EMBEDDED SYSTEMS SOLUTIONS

axivion
stopping software erosion

# Axivion Suite – Webinar India 20/11/2019

## Sustaining Maintainability of Your Software Applications

## Agenda

- Presentation Axivion (14min)
  - Company
  - Technology
- Demonstration (6min)
  - Walk through Axivion Suite
- Q&A (5-15min)
  - Please raise your questions in the Q&A section of this Webinar
  - The Chat is disabled
  - We will start answering the questions in this session

## About Axivion: Software Erosion Protection Made in Germany

axivion
stopping software erosion

Axivion is a 100% owner-managed technology company with profitable and sustainable growth and …

… more than 10 years of experience,

sound academic background,

development in Germany, support and service local

more than 100 customers worldwide,

several thousand users, and …

… references across all different industries with technical software.

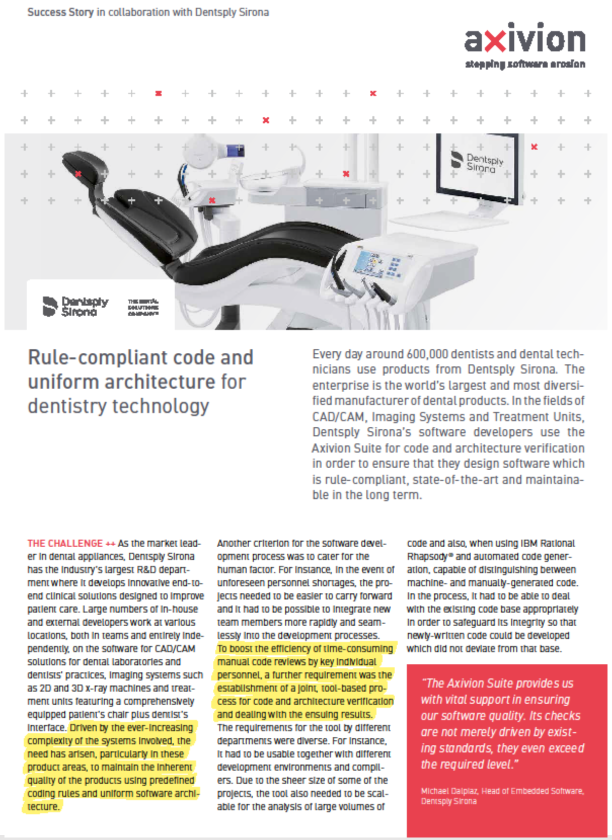
## About us: Some of Our Customers
## Software Erosion Matters Across All Technologies

axivion
stopping software erosion

| | | | | | | |
|---|---|---|---|---|---|---|
| annax | AVL SOFTWARE AND FUNCTIONS | BALLUFF | BASLER | BOSCH Technik fürs Leben | Bühler Motor Bühler | CETITEC connective technologies |
| CI Tech Components AG | Dallmeier | Dentsply Sirona | dunkermotoren | DÜRKOPP ADLER | Elektrobit | eos |
| FRESENIUS MEDICAL CARE | haas | HELLA | HENSOLDT Detect and Protect. | HILTI | ibeo automotive | ifm |
| JENOPTIK | JUNGHEINRICH | KNORR-BREMSE | Leuze electronic the sensor people | MAGNA | iro MEMMINGER-IRO | ottobock. |
| PHOENIX CONTACT | sartorius | SCHAEFFLER | semcon | SENNHEISER | SIEMENS | SMA |
| TQ | Relectronic | TRUMPF | VECTOR | VEGA | WIBU SYSTEMS | ZF |

# Customer References 1/2

Wir unterstützen unsere Kunden,
damit in der Küche nichts anbrennt.
B/S/H/
axivion
stopping software erosion



**Success Story** in collaboration with Dentsply Sirona

axivion
stopping software erosion

Dentsply Sirona

## Rule-compliant code and uniform architecture for dentistry technology

Every day around 600,000 dentists and dental technicians use products from Dentsply Sirona. The enterprise is the world's largest and most diversified manufacturer of dental products. In the fields of CAD/CAM, Imaging Systems and Treatment Units, Dentsply Sirona's software developers use the Axivion Suite for code and architecture verification in order to ensure that they design software which is rule-compliant, state-of-the-art and maintainable in the long term.

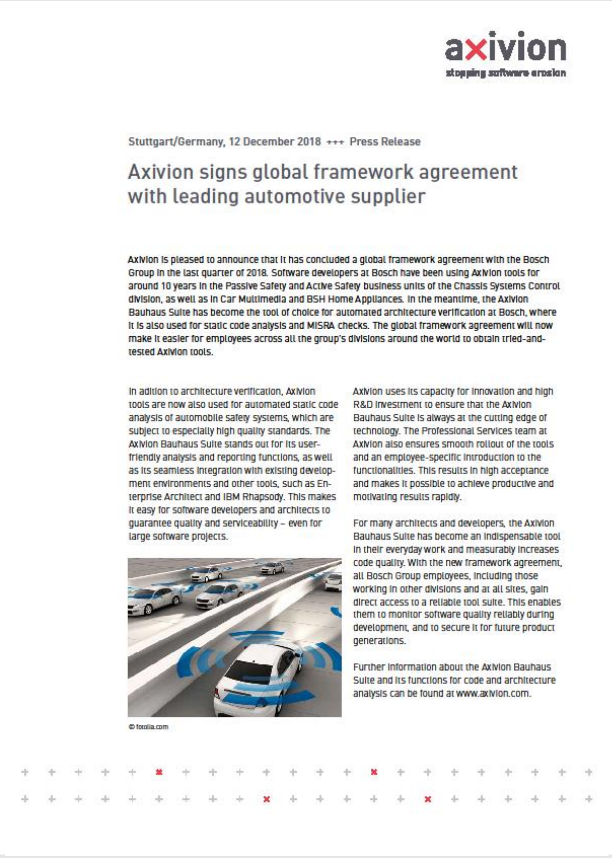
THE CHALLENGE ++ As the market leader in dental appliances, Dentsply Sirona has the industry's largest R&D department where it develops innovative end-to-end clinical solutions designed to improve patient care. Large numbers of in-house and external developers work at various locations, both in teams and entirely independently, on the software for CAD/CAM solutions for dental laboratories and dentists' practices, imaging systems such as 2D and 3D x-ray machines and treatment units featuring a comprehensively equipped patient's chair plus dentist's interface. Driven by the ever-increasing complexity of the systems involved, the need has arisen, particularly in these product areas, to maintain the inherent quality of the products using predefined coding rules and uniform software architecture.

Another criterion for the software development process was to cater for the human factor. For instance, in the event of unforeseen personnel shortages, the projects needed to be easier to carry forward and it had to be possible to integrate new team members more rapidly and seamlessly into the development processes. To boost the efficiency of time-consuming manual code reviews by key individual personnel, a further requirement was the establishment of a joint, tool-based process for code and architecture verification and dealing with the ensuing results. The requirements for the tool by different departments were diverse. For instance, it had to be usable together with different development environments and compilers. Due to the sheer size of some of the projects, the tool also needed to be scalable for the analysis of large volumes of code and also, when using IBM Rational Rhapsody® and automated code generation, capable of distinguishing between machine- and manually-generated code. In the process, it had to be able to deal with the existing code base appropriately in order to safeguard its integrity so that newly-written code could be developed which did not deviate from that base.

*"The Axivion Suite provides us with vital support in ensuring our software quality. Its checks are not merely driven by existing standards, they even exceed the required level."*

Michael Dalplaz, Head of Embedded Software, Dentsply Sirona

# Customer References 2/2



Stuttgart/Germany, 12 December 2018 +++ Press Release

## Axivion signs global framework agreement with leading automotive supplier

Axivion is pleased to announce that it has concluded a global framework agreement with the Bosch Group in the last quarter of 2018. Software developers at Bosch have been using Axivion tools for around 10 years in the Passive Safety and Active Safety business units of the Chassis Systems Control division, as well as in Car Multimedia and BSH Home Appliances. In the meantime, the Axivion Bauhaus Suite has become the tool of choice for automated architecture verification at Bosch, where it is also used for static code analysis and MISRA checks. The global framework agreement will now make it easier for employees across all the group's divisions around the world to obtain tried-and-tested Axivion tools.

