A Preliminary Evaluation of the Benefits of AOSD

Tom Tourwé
CWI/SEN & VUB/PROG
EvA Symposium and Contact Day

in collaboration with
Magiel Bruntink &
Arie van Deursen

Overview

- Context
- Problem Statement
- First Experiment
  - Verifying & Reasoning about CCC
- Conclusion & Future Work

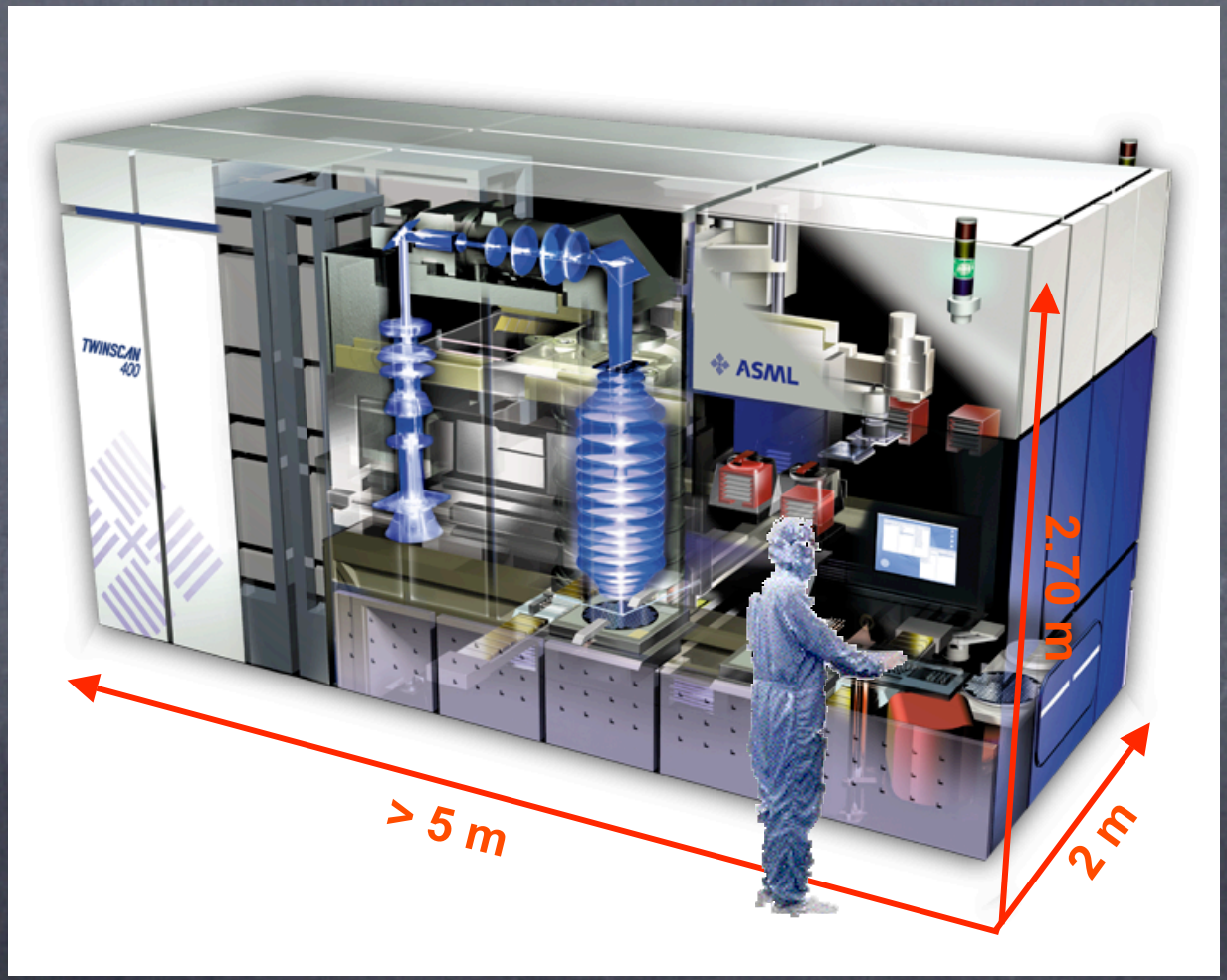
Overview

- Context
- Problem Statement
- First Experiment
  - Verifying & Reasoning about CCC
- Conclusion & Future Work

