The Impact of Model-based Development in the Software Lifecycle

John Favaro

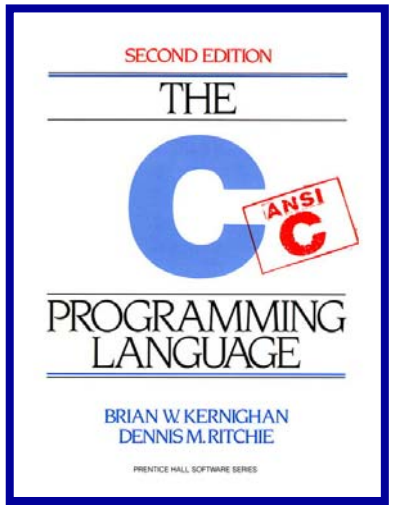
AutoSPIN Italia
Milano
4 June 2009

Prologue

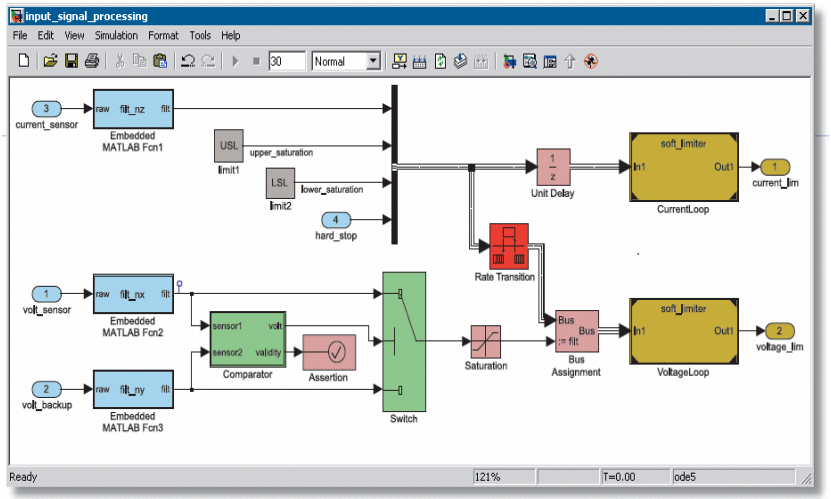
"In many ways, the move to model-based code generation parallels the move from assembly to high-level languages. Each move along the path is **a step up the abstraction ladder.** Each step frees the developer from some of the gritty details of programming."

- NASA Safety Guidebook (March 2004)

State of the Practice



- ANSI C
- MISRA C
- Standard V&V tools



- Simulink / Stateflow (Mathworks)
- Targetlink (dSPACE)
- MISRA SLSF, MISRA TL
- V&V Toolbox (Mathworks)