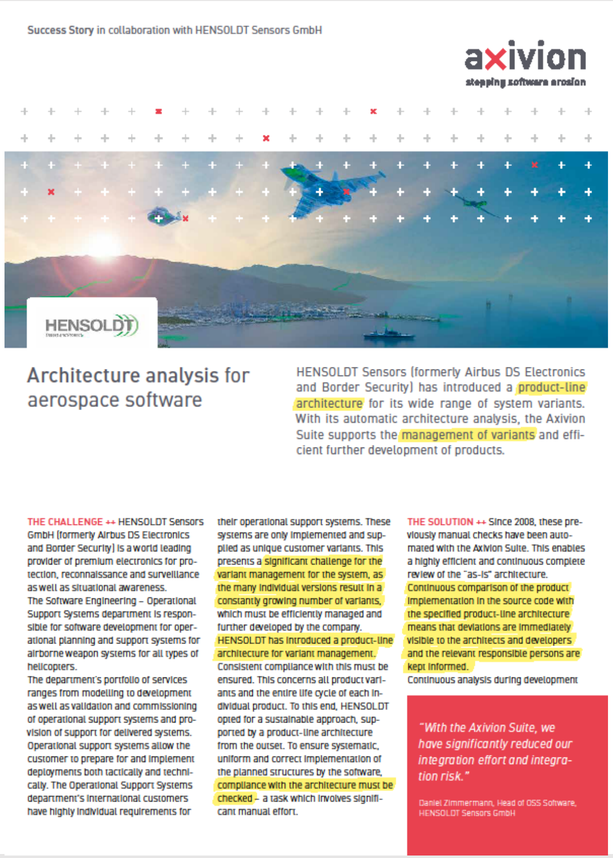
In adition to architecture verification, Axivion tools are now also used for automated static code analysis of automobile safety systems, which are subject to especially high quality standards. The Axivion Bauhaus Suite stands out for its user-friendly analysis and reporting functions, as well as its seamless integration with existing development environments and other tools, such as Enterprise Architect and IBM Rhapsody. This makes it easy for software developers and architects to guarantee quality and serviceability – even for large software projects.

© fotolia.com

Axivion uses its capacity for innovation and high R&D investment to ensure that the Axivion Bauhaus Suite is always at the cutting edge of technology. The Professional Services team at Axivion also ensures smooth rollout of the tools and an employee-specific introduction to the functionalities. This results in high acceptance and makes it possible to achieve productive and motivating results rapidly.

For many architects and developers, the Axivion Bauhaus Suite has become an indispensable tool in their everyday work and measurably increases code quality. With the new framework agreement, all Bosch Group employees, including those working in other divisions and at all sites, gain direct access to a reliable tool suite. This enables them to monitor software quality reliably during development, and to secure it for future product generations.

Further information about the Axivion Bauhaus Suite and its functions for code and architecture analysis can be found at www.axivion.com.



Success Story in collaboration with HENSOLDT Sensors GmbH

## Architecture analysis for aerospace software

HENSOLDT Sensors (formerly Airbus DS Electronics and Border Security) has introduced a product-line architecture for its wide range of system variants. With its automatic architecture analysis, the Axivion Suite supports the management of variants and efficient further development of products.

THE CHALLENGE ++ HENSOLDT Sensors GmbH (formerly Airbus DS Electronics and Border Security) is a world leading provider of premium electronics for protection, reconnaissance and surveillance as well as situational awareness.
The Software Engineering – Operational Support Systems department is responsible for software development for operational planning and support systems for airborne weapon systems for all types of helicopters.
The department's portfolio of services ranges from modelling to development as well as validation and commissioning of operational support systems and provision of support for delivered systems. Operational support systems allow the customer to prepare for and implement deployments both tactically and technically. The Operational Support Systems department's international customers have highly individual requirements for their operational support systems. These systems are only implemented and supplied as unique customer variants. This presents a significant challenge for the variant management for the system, as the many individual versions result in a constantly growing number of variants, which must be efficiently managed and further developed by the company.
HENSOLDT has introduced a product-line architecture for variant management.
Consistent compliance with this must be ensured. This concerns all product variants and the entire life cycle of each individual product. To this end, HENSOLDT opted for a sustainable approach, supported by a product-line architecture from the outset. To ensure systematic, uniform and correct implementation of the planned structures by the software, compliance with the architecture must be checked – a task which involves significant manual effort.

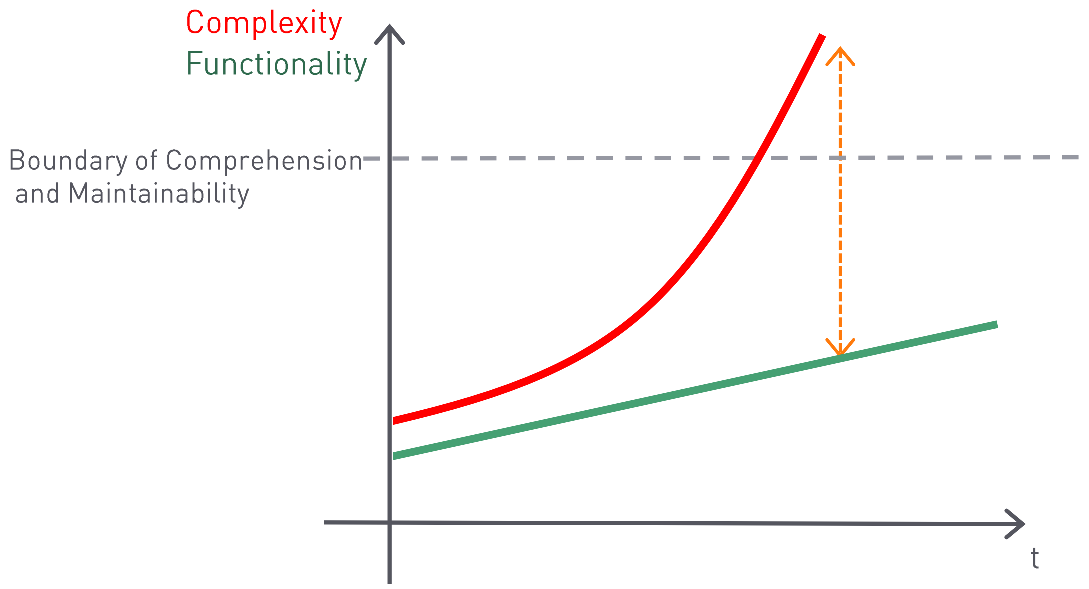
THE SOLUTION ++ Since 2008, these previously manual checks have been automated with the Axivion Suite. This enables a highly efficient and continuous complete review of the "as-is" architecture.
Continuous comparison of the product implementation in the source code with the specified product-line architecture means that deviations are immediately visible to the architects and developers and the relevant responsible persons are kept informed.
Continuous analysis during development

*"With the Axivion Suite, we have significantly reduced our integration effort and integration risk."*

Daniel Zimmermann, Head of OSS Software, HENSOLDT Sensors GmbH

## Competitive Advantage – Stopping Software Erosion

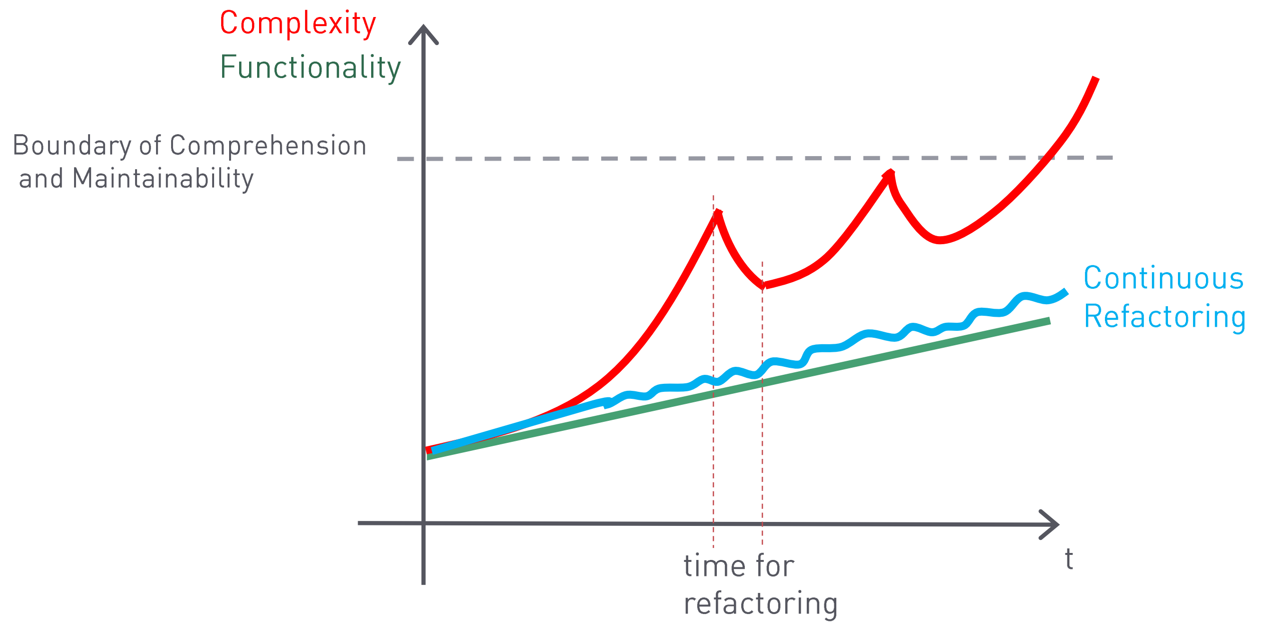
- Architecture Verification against a model (UML or drawn in Gravis)
- Extremely low number of FALSE POSITIVES
- seamless Integration into DevOps / CI
- IDE Integration (deep integration into Eclipse and Visual Studio, connection to almost all IDEs possible)
- Support of the latest language features; C++17, AUTOSAR C++14 (19/03)
- Performance
    - even multi-million LOC could be checked in a nightly build
- Delta Analysis
    - show changes in a selected time window
- Open - well documented Python API to the data
- Service Capability local to the customer



