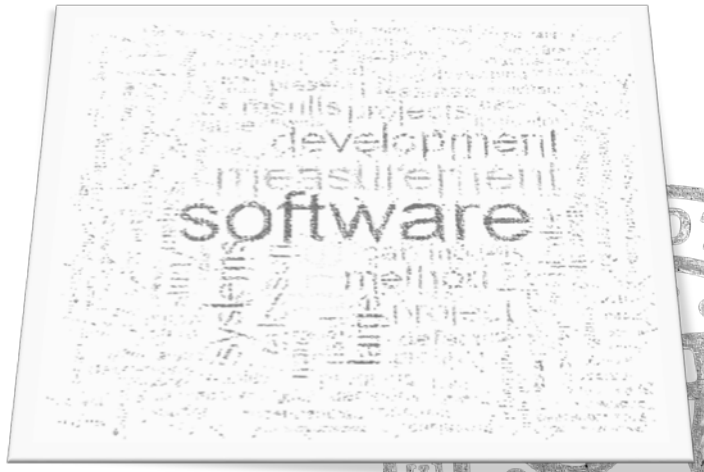
GÖTEBORGS UNIVERSITET

SOFTWARE COMPLEXITY METRICS IN GENERAL AND IN THE CONTEXT OF ISO 26262 SOFTWARE VERIFICATION REQUIREMENTS

MIROSLAW STARON, ASSOCIATE PROFESSOR, SOFTWARE ENGINEERING
CHALMERS | UNIVERSITY OF GOTHENBURG



VINNOVA

VOLVO

Software Center

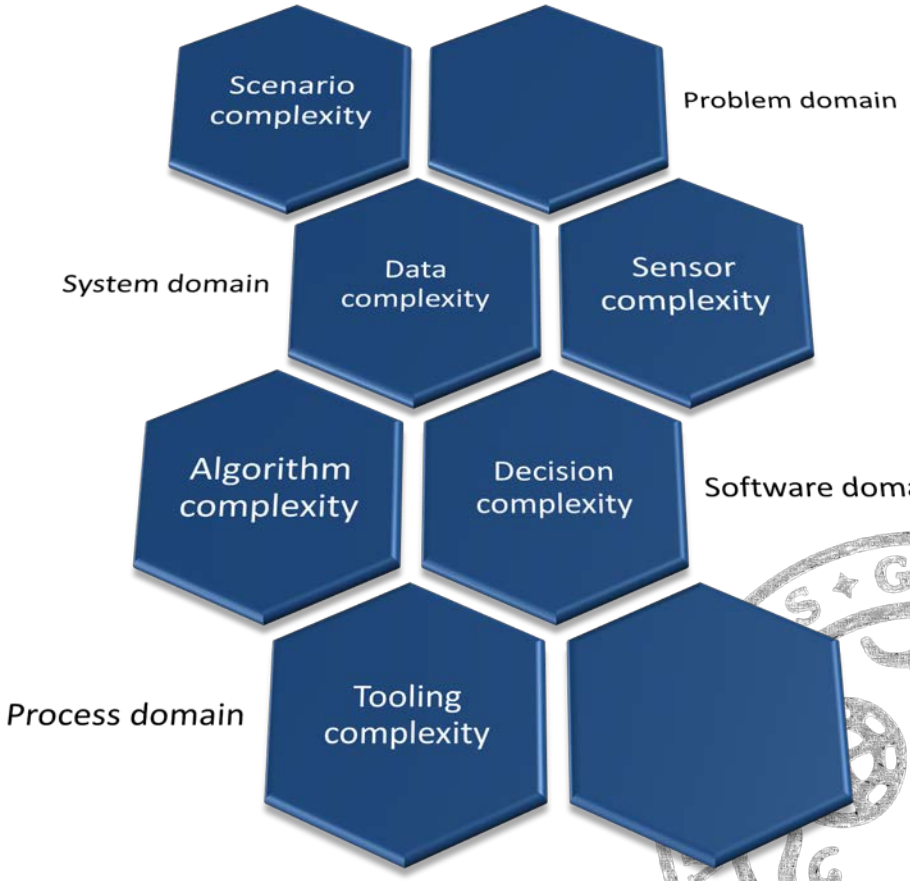
Motivation for our research – safe cars

- The number of functions that are software steered grows as well
  - Autonomous driving >> 50 pure software functions
- Exponential growth of vehicle's software size
  - The number of ECUs grows exponentially (2 ECUs in 1970 to over 130 in 2016)
  - The amount of software grows exponentially
  -
- **We face new challenges**
  - **How to verify and validate all the software?**
  - **How to increase sw dev. speed if the sw. complexity grows?**

