Improve Safety and Reliability of Medical Devices with Modeling and Simulation

Arvind Ananthan
Medical Devices Industry Manager
MathWorks
Natick, USA
“Modeling and simulation (M&S) is getting information about how something will behave without actually testing it in real life.”

– Wikipedia
Software in Medical Devices

Complex software increasing rapidly in Medical Devices

80,000 LOC*

170,000 LOC*

7 Million LOC*

Apps – software is a medical device

* LOC – Lines of Code.

Source: Economist June 2012 article “When code can kill or cure”, http://www.economist.com/node/21556098
Medical Device Recalls
1999 – 2005

- “1 in 3 of all software-based medical devices sold in America between 1999 and 2005 had been recalled for software failures”**

- Frequent Devices for Class 1 Recall
  - Infusion Pumps
  - AED
  - Continuous Ventilators
  - Blood Glucose Monitors (OTC)

** Economist June 2012 article “When code can kill or cure”, referring to study done by University of Patras in Greece on Medical Device recalls
FDA Recalls

- # of Recalls **doubled** since 2003
- Increasing # of Class 1 and 2 recalls

Source: Medical Device Recall Report FY 2003-FY2012, CDRH, Office of Compliance, FDA
Recall Cause Categories

- Majority of Recall Reasons related to 820.30 Design Controls
  - Design and revision of that design during the lifetime of the device
Traditional Development Process

- RESEARCH
- REQUIREMENTS
- SPECIFICATIONS
- DESIGN AND IMPLEMENTATION
- INTEGRATION AND TEST

- C/C++
- Algorithm Design
- MCAD/MCAE
- EDA

- Embedded Software
- Embeddable Algorithms
- Mechanical Components
- Electrical Components
Problems in Traditional Development

**RESEARCH**

**REQUIREMENTS**

**SPECIFICATIONS**

**DESIGN AND IMPLEMENTATION**

**INTEGRATION AND TEST**

- **Requirement Documents**: Difficult to analyze and maintain.
- **Paper Specifications**: Incomplete, easy to misinterpret, and often out-of-date.
- **Physical Prototypes**: Expensive!
- **Manual Coding**: Expensive, time consuming, and buggy!
- **Traditional Testing**: Specification and design issues found late in project.
Late Errors Cost More

Relative cost to fix a software error

Type of error:
- requirements
- design
- coding

Source: Return on Investment for Independent Verification & Validation, NASA, 2004
Solutions

1. Multi-domain System Modeling
Solutions

1. Multi-domain System Modeling
2. Model-Based Design
Solutions

1. Multi-domain System Modeling
Modeling

Building Modeling

Solid Modeling

Multi-Domain System Modeling
Multi-domain System Modeling
6 Modeling Domains

CONTINUOUS-TIME MODELS
Dynamic systems, Plant models, Controller models

DISCRETE-TIME MODELS
Digital control, DSP, Image/video

DISCRETE-EVENT MODELS
Architecture, Latency, Resource, Performance modeling

STATE MACHINE MODELS
Control logic, Mode logic

PHYSICAL MODELS
Electronics, Mechanics, Hydraulics, Thermal

TEXT-BASED MODELS
Building a Robotic Surgical Arm

How a differential equation becomes a moving robot arm
What makes a robot arm move?

Controller code
How to design and build a robot arm?

If possible, break down a big problem into “smaller” (i.e. more manageable) problems

*** Use a “divide and conquer” approach ***
So, where do we really start?

If possible, break down a big problem into “smaller” (i.e., more manageable) problems

*** Use a “divide and conquer” approach ***
So, where do we really start?

Understand the underlying mathematics/physics of the problem
So, where do we really start?

