What Goes Wrong With Software Development And Why?*

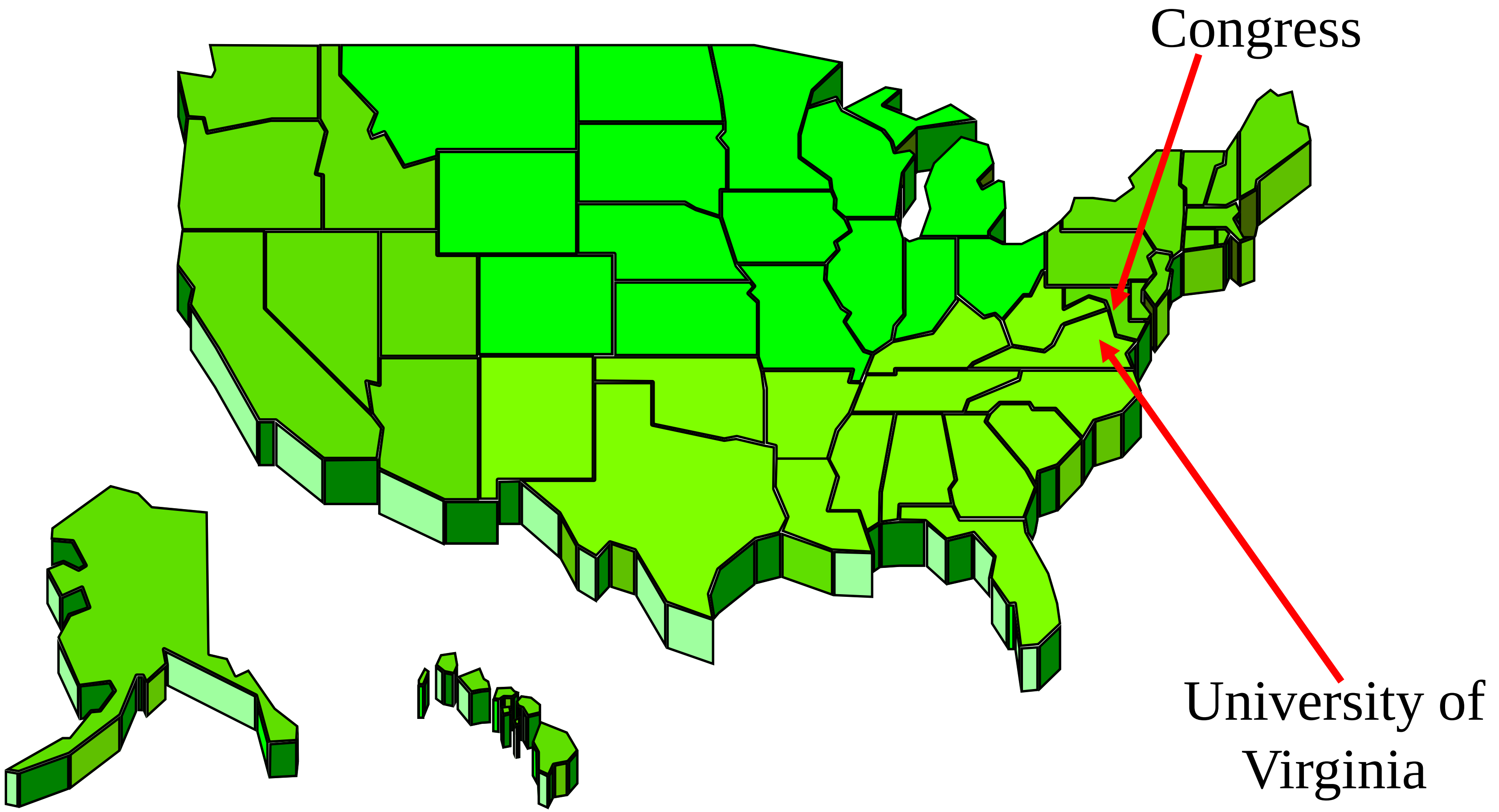
John C. Knight

Department of Computer Science
University of Virginia

November 10, 2011

***Supported in part by: National Science Foundation, AdaCore**

Where Am I From?