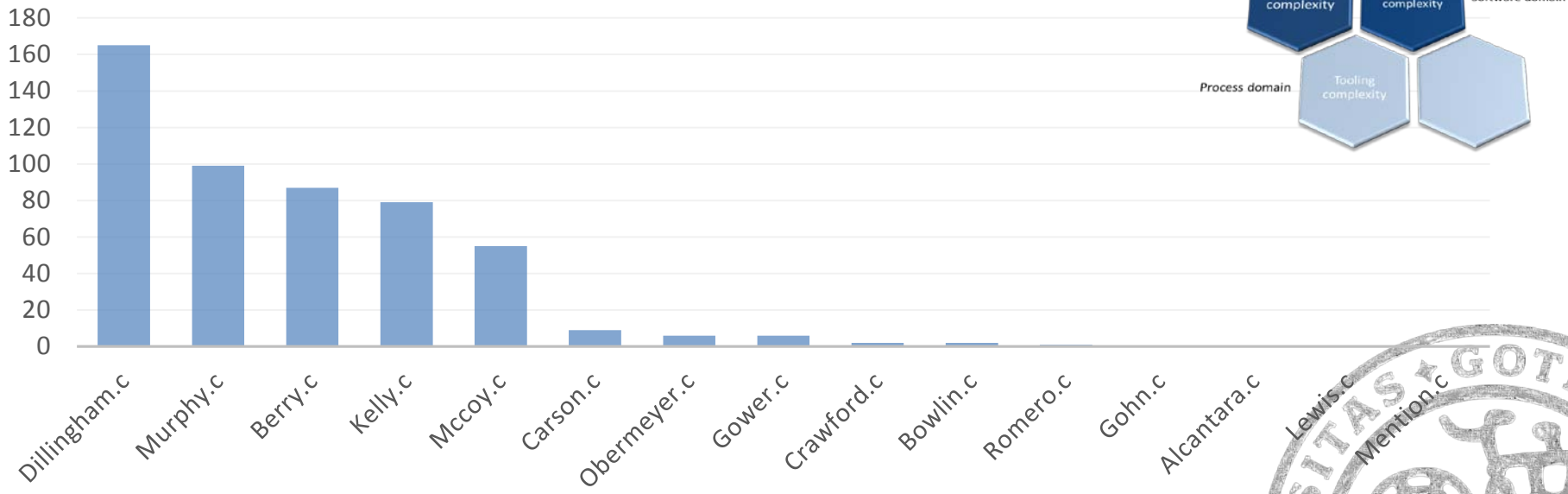
Complexity in the software of modern cars

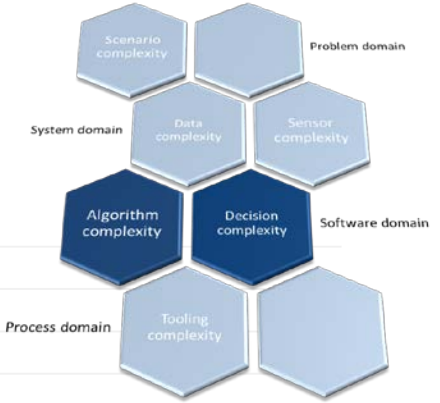
- Software complexity
  - The degree of connectivity between entities in a program

- Metrics (examples)
  - Cyclomatic complexity metric (McCabe)
  - Software science metrics (Halstead)
  - Software Structure Metrics (Henry and Kafura)
  - Metrics Suite for Object Oriented Design (Chidamber and Kamerer)
  - Branching complexity (Sneed)
  - Data access complexity (Card)
  - Data complexity (Chapin)
  - Data flow complexity (Elshof)
  - Decisional complexity (McClure)



Complexity of decision algorithms in practice





Altinger, H., Siegl, S., Dajsuren, Y., & Wotawa, F. (2015, May). A novel industry grade dataset for fault prediction based on model-driven developed automotive embedded software. In 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories (MSR), pp. 494-497, IEEE Computer Society Press.