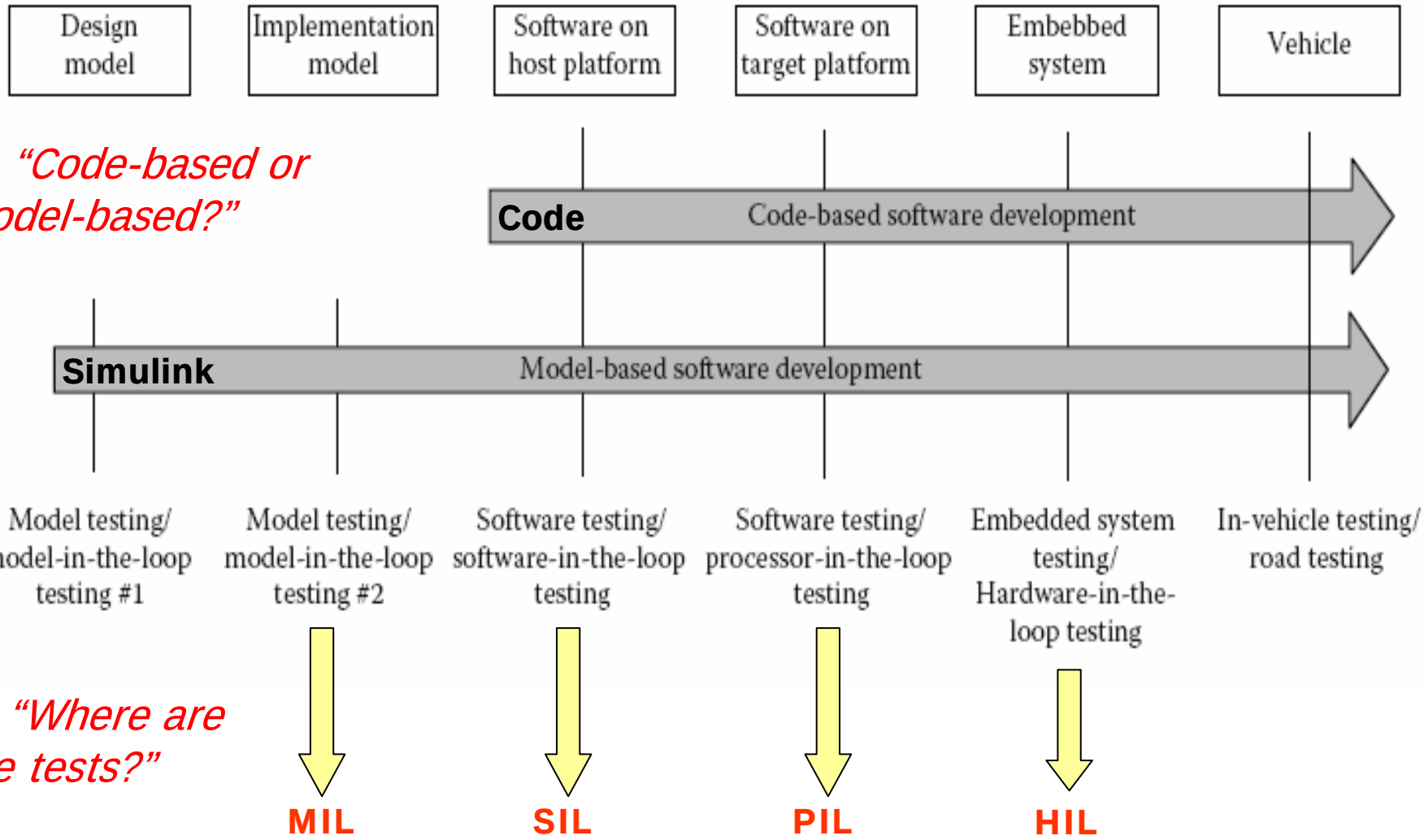
Lifecycles

In the **Automotive** domain we typically find today:

- The '**code based**' software lifecycle:
  - **Unit Test** on host where unit of reference is **.c** file
  - **Integration Test** on host where unit of reference is a subset of **.c** files
  - **Software Test** on target of entire software application integrated with reference hardware (*ECU-level test*)

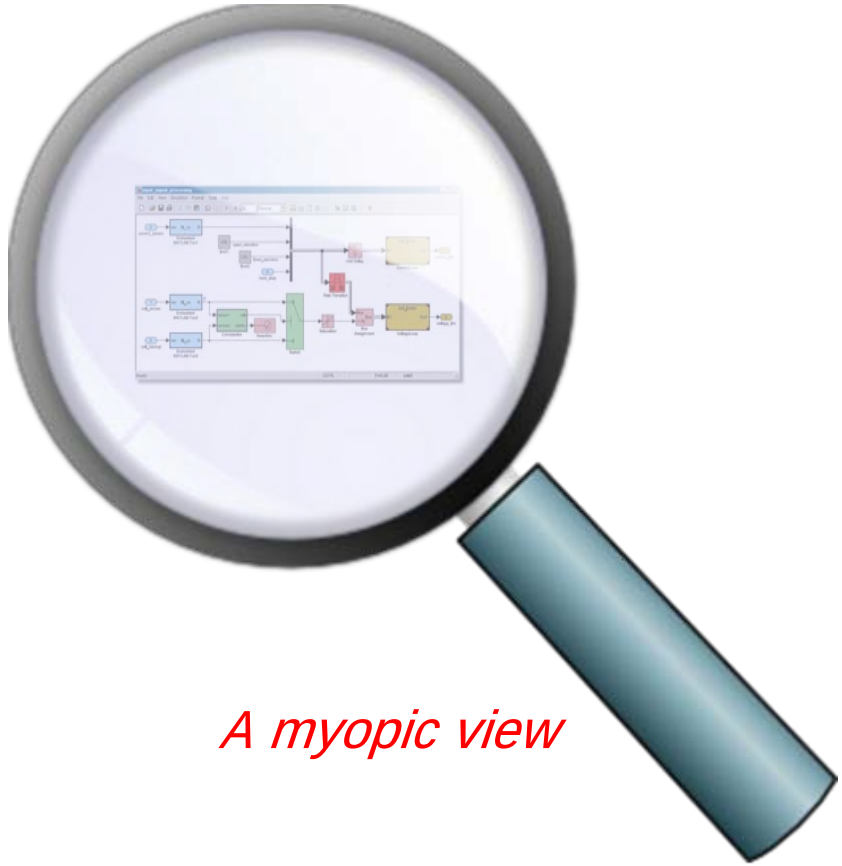
- The '(**Simulink) Model Based**' software lifecycle (requirements-based test):
  - **Unit Test** on host where unit of reference is Simulink model **.mdl** file, executed both on model and on software (*back-to-back testing*) → MIL, SIL
  - **Integration Test** on host where unit of reference is subset of **.mdl** files, executed only on model → MIL
  - **Software Test** on target of entire software application integrated with reference hardware (*ECU-level test*) → HIL