## Drivers of Complexity

- Architecture Violation/ Hidden Dependency
- Clone
- Metric Violation
- Style Violation
- Dead Code
- Cycle

## Stopping Software Erosion



- Architecture Violation/ Hidden Dependency
- Clone
- Metric Violation
- Style Violation
- Dead Code
- Cycle

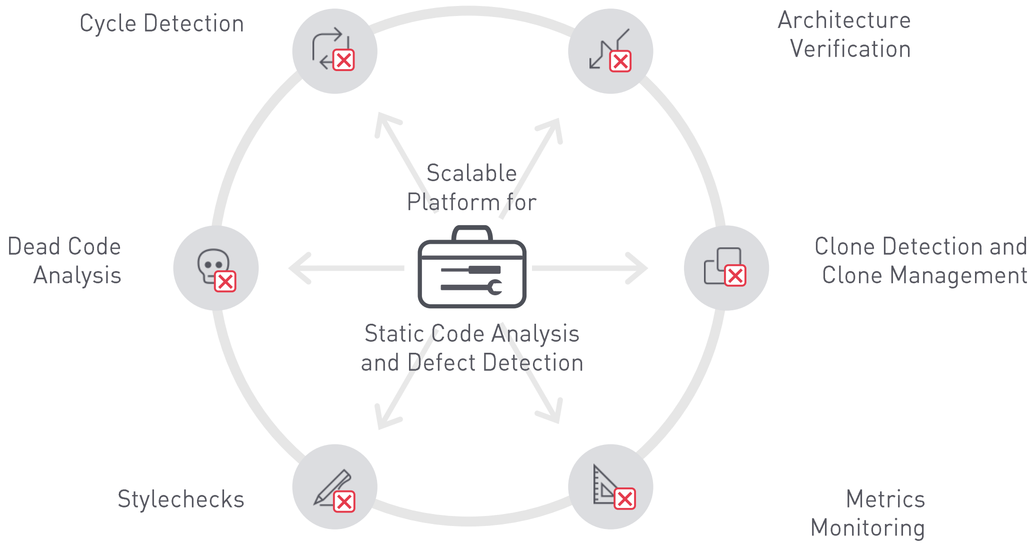
# Critical Success Factor #1: Integrated, tailored, and holistic Approach to mitigate ALL Erosion Categories

axivion
stopping software erosion

- Calls
- Module Dependencies
- Includes/Imports
- Customized Checks

- Reachability Analysis
- Analysis of Libraries

- Misra C:2004 and C:2012
- Misra C++:2008
- AUTOSAR C++14
- CERT®
- Best Practice
- Race Conditions
- Customized Checks



Cycle Detection
Architecture Verification
Scalable Platform for
Dead Code Analysis
Clone Detection and Clone Management
Static Code Analysis and Defect Detection
Stylechecks
Metrics Monitoring

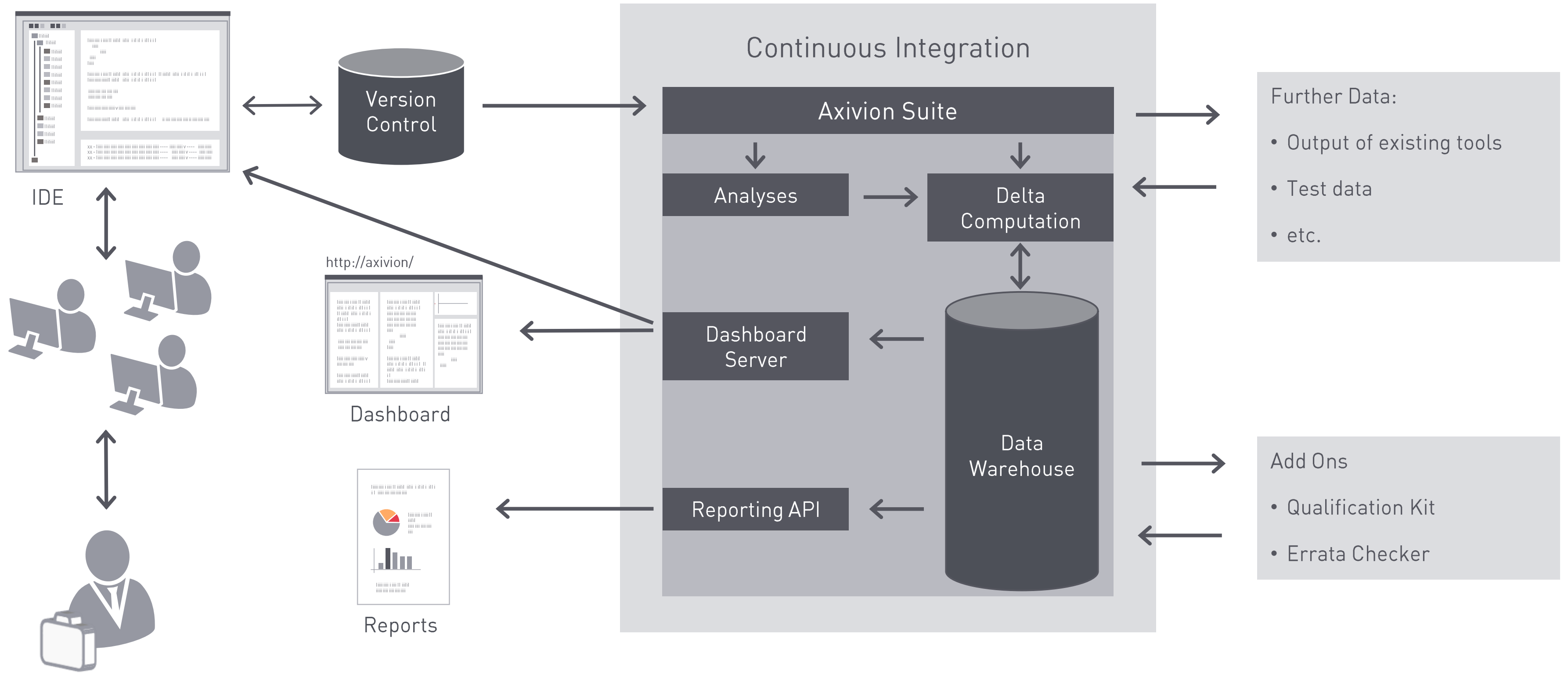
- Integrated Modeler
- Interfaces to UML-Tools via XMI
- AUTOSAR XML Import

- Type I (1:1)
- Type II (Changes)
- Type III (Changes + Add/Delete)

