Where All Projects Start: Requirements

Michael Masin, IBM Research - Haifa
Luca Pulina, Università degli Studi di Sassari
Where All Projects Start: Requirements

Michael Masin, IBM Research - Haifa
Luca Pulina, Università degli Studi di Sassari
Outline

- Model Based Engineering Background
- PORTALS
  - Architecture
  - Interaction
  - Scenarios
  - Demo
- Requirements Formalization and Verification
Waterfall Model (from 1950s)

Start of Systems Engineering
V model (1980s)

Concept of Operations → Validation

Requirements → Verification

System Design → Integration

Subsystem Design → Testing

Implementation

Systems Engineering today
Model Based Engineering (< 10 years)
Continuous Model Based Engineering

CMBE: Models create models


Virtual models indicated with a purple label.

MBE label at bottom left of diagram.

CMBE: Models create models label at bottom right of diagram.

Slide number 7 of presentation.
❑ to create tools to assist requirements engineers in incrementally raising the formalization level of system requirements, and

❑ to use formalized requirements to
  o provide feedback on the quality of the requirements (e.g., identifying omissions and contradictions), and
  o create downstream artifacts (e.g., models, monitors, tests, code)

PORTALS Architecture

Input Methods
- Restricted Input
- Free Input, Watson Parser

Heavier \rightarrow Lighter

Knowledge Representation
- Formalized Requirements
- Engineering Knowledge Base

Analytics
- Requirement Analytics
- IoT Automation
- Asset Management Automation

Salient features:
- Heavier vs Lighter
- Input Methods
- Formalized Requirements
- Engineering Knowledge Base
- Analytics
Personas and Interactions

- **Requirements Engineer**
  - Compose requirements
  - Traceability

- **Systems Engineer**
  - Feedback on consistency
  - Make design decisions
  - Reason about requirements and models

- **Project Manager**
  - Project planning and control
  - Generate Artifacts

- **DOORS NG**

- **PORTALS**

- **Node-RED**
  - Optimization
  - Hybrid Simulation

- **Rhapsody**
Scenario 1: IBM IoT Safer Workplace

1. Requirement in DOORS

If an employee falls, the system shall send an SMS to the employee’s manager.

2. Paraphrase by PORTALS

if "an employee" falls then "the system" shall send [an abstract entity] "an SMS" (direction) "manager" of "the employee's"

3. Process Model

4. Implementation in Node-RED
Scenario 2: IoT Pump

1. Requirement in DOORS
   
   487: Too much vibration

   If the pump's vibration exceeds 100 Hz for two minutes, a technician shall be sent to the pump within 24 hours.

2. Paraphrase by PORTALS
   
   if "vibration" of "the pump's" is greater than 100 Hz (duration) 2 min then "?" shall send [a role entity] "a technician" (duration) 24 hr; (direction) "the pump"

3. Process Model

4. Implement in Node-RED
PORTALS: Enhanced Requirements
IBM Research - Haifa

Yishai Feldman
Vladimir Lipets
Aviad Sela
Evgeny Shindin
Scenario 3: Door Management System

- The weight of the Doors Management System shall not exceed 500 kg.
- The target mass of the locking system shall not exceed 260 kg.
- The target mass of the latching system shall not exceed 250 kg.
- The volume of the Doors Management System shall not exceed 1000 ft$^3$.
- The volume of the latching system shall not exceed 30 m$^3$.
- The volume of the latching system shall not exceed 35 m$^3$. 
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Primary Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>878</td>
<td>DMS-019</td>
<td>The weight of the Doors Management System shall not exceed 500 kg.</td>
</tr>
<tr>
<td>879</td>
<td>DMS-039</td>
<td>The target mass of the latching system shall not exceed 250 kg.</td>
</tr>
<tr>
<td>880</td>
<td>DMS-040</td>
<td>The target mass of the locking system shall not exceed 260 kg.</td>
</tr>
<tr>
<td>881</td>
<td>DMSX-1</td>
<td>The volume of the Doors Management System shall not exceed 1000 l.</td>
</tr>
<tr>
<td>882</td>
<td>DMSX-2</td>
<td>The volume of the latching system shall not exceed 30 m³.</td>
</tr>
<tr>
<td>883</td>
<td>DMSX-3</td>
<td>The volume of the latching system shall not exceed 35 m³.</td>
</tr>
<tr>
<td></td>
<td>Locking:Locking_System</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----------------------</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Requirements Analysis