The Ideals Project

- Ideals = Idiom Design for Embedded Applications on Large Scale
- Participants
  - ASML, Embedded Systems Institute (ESI), Universiteit Twente (UT), Universiteit Eindhoven (TU/e), Centrum voor Wiskunde & Informatica (CWI)

ASML Wafer Scanners



Goal

Improve the handling of cross-cutting concerns in embedded software systems

in order to

- reduce the design effort, errors & code size
- improve consistency, maintainability, evolvability and developer productivity

Context

- Approx. 12 MLoc C code
- Decomposed into components, modules, functions, ...
- Idiomatic approach to implement cross-cutting concerns
  - Coding conventions & idioms

Overview

- Context
- Problem Statement
- First Experiment
  - Verifying & Reasoning about CCC
- Conclusion & Future Work

Problem Statement

Disproportional amount of time spent on implementing "boiler plate" code

- Code scattering
  - Part of the code of each function is nearly the same
- Code tangling
  - Each function addresses multiple concerns

Concern code "obstructs" functional code

Problem Statement

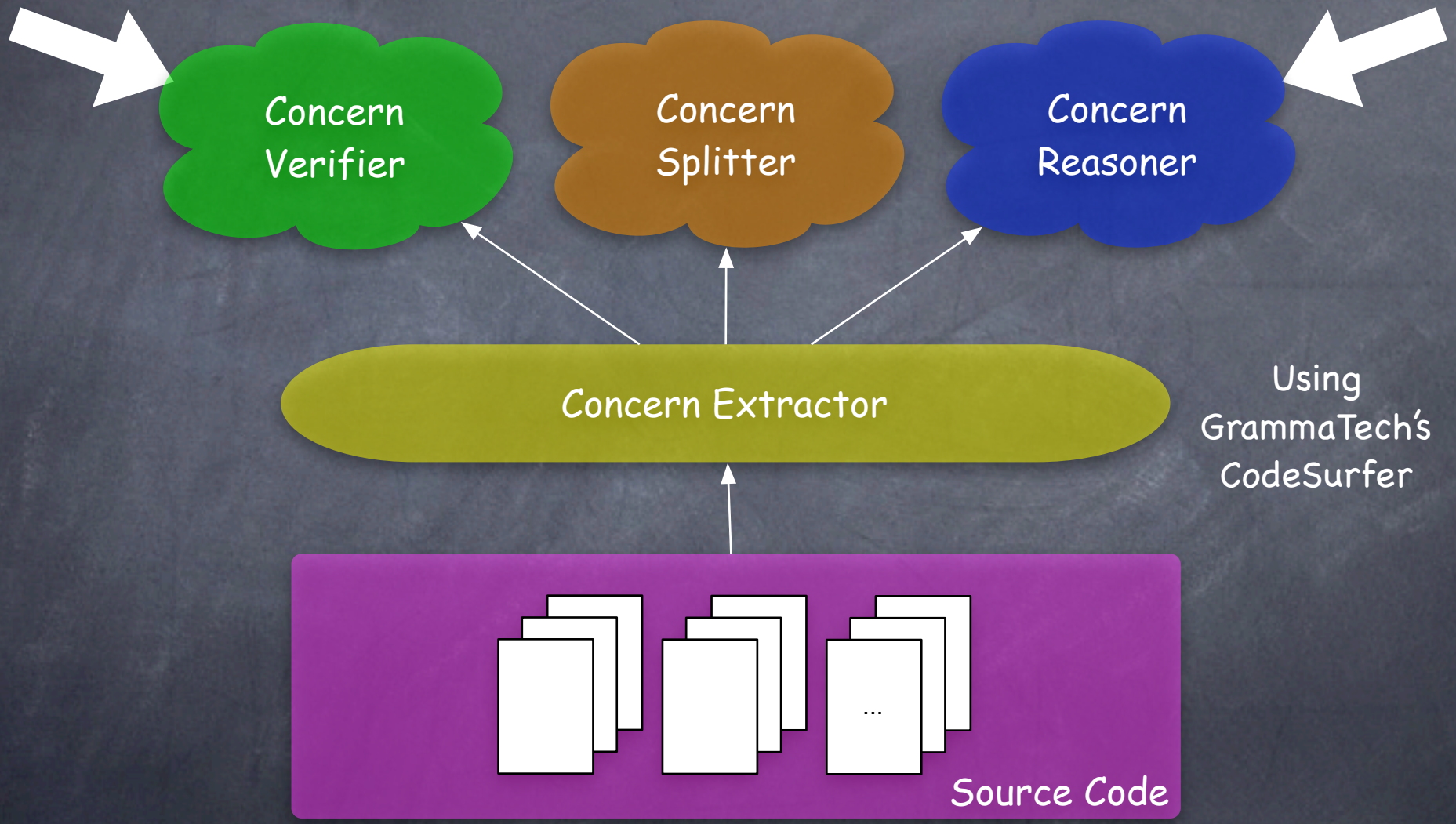
Concern	LoC	Fraction
Error Handling	1716	9%
Dynamic Execution Tracing	1539	8%
Parameter Checking	1441	7%
Memory Allocation Handling	1110	6%
Total	5806	31%