Understand the underlying mathematics/physics of the problem

*** Gain insight into the problem ***
Now we need pen and paper...

\begin{align*}
\text{Kinematics} \\
x &= P \cos \theta \\
y &= P \sin \theta \\
x' &= -P \cos \theta \dot{\theta} \\
y' &= P \sin \theta \dot{\theta}
\end{align*}

\begin{align*}
\text{Dynamics} \\
I_2 &= \frac{mL^2}{12} + m\left(\frac{L}{2}\right)^2 \\
&= \frac{mL^2}{12} + \frac{mL^2}{4} \\
&= \frac{2mL^2}{12} = \frac{mL^2}{6}
\end{align*}

Newton's Law
\begin{align*}
\sum \text{Torques} &= I_2 \dot{k} \\
- \text{Mg cos} \theta \cdot \frac{1}{2} &= I_2 \ddot{\theta}
\end{align*}

Then:
\begin{align*}
\ddot{\theta} &= -\frac{3G \cos \theta}{2L}
\end{align*}
Or, we can use the Symbolic Math Toolbox notebook interface…

- To help us derive the dynamic equations analytically
- To find analytic (when possible) and/or numerical solutions to the problem
Or, we can use Simulink to build a graphical representation of the equation...

Dynamic differential equation for an ideal single link

$$\ddot{\theta} = -\frac{3g}{2\rho} \cos \theta$$
Now, let's return to our original problem…

We need to “extend” the approach.
Now, let's return to our original problem...

We need to "extend" the approach
Now, let's return to our original problem...

We need to “extend” the approach
And quickly, things start to get ugly…
We can solve this problem too...

In general, more complex equations will require more complex solution models.
But, there HAS to be a more efficient way of doing this…

Often times, it helps looking at a problem from a slightly different perspective
*** i.e. any mechanism can be defined as a combination of rigid bodies and joints ***
We can model the dynamics of 3D mechanisms…

Using the appropriate tool for the job can sometimes make our lives much easier
And manually create and combine any number of bodies and joints...

SimMechanics will allow us to easily extend our 3D mechanical model
Or, we can import the entire mechanical design directly from a CAD tool…

The translator automatically generates the SimMechanics model using CAD information
Now, for our robot to be of any use, it has to be able to move...

We can include the effects of actuator and sensor dynamics in the simulation model.
So, “something” has to actuate those mechanical joints…

Take advantage of the flexibility provided by *Simscape* and the advanced physical component libraries in Simulink to create models for all kinds of actuators and sensors.
We can balance model fidelity and simulation speed based on our needs…

Easily compare results and control the overall level of detail in your simulation model by using configurable subsystems or switching between subsystem variants

Average voltage H-bridge amplifier model

Transistor level detail H-bridge amplifier model
If needed, we can quickly validate our component models using test data...

\[ R = \text{Resistance} \]
\[ L = \text{Inductance} \]
\[ J = \text{Inertia} \]
\[ B = \text{Friction} \]
\[ K = \text{Back EMF Constant} \]

Use Simulink Design Optimization to automatically tune model parameters by taking advantage of the capabilities in the MATLAB Optimization Toolbox directly in Simulink.
We can store our customized component models in libraries for easy reuse...

Develop component models and build your own custom project libraries for future use and/or sharing with other members of your team.
So far, we have built a dynamic model of our plant...
And added the actuators and sensors…
And added the appropriate actuators and sensors...
Now it is time to add a controller and “close the loop”...

Modeling both, plant and controller in a single environment allows us to better understand and optimize the performance of the entire system.
And we can test and verify the overall system performance in simulation...

Generate a battery of test vector arrays and verify the controller design against the performance specifications given for the system.
医療・介護機器の制御システム設計
〜 モデルベースデザインにトライ 〜

概要: ロボットアームのプラントモデリング、コントローラ設計
講演者: 福井慶一
セッション番号: E4
会場: E
時間: 16:20 – 17:10
Modeling Blood Flow in Capillary Networks at Draper Labs, Cambridge, MA

Simulating physiological network system for synthetic organ development
Vasculature Modeling for MEMS fabrication

- Microfluidics simulation to develop Kidney/Lungs

Each step requires development of models, algorithms, and solvers.
Blood Capillary Network Model
Whole Blood Testing

GEM 3000 Blood Gas Analyzer
- $O_2$ pressure
- $CO_2$ pressure

Avoximeter 400 (ITC)
- tHb content
- pHb $O_2$
- pHb $CO_2$
Mock Circulatory Loop (MCL) Model for Testing Heart Pumps (or VADs)
Automating the Mock Circulatory Loop