- HIS
- OO Design Metrics
- Best Practice
- Customized Checks

## Critical Success Factor #2: Fast and Focused Feedback by Integration into Development Environments and Processes

axivion
stopping software erosion

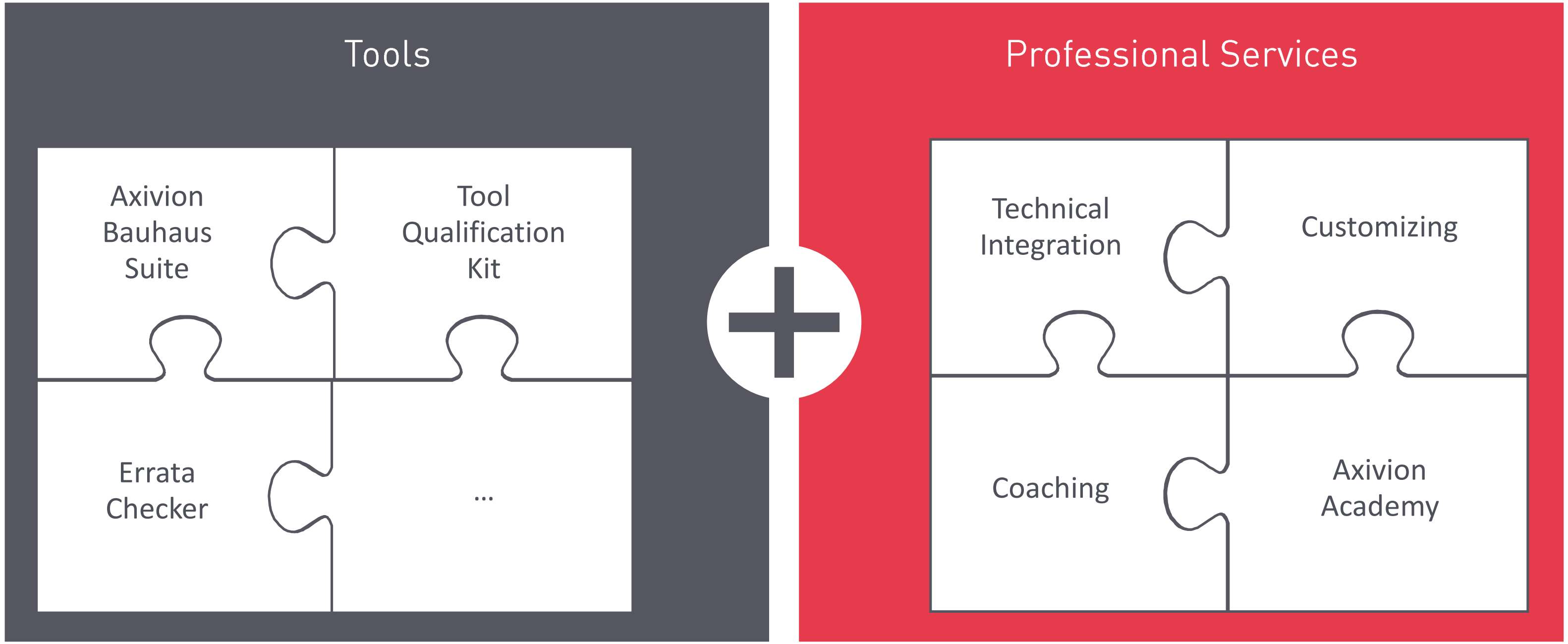


## Axivion is Full Service Provider: Combining Tools, Service, and Training Leads to Long Term Success

axivion
stopping software erosion

Complete Solution for Stopping Software-Erosion

EMBEDDED SYSTEMS SOLUTIONS



# ISO26262 – Safety in automotive

axion
stopping software erosion

- Axivion will help to cover required modelling and coding guidelines
- Style checks consist of MISRA, CERT, AUTOSAR C++14 and customer's rules
- Architecture Verification allows for consistency between architecture, design and implementation

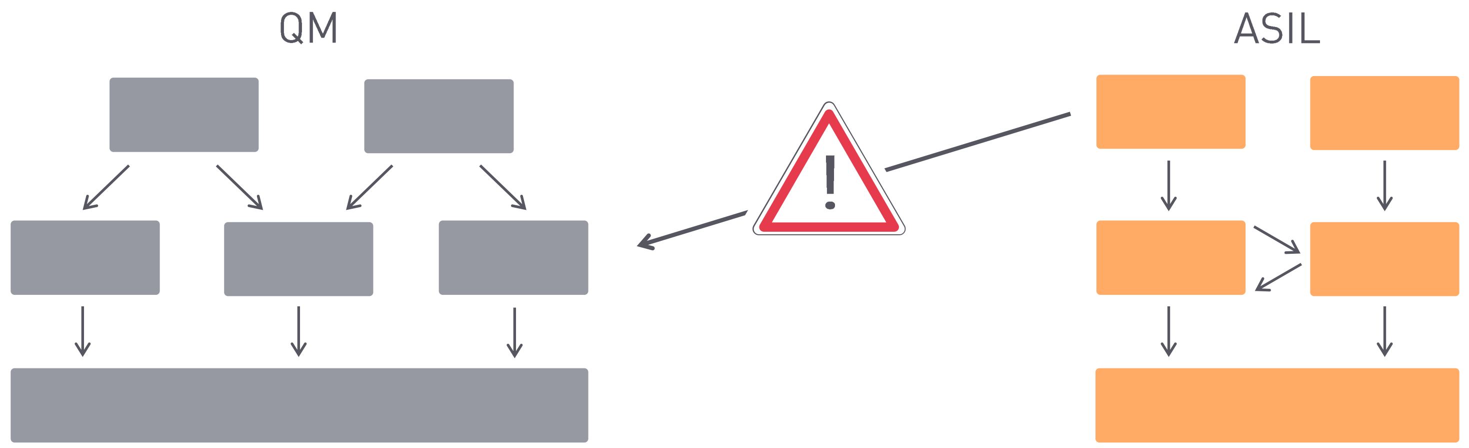
| Topics | | ASIL | | | | Axivion Suite solutions | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A | B | C | D | Architecture Verification | Clone Detection & Management | Cycle Detection | Dead Code Analysis | Metrics Monitoring | Stylechecks | Static/Semantic Code Analysis | Data Races |
| 1a | Enforcement of low complexity | ++ | ++ | ++ | ++ | ● | ● | ● | ● | ● | ● | | |
| 1e | Use of well-trusted design principles | + | + | ++ | ++ | ● | ● | ● | | ● | | | |
| 1f | Use of unambiguous graphical representation | + | ++ | ++ | ++ | ● | | | | | | | |
| 1b | Use of language subset | ++ | ++ | ++ | ++ | | | | | | ● | | |
| 1c | Enforcement of strong typing | ++ | ++ | ++ | ++ | | | | | | ● | ● | |
| 1d | Use of defensive implementation techniques | + | + | ++ | ++ | | ● | | | | ● | ● | |
| 1g | Use of style guides | + | ++ | ++ | ++ | | | | | | ● | | |
| 1h | Use of naming conventions | ++ | ++ | ++ | ++ | ● | | | | | ● | | |
| 1i | Concurrency aspects | + | + | + | + | | | | | | | ● | ● |

from: ISO 26262-6:2018, *Road vehicles — Functional safety — Part 6:* ***Product development at the software level, chapter 5***

## Mixed Criticality – Software Components with different ASIL Levels

*Static View on C / C++*

axion
stopping software erosion

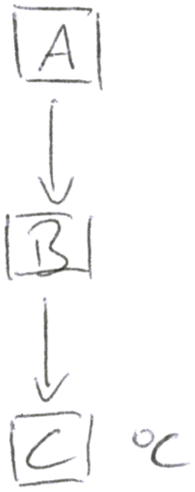


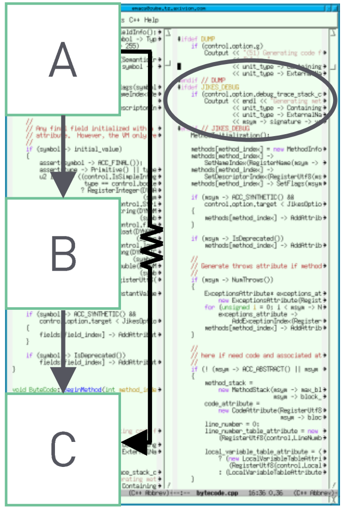
- Requirement: ASIL Code shall not call *(synchronous)* QM functionality
- Axivion helps to ensure Freedom from Interference
  - identifies calls from ASIL code into QM code or into code with lower criticality

