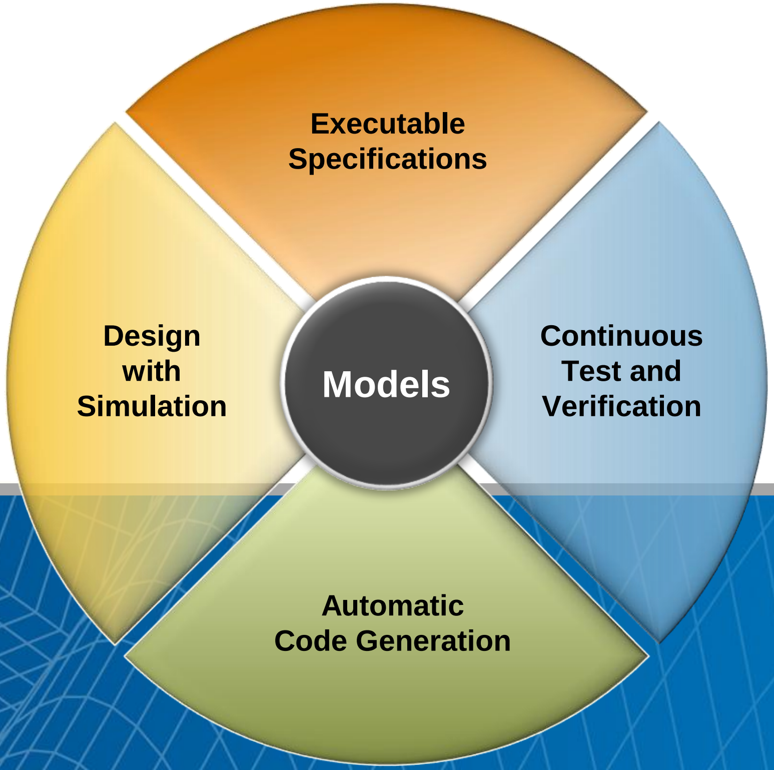
MATLAB/Simulink in der Mechatronik

So einfach geht‘s!


Executable Specifications
Design with Simulation
Models
Continuous Test and Verification
Automatic Code Generation

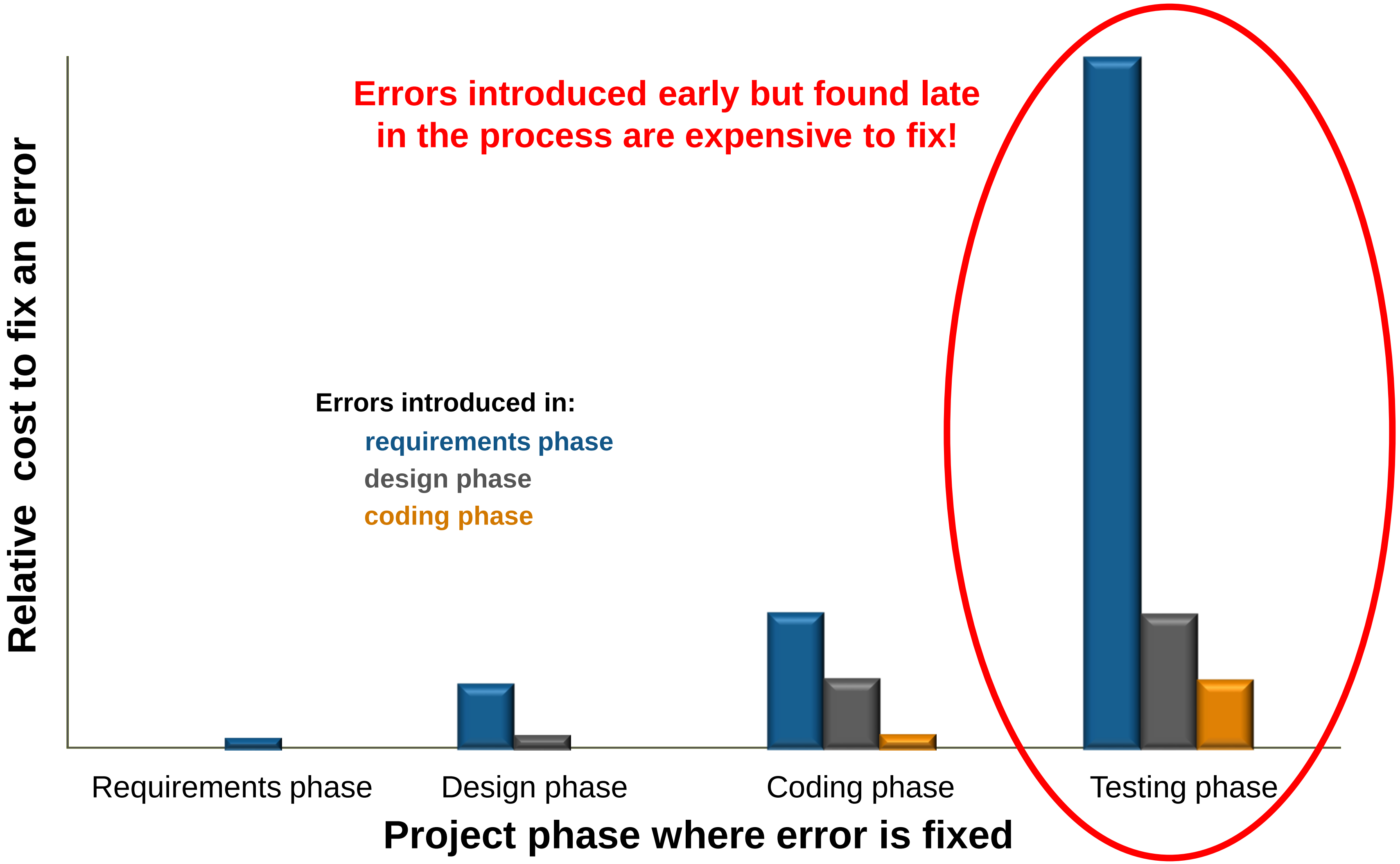
Tobias Kuschmider
Applikationsingenieur

MathWorks?
Was ist das?

Engineering Challenges Today

- Ambitious, highly-complex projects with short development cycles
- False implementation of (often incomplete) requirements
- Discovery of errors late in development process
  Costly and time consuming to fix
- Time delays and cost overruns
  Resulting in loss of reputation/market shares

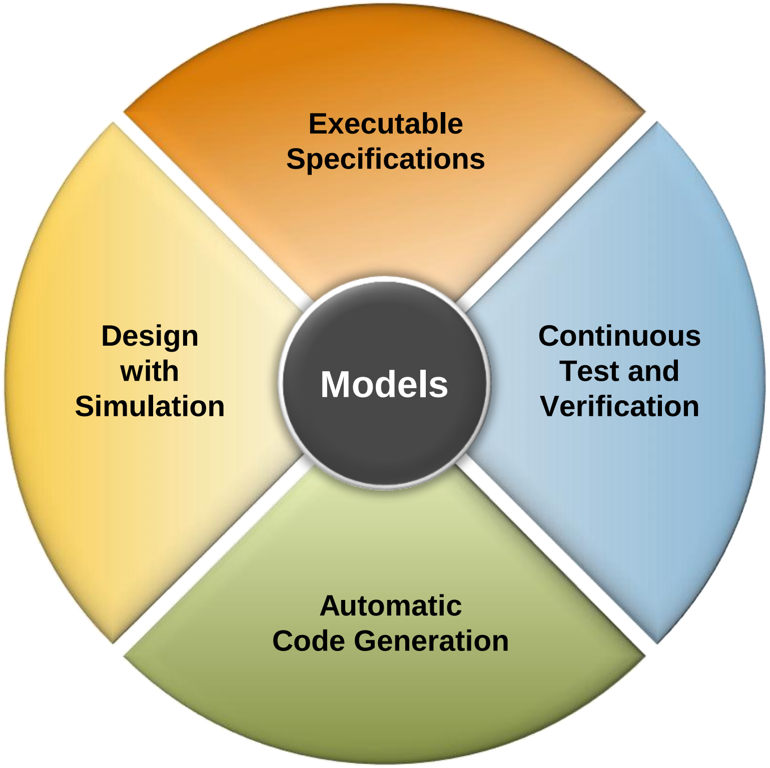
What is the Most Expensive Project Stage to Find Errors In?



Errors introduced early but found late in the process are expensive to fix!