Where are the tests?



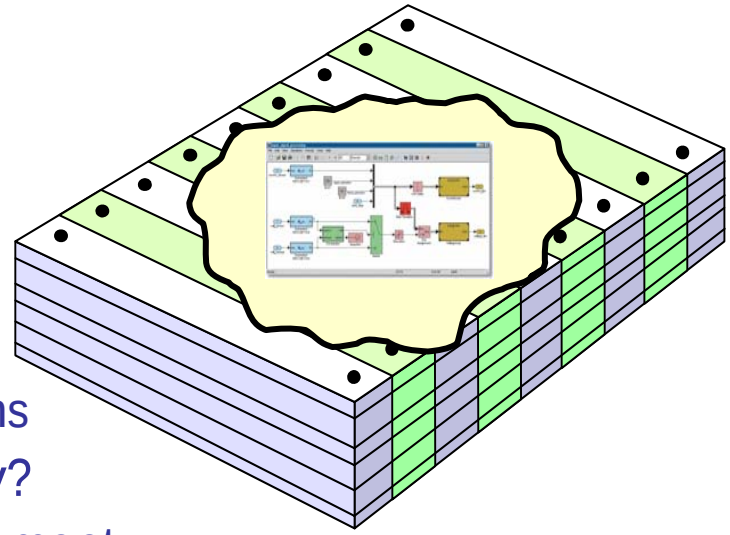
But … a Myopic View

- Thanks to Simulink, Model Based Development is widely and successfully deployed in the automotive industry today
  - High level of abstraction w.r.t. textual descriptions, great for intra- and inter-team communication
  - Automated coding implementation
  - Formalization of test environment
- An **enormous achievement and advantage**: we are comfortable with a model-based approach in the automotive industry
- **But ultimately it is a myopic view**



A myopic view

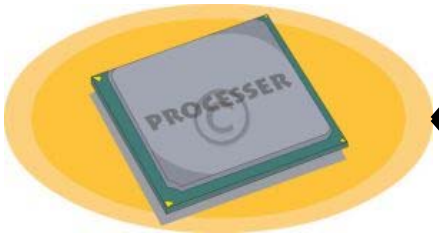
Broadening our Perspective



- Traditionally embedded systems in the automotive industry have been
  - Relatively small
  - Monolithic
  - Oriented toward control of dynamic systems
- But where are new problems arising today?
  - Inadequate software configuration management
  - Difficulties in managing system/software families
  - Difficulties in managing reuse of components
  - Difficulties in managing distributed development of large, software intensive systems
  - Necessity to deal with multiple platforms
- **These new problems are systems and software engineering problems**!

The traditional domination of the control element is decreasing

ECU Categories Today



High End

- 1 Gb memory
- Multimedia & Infotainment

Mid-Range

- 8 Mb memory
- Network gateways

Low End