## Impact of Architecture Violations and Hidden Dependencies

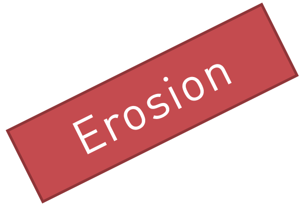
axivion
stopping software erosion

Architecture Specification → Implementation → Actions

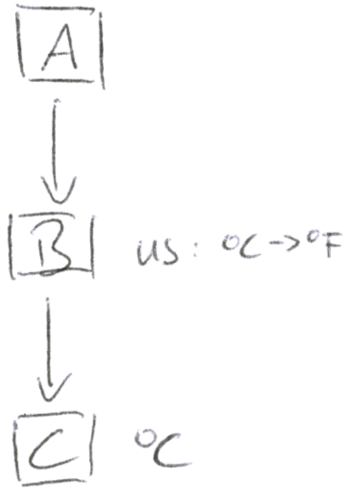




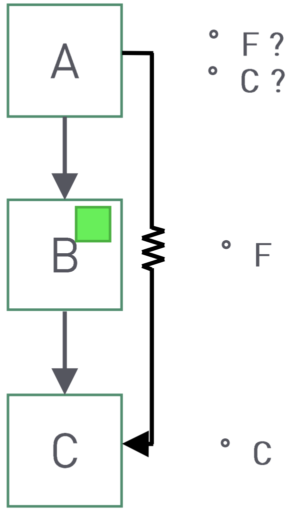
- None, as long as the divergence does not lead to errors



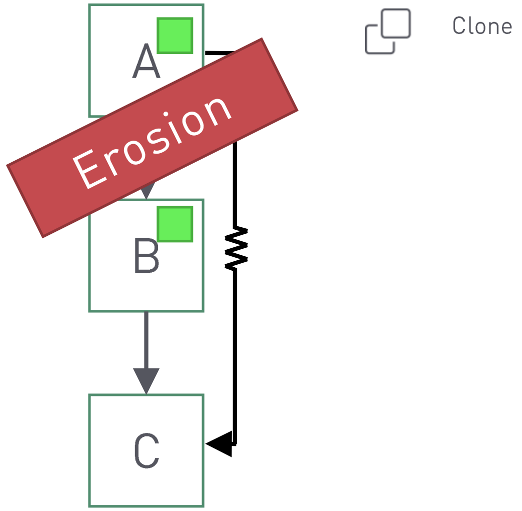
Redesign with false assumptions



Impacts of design



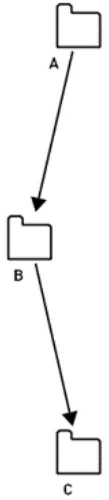
Workaround



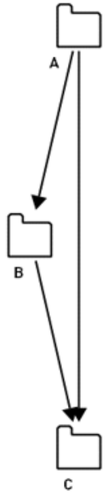
Preventing Software Erosion during Development
Avoiding Architecture Violations and Hidden Dependencies

axivion
stopping software erosion

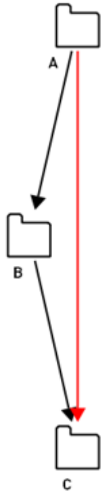
Case 1 (Deviation corrected in Source Code)



Case 2 (Update of the architecture specification)



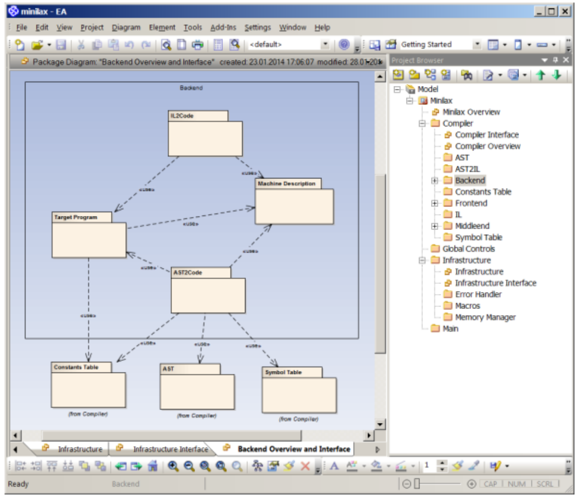
Case 3 (Deviation is accepted temporarily)



## Architecture Model exists (CASE-Tool, XMI, XML, etc.)
## Automated Import of relevant model parts

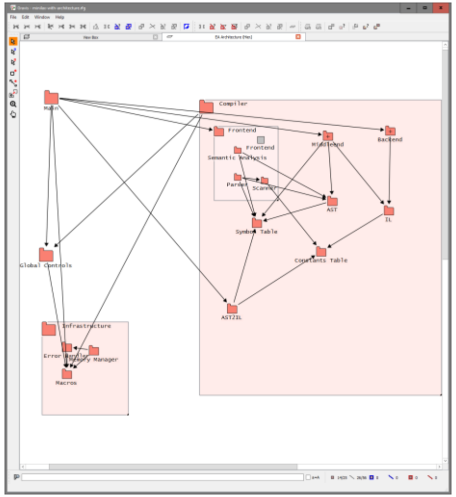
axivion
stopping software erosion

Architecture
e.g. in Enterprise Architect



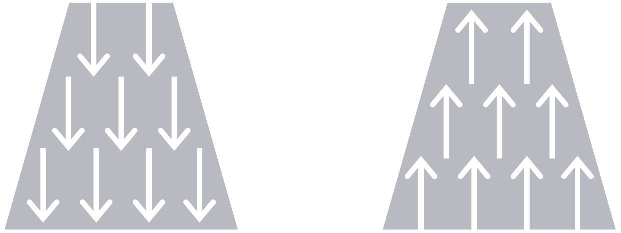
Automated Transfor-mation

Imported Architecture

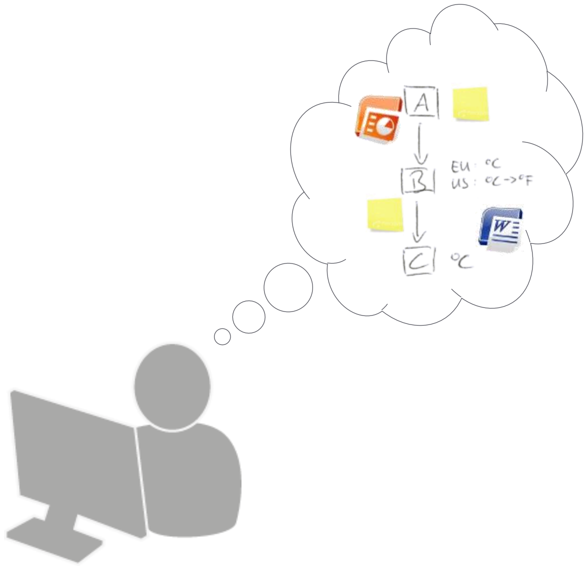


## Architecture Documentation does not exist (Live Project)
## Architecture Recovery

Approach: Top-Down or Bottom-Up



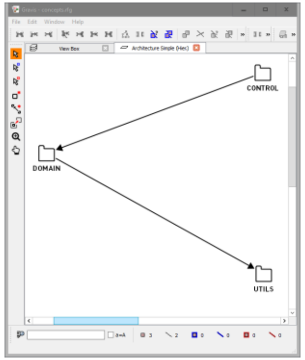
Developers usually have a conceptual model in mind. It makes perfect sense to explicit this model and check for “hidden” dependencies.