GÖTEBORGS UNIVERSITET

Overview of V&V requirements from ISO 26262

Software design and implementation

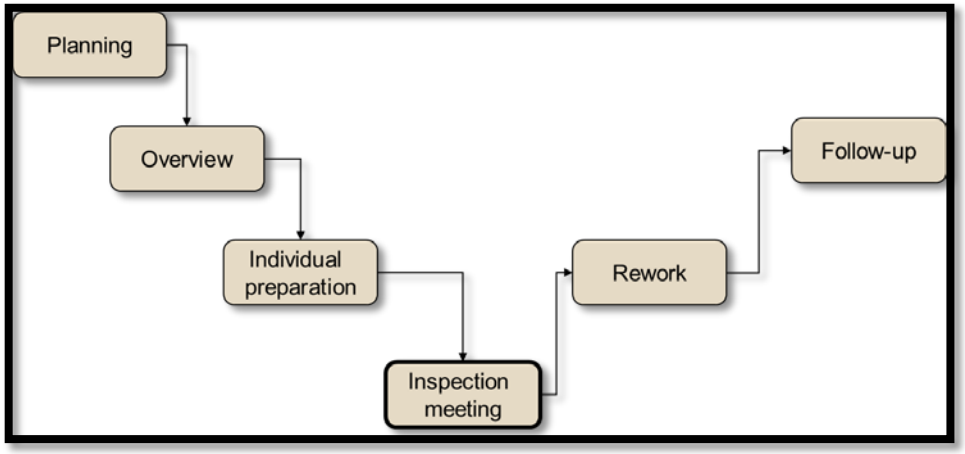
- Walkthrough
- Inspection
- Semi-formal verification
- Control-flow analysis
- Data-flow analysis
- **Static code analysis**
- Semantic code analysis

GÖTEBORGS UNIVERSITET

Overview of V&V requirements from ISO 26262

Software design and implementation

- Walkthrough
- **Inspection**
- Semi-formal verification
- Control-flow analysis
- Data-flow analysis
- Static code analysis
- Semantic code analysis



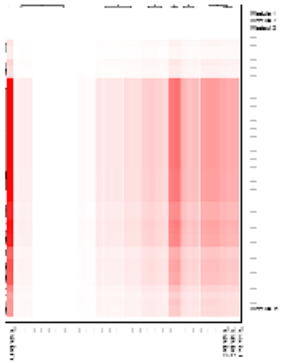
- Efficiency
  - 125 source statement/hour during individual preparation
  - 90-125 statements/hour can be inspected during inspection meeting
- Inspection is therefore an expensive process
  - Inspecting 500 lines costs about 40 man/hours effort – about €2000

The big questions are

- When will we stop being able to secure the safety of the software?
- How can we release software that we cannot safety-assure?
- When will we experience ”emergent” behaviour caused by the lack of control over complexity?

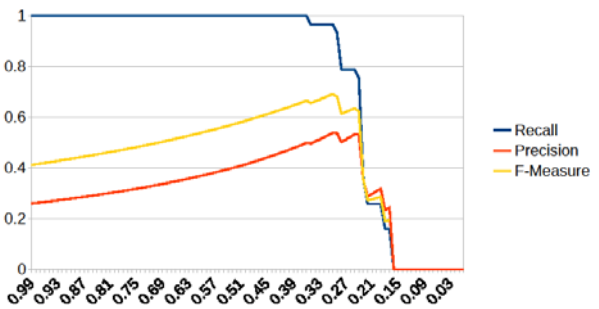
Q1: When will we stop being able to secure the safety of the software?

- Motivating case: Software testing



- 1 execution/control path >= 1 test case

- Modern software > 1000 execution/control paths per module!
- Modern software >> 1000 modules



Knauss, Eric, Miroslaw Staron, Wilhelm Meding, Ola Söder, Agneta Nilsson, and Magnus Castell. "Supporting continuous integration by code-churn based test selection." In Proceedings of the Second International Workshop on Rapid Continuous Software Engineering, pp. 19-25. IEEE Press, 2015.

Q2: How can we release software that we cannot safety-assure?

- Motivating case: Tesla's autopilot

- *A handful of times I instinctively grabbed the wheel or hit the brakes when a few impatient New York drivers cut me off, not really sure if the car would figure out what was happening. I'm sure the car would have, but I didn't want to be responsible for crunching up a $120,000 car I didn't own. Only once did the car ask me to retake control, ostensibly because it couldn't read the nearly nonexistent lane markings.*
  - *Chris Perkins, Mashable.com, about driving in Manhattan*


UNIVERSITAS GRONINGENSIS