Software Process Synthesis in Assurance Based Development of Dependable Systems

Patrick J. Graydon
John C. Knight

University of Virginia
Certification By Safety Argument

System Requirements and Goals

Goals

Does the evidence imply the goals?

Rigorous Argument

Evidence

Standard Development Process

“I believe that the evidence I have obtained justifies the claim that the goals are met”
Can We Exploit Argument?

Argument tells us what is important

Mechanism goal: generate the necessary artifacts (including *software*) and evidence

Artifacts provide services and evidence

Is there a mechanism?
### Turning the Knobs to Maximum

- Extreme Programming “turned up the knobs”
- ABD turns up the knobs on rational development

<table>
<thead>
<tr>
<th>Good</th>
<th>➡️</th>
<th>Better</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety arguments</td>
<td>➡️</td>
<td>Broader argument</td>
</tr>
<tr>
<td>“Early and often”</td>
<td>➡️</td>
<td>Continuous argument update</td>
</tr>
<tr>
<td>Rational choices</td>
<td>➡️</td>
<td>A completely rational process</td>
</tr>
</tbody>
</table>
Comprehensive Argument

• Problem:
  – Argument must address all goals of all stakeholders

• There will be tradeoffs
  – Must appear explicitly in the argument
    • Must be addressed
    • Seen to be addressed
    • Impact of each must be clear

• Without this, consequences could be severe

• As a driver, safety argument does not do this
Software That Is Fit For Use

- Not just safety, but *fitness for purpose*:
  - Adequate safety, *and*
  - Adequate security, *and*
  - Desired functionality, *and*
  - *Everything else*

**Main fitness claim:**

*The system is adequately fit for use in the context(s) in which it will be operated.*
The Merit Of Envelope Protection

“Darn, the wings *did* fall off.”
• The *fitness argument* captures data about the *product*
• The *success argument* captures data about the *process*

• Development schedule, *and*
• Budget and other resource constraints, *and*
• Pragmatic development constraints
Assurance Based Development

Development

Success Argument

Obligation

Support

Synthesized Development Process

Fitness Argument

Obligation

Support

Shrinks

Grows

Software Process Synthesis in Assurance Based Development of Dependable Systems
**Process Synthesis: Select Goals to Address**

- **First step:** select a goal or goals to address

  - Fitness argument
  - Success argument

  - Assurance obligations
  - Options

  - Pattern library and other literature
  - Experience (both personal and that of colleagues)
  - Option 1
  - Option 2
  - ... Option n

- Area of expertise
- Perceived risk of infeasibility
- Minimize interdependency
Process Synthesis: Gather Options

- Second step: gather options that meet the selected obligations

• Gathering options takes time
• Must balance effort spent versus perceived risk of making a poor choice

Fitness argument

Pattern library and other literature

Success argument

Experience (both personal and that of colleagues)

Assurance obligations

Assurance obligations

Options

Options

Options

Option 1

Option 2

...

Option n
Example Options

• Obligations:
  – **Success**: Timing goals can be met demonstrably
  – **Fitness**: Real-time requirements met

• Options considered:
  – Analyze WCET using a tool to be chosen later
  – Utilize a watchdog timer to re-issue last frame’s control outputs if deadline would be missed
Process Synthesis: Evaluating Options

Consider:
- Functionality
- Restrictions on later choices

- Cost
- Feasibility
- Applicable standards
- Non-functional requirements
Evaluation of Example Option

• Use WCET analysis
  – Would supply strong evidence that the hard real-time deadlines would be met

• Re-issue last frame’s control outputs if late
  – Unacceptable — how would we demonstrate:
    • re-issuing the last frame’s outputs would be rare
    • doing so rarely would keep the system safe
Process Synthesis: *Recording A Choice*

If a choice is expected to produce evidence, the evidence added to the argument is marked as forthcoming (e.g. using a diamond in Goal Structuring Notation)

Fragments associated with chosen option

Process elements

Arg. fragment(s)

Planned process description

Updated success argument

Updated fitness argument

Evaluation
Process Execution

Confirmation or exception

Process synthesis mechanism

Process description

Process execution

Experience

Artefacts

Evidence
A Case Study of ABD

• Controlled, replicated experiment impractical
  – Instead: conduct carefully a case study
• Look for evidence of four kinds of problems:
  – Infeasibility: developers might be incapable; additional effort might be infeasible
  – Obligations are not appropriate choice drivers
  – Can’t judge options by argument impact alone
  – No decision criteria is missing or superfluous
• Study protocol includes questionnaire
  – 23 questions answered per synthesis step
The UVA LifeFlow LVAD

- **Left Ventricular Assist Device**
- Continuous-flow axial design
- Magnetic bearings
- 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
Resulting Synthesized Process

- **Formal specification** in PVS
- Design a cyclic executive to manage the real-time tasks
- Design bearing control task routines
- **Implement** MBCS in **SPARK Ada** (2,510 lines)
- Implement bootstrap (106 assembly instructions)
- Use AdaCore's GNAT **Pro High-Integrity Edition** compiler
- **Formally verify the implementation**
  - Used Echo approach, PVS, and SPARK Tools
- Analyze Worst-Case Execution Time (**WCET**) and stack usage
- Requirements-based **functional testing** to Modified Condition / Decision Coverage (MC/DC) (not completed)
Fitness argument

Context elaborations

G_Fitness The system adequately solves the problem it is intended to solve in the context it is intended to be operated in