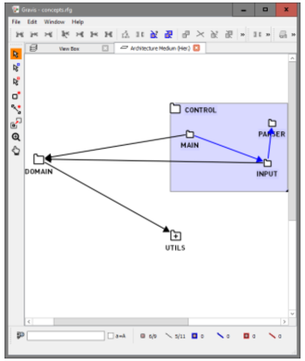


## Example for hypothesis-driven architecture recovery: 4. Complement the hypothesis and check again

Checking of Hypothesis

Describe Hypothesis

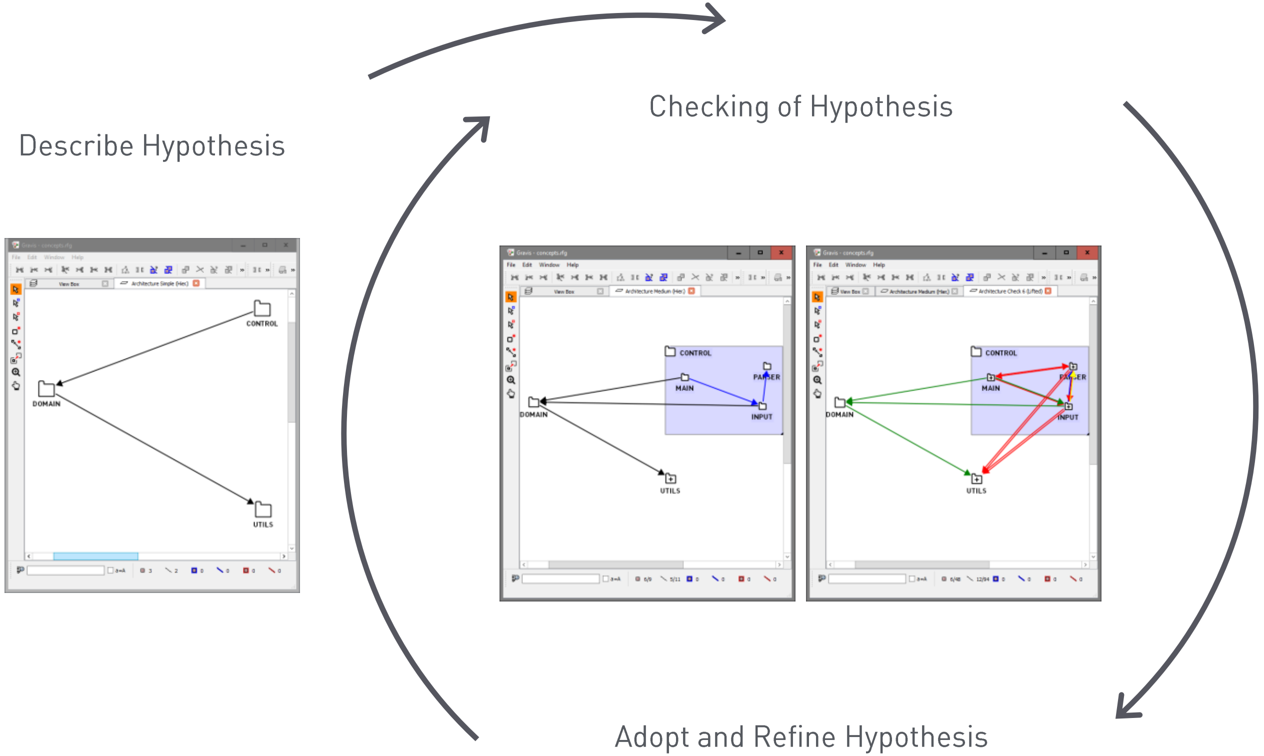




Adopt and Refine Hypothesis

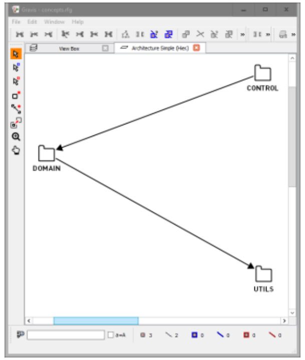
## Example for hypothesis-driven architecture recovery:
## 4. Complement the hypothesis and check again

axion
stopping software erosion

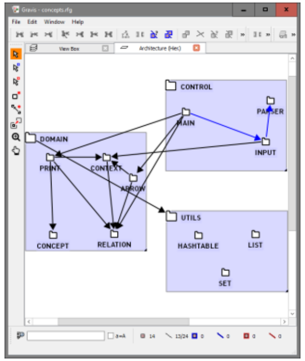


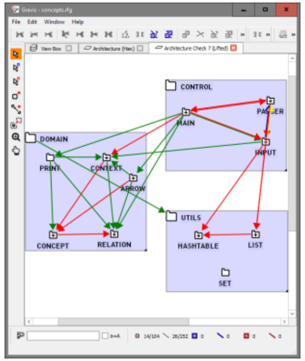
## Example for hypothesis-driven architecture recovery:
## 5. Prioritize remaining architecture violations and create issues to remove them

Describe Hypothesis

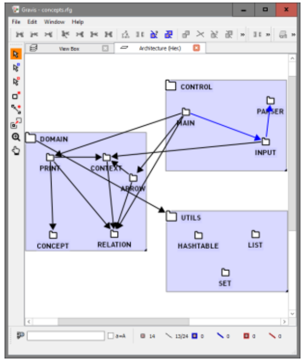


Checking of Hypothesis



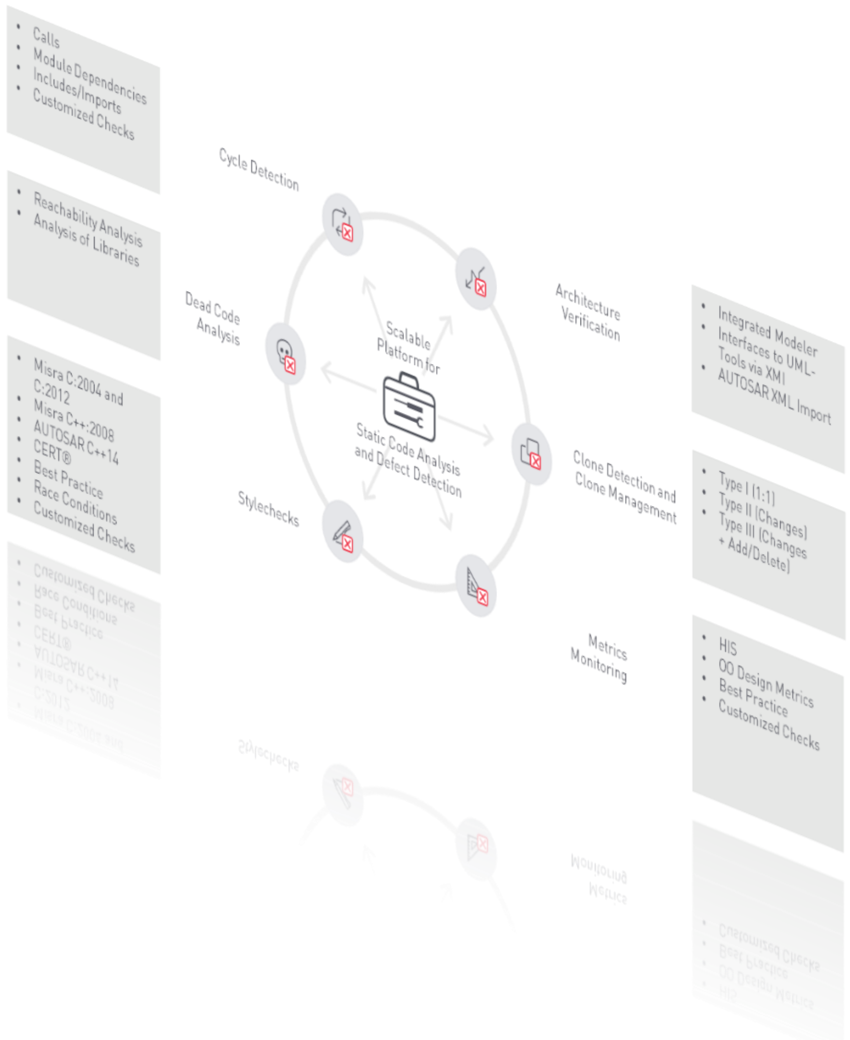


Validated Architecture



Adopt and Refine Hypothesis

## Demonstration of the Axivion Suite

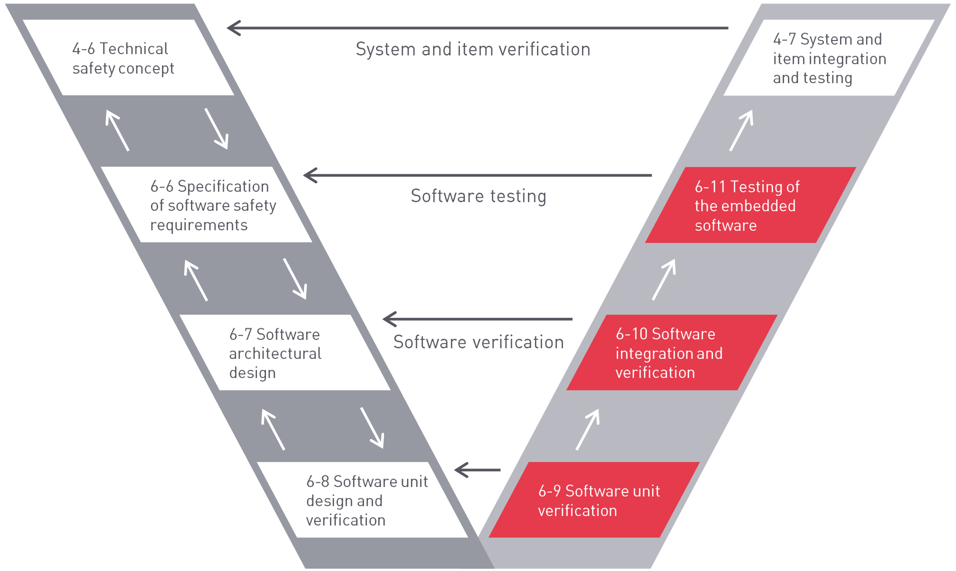


# ISO 26262 – Parts and Processes supported by Axivion

axivion
stopping software erosion

## ISO 26262

| | |
|---|---|
| 1. | |
| 2. | |
| 3. | |
| 4. | 6-8 Software unit design and implementation |
| 5. | 6-9 Software unit verification |
| 6. | 6-10 Software integration and verification |
| 7. | |
| 8. | 8-11 Confidence in the use of software tools |
| 9. | |
| 10. | |
| 11. | |
| 12. | |