Software In Operation
A Mixed Record

London Ambulance Service, 1992

Ariane 5, 1

Korean Air

Mars Polar Lander, 1999

Boeing

Airbus A330, October 2008

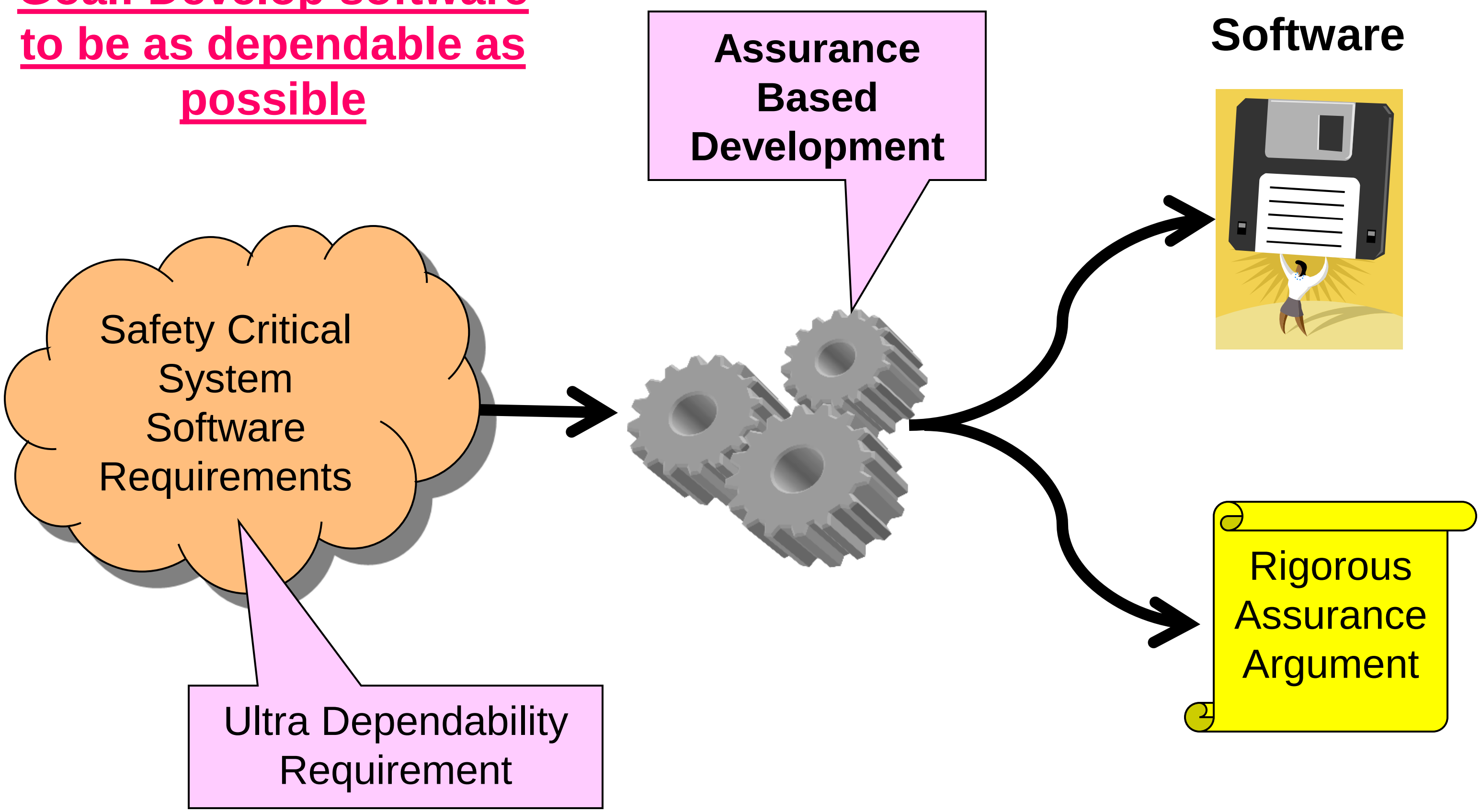
What Is The Best We Could Do?

- Many accidents and incidents have had software as a causative factor
- ***Why*** is software imperfect?
- Would “better” development and analysis techniques help?
- Is software somehow ***inherently*** less dependable than we would like?
- Where should we look for issues to address in certification?
- Let’s not speculate,

Let’s do an experiment (case study) and see what we can find out

Design of the Case Study – 1

Goal: Develop software to be as dependable as possible



Rigorous Assurance Argument

- Informally, basis of rigorous argument is:

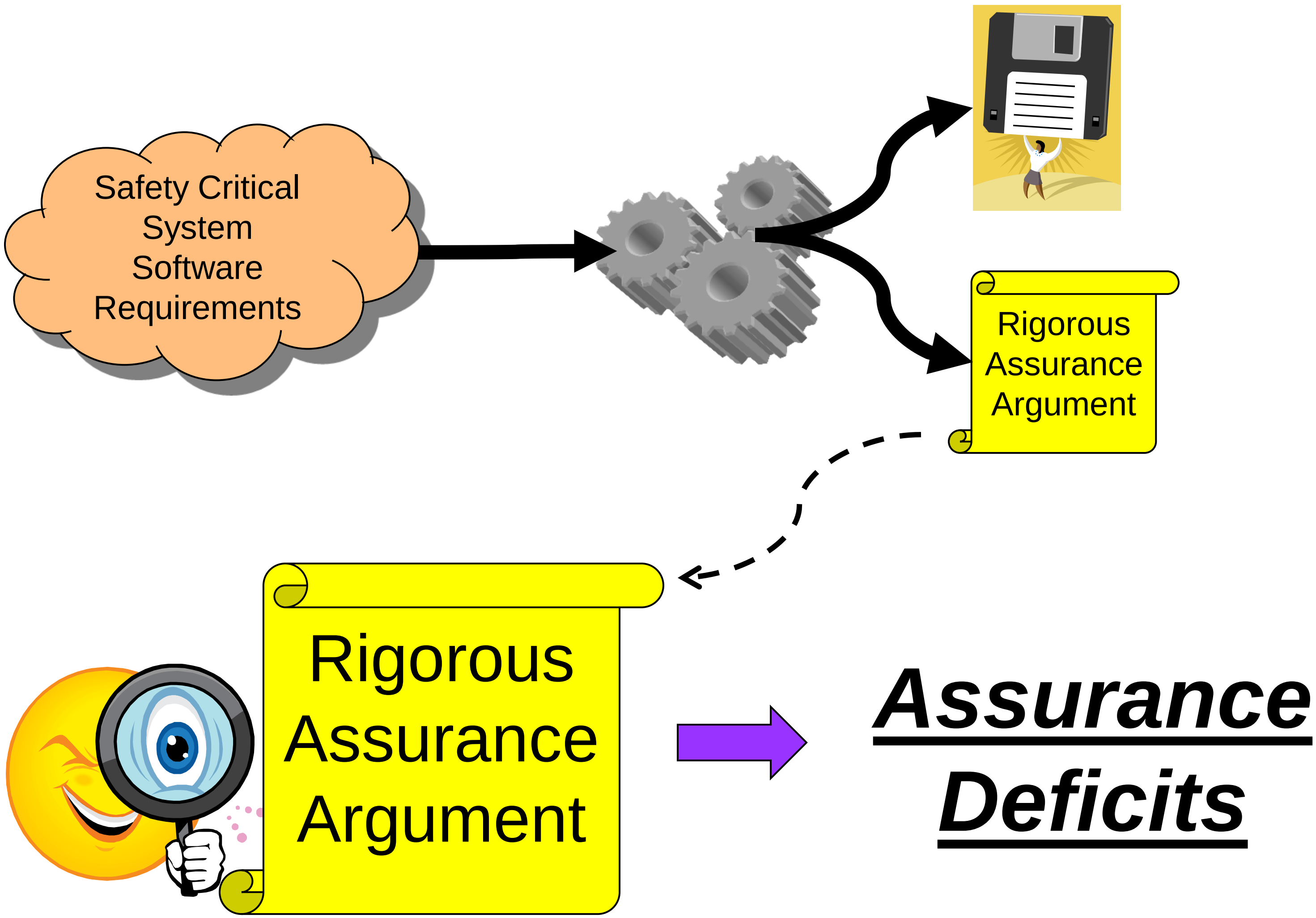
Systematically document rationale for belief in assurance claim

- Assurance deficits:

Aspects of the argument where doubt remains

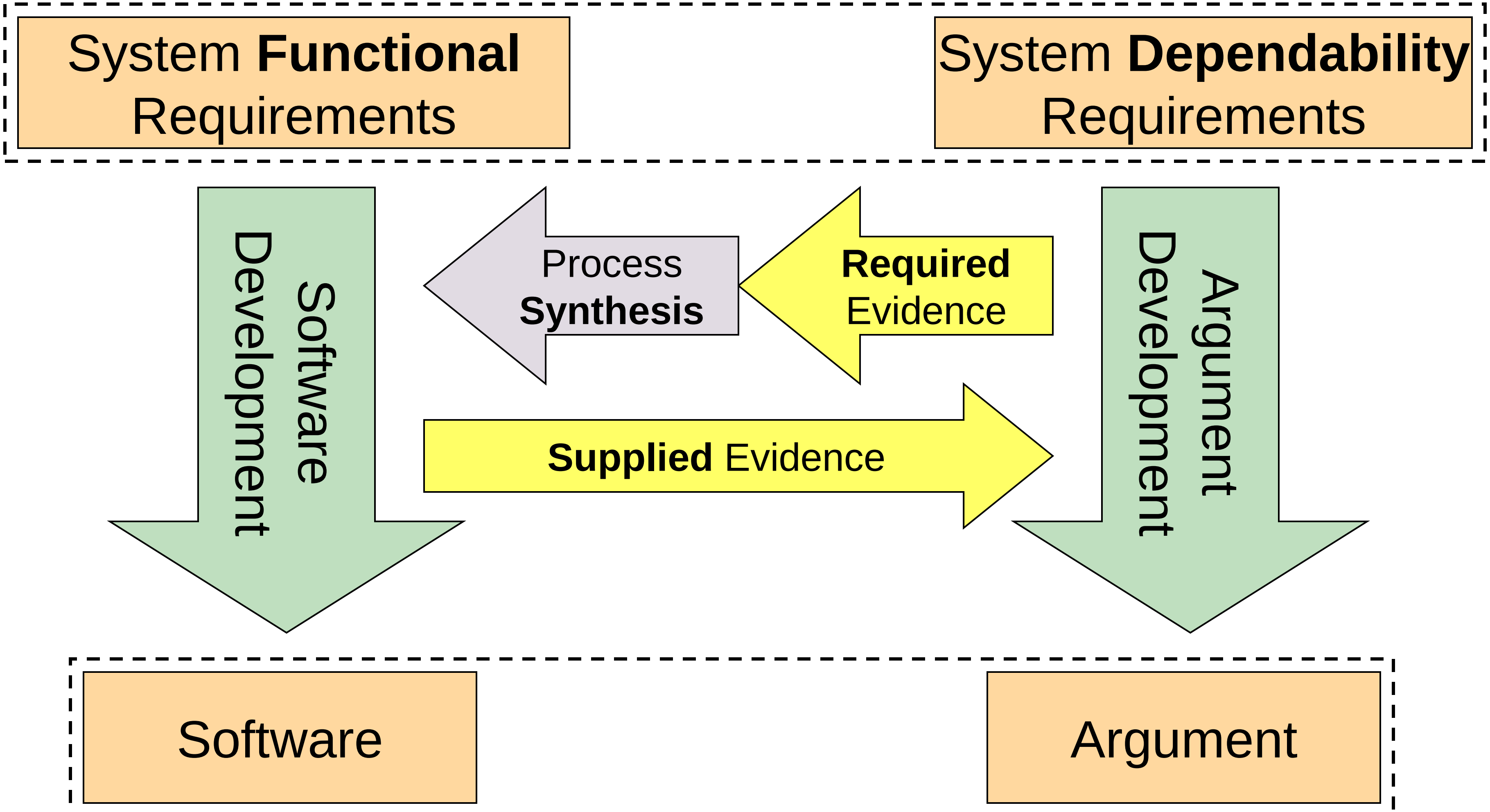
- Analyze ***argument*** to determine how well we achieved our goal

