results (graph, statistics,...) for later review
  - Workspace export in alauncher (Option: `--export-workspace ws.apx` )
- Generating AIS annotations from
  - Graph/GUI
  - Message View (as annotation hints)
  - AIS Editor
- Loop Bound View in Statistics
  - AIS annotation generation
- Project File Generator from built-in Editor and Information Views
- Stack area specification in GUI
- a<sup>3</sup> Jenkins Plugin 
- Extended AIS2 annotations for
  - Initialization Value Analysis
  - Handling Tail Calls
  - New end/offset functor

# Collect Initializations

- If **ValueAnalyzer Add-On** is available
- Use initialization (value) analysis to collect (pointer) variable initializations:



```
1 #
2 # AbsInt Scenarios Example
3 #
4 # (C) AbsInt Angewandte Informatik GmbH
5 #
6 # Local AIS file for value analysis task 'init'
7
8 ais2 {
9
10 # Initialization information needed for analysis of
11 # task_5ms_CAN_wcet and task_5ms_CAN_stack
12 #
13 collect initialization:
14 "msg_length", "reg_err_handler", "can_msg_buffer", "reg_msg_handlers";
15
16 }
17
18 |
```

# New AIS2 Features (1)

- New types for collecting initialization values:

- *For all structure members at once*

```
collect initialization: ("<name>".);
```

- *All variables of a certain type*

```
collect initialization: type(void () *);
```

- *All function pointer variables*

```
collect initialization: type(function pointer);
```



## New AIS2 Features (2)

- Annotating tail calls now more convenient

```
instruction <pp> tail calls: <targets>;
```

instead of

```
instruction <pp> {  
    calls: ...;  
    type: tail call;  
}
```

# New AIS2 Features (3)

- New functors available:

Use

```
end(<area>)
```

instead of  $\text{address}(\langle\text{area}\rangle) + \text{width}(\langle\text{area}\rangle) - 1$

- Compute the offset of a structure field "data" in a global structure variable "fTable" with the following functor:

```
offset(("fTable"."data"));
```

# a<sup>3</sup>/Astrée Release Notes for 16.10

<http://www.absint.com/releasenotes/a3/16.10/>

<http://www.absint.com/releasenotes/astree/16.10/>



email: [info@absint.com](mailto:info@absint.com)

<http://www.absint.com>