Conflicting values in similar requirements

Subsystem budgets exceed system budget

Missing budget for a subsystem

Conflict due to different units (cf. Mars Climate Orbiter)
Requirements Analysis

Conflicting Requirements

- Inconsistent volume requirements for Latching System; values are 20 m³, 35 m³
- The weight budget for system Doors Management System, 100 kg, is exceeded by budgets for subsystems (Latching System = 250 kg, Locking System = 250 kg)
- The target mass of the locking system shall not exceed 250 kg
- The weight of the Doors Management System shall not exceed 150 kg
- The target mass of the latching system shall not exceed 150 kg
- A volume budget requirement for subsystem Locking System was not found; budget for containing system (Doors Management System) is 1000 ft³
- The volume budget for system Doors Management System, 1000 ft³, is exceeded by budgets for subsystems (Latching System = 50 m³)
Requirements Analysis

- Inconsistent volume requirements for Latching System: values are 20 m³, 35 m³.
- The weight budget for system Doors Management System, 100 kg, is exceeded by budgets for subsystems (Latching System = 250 kg; Locking System = 250 kg).
- The target mass of the locking system shall not exceed 250 kg.
- The weight of the Doors Management System shall not exceed 500 kg.
- The target mass of the latching system shall be less than or equal to 250 kg.
- A volume budget requirement for subsystem Locking System was not found; budget for containing system (Doors Management System) is 1000 ft³.
- The volume budget for system Doors Management System, 1000 ft³, is exceeded by budgets for subsystems (Latching System = 20 m³).
In the context of adaptive CPSs, checking the consistency of requirements is an indisputable, yet challenging task.

- Requirements written in natural language call for time-consuming and error-prone manual reviews, BUT
- enabling automated consistency verification often requires overburdening formalizations.

Given the increasing pervasiveness of CPSs, their stringent time-to-market and product budget constraints, practical solutions to enable automated verification of requirements are in order.

Desiderata: Unambiguous language with high expressiveness, that can be automatically translated in some logic and then used for verification/validation.

Expressiveness vs Unambiguity!
Property Specification Patterns (PSPs) offer a viable path towards this goal.

- PSP: collection of parameterizable, high-level, formalism-independent specification abstractions, originally developed to capture recurring solutions to the needs of requirement engineering.
- Each pattern can be directly encoded in a formal specification language, such as linear time temporal logic (LTL), computational tree logic (CTL), or graphical interval logic (GIL).
- Because of their features, PSPs may ease the burden of formalizing requirements, yet enable their verification using current state-of-the-art automated reasoning tools (e.g., for LTL).
Modal temporal logic with *modalities* referring to time

- One can encode formulae about the future of *paths*, e.g., a condition will eventually be true, a condition will be true until another fact becomes true, etc.

**Syntax:**

- LTL is built up from a finite set of *propositional variables* $AP$, the *logical operators* $\neg$ and $\lor$, and the *temporal modal operators* $X$ (next) and $U$ (until).

- The set of LTL formulas over $AP$ is inductively defined as follows:
  - if $p \in AP$ then $p$ is an LTL formula;
  - if $\psi$ and $\phi$ are LTL formulas then $\neg \psi$, $\phi \lor \psi$, $X \psi$, and $\phi U \psi$ are LTL formulas.