Design of the Case Study – 2

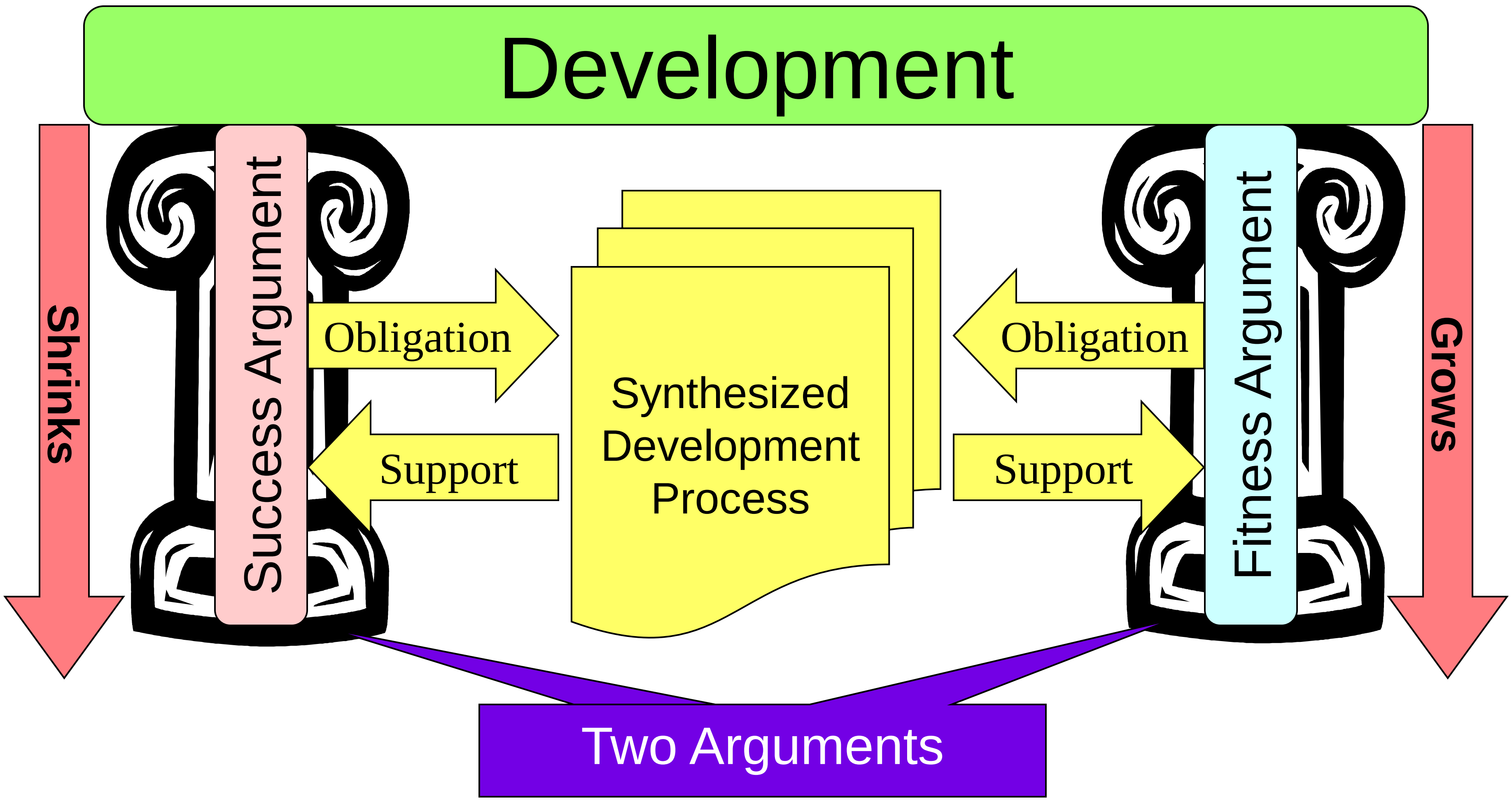


Assurance Based Development

The Principle



Assurance Based Development



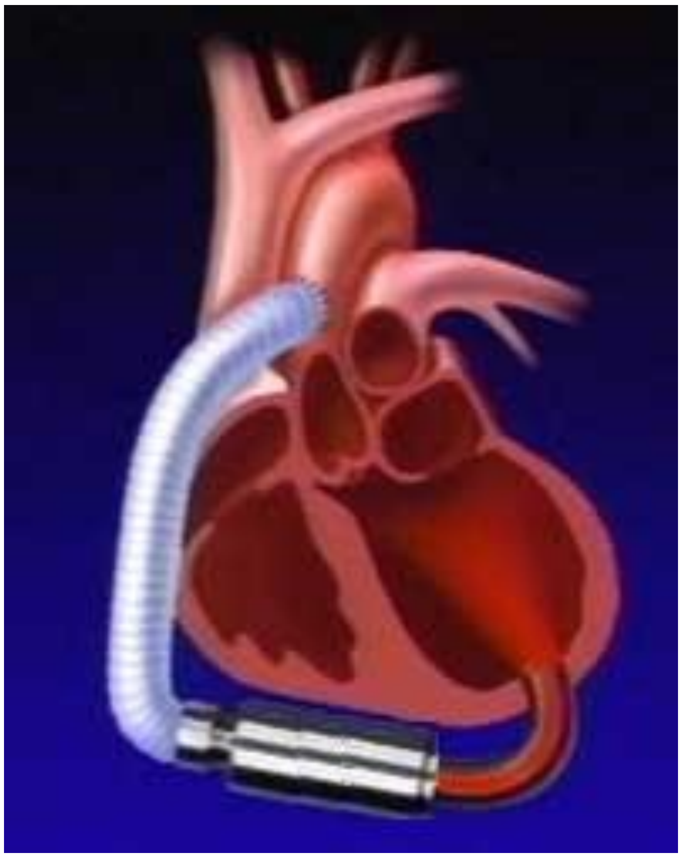
Case Study

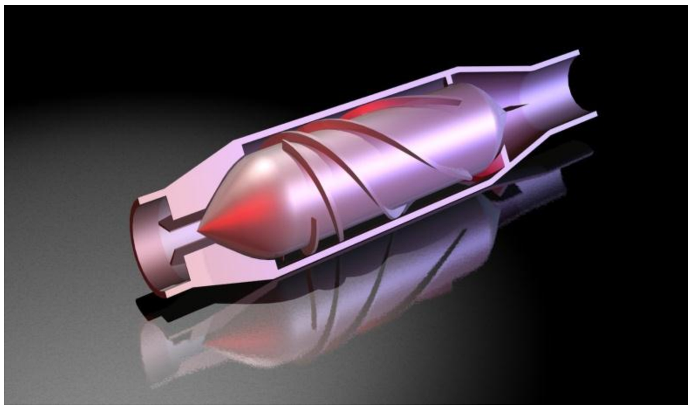
Target: Left Ventricular Assist Device