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

Model-Based Design



Executable Specifications

Design with Simulation

Models

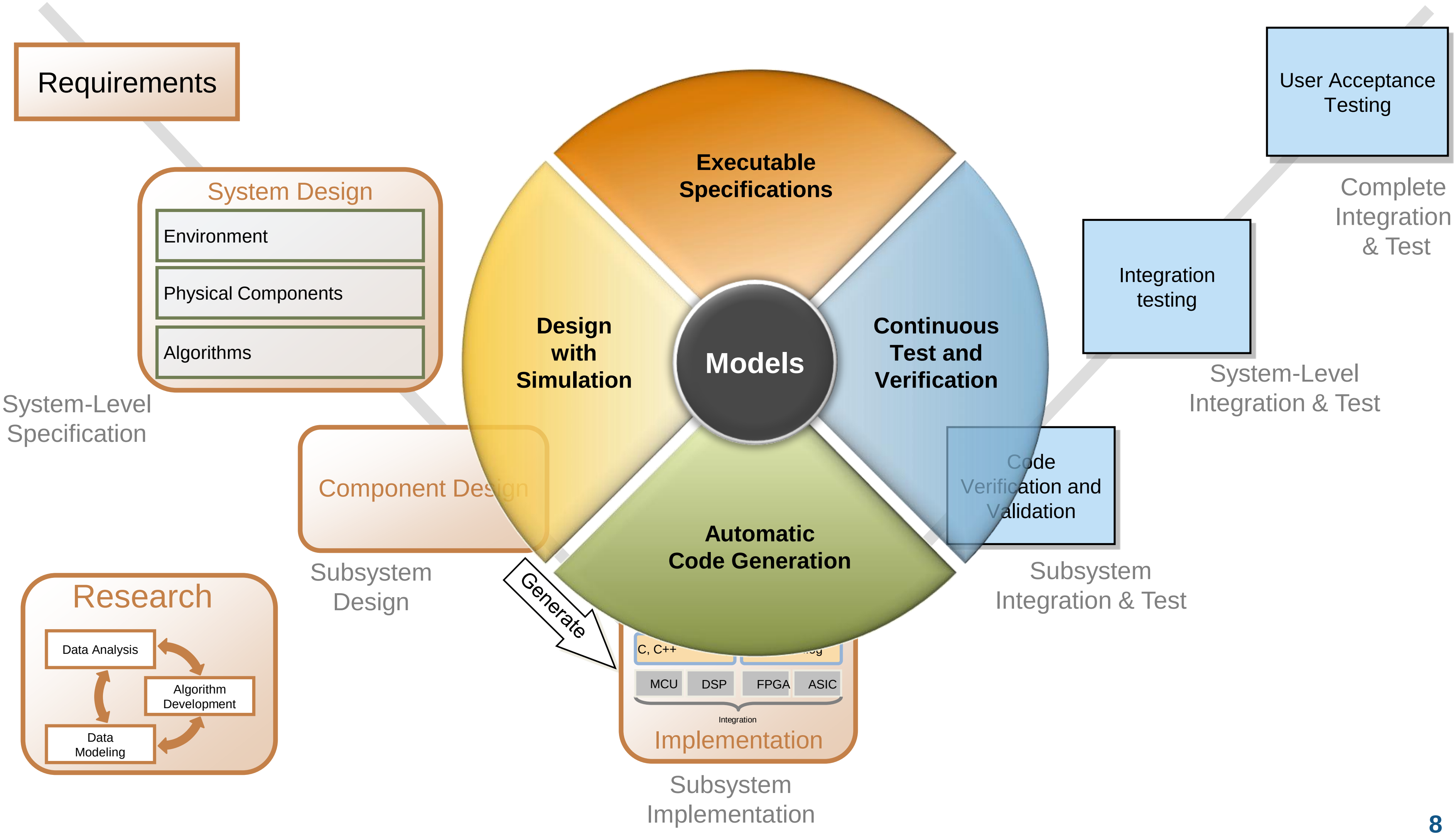
Continuous Test and Verification

Automatic Code Generation

Demonstration Motor-Control

Model-Based Design

Development Process



Requirements

System Design

Environment

Physical Components

Algorithms

System-Level Specification

Component Design

Subsystem Design

Research

Data Analysis

Algorithm Development

Data Modeling

Generate

C, C++

MCU

DSP

FPGA

ASIC

Integration

Implementation

Subsystem Implementation

Executable Specifications

Design with Simulation

Models

Continuous Test and Verification

Automatic Code Generation

User Acceptance Testing

Complete Integration & Test

Integration testing

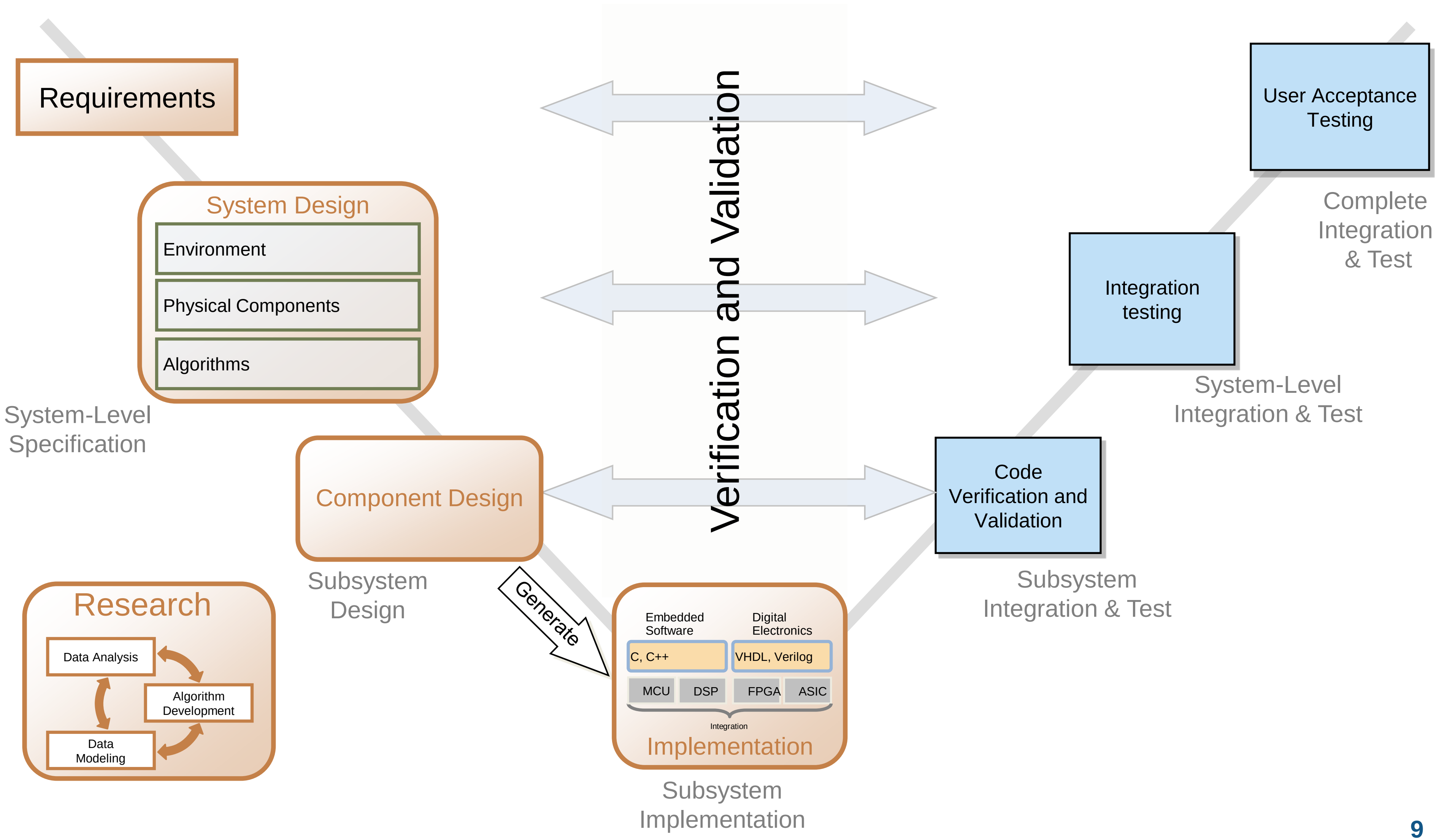
System-Level Integration & Test

Code Verification and Validation

Subsystem Integration & Test

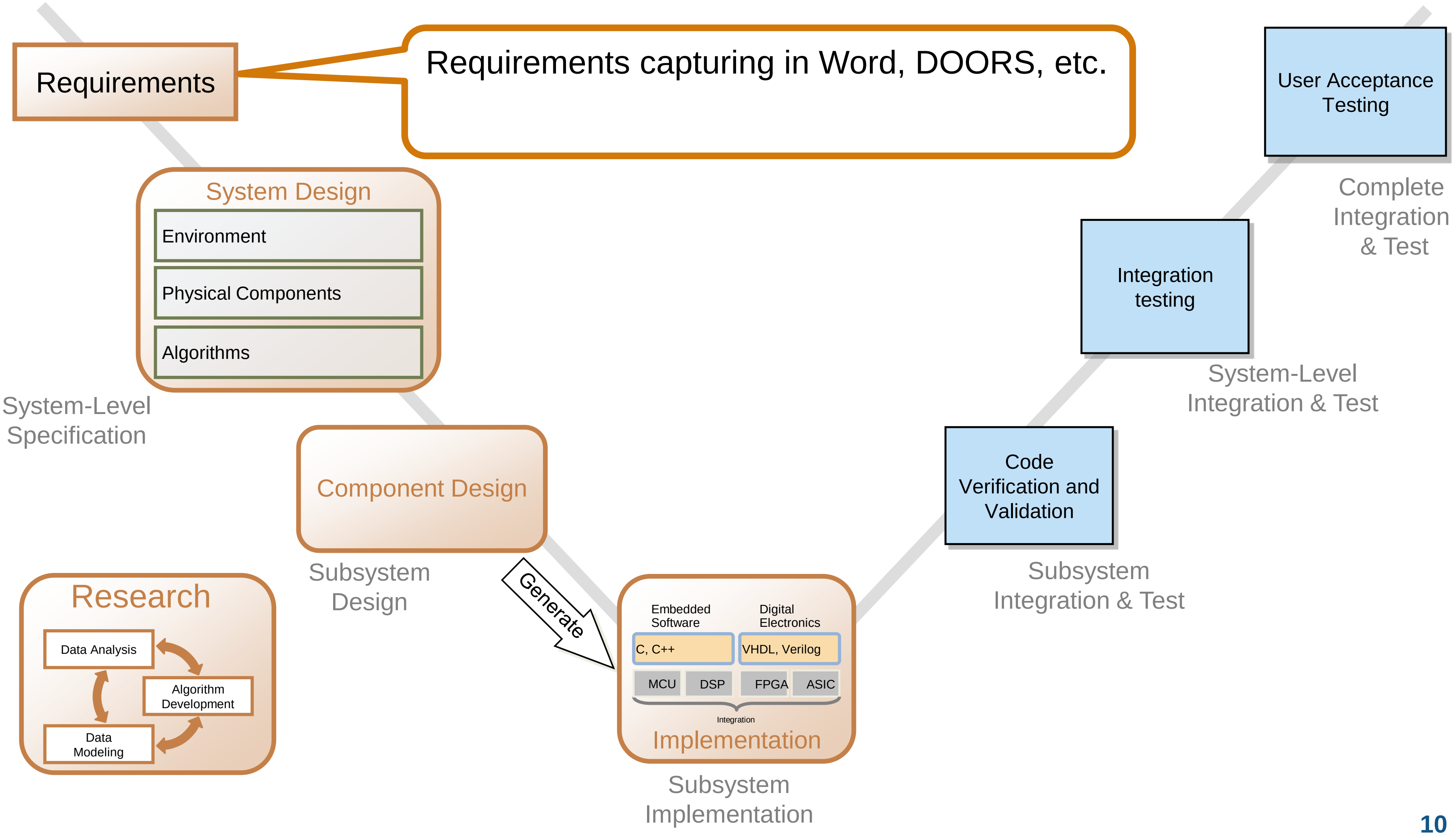
Model-Based Design

Continuous Verification and Validation



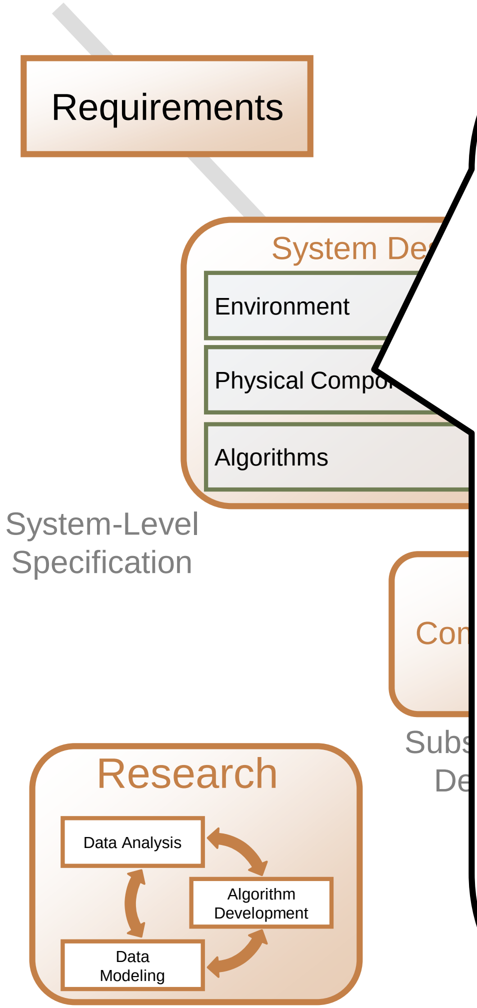
Model-Based Design

Development Process

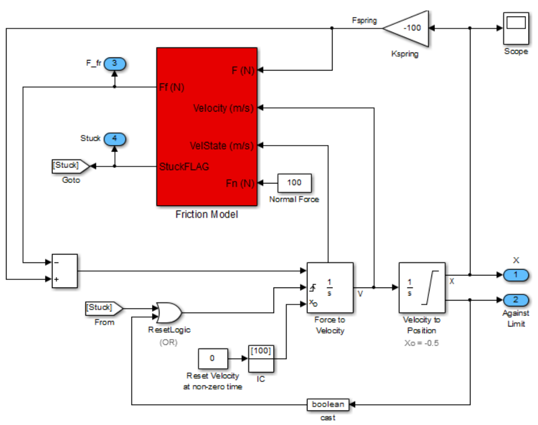


Model-Based Design

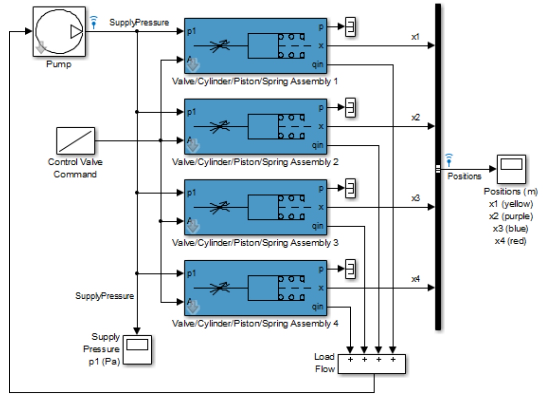
Multi-Domain Modeling and Algorithm Development



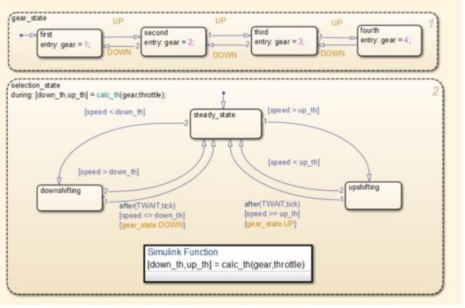
Methods for modeling systems in different domains