Concerns in a 19 KLoc component

Overview

- Context
- Problem Statement
- First Experiment
  - Verifying & Reasoning about CCC
- Conclusion & Future Work

Handling CCC



An Example Concern

All parameters should be checked

- input & output parameters?
- how should they be checked?
- where should they be checked?

Concern Verification

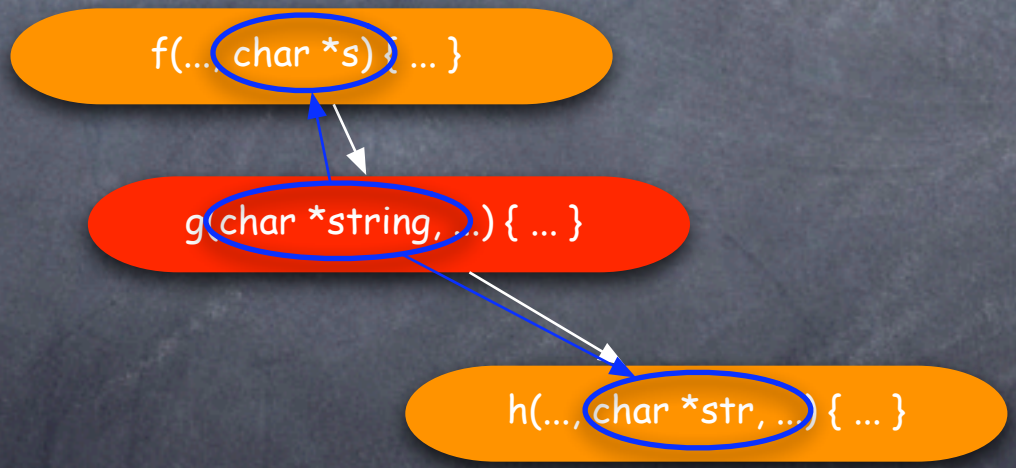
Evaluate correctness & consistency

- Input parameters should always contain a value
- Output parameters should never contain a value, but pointer should not point to NULL
- Should be checked "whenever appropriate"