(Joint work with Departments of Mechanical & Aerospace Engineering and Electrical & Computer Engineering)

Example: LVAD

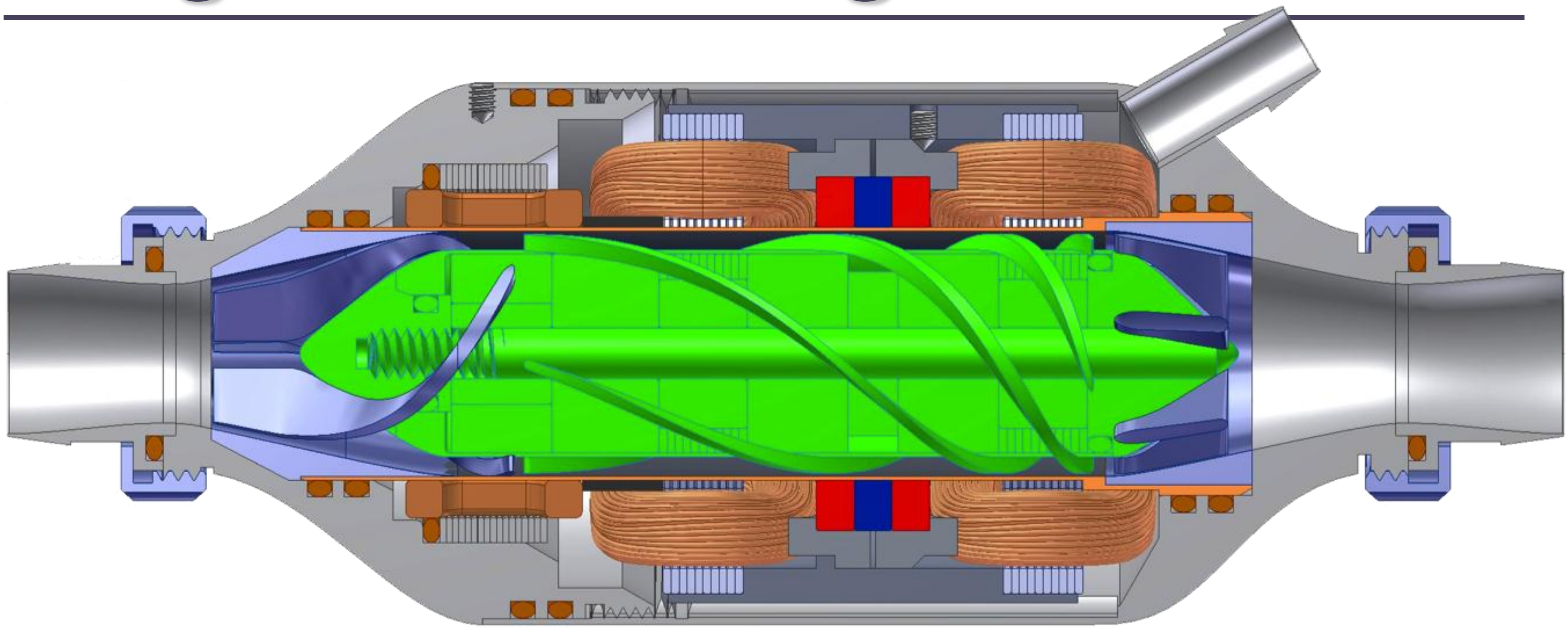
- Left Ventricular Assist Device





- **Magnetic bearings**
- Continuous-flow axial design
- Less blood damage than current models