ST_ArgByIntegrationTesting Argument by appeal to integration testing

G_ArgByIndSubArgs1 Argument by independent sub-arguments

Context elaborations and justification

G_ArgBySatisfactionOfReqs Argument by appeal to demonstrable satisfaction of requirements

G_ReqSatisfied The delivered system satisfies its requirements

ST_ArgBySatisfactionOfReqs Satisfied The system satisfies its real-time requirements

G_RealTimeReqsSatisfied The system satisfies its real-time requirements

G_ArgByIndSubArgs2 Argument by independent sub-arguments

G_ArgOverRefinement Argument by showing that successive forms refine an original

G_ExecRefinesCode The executable refines the code

G_CodeRefinesSpec The code refines the non-real-time portions of the specification

G_SpecRefinesRequirements The specification refines the requirements

G_TestingShowsNon-TimingReqs Satisfied Testing shows that the system satisfies its non-timing requirements

S_Int TestReport Report from integration testing

Appeal to correct use of correct tool to establish WCET within schedule limits

Appeal to requirements-based functional testing with MC/DC of code

Appeal to separate argument and inspection of that argument

G_LLSSpecRefinesFormalSpec The low-level specification refines the formal portion of the specification

Implication proof checked by PVS proof checker

SPARK tools used to show that the Ada source code refines the low-level specification embodied in the SPARK-annotated subprogram specifications

G_TransformedCodeRefinesLL Spec The transformed code refines the low-level specification

G_TransformedCodeSemantic Equivalence The transformed code is semantically equivalent to the code

G_CodeRefinesSpec Contextural information is supported by

Compiler assumed correct

Appeal to correct use of correct tool to confirm stack usage within limits set in linker script

Legend:

A goal (claim)

An argument strategy

A solution (evidence)

An assumption

In the context of
Fitness argument

- 348 GSN elements
- Widest step: 5 child elements
- Longest path: 26 elements
- General form:
  - Integration testing and
  - Appeal to satisfied requirements
- Timing demonstrated by WCET analysis
- Functionality demonstrated by testing and formal proof

Legend:

- A goal (claim)
- An assumption
- An argument strategy
- Contextual information
- A solution (evidence)
- In the context of
- Is supported by
- Compiler assumed correct
- Appeal to correct use of correct tool to confirm stack usage within limits set in linker script
- SPARK tools used to show that the Ada source code refines the low-level specification embodied in the SPARK-annotated subprogram specifications
- Appeal to correct use of correct tool to confirm stack usage within limits set in linker script
Success Argument

C_System
"System" is the magnetic bearing control software
DC-001

C_OperatingContext
The system is a component of the LifeFlow LVAD First Prototype
DC-001

C_AcceptableCost
Presently available resources and staff plus target hardware costs
DC-001

G_Success
The development effort will lead to an adequate system in acceptable time and at acceptable cost

G_PlanDeliversOnTime
The development effort will be completed on time
DC-003

G_ScheduledDeliveryDateAccurate
The schedule accurately predicts the delivery date
DC-003

C_DevelopmentSchedule
The development schedule as recorded in /project-docs/plan/trunk/vad_ctl_sw.plan

G_ScheduledDeliveryDateAcceptable
The predicted delivery date is acceptable
DC-003

C_Requirements
Requirements imposed by the LifeFlow LVAD First Prototype are recorded in /project_docs/requirements/tags/v00.05.02_20090716/vad.cntrl_sw.reqs.pdf
DC-001

C_AcceptableTime
LifeFlow LVAD First Prototype delivery date
DC-001

C_AssumedDepGoals
The assumed dependability goals, as stated in /project_docs/assumed_dep_goals/tags/v00.01_20081212/assumed_dep_goals.pdf
DC-010

G_DevRisksMitigated
All credible development risks have been adequately mitigated
DC-002

G_ArgOverRisks
Argument over all credible development risks
DC-002
Success Argument

- 49 GSN elements
- Widest step: 10 child elements
- Longest path: 9 elements
- General form:
  - Appeal to planning **and**
  - Argument over enumerated development risks
- Evolution:
  - Starts small, grows early, becomes moot
### Case Study Results

<table>
<thead>
<tr>
<th>Feasibility</th>
<th>No difficulties observed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obligations are not appropriate choice drivers</td>
<td>28 choices:</td>
</tr>
<tr>
<td></td>
<td>• 19 followed direct reasoning</td>
</tr>
<tr>
<td></td>
<td>• 6 others addressed an obligation</td>
</tr>
<tr>
<td></td>
<td>• 2 addressed unnoted development risk</td>
</tr>
<tr>
<td></td>
<td>• 1 remaining case was an implicit choice</td>
</tr>
<tr>
<td>Can’t judge options by argument impact alone</td>
<td>We observed no value that could not be represented in the fitness or success argument</td>
</tr>
<tr>
<td>No decision criteria is missing or superfluous</td>
<td>We observed that impact on schedule was not covered in the choice criteria; it has been added</td>
</tr>
</tbody>
</table>
• Case study reimplementation of *Tokeneer*
  – NSA security challenge problem
  – Common Criteria
  – Evaluating ABD repair mechanism
  – Evaluating ABD pattern library
  – Collecting developer effort metrics

• Developing argument-based certification mechanism

• Working with philosophy department on argument quality
Questions?

Software Process Synthesis in Assurance Based Development of Dependable Systems

Patrick J. Graydon
John C. Knight

University of Virginia