4-6 Technical safety concept
System and item verification
4-7 System and item integration and testing
6-6 Specification of software safety requirements
Software testing
6-11 Testing of the embedded software
6-7 Software architectural design
Software verification
6-10 Software integration and verification
6-8 Software unit design and verification
6-9 Software unit verification

from: ISO 26262-6:2018, *Road vehicles — Functional safety — Part 6: Product development at the software level*
ISO 26262-8:2018, *Road vehicles — Functional safety — Part 8: Supporting processes*

# ISO26262 – Safety in automotive

axivion
stopping software erosion

- Axivion will help to cover required modelling and coding guidelines
- Style checks consist of MISRA, CERT, AUTOSAR C++14 and customer's rules
- Architecture Verification allows for consistency between architecture, design and implementation

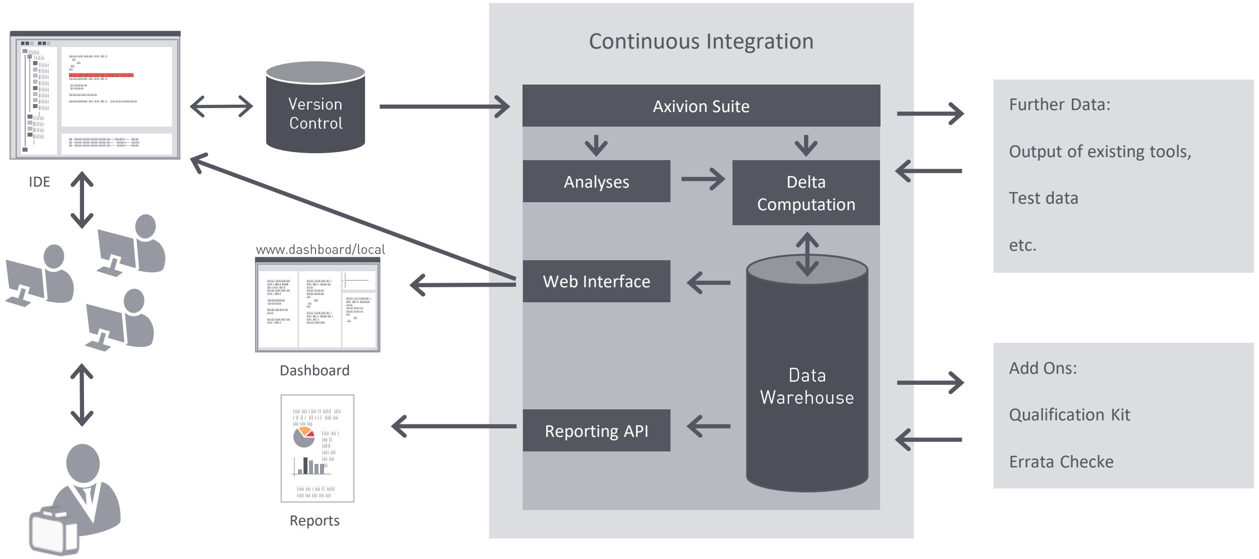
| Topics | | ASIL | | | | Axivion Suite solutions | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A | B | C | D | Architecture Verification | Clone Detection & Management | Cycle Detection | Dead Code Analysis | Metrics Monitoring | Stylechecks | Static/Semantic Code Analysis | Data Races |
| 1a | Enforcement of low complexity | ++ | ++ | ++ | ++ | ● | ● | ● | ● | ● | ● | | |
| 1e | Use of well-trusted design principles | + | + | ++ | ++ | ● | ● | ● | | ● | | | |
| 1f | Use of unambiguous graphical representation | + | ++ | ++ | ++ | ● | | | | | | | |
| 1b | Use of language subset | ++ | ++ | ++ | ++ | | | | | | ● | | |
| 1c | Enforcement of strong typing | ++ | ++ | ++ | ++ | | | | | | ● | ● | |
| 1d | Use of defensive implementation techniques | + | + | ++ | ++ | | ● | | | | ● | ● | |
| 1g | Use of style guides | + | ++ | ++ | ++ | | | | | | ● | | |
| 1h | Use of naming conventions | ++ | ++ | ++ | ++ | ● | | | | | ● | | |
| 1i | Concurrency aspects | + | + | + | + | | | | | | | ● | ● |

from: ISO 26262-6:2018, *Road vehicles — Functional safety — Part 6:* ***Product development at the software level, chapter 5***

# ISO26262 – Continuous Integration and Fast Roundtrip Times

EXAMPLE 2 Continuous integration based on an automated build system can support consistency of sub-phases and facilitate regression tests. Such a build system typically performs code generation, compiling and linking, static code analysis, documentation generation, testing and packaging. It allows, subject to tool chain and tool configuration, repeatable and, after changes, comparable production of software, documentation and test results.
from: ISO 26262-6:2018, *Road vehicles — Functional safety — Part 6: Product development at the software level*

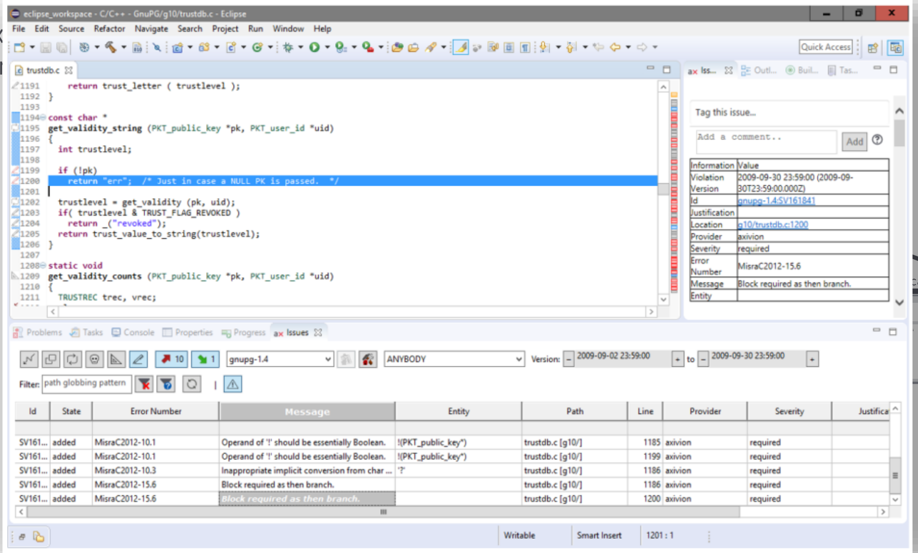
Axivion will also be installed locally to allow for fast checks and short roundtrip times



## ISO26262 – Continuous Integration and Fast Roundtrip Times

axion
stopping software erosion

EXAMPLE 2 **Continuous integration** based on an automated build system can support consistency of sub-phases and facilitate regression tests. Such a build system typically performs code generation, compiling and linking, **static code analysis, documentation generation**, testing and packaging. It allows, subject to tool chain and tool configuration, repeatable and, after changes, comparable production of software, documentation and test results.
from: ISO 26262-6:2018, *Road vehicles — Functional safety — Part 6: Product development at the software level*



eclipse_workspace - C/C++ - GnuPG/g10/trustdb.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
trustdb.c
Problems Tasks Console Properties Progress Issues
Iss... Outl... Buil... Tas...
Tag this issue...
Add a comment..
Add