Magnetic Bearing Control



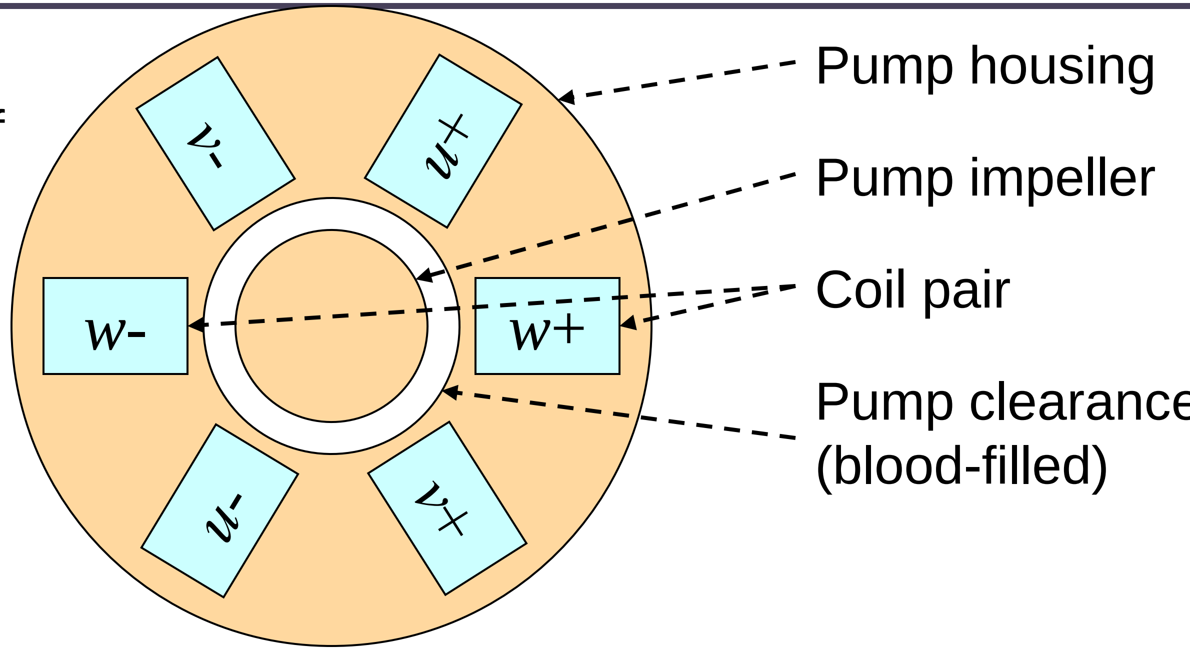
- Compute control updates in hard-real-time (5 kHz)
  - State-space control model, 16 states
- **No more than 10^{-9} failures per hour of operation**

Active Mag Bearing Controller

Magnetic bearing controller is part of larger LVAD system.

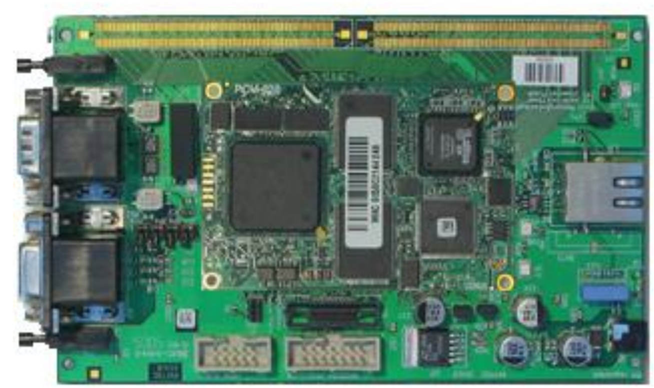
LVAD's goal: adequately support patient's circulation.

Some responsibility falls on magnetic bearings.



Target:

Freescale MPC5554
+ custom DACs
No system software



LVAD System Requirements

Functionality

1. Trigger and read Analog-to-Digital Converters (ADCs) to obtain impeller position vector *u*.
2. Determine whether **reconfiguration** is necessary. If so, select appropriate gain matrices **A**, **B**, **D**, and **E**. (**reconfiguration to cope with coil failure**)
3. Compute target coil current vector *y* and next controller state vector *x*:

   $y_k = \mathbf{D} \times x_k + \mathbf{E} \times u_k$

   $x_{k+1} = \mathbf{A} \times x_k + \mathbf{B} \times u_k$
4. Update DACs to output *y* to coil controller.

