Correct by construction software development

Saddek Bensalem

BIP team @ VERIMAG

VERIMAG Laboratory (Grenoble, France)

May, 12th 2015
• Introduction

• System Design

• Correct by Construction

• Discussion
The Evolution of Computer Science

Scientific Computing – Defence Applications

- Convergence between Computing and Telecommunications
- Graphic Interfaces, Mouse

WEB – Information Society

Multi-core Systems

1945

1936

Foundations - Alan Turing, Kurt Gödel

1970

1980

Embedded Systems:
Computing + Physicality
- Seamless revolution
- 95% of chips are embedded

1990

Information Systems:
Commercial Applications
- Integrated circuits

2000

The Internet of Things:
Convergence between Embedded Systems and the Web

2010

2015
## Systems vs Programs – The Hardest and the Most Important

### Significant differences

<table>
<thead>
<tr>
<th>Programs</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminating</td>
<td>non-terminating</td>
</tr>
<tr>
<td>deterministic</td>
<td>non-predictable</td>
</tr>
<tr>
<td>platform-independent</td>
<td>platform-dependent</td>
</tr>
<tr>
<td>Theory</td>
<td>No theory!</td>
</tr>
</tbody>
</table>

### Systems

- **are hard to design** due to unpredictable and subtle interactions with the environment, emergent behaviors, and occasional catastrophic cascading failures rather than to complex data and algorithms
- **are increasingly important** in modern computing systems: embedded systems, cyber-physical systems, mobile systems, web-services
What is needed?

We need new methods for designing and verifying systems in which physical characteristics are determined (or influenced) computationally and the other way around.
System Design – About Design

Design is a universal concept, a par excellence intellectual activity leading to artifacts meeting given requirements.

**Easy Apple Pie**

**Proceduralization**

**Recipe** (Program)

- Put apples in pie plate;
- Sprinkle with cinnamon and 1 tablespoon sugar;
- In a bowl mix 1 cup sugar, flour and butter;
- Blend in unbeaten egg, pinch of salt and the nuts;
- Mix well and pour over apples;
- Bake at 350 degrees for 45 minutes

**Materialization**

**Ingredients** (Resources)

- 1 pie plate buttered
- 5 or 6 apples, cut up
- ¾ c. butter, melted
- 1 c. flour
- ½ c. chopped nuts
- 1 tsp cinnamon
- 1 tbsp sugar
- 1 c. Sugar
System Design – Two Main Gaps

Requirements (declarative) → Application SW (executable) → System (HW+SW)

Correctness?
System Design – Correctness: Reminder

**Trustworthiness requirements** express assurance that the designed system can be trusted that it will perform as expected despite

- **HW failures**
- **Design Errors**
- **Environment Disturbances**
- **Malevolent Actions**

**Optimization requirements** dealing with optimization of functions subject to quantitative constraints on

1) **Performance**: how well the system does wrt user demands e.g. throughput, jitter, latency, quality.
2) **Cost**: how well resources are used wrt economic demands e.g. storage efficiency, processor utilizability, energy efficiency.

Usually they determine tradeoffs between performance and cost.
The V-model of the traditional Systems Engineering process

1. assumes that all the system requirements are initially known, can be clearly formulated and understood.

2. assumes that system development is top-down from a set of requirements. Nonetheless, systems are never designed from scratch; they are built by incrementally modifying existing systems and component reuse.

3. considers that global system requirements can be broken down into requirements satisfied by system components. Furthermore, it implicitly assumes a compositionality principle: if components are proven correct with respect to their individual requirements, then correctness of the whole system can be inferred from correctness of its components.

4. relies mainly on correctness-by-checking (verification or testing)
Correctness: designed artifact meets its requirements specifications.

Ideal for many designers: correctness by checking.

Usually impossible: verification method explodes exponentially with the number of components in the artifact.

The best we can do is limit automatic verification to small or medium size models and to specific properties.
The Limits of Correctness-by-checking: Conclusion

- **correctness-by-checking** of trustworthiness: limited to requirements that can be formalized and checked efficiently.

- Application to optimization requirements: limited to the validation of scheduling and resource management policies on abstract system models (medium size systems).

- For optimization: a more natural approach for their satisfaction is **by enforcing** rather than by checking.
Correct-by-Construction Approaches

- At the root of any mature engineering discipline.
  - scalable and do not suffer limitations of correctness-by-checking.
    - Testing may be still necessary, but its role is to validate the correct-by-construction process rather than to find bugs.

- Methodology to ensure correctness-by construction gradually by acting in two different directions:
  - **Horizontal correctness**, within a design step, by providing rules for enforcing global properties of composite components while preserving essential properties of atomic components;
  - **Vertical Correctness**, between design steps to guarantee that if some property is established at some step then it will be preserved at all subsequent steps.
Build a component $C$ satisfying a given property $P$, from

- $C_0$ a set of atomic components modeling behavior
- $GL = \{gl_{i,1}, ..., gl_{i,i}, ...\}$ a set of glue operators on components

Glue operators

- model mechanisms used for communication and control such as protocols, controllers, buses.
- restrict the behavior of their arguments, that is

$$gl(C_1, C_2, ..., C_n)| A_1 \text{ refines } C_1$$
Rule 1: Property Enforcement

Architecture for Mutual Exclusion

Horizontal Correctness (1)

Components

Architecture for Mutual Exclusion satisfies Mutex
Horizontal Correctness: Property Composability

Make the new without breaking the old: Rules guaranteeing non interference of solutions

Property stability phenomena are poorly understood. We need composability results e.g. feature interaction in middleware, composability of scheduling algorithms, theory for reconfigurable systems.
Feature interaction in telecommunication systems, interference among web services and interference in aspect programming are all manifestations of a lack of composability.
Rule 1: Property Preservation

Deadlock-free Routing Protocol → Deadlock-free components

Deadlock-free Routing Protocol

Deadlock-free composite component
The refinement relation \( \geq \) is defined as:

- All traces of \( S_2 \) are traces of \( S_1 \) (modulo some observation criterion).
- If \( S_1 \) is deadlock-free, then \( S_2 \) is also deadlock-free.

The question is: Propose adequate refinement relations.
Vertical Correctness: Preservation of $\geq$ substitution
Putting Correctness-by-Construction into Practice: The BIP Design Flow

- **D-Finder**
  - compositional verification
  - correctness checking

- **Application Software (Programming Models)**
  - translation
  - BIP

- **Application Software Model**
  - BIP
  - transformation
  - correct-by-construction
  - decentralization + integration of communication protocols

- **Distributed Software Model**
  - BIP
  - transformation
  - correct-by-construction
  - integration of platform and mapping constraints

- **Distributed System Model**
  - BIP
  - transformation
  - correct-by-construction
  - calibration w.r.t. execution times

- **Instrumented System Model**
  - BIP
  - performance evaluation

- **SMC-BIP**
  - statistical model-checking

- **(Many-Core) Platform Model**
  - Mapping
  - Functional Code
  - Glue Code
  - Runtime
  - HW Platform
  - e.g. manycore STHORM/STMicroelectronics
  - MPPA/Kalray
module Motion {
    number: 9600;
    SDI: MOTION_DATA;
}
request SetPos {
    type: control;
    input: pos::pos;
    control: controlPos;
    report: BAD_PARAM;
}
task Move {
    period: 25;
    priority: 15;
}
• Thanks for your attention.

Questions?