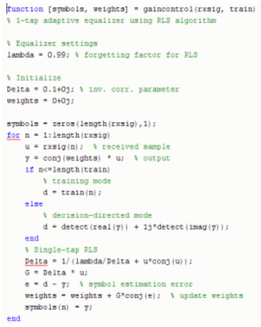
Data Flow (Block diagram)



Physical Modeling (Schematic)



Modeling of Event-Driven Systems (State - Machines)



```
function [symbols, weights] = gaincontrol(rxsig, train)
% 1-tap adaptive equalizer using RLS algorithm

% Equalizer settings
lambda = 0.99; % forgetting factor for RLS

% Initialize
Delta = 0.1+0j; % inv. corr. parameter
weights = 0+0j;

symbols = zeros(length(rxsig),1);
for n = 1:length(rxsig)
    u = rxsig(n);  % received sample
    y = conj(weights) * u;  % output
    if n<=length(train)
        % training mode
        d = train(n);
    else
        % decision-directed mode
        d = detect(real(y)) + 1j*detect(imag(y));
    end
    % Single-tap RLS
    Delta = 1/(lambda/Delta + u*conj(u));
    G = Delta * u;
    e = d - y;  % symbol estimation error
    weights = weights + G*conj(e);  % update weights
    symbols(n) = y;
end
```

Programing Language (Textual)

Model-Based Design

Early Concept Verification

Requirements

User Acceptance Testing

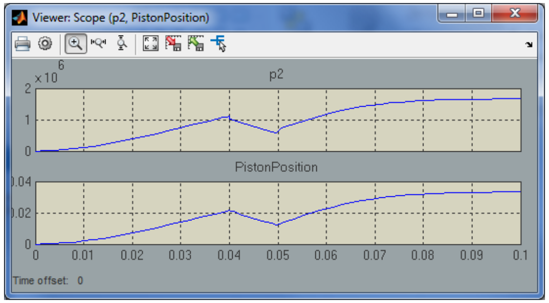
System-Level Specification

Researc

Data Analysis

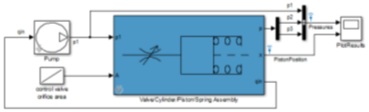
Data Modeling

- Executable specifications
- Predict dynamic system behaviour by simulation
  - System & environment models
  - Less physical prototypes
- Use of simulation results for system design
  - Fast What-/If studies
  - Short iteration cycles