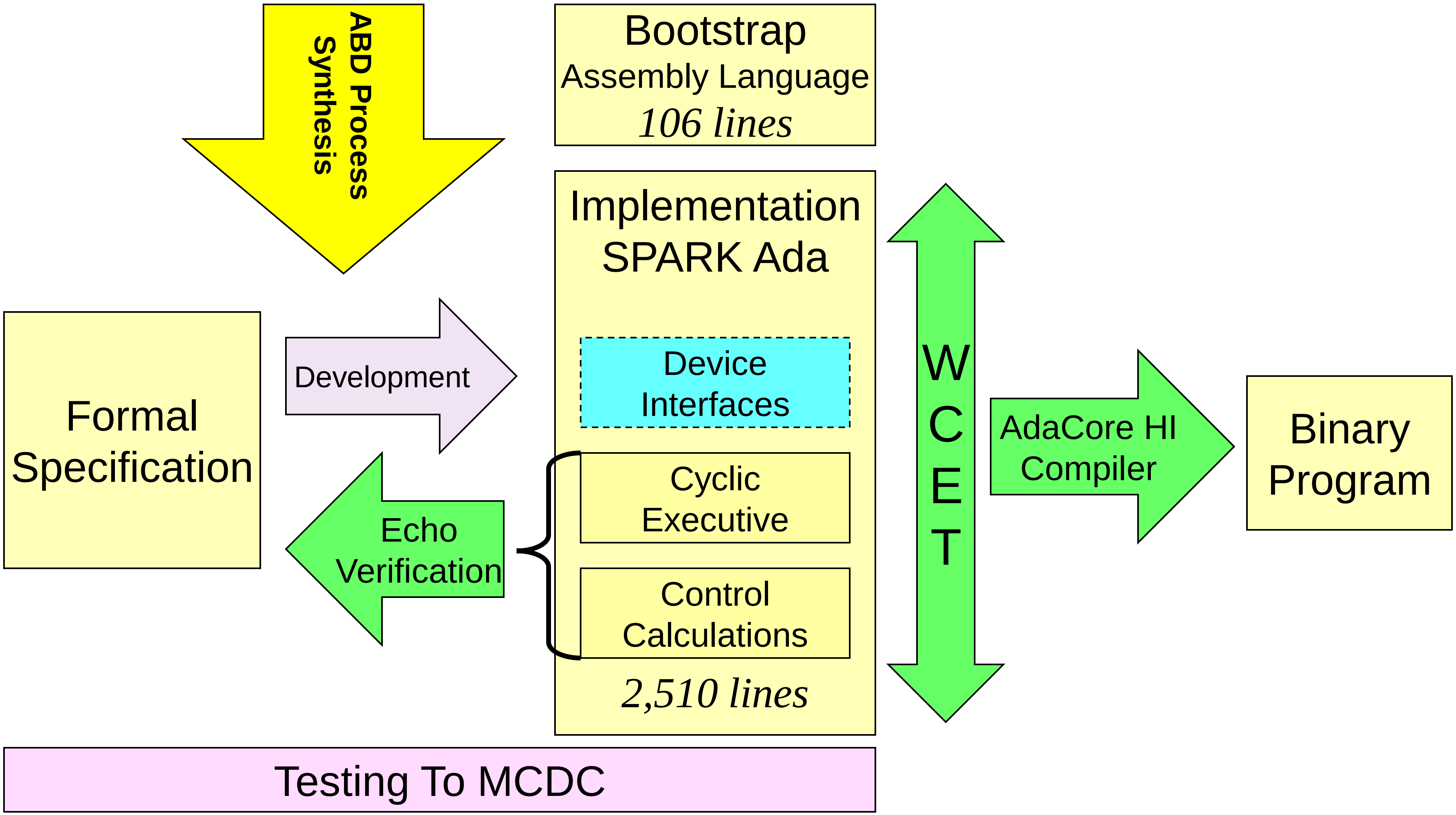
Timing

Execute control in hard-real-time with a frame rate of 5 kHz.