Blood Pump

Compliance Chamber

Peripheral Resistance
Implementing a Fully Automated Mock Circulatory Loop to Simulate Cardiovascular Conditions

By Charles E. Taylor, Virginia Commonwealth University

Mock circulatory loops (MCLs) simulate the human circulatory system to enable testing of ventricular assist devices and other cardiac assist technologies. When conducting tests using an MCL, investigators must adjust the settings on each component in the loop. Components typically include a mechanical pump, a compliance chamber, and a peripheral resistance valve (Figure 1). In the past, these setting adjustments were performed manually, a method that was imprecise, time-consuming, and susceptible to human error.

Figure 1. The mock circulatory loop, including a modified Harvard apparatus pump (1), flow meter (2), compliance chamber (3), peripheral resistance valve (4), reservoir tank (5), and centrifugal pump (6).
Solutions

1. Multi-domain System Modeling
2. Model-Based Design
Traditional System Development Process
“Waterfall”

1. Research
2. Requirements and Specifications
3. Design
4. Implementation
5. Test and Verification

Model-Based Design Process
Traditional System Development Process
“Waterfall”

Model → specification
Requirements → in the model
Design is iterative
Code is generated automatically
Testing and verification is continuous

Model-Based Design Workflow

RESEARCH → REQUIREMENTS → DESIGN → IMPLEMENTATION → TEST & VERIFICATION

- Environment Models
- Physical Component Models
- Algorithms

IMPLEMENTATION
- C, C++
- VHDL, Verilog
- Structured Text
- MCU, DSP, FPGA, ASIC, PLC, PAC

INTEGRATION
Artificial Pancreas Project
Developing the next generation of closed-loop insulin infusion system
Carbohydrate Metabolism with T1DM
Type 1 Diabetes with Insulin Therapy
Glycemic System Model & Insulin Pump Modeling Device and Patient
Requirements: Linking and Tracing
# Model-Based Design mapping with IEC-62304

## Software Development Process

<table>
<thead>
<tr>
<th>Planning</th>
<th>Requirements Analysis</th>
<th>Architectural Design</th>
<th>Detailed Design</th>
<th>Implement and Verify Units</th>
<th>Integration and Integration Testing</th>
<th>(Software) System Testing</th>
<th>Release</th>
</tr>
</thead>
</table>

## Model-Based Design Tools
- Simulink and Stateflow
- Simulink Verification and Validation
- Simulink Report Generator
- Model Advisor Checks
- SignalBuilder
- Model Coverage
- Simulink Design Verifier
- SystemTest
- Real-Time Workshop Embedded Coder
- Code Generation Traceability Report
- Embedded IDE Link
- PolySpace

## Typical Documentation Artifacts

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Detailed Design (Class C)</td>
<td>Unit Test Report</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

[Image of diagram showing the mapping of the software development process with IEC-62304 standards using Model-Based Design tools and typical documentation artifacts.]
IEC 61508 TÜV Certification of MathWorks Tools

Certificate based on:

- Focused audit by TÜV of The MathWorks development and quality assurance processes
- Review by TÜV of MathWorks document describing example workflow for verification and validation of models and generated code
- Test cases and test procedures to support tool validation