- Additional temporal operators: $G$ (globally), $F$ (eventually), $R$ (release)
**Linear Temporal Logic (LTL) - Semantics**

<table>
<thead>
<tr>
<th>Textual</th>
<th>Symbolic</th>
<th>Explanation</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unary operators:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( X \phi )</td>
<td>( \circ \phi )</td>
<td>\text{next}: ( \phi ) has to hold at the next state.</td>
<td>![Diagram for X phi]</td>
</tr>
<tr>
<td>( F \phi )</td>
<td>( \Diamond \phi )</td>
<td>\text{Finally}: ( \phi ) eventually has to hold (somewhere on the subsequent path).</td>
<td>![Diagram for F phi]</td>
</tr>
<tr>
<td>( G \phi )</td>
<td>( \Box \phi )</td>
<td>\text{Globally}: ( \phi ) has to hold on the entire subsequent path.</td>
<td>![Diagram for G phi]</td>
</tr>
<tr>
<td><strong>Binary operators:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \psi U \phi )</td>
<td>( \psi U \phi )</td>
<td>\text{Until}: ( \psi ) has to hold at least until ( \phi ) becomes true, which must hold at the current or a future position.</td>
<td>![Diagram for U phi]</td>
</tr>
<tr>
<td>( \psi R \phi )</td>
<td>( \psi R \phi )</td>
<td>\text{Release}: ( \phi ) has to be true until and including the point where ( \psi ) first becomes true; if ( \psi ) never becomes true, ( \phi ) must remain true forever.</td>
<td>![Diagram for R phi]</td>
</tr>
</tbody>
</table>
PSPs are meant to describe the essential structure of system's behaviours and provide expressions of such behaviors in a range of common formalisms.

A pattern is comprised of a

- name;
- an informal statement describing the behaviour captured by the pattern;
- a (structured English) statement that should be used to express requirements.
The LTL mappings corresponding to different declinations of the pattern are also given, where capital letters (P, Q, R, ...) stands for Boolean states/events.

A complete list of patterns is available at http://patterns.projects.cs.ksu.edu
The original formulation of PSPs caters for temporal structure over Boolean variables: for most practical applications, such expressiveness is too restricted.

Example: embedded controller for robotic manipulators (from CERBERO use case)
  - With original PSPs, requirements such as "The angle of joint1 shall never be greater than 170 degrees" cannot be expressed.

Solution proposed in CERBERO: PSPs with Boolean and Constrained Numerical Signals (with sound translation to LTL).
Controller for a Robotic Manipulator

Let consider a set of requirements from the design of an embedded controller for a robotic manipulator:

• the controller should direct a properly initialized robotic arm to look for an object placed in a given position and move to such position in order to grab the object;
• once grabbed, the object is to be moved into a bucket placed in a given position and released without touching the bucket.
• The robot must stop also in the case of an unintended collision with other objects or with the robot itself.
• collisions can be detected using torque estimation from sensors placed in the joints.

The manipulator is a 4 degrees-of-freedom Trossen Robotics WidowX Arm equipped with a gripper.
Constrained numerical signals are used to represent requirements related to various parameters:

- angle, speed, acceleration, and torque of the 4 joints, size of the object picked, and force exerted by the end-effector.

75 requirements in total.

Globally, it is never the case that joint1_angle < -170 or joint1_angle > 170 holds.

Globally, it is always the case that if ef_idle holds, then ef_speed = 0 and ef_acc = 0 holds as well.

After state_init until state_scanning, it is never the case that state_moving_to_target holds.

The complete list is available at https://github.com/SAGE-Lab/robot-arm-usecase
The formal representation of all requirements is "glued" together. The resulting formula is checked with a Model Checker or Theorem Prover. If the formula is satisfiable, then the system can be realized. Otherwise, inconsistency $\Rightarrow$ Impossible to build a system that satisfy all the requirements!
The ReqV tool

- ReqV (with NuSMV as back engine) checked automatically the requirements in about 37 seconds.

Available at
- https://github.com/SimoV8/ReqV-webapp
- https://github.com/SimoV8/ReqV-backend
The ReqV tool
### GA demo