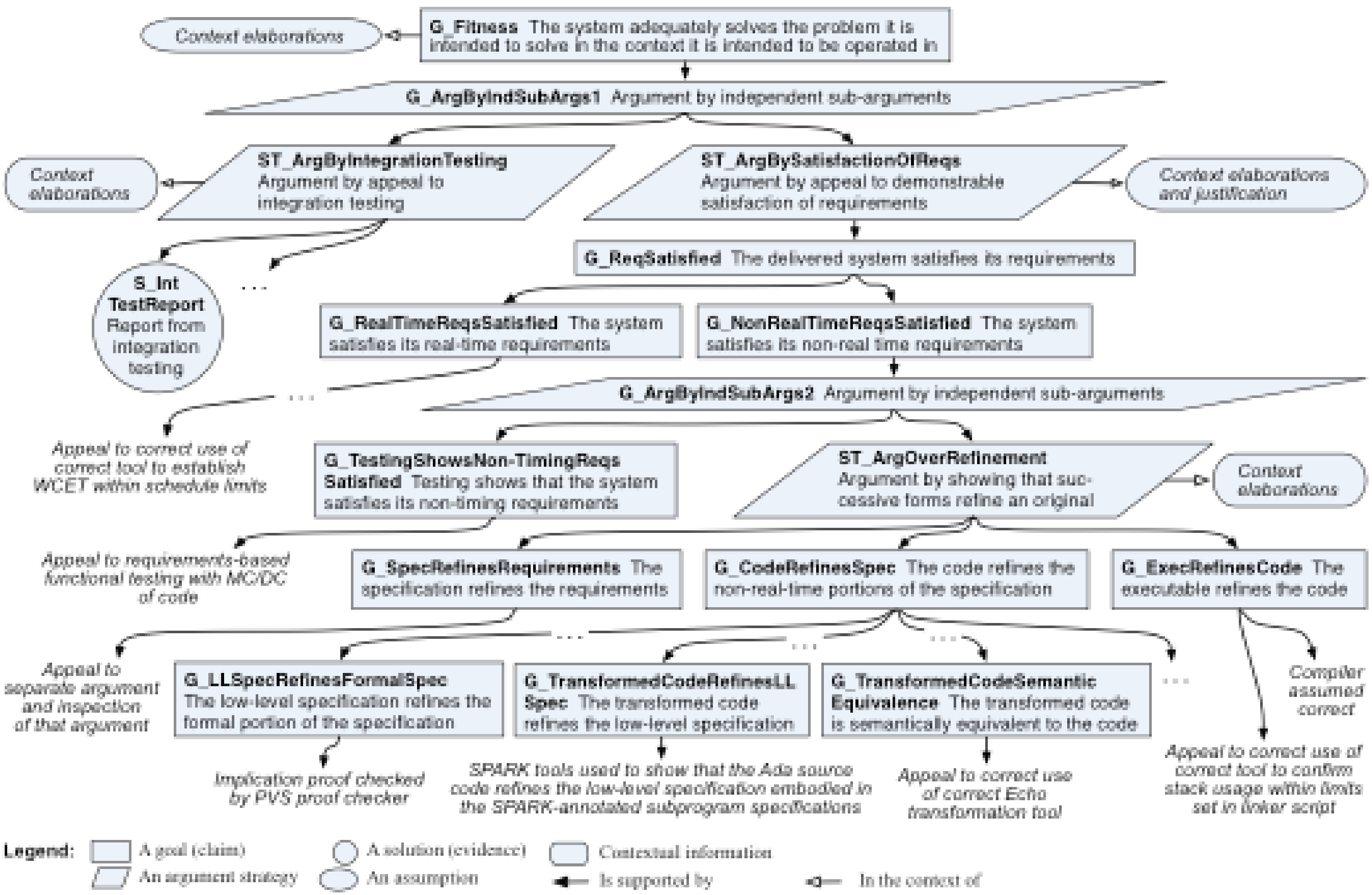
Reliability

No more than 10^{-9} failures per hour of operation.

Overall Development Process



Fitness Argument *Fragment*



Assurance Deficits

- Reliance upon:
  - Correct requirements
  - Reliable human-to-human communication
  - Understanding the semantics of formalisms
  - Reviews or inspections
  - Human compliance with protocols
  - Unqualified tools
  - Tools that lack complete hardware models
  - Testing
  - Human assessment of dependability
- The unavoidable use of low-level code
- The ability to verify floating-point arithmetic

Human-To-Human Communication

- Problem:
  - Communication of technical concepts from one individual to another
    - Systems to software engineer, medical professionals, etc.
  - Those involved ***frequently unaware of the error***
- MBCS manifestations:
  - Use of documents in English
- Potential mitigations:
  - Formal languages
  - Rigorous use of natural language (CLEAR method)

Verification of Floating Point

- Problem:
  - Comprehensive formal verification unavailable
- MBCS manifestations:
  - Control equations fundamentally computational
  - Verification using SPARK Ada tools assuming real arithmetic in bounded range
- Potential mitigations:
  - Avoid problem areas such as tests for equality
  - Switch to fixed point
  - Fund more research

Unqualified Tools

- Tools included:
  - SPARK Ada tools
  - Commercial WCET analysis tools
  - AdaCore high integrity Ada compiler
  - (Echo verification tools)
  - Assembler
  - PVS
  - Etc.
- How trustworthy?
- How would assurance in tools be established?

Incomplete Hardware Models

- Freescale MPC5554:
  - Powerful processor for embedded applications
  - Based on Power PC
  - Many additional “features” (A/D, timers, coprocessors)
- Processor configuration required
- But no formal semantics of processor extensions:
  - Natural language definitions and best-effort engineering
  - Significant opportunity for research:
    - Complex logic
    - Complex interactions

Use of Low-Level Code

- Problem:
  - Direct access to hardware
  - Setting processor states & controlling peripherals
- MBCS manifestations:
  - Freescale MPC5554 processor control registers
  - PowerPC assembly language with no verification technology
- Potential mitigations:
  - Human inspection
  - Testing
  - Tool development and integration

Conclusion

- Assurance of dependability is crucial:
  - We need to “know” that the system will operate properly
- Case study used the best software technology that we could think of
- Assurance deficits were many and subtle:
  - Many were expected, some were not
  - Complete list is surprising
- In practice, need to:
  - Search for sources of assurance deficit
  - Add additional vigilance – be on our guard!

Contact

- E-mail address:

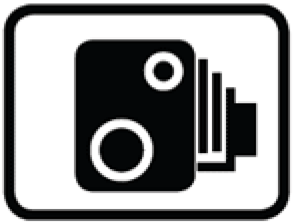
  knight@cs.virginia.edu

- For more information see:

  http://www.cs.virginia.edu/knight/

  http://dependability.cs.virginia.edu/

Questions?



U.K.

U.S.