Concern Verification

Evaluate correctness & consistency

Take into account data dependencies between callers/callees



Concern Verification

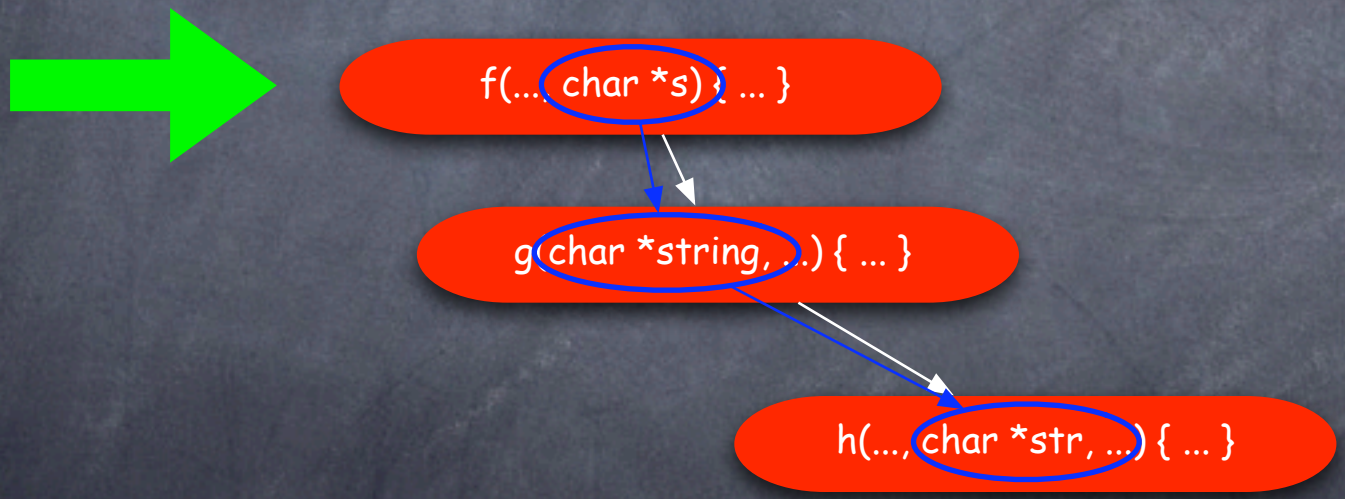
Evaluate correctness & consistency

Concern	Not Checked
input parameter != NULL	65%
output parameter != NULL	21,5%
*output parameter == NULL	65,5%

Concern Reasoning

Detect improvements

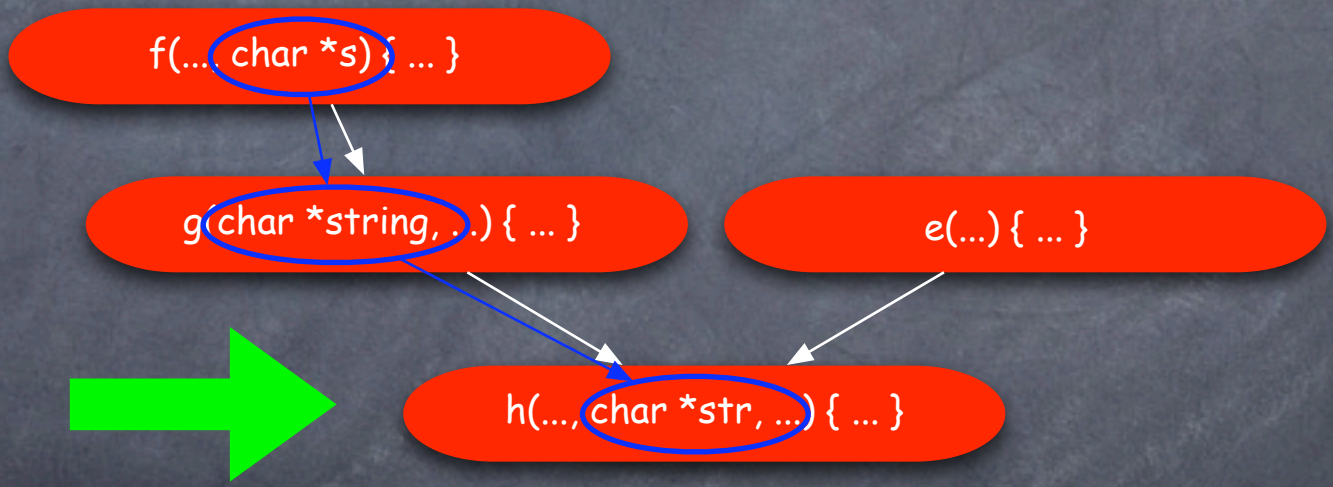
Where should additional parameter checks be inserted?