Certificate report
Workflow description
# Model-Based Design in Medical Devices

<table>
<thead>
<tr>
<th>Philips</th>
<th>Smart Digital RF Power Subsystems for MRI Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Design issues resolved early in development</td>
</tr>
<tr>
<td></td>
<td>Tradeoffs rapidly assessed and implemented</td>
</tr>
<tr>
<td></td>
<td>Process consistency and predictability improved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Infraredx</th>
<th>Accelerate FPGA Development for IVUS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>System constructs a 3-D model of the area around the car</td>
</tr>
<tr>
<td></td>
<td>Uses 2-D images generated by a single side-view camera</td>
</tr>
<tr>
<td></td>
<td>▪ Image processing algorithms developed 3-4 times faster than in C or C++</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weinmann</th>
<th>Software for Emergency Ventilators</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Code development and reviews accelerated by 50%</td>
</tr>
<tr>
<td></td>
<td>Dozens of design alternatives explored</td>
</tr>
<tr>
<td></td>
<td>60% of core design reused</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sonova/PHONAK</th>
<th>Hearing Aids and Implants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>▪ Real-time prototypes updated in minutes, not days</td>
</tr>
<tr>
<td></td>
<td>▪ 80% of software libraries reused in platform</td>
</tr>
<tr>
<td></td>
<td>▪ Software quality improved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doheny Eye Inst. (Second Sight)</th>
<th>Image processing algorithms for retinal prosthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>▪ Development time reduced from months to weeks</td>
</tr>
<tr>
<td></td>
<td>▪ DSP deployment streamlined</td>
</tr>
<tr>
<td></td>
<td>▪ Patient testing improved</td>
</tr>
</tbody>
</table>
Model-Based Design
– where is it being used?

ISO 26262
GM USA
Hybrid Powertrain

EN-50128
Alstom France
Propulsion Control Systems

DO-178
Airbus Helicopters
EC130 Helicopter

IEC-60880
MTU Germany
Nuclear Emergency Generators

IEC 61508
Alstom Grid UK
HDVC Power Systems

IEC 62304
Weinmann Medical Germany
Transport ventilator
Recall Cause Categories

- Majority of Recall Reasons related to 820.30 Design Controls
  - Design and revision of that design during the lifetime of the device
Model-Based Design Impact

More time on design (Increase Safety)
Less time on implementation and test

Source: Arthur D. Little GmbH
[chart translated]
FDA & Model-Based Design

Infusion Pump Software Safety Research at FDA

- Software Safety
  - Model-Based Design of Infusion Pumps
  - Generic Infusion Pump Project
  - Static Analysis
  - Recent Publications About Model-Based Software Development

Software Safety

Many infusion pumps are controlled by software that governs key aspects of the user interface, controls the pumping mechanism to maintain the prescribed infusion rate, and performs key safety functions. The purpose of software is to make the device safer and easier to use. Users often do not realize the extent to which software determines many of the key functional and performance characteristics of the system until something goes wrong.

Unfortunately, in some cases, the software does “go wrong” and compromises patient safety. Because software is inevitably complex, abstract, and intangible, design errors can be difficult to detect. Users and patients should expect infusion pump software to be free from errors. The occurrence of a software error should be a highly unusual event.

Historically, software safety has been focused on the software development process and system-level testing. Not coincidentally, the FDA has focused its regulatory oversight on these two elements as well. Having skilled engineers and rigorous software development processes, as required by FDA’s Quality System Regulation, is important to help minimize errors. However, assessing the software development process provides only indirect insight into the “goodness” of the resulting software. And, system-level testing that would comprehensively test the software is currently beyond the state of the art for all but the simplest of systems. Moreover, it is very difficult to actually test the software under real-world conditions until it has been married with the hardware in a finished prototype, and that typically doesn’t happen until late in the product development cycle.

Model-Based Design of Infusion Pumps

The FDA has recognized that if product developers had tools that enable them to examine and evaluate software earlier in the development cycle, then there would be a greater likelihood that the resulting software
FDA-MathWorks MBD Workshop
Experiential Learning Program

- 2-Day Workshop conducted in Natick at MathWorks

- 10 attendees from CDRH, Ventilator and Anesthesia division

- Collaboration with Dräger Medical
  - Presentation of closed-loop controller used in Dräger Devices
  - Demo of Ventilator and Anesthesia machines
医療機器開発向け
モデルベースデザインによる開発フローのご紹介

会場：E
時間：15:20-16:10
講演者：MathWorks Japan
大塚慶太郎

要求仕様書からモデリング、検証、コード生成まで、モデルベースデザインによる開発フローの一連の流れをご紹介いたします。

“実行可能な仕様書”
医療・介護機器の制御システム設計～モデルベースデザインにトライ～

概要：ロボットアームのプラントモデリング、コントローラ設計
講演者：福井慶一
セッション番号：E4
会場：E
時間：16:20－17:10