<table>
<thead>
<tr>
<th>id</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>939</td>
<td>Globally, it is never the case that joint1_angle &lt; -1.30 or joint1_angle &gt; 1.70 holds.</td>
</tr>
<tr>
<td>940</td>
<td>Globally, it is never the case that joint2_angle &gt; 1.30 or joint2_angle &gt; 1.30 holds.</td>
</tr>
<tr>
<td>941</td>
<td>Globally, it is never the case that joint3_angle &lt; -1.30 or joint3_angle &gt; -1.30 holds.</td>
</tr>
<tr>
<td>942</td>
<td>Globally, it is never the case that joint4_angle &lt; -0.30 or joint4_angle &gt; 0.30 holds.</td>
</tr>
<tr>
<td>943</td>
<td>Globally, it is never the case that joint1_speed &gt; 30 holds.</td>
</tr>
<tr>
<td>944</td>
<td>Globally, it is never the case that joint2_speed &gt; 30 holds.</td>
</tr>
<tr>
<td>945</td>
<td>Globally, it is never the case that joint3_speed &gt; 30 holds.</td>
</tr>
<tr>
<td>946</td>
<td>Globally, it is never the case that joint4_speed &lt; 30 holds.</td>
</tr>
<tr>
<td>947</td>
<td>Globally, it is never the case that joint1_acc &gt; 10 holds.</td>
</tr>
<tr>
<td>948</td>
<td>Globally, it is never the case that joint2_acc &gt; 10 holds.</td>
</tr>
<tr>
<td>949</td>
<td>Globally, it is never the case that joint3_acc &gt; 10 holds.</td>
</tr>
<tr>
<td>950</td>
<td>Globally, it is never the case that joint4_acc &gt; 10 holds.</td>
</tr>
<tr>
<td>951</td>
<td>Globally, it is never the case that e_force &gt; 1.5 holds.</td>
</tr>
<tr>
<td>952</td>
<td>Globally, it is never the case that proximity_sensor &lt; 0 holds.</td>
</tr>
</tbody>
</table>
The ReqV tool
The ReqV tool

GA demo

Computing Minimum Unsatisfiable Core of 75 requirements
13-02-2019 08:46:50
Log:

Consistency checking 75 requirements
13-02-2019 08:42:22
Log:

Translating requirements...
Starting model checking...

*** This is NuSMV 2.6.0 (compiled on Wed Oct 1 15:29:00 2015)
*** Grid limit reached. This is a common issue with large models.
*** For more information on NuSMV see: http://nusmv.fbk.eu

40
The ReqV tool

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>939</td>
<td>Globally, it is never the case that joint1_angle &lt; -130 or joint1_angle &gt; 170 holds.</td>
</tr>
<tr>
<td>940</td>
<td>Globally, it is never the case that joint2_angle &lt; -130 or joint2_angle &gt; 170 holds.</td>
</tr>
<tr>
<td>941</td>
<td>Globally, it is never the case that joint3_angle &lt; -130 or joint3_angle &gt; 170 holds.</td>
</tr>
<tr>
<td>942</td>
<td>Globally, it is never the case that joint4_angle &lt; -30 or joint4_angle &gt; 90 holds.</td>
</tr>
<tr>
<td>943</td>
<td>Globally, it is never the case that joint1_speed &gt; 90 holds.</td>
</tr>
<tr>
<td>944</td>
<td>Globally, it is never the case that joint2_speed &gt; 90 holds.</td>
</tr>
<tr>
<td>945</td>
<td>Globally, it is never the case that joint3_speed &gt; 90 holds.</td>
</tr>
<tr>
<td>946</td>
<td>Globally, it is never the case that joint4_speed &gt; 90 holds.</td>
</tr>
<tr>
<td>947</td>
<td>Globally, it is never the case that joint1_acc &lt; 10 holds.</td>
</tr>
<tr>
<td>948</td>
<td>Globally, it is never the case that joint2_acc &lt; 10 holds.</td>
</tr>
<tr>
<td>949</td>
<td>Globally, it is never the case that joint3_acc &lt; 10 holds.</td>
</tr>
<tr>
<td>950</td>
<td>Globally, it is never the case that joint4_acc &lt; 10 holds.</td>
</tr>
<tr>
<td>951</td>
<td>Globally, it is never the case that e_form &gt; 2.5 holds.</td>
</tr>
<tr>
<td>952</td>
<td>Globally, it is never the case that proximity_sensor &lt; 0 holds.</td>
</tr>
</tbody>
</table>
Enabling the automated (formal) verification of requirements is one of the key aspects towards the development of safety- and security-critical CPSs.

The expressiveness of original PSPs is often too restricted for practical applications.

- Hybrid systems? Probabilistic models? Real-time constraints?

Main issue: scalability!
Some references