Concern Reasoning

Detect improvements

but ...

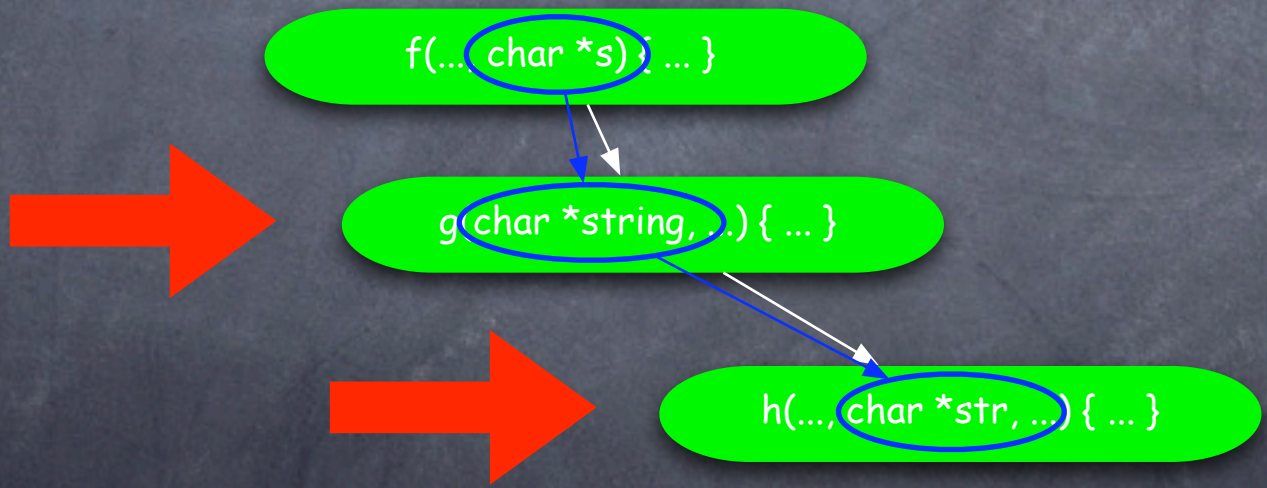


support for evolution!

Concern Reasoning

Detect improvements

Where should redundant parameter checks be removed?



Concern Reasoning

Detect improvements

Where should redundant parameter checks be removed?

Concern	Reduced checks
input parameter != NULL	27%
output parameter != NULL	8%
*output parameter == NULL	7%

Overview

- Context
- Problem Statement
- First Experiment
  - Verifying & Reasoning about CCC
- Conclusion & Future Work

Conclusion

- AOSD benefits in this context ...
  - Reduced code size, improved consistency, improved performance, improved evolvability a.o.
- ... but
  - some concerns more difficult to separate into aspects
  - requires domain-specific aspect languages

Future Work

- Consider other 3 concerns
- Design domain-specific aspect language(s)
- Implement concern splitter
  - Extract code & define corresponding aspect
- Verify results for other components