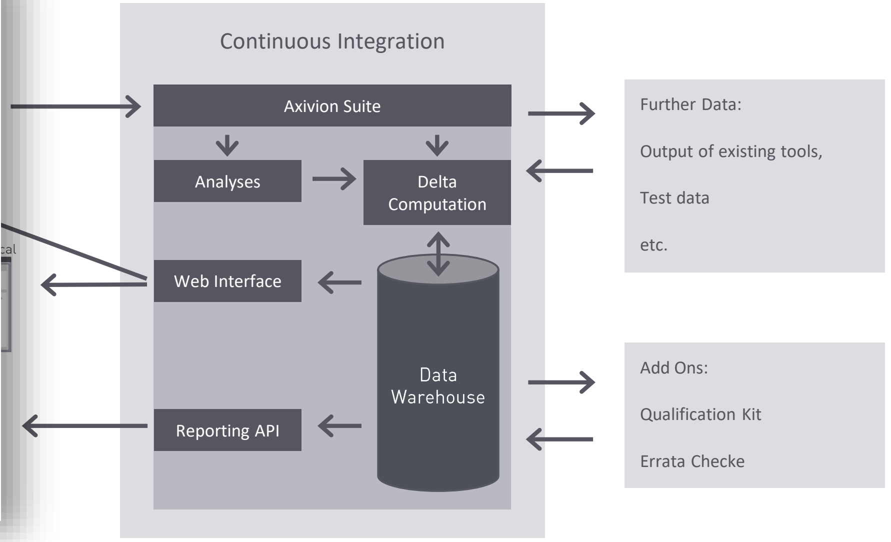
```
    return trust_letter ( trustlevel );
}

const char *
get_validity_string (PKT_public_key *pk, PKT_user_id *uid)
{
  int trustlevel;

  if (!pk)
    return "err";  /* Just in case a NULL PK is passed.  */

  trustlevel = get_validity (pk, uid);
  if( trustlevel & TRUST_FLAG_REVOKED )
    return _("revoked");
  return trust_value_to_string(trustlevel);
}

static void
get_validity_counts (PKT_public_key *pk, PKT_user_id *uid)
{
  TRUSTREC trec, vrec;
```

| Information | Value |
|---|---|
| Violation Version | 2009-09-30 23:59:00 (2009-09-30T23:59:00.000Z) |
| Id | gnupg-1.4:SV161841 |
| Justification | |
| Location | g10/trustdb.c:1200 |
| Provider | axivion |
| Severity | required |
| Error Number | MisraC2012-15.6 |
| Message | Block required as then branch. |
| Entity | |

10 1 gnupg-1.4 ANYBODY Version: 2009-09-02 23:59:00 to 2009-09-30 23:59:00
Filter: path globbing pattern

| Id | State | Error Number | Message | Entity | Path | Line | Provider | Severity | Justifica |
|---|---|---|---|---|---|---|---|---|---|
| SV161... | added | MisraC2012-10.1 | Operand of '!' should be essentially Boolean. | !(PKT_public_key*) | trustdb.c [g10/] | 1185 | axivion | required | |
| SV161... | added | MisraC2012-10.1 | Operand of '!' should be essentially Boolean. | !(PKT_public_key*) | trustdb.c [g10/] | 1199 | axivion | required | |
| SV161... | added | MisraC2012-10.3 | Inappropriate implicit conversion from char ... | '?' | trustdb.c [g10/] | 1186 | axivion | required | |
| SV161... | added | MisraC2012-15.6 | Block required as then branch. | | trustdb.c [g10/] | 1186 | axivion | required | |
| SV161... | added | MisraC2012-15.6 | Block required as then branch. | | trustdb.c [g10/] | 1200 | axivion | required | |

Writable Smart Insert 1201 : 1



# ISO26262-6: Chapter 7 – Software architectural design

axivion
stopping software erosion

- The description of the software architectural design shall address the following characteristics: comprehensibility, consistency, simplicity, verifiability, modularity, abstraction, encapsulation and maintainability
- Testability of the software architecture during software integration testing ✓
- Maintainability of the software architectural design ✓

| Notations | | | ASIL | | | |
|---|---|---|---|---|---|---|
| | | | A | B | C | D |
| 1a | Natural language * | ✓ | ++ | ++ | ++ | ++ |
| 1e | Informal notations | ✓ | ++ | ++ | + | + |
| 1f | Semi-formal notations ** | ✓ | + | + | ++ | ++ |
| 1b | Formal notations | | + | + | + | + |

* Natural language can complement the use of notations for example where some topics are more readily expressed in natural language or providing explanation and rationale for decisions captured in the notation.

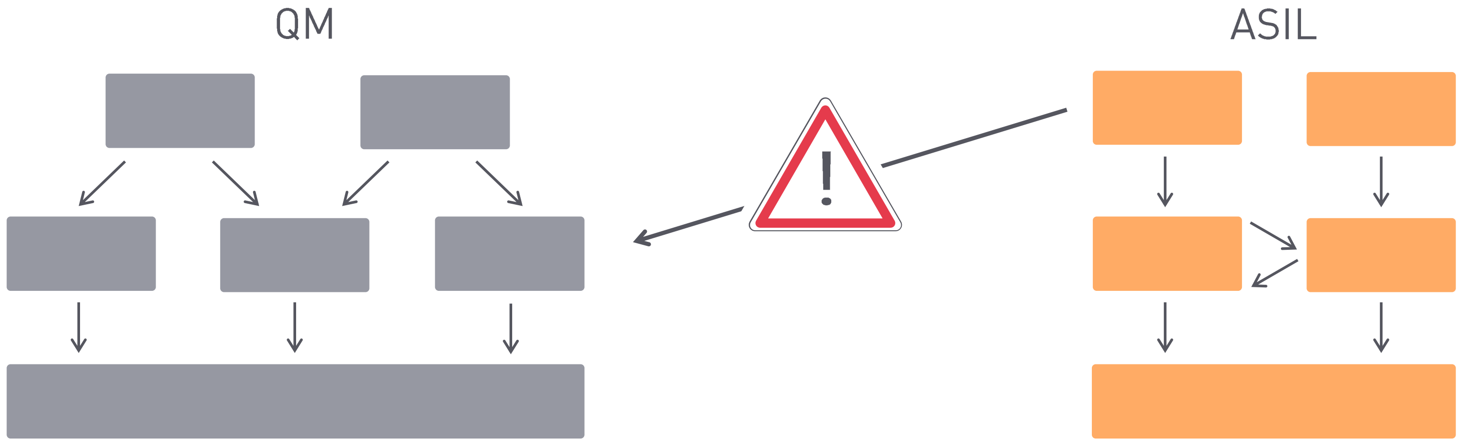
** Semi-formal notations can include pseudocode or modelling with UML®, SysML®, Simulink® or Stateflow®

Note: UML®, SysML®, Simulink® or Stateflow® are examples of suitable products available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of these products.

## Mixed Criticality – Software Components with different ASIL Levels

*Static View on C / C++*

axion
stopping software erosion



- Requirement: ASIL Code shall not call *(synchronous)* QM functionality
- Axivion helps to ensure Freedom from Interference
  - identifies calls from ASIL code into QM code or into code with lower criticality

# Confidence in the use of software tools

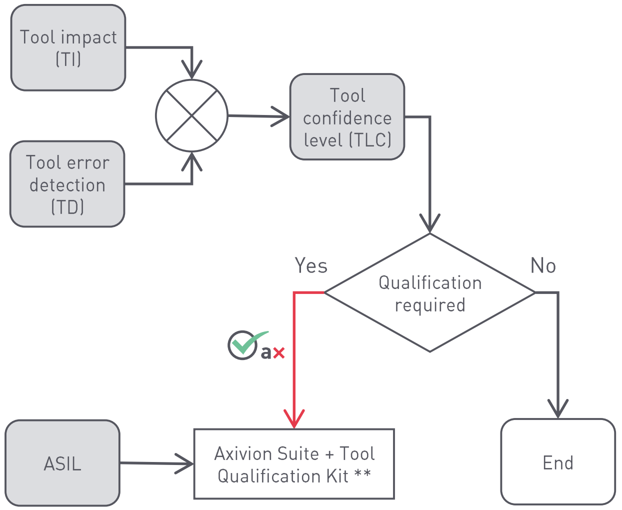
axivion
stopping software erosion

- Axivion integrates in almost any CI workflow
- Axivion's *Tool Qualification Kit* ensures confidence of usage in customers environment
- Depending on your use case any TCL level could be achieved
- *Tool Qualification Kit* becomes part of your CI process
  - managed qualification

| Methods | | | ASIL | | | |
|---|---|---|---|---|---|---|
| | | | A | B | C | D |
| 1a | Increased confidence from use in accordance with 11.4.7 | | ++ | ++ | ++ | + |
| 1e | Evaluation of the tool development process in accordance with 11.4.8 | ✓ax | ++ | ++ | ++ | + |
| 1f | Validation of the software tool in accordance with 11.4.9 | ✓ax | + | + | + | ++ |
| 1b | Development in accordance with a safety standard* | | + | + | + | + |

* No safety standard is fully applicable to the development of software tools. Instead, a relevant subset of requirements of the safety standard can be selected.

Example: Development of the software tool in accordance with ISO 26262, IEC 61508, EN 50128 or RTCA DO-178C


Tool impact (TI)
Tool error detection (TD)
Tool confidence level (TLC)
Qualification required
Yes
No
ax
ASIL
Axivion Suite + Tool Qualification Kit **
End

** Quality tool with appropriate degree of rigour