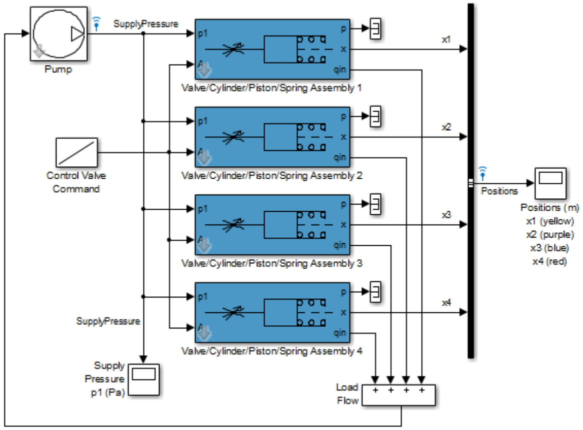
Idea

→



Simple Model

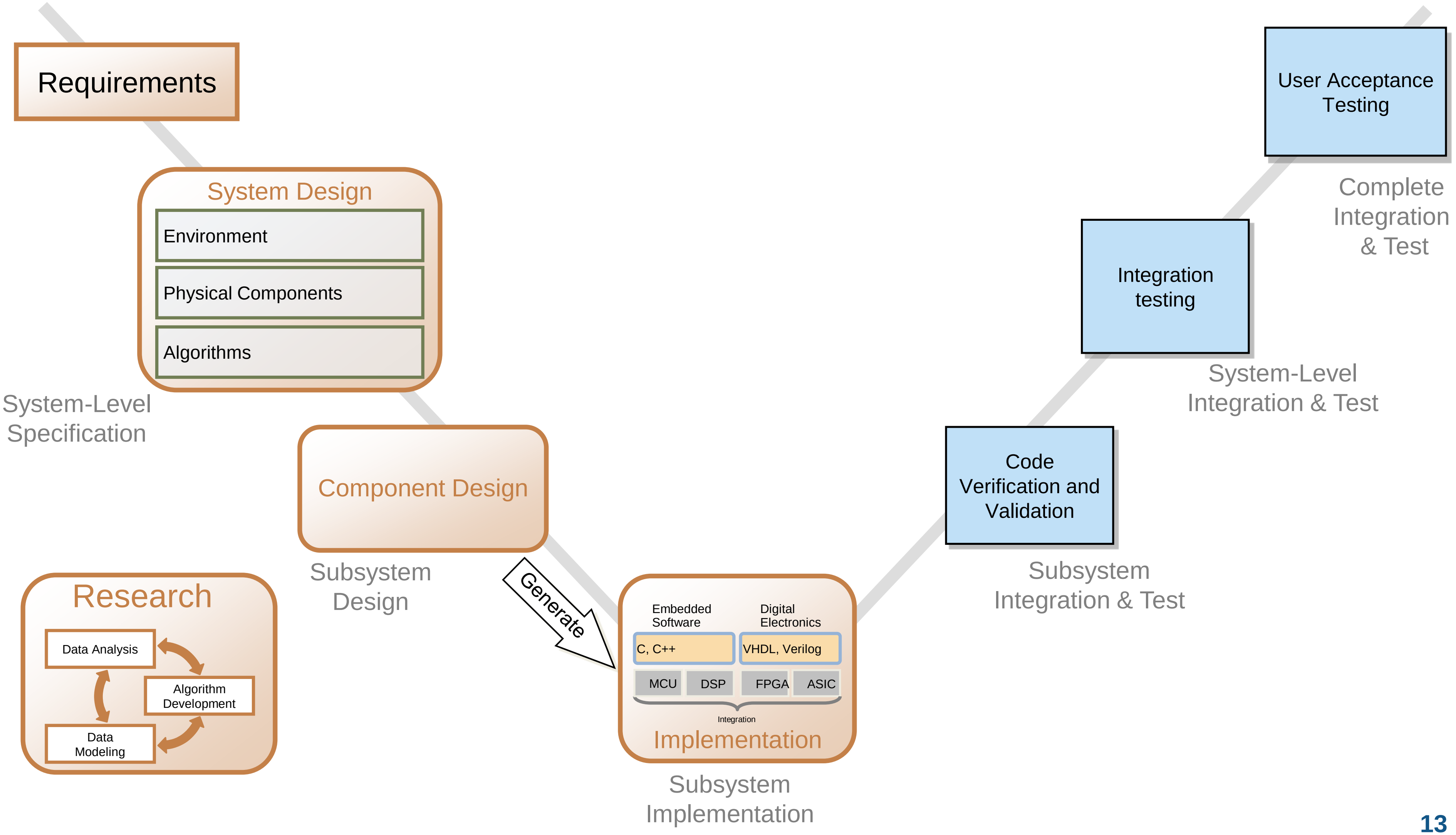
→



Detailed Model

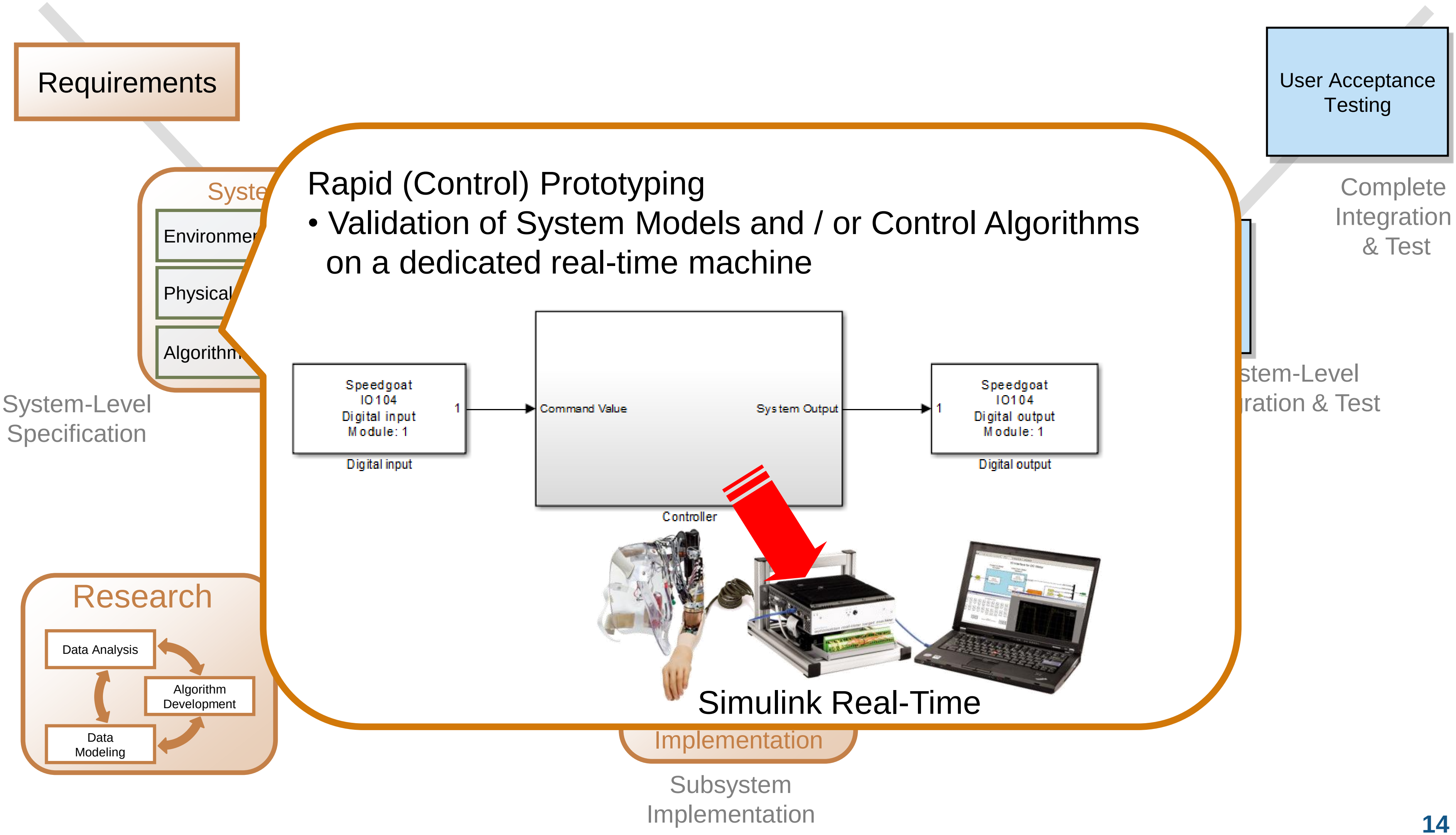
Model-Based Design

Development Process



Model-Based Design

Rapid Prototyping

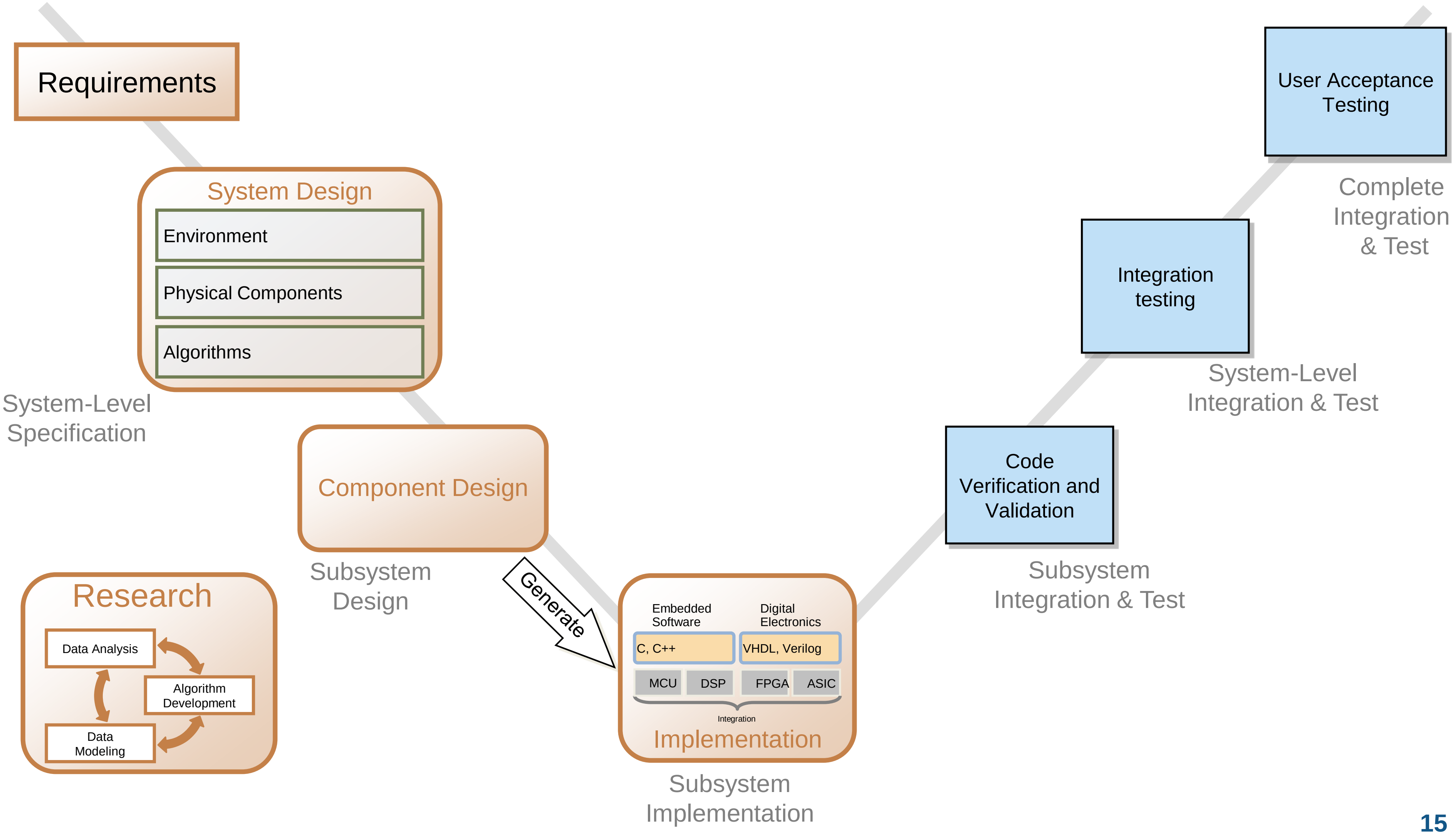


Rapid (Control) Prototyping

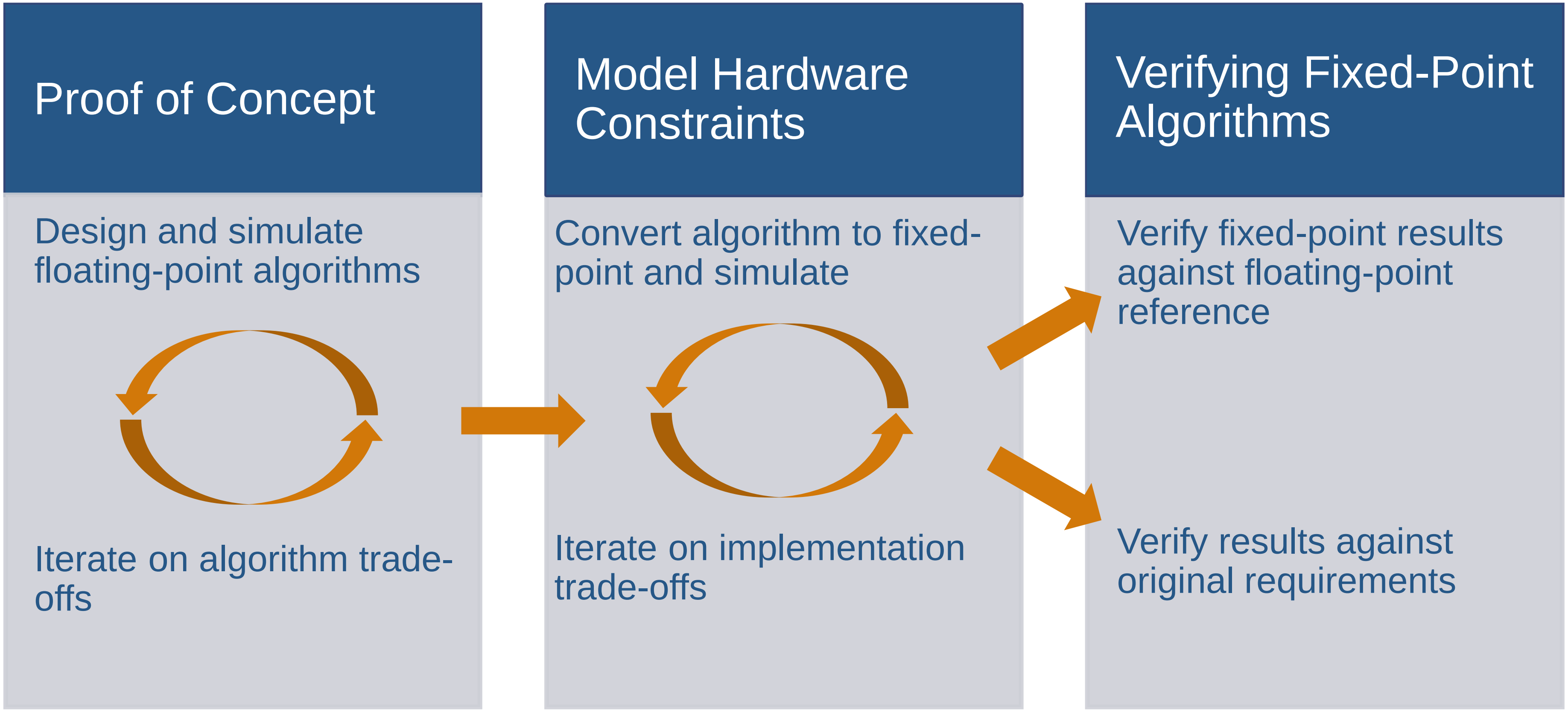
- Validation of System Models and / or Control Algorithms on a dedicated real-time machine

Model-Based Design

Development Process



Floating-Point to Fixed-Point Workflow



Proof of Concept

Design and simulate floating-point algorithms

Iterate on algorithm trade-offs

Model Hardware Constraints

Convert algorithm to fixed-point and simulate

Iterate on implementation trade-offs

Verifying Fixed-Point Algorithms

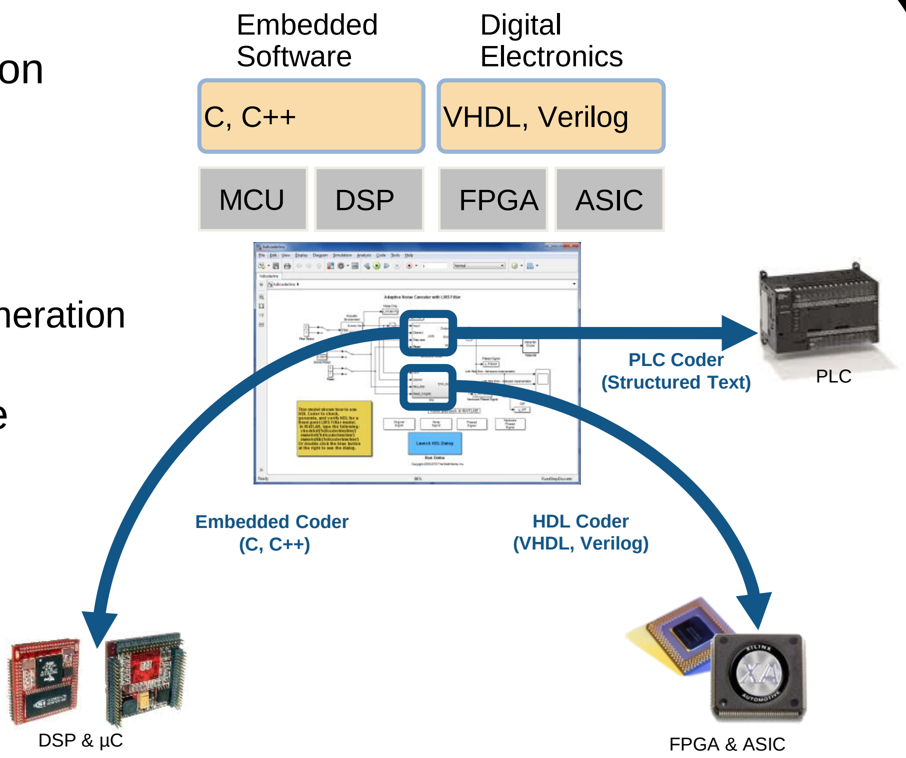
Verify fixed-point results against floating-point reference

Verify results against original requirements

Model-Based Design

Automatic Code Generation

- C/C++, VHDL and PLC-Code Generation from **one model**
- Support for Fixed Point Data Format
  - Automatic scaling
  - Supported in Simulation and Code-Generation
- Easy integration of legacy C/C++-Code
- System development independent of the target

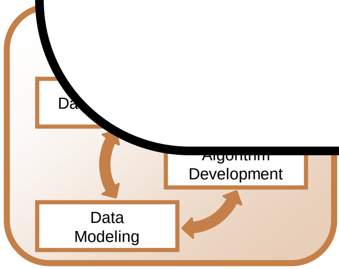


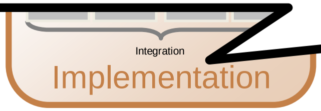
F

nce

plete
ation
est

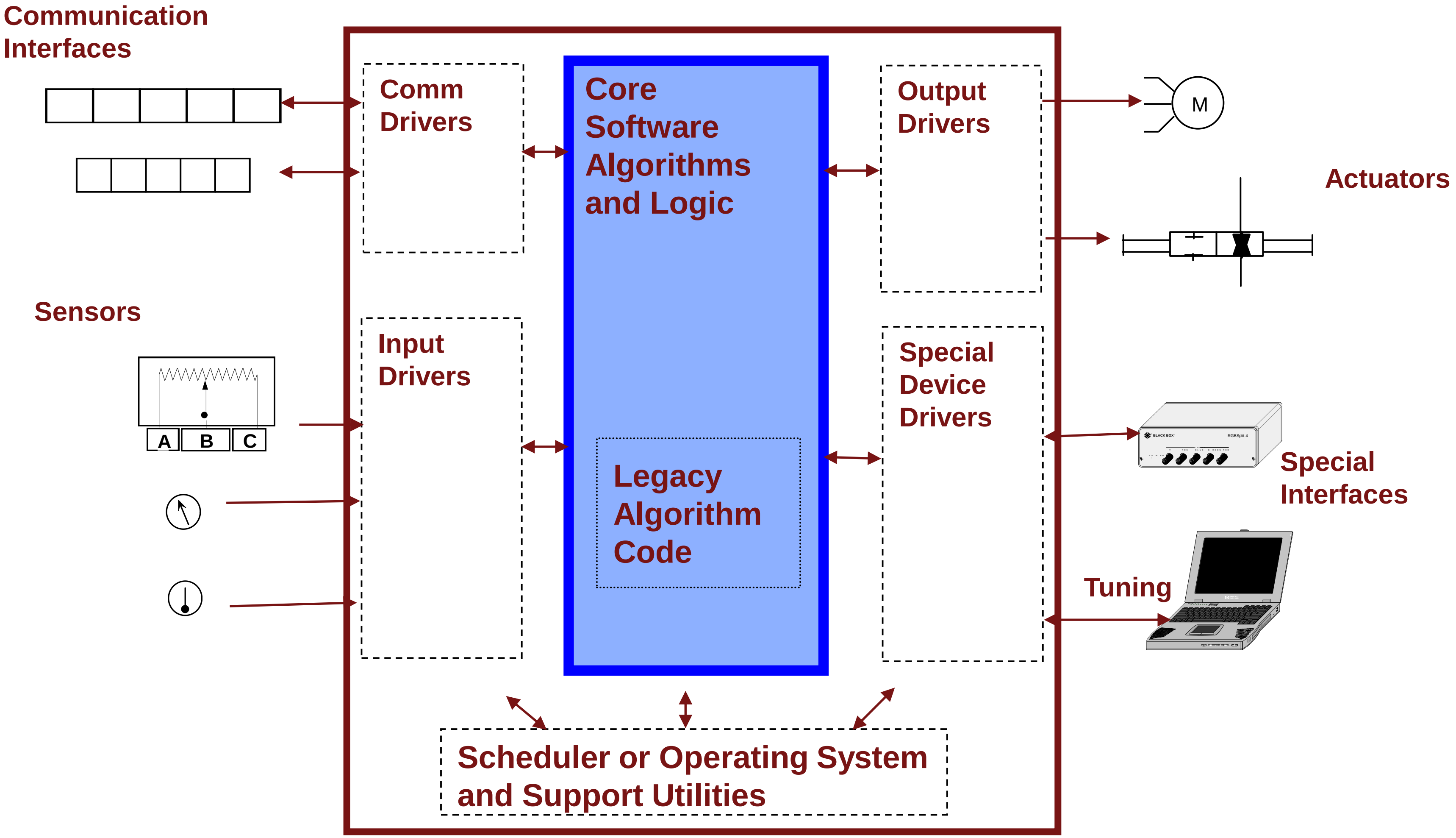
Syst
Spe





Subsystem Implementation

Integration



Communication Interfaces

Comm Drivers

Core Software Algorithms and Logic

Output Drivers

Actuators

Sensors

Input Drivers

Legacy Algorithm Code

Special Device Drivers

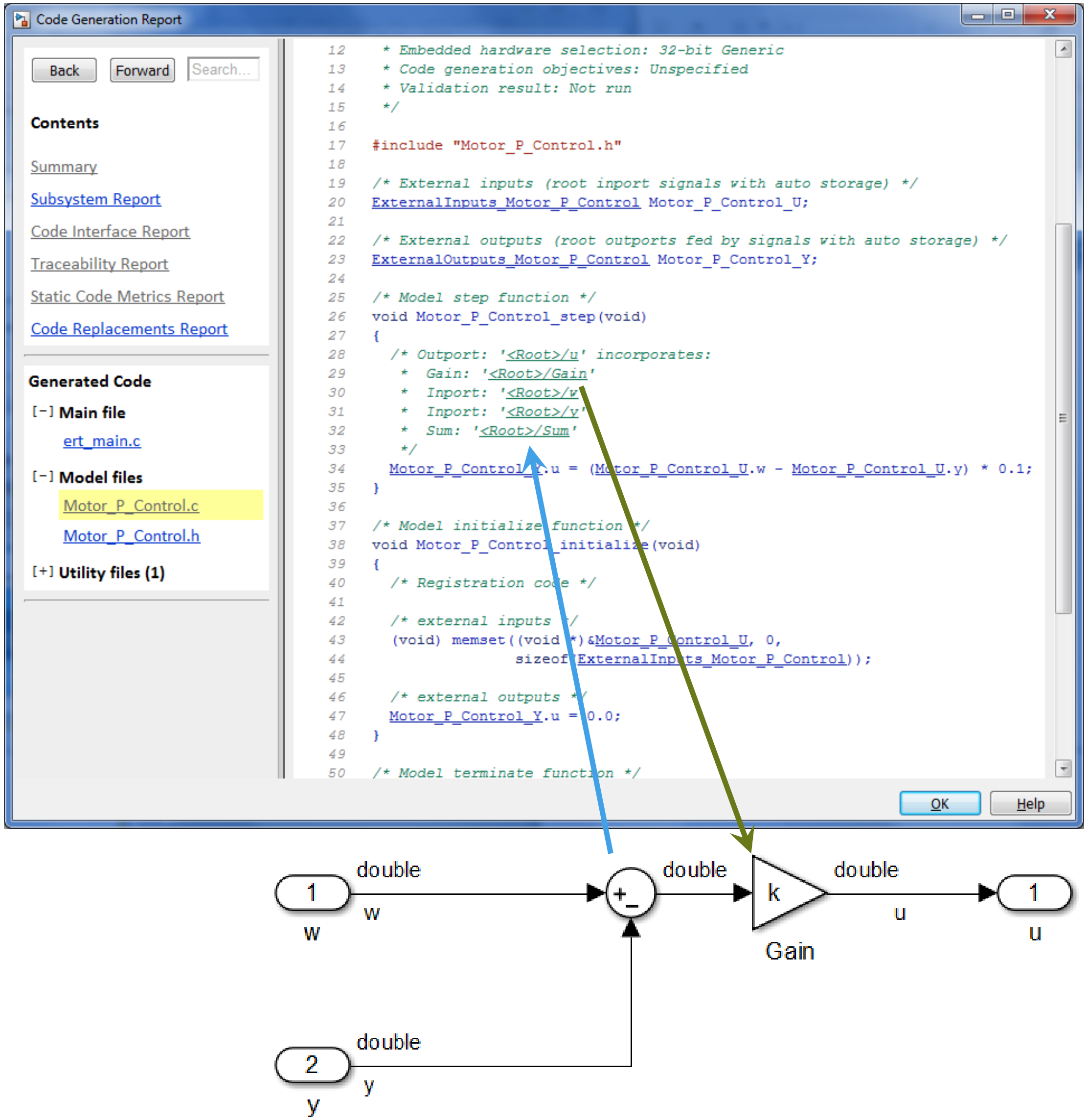
Special Interfaces

Tuning

Scheduler or Operating System and Support Utilities

HTML Code Generation Report

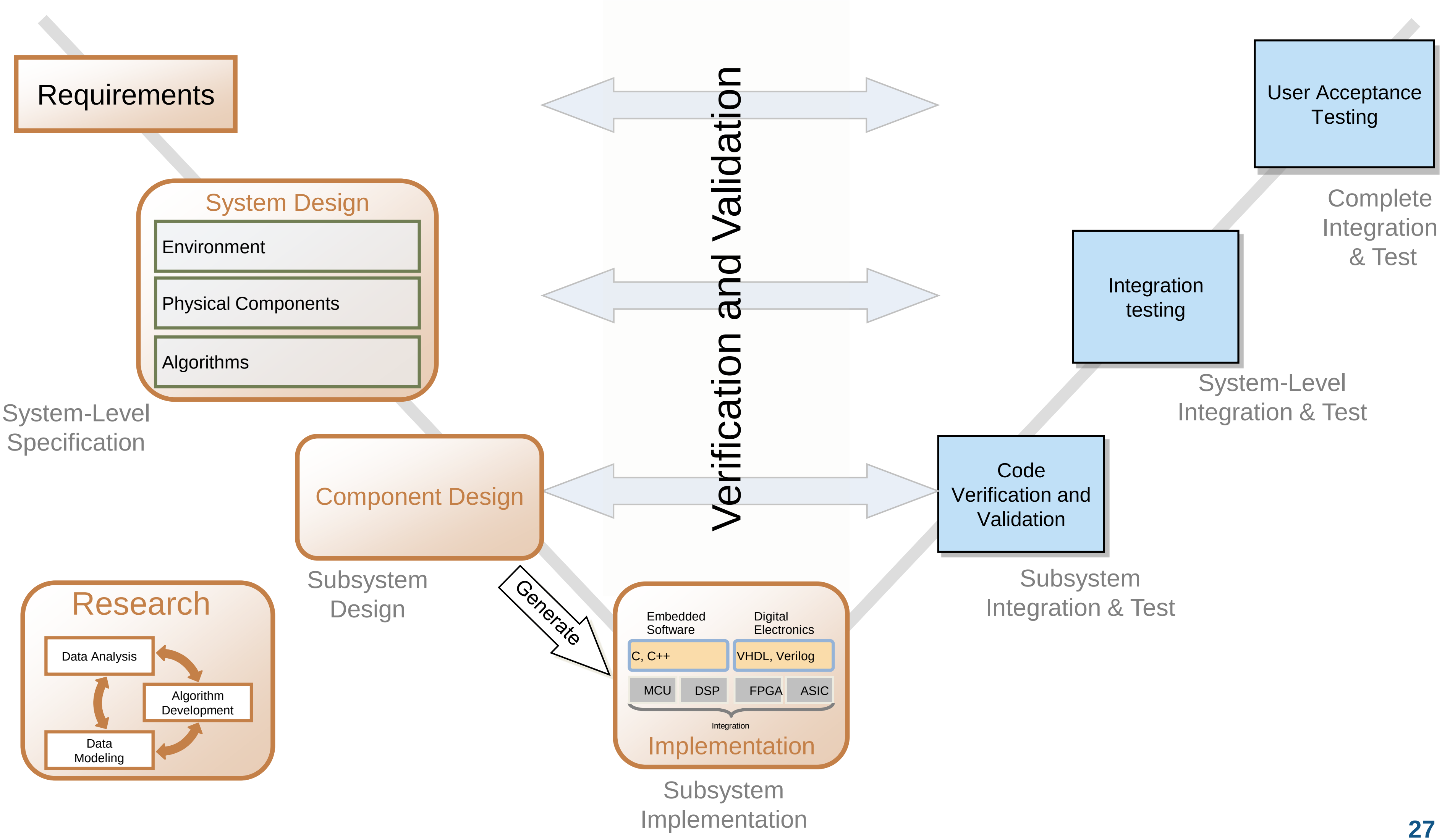
- Hyperlinks
  - •Code to Model
  - •Model to Code



```
 * Embedded hardware selection: 32-bit Generic
 * Code generation objectives: Unspecified
 * Validation result: Not run
 */

#include "Motor_P_Control.h"

/* External inputs (root inport signals with auto storage) */
ExternalInputs_Motor_P_Control Motor_P_Control_U;

/* External outputs (root outports fed by signals with auto storage) */
ExternalOutputs_Motor_P_Control Motor_P_Control_Y;

/* Model step function */
void Motor_P_Control_step(void)
{
  /* Outport: '<Root>/u' incorporates:
   *  Gain: '<Root>/Gain'
   *  Inport: '<Root>/w'
   *  Inport: '<Root>/y'
   *  Sum: '<Root>/Sum'
   */
  Motor_P_Control_Y.u = (Motor_P_Control_U.w - Motor_P_Control_U.y) * 0.1;
}

/* Model initialize function */
void Motor_P_Control_initialize(void)
{
  /* Registration code */

  /* external inputs */
  (void) memset((void *)&Motor_P_Control_U, 0,
                sizeof(ExternalInputs_Motor_P_Control));

  /* external outputs */
  Motor_P_Control_Y.u = 0.0;
}

/* Model terminate function */
```

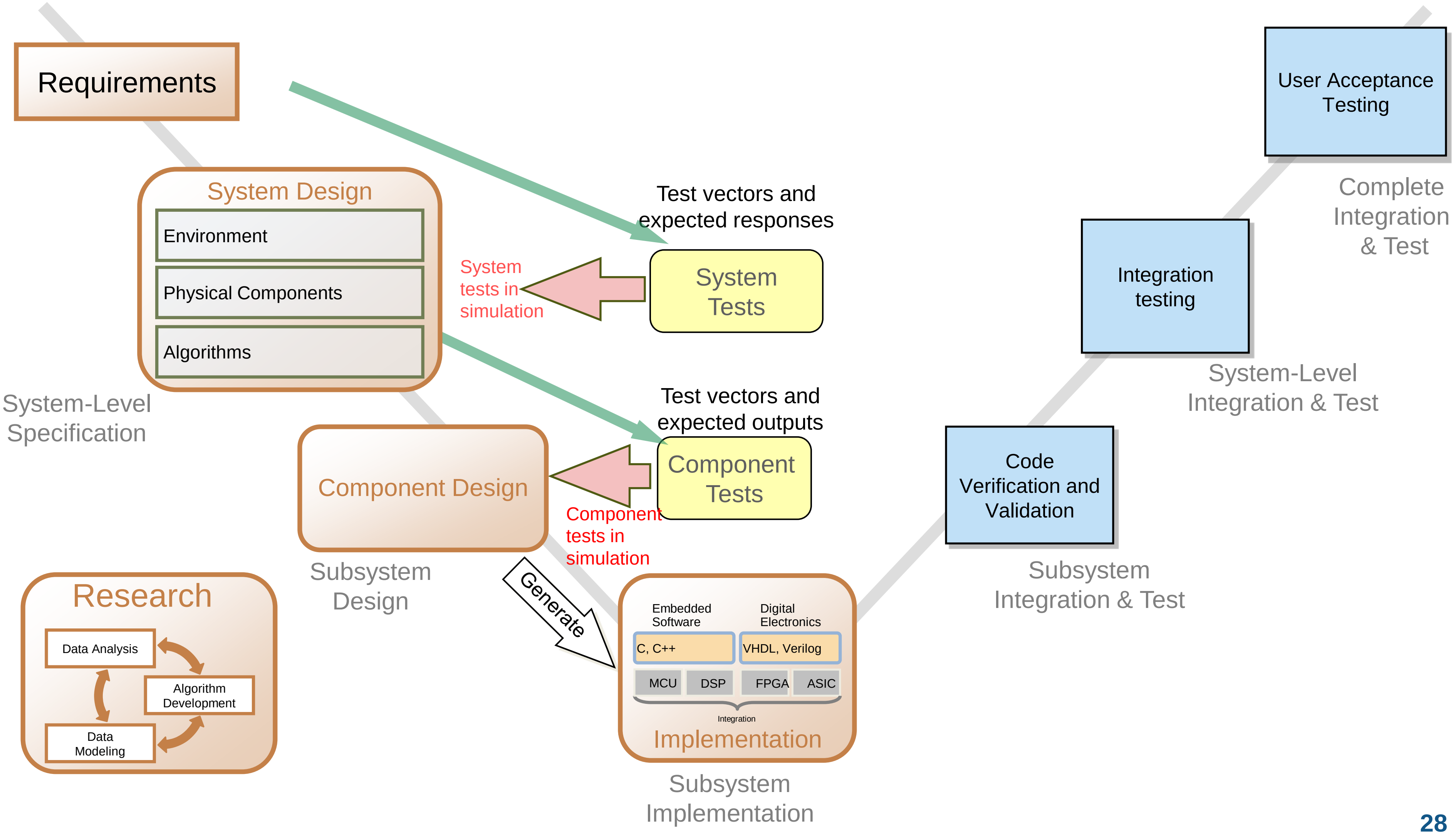
Model Based Design

Continuous Verification and Validation



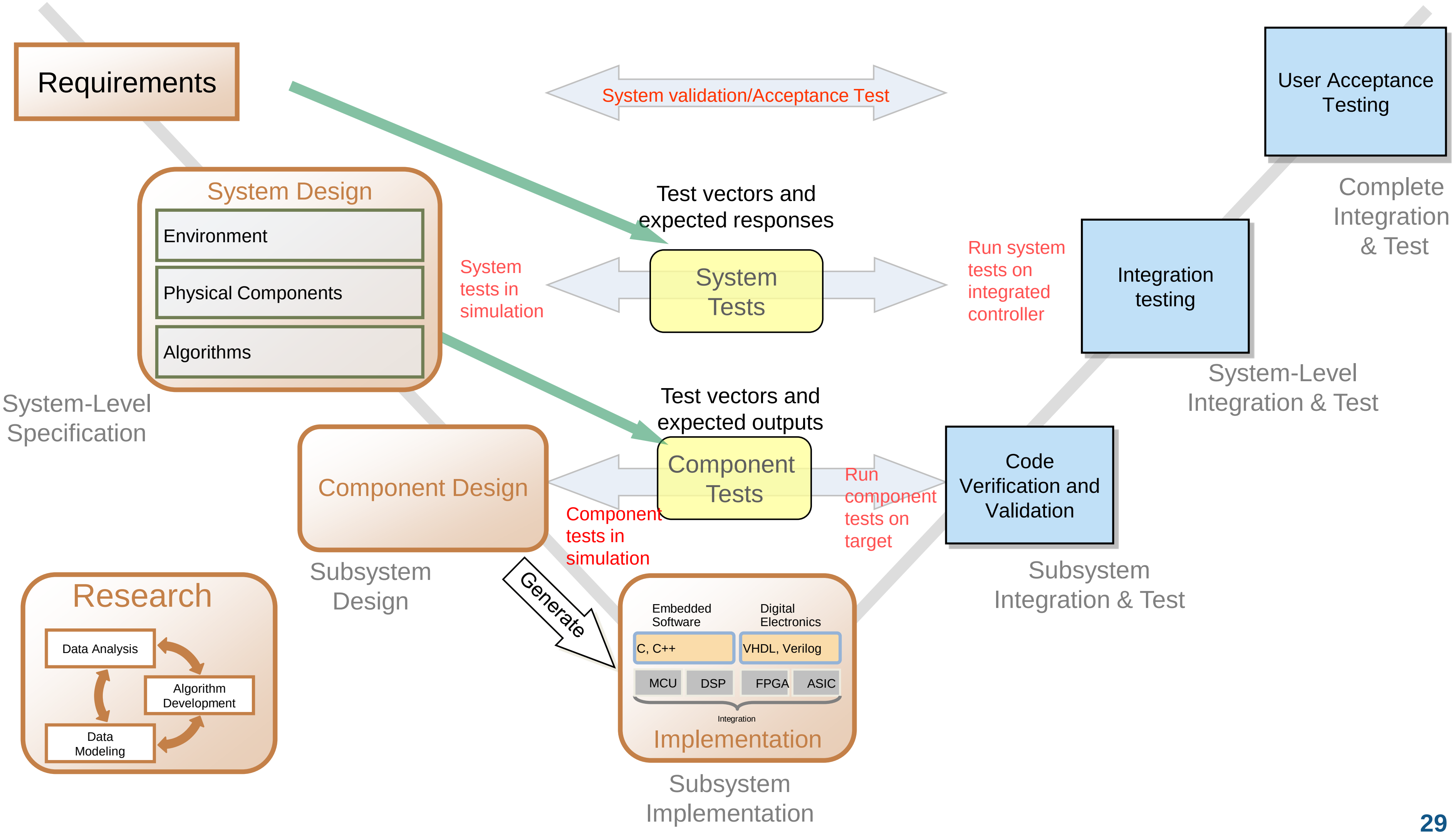
Model Based Design

Continuous Verification and Validation



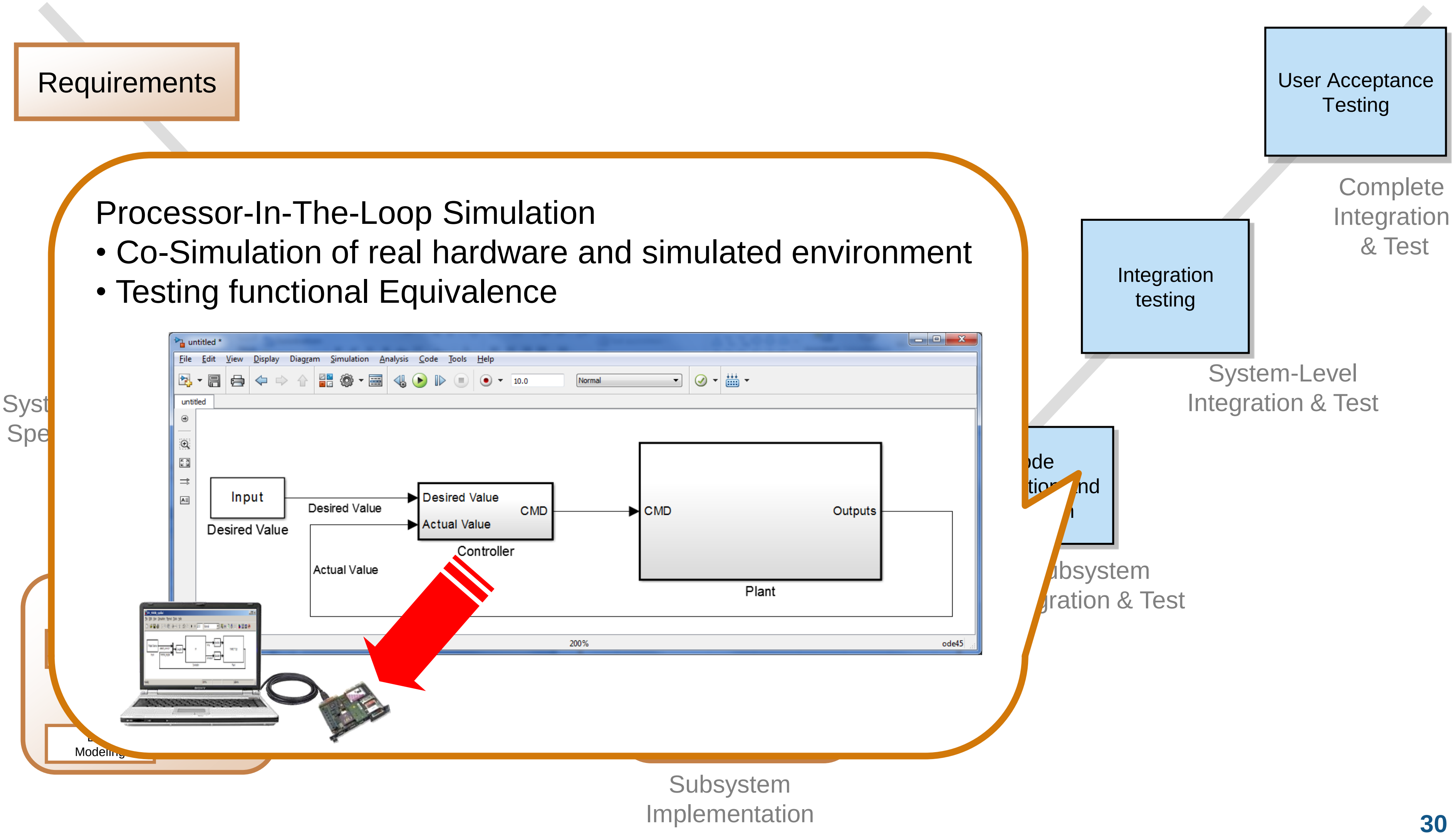
Model Based Design

Continuous Verification and Validation


Requirements
System validation/Acceptance Test
User Acceptance Testing
Complete Integration & Test
Test vectors and expected responses
System Design
Environment
Physical Components
Algorithms
System tests in simulation
System Tests
Run system tests on integrated controller
Integration testing
System-Level Integration & Test
System-Level Specification
Test vectors and expected outputs
Component Design
Component Tests
Component tests in simulation
Run component tests on target
Code Verification and Validation
Subsystem Integration & Test
Subsystem Design
Generate
Research
Data Analysis
Algorithm Development
Data Modeling
Embedded Software
Digital Electronics
C, C++
VHDL, Verilog
MCU
DSP
FPGA
ASIC
Integration
Implementation
Subsystem Implementation

Model Based Design

Subsystem-Level Integration & Testing

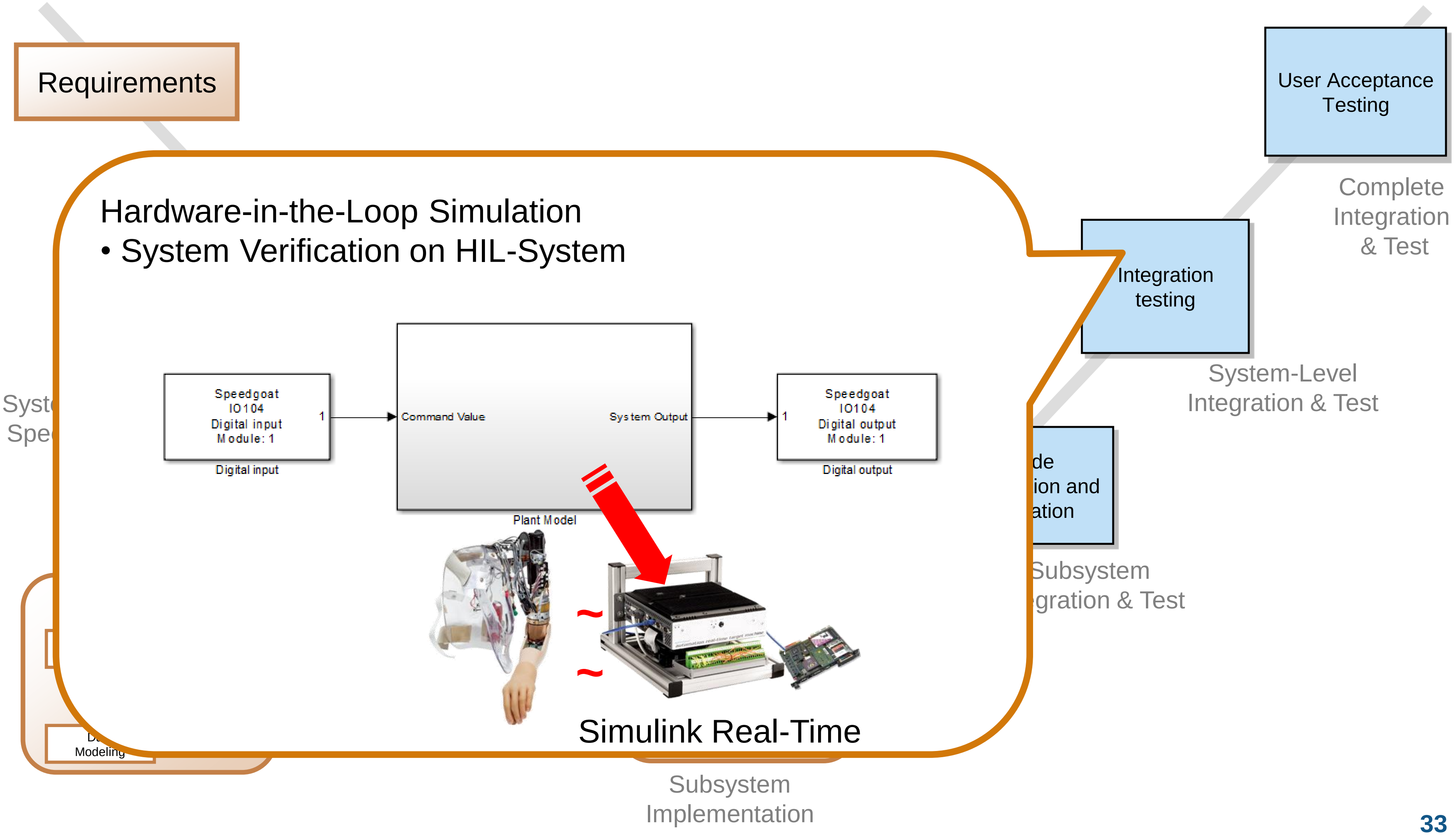


Processor-In-The-Loop Simulation

- Co-Simulation of real hardware and simulated environment
- Testing functional Equivalence

Model Based Design

System-Level Integration & Testing



Hardware-in-the-Loop Simulation

- System Verification on HIL-System

Simulink Real-Time

Requirements

User Acceptance Testing

Complete Integration & Test

Integration testing

System-Level Integration & Test

de ion and ation

Subsystem gration & Test

Subsystem Implementation

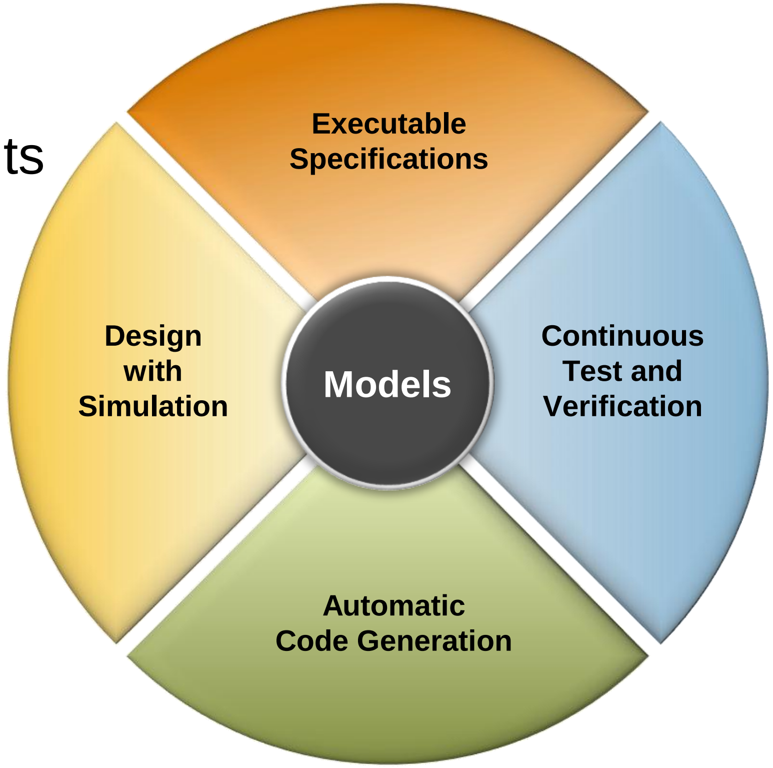
Syst Spe

Modeling

Speedgoat IO104 Digital input Module: 1
1
Digital input
Command Value
System Output
Plant Model
1
Speedgoat IO104 Digital output Module: 1
Digital output

Benefits of Model-Based Design

- Models
  - Core of the Development Process
- Unambiguous Description of Requirements
  - Executable Specification
- Fast Evaluation of Design Variants
  - Simulation
- Early Test and Verification
- Automatic Code Generation


Executable Specifications
Design with Simulation
Models
Continuous Test and Verification
Automatic Code Generation

⇨ Better Cooperation, Communication and Collaboration

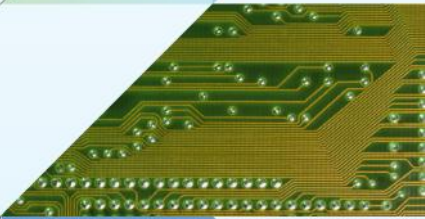
⇨ means for quick what/if scenarios

⇨ Higher Product Quality

MATLAB EXPO 2014
DEUTSCHLAND

9. Juli – *München*

Jetzt anmelden: matlabexpo.de



MathWorks

Change the world by

Accelerating the pace

of discovery, innovation, development, and learning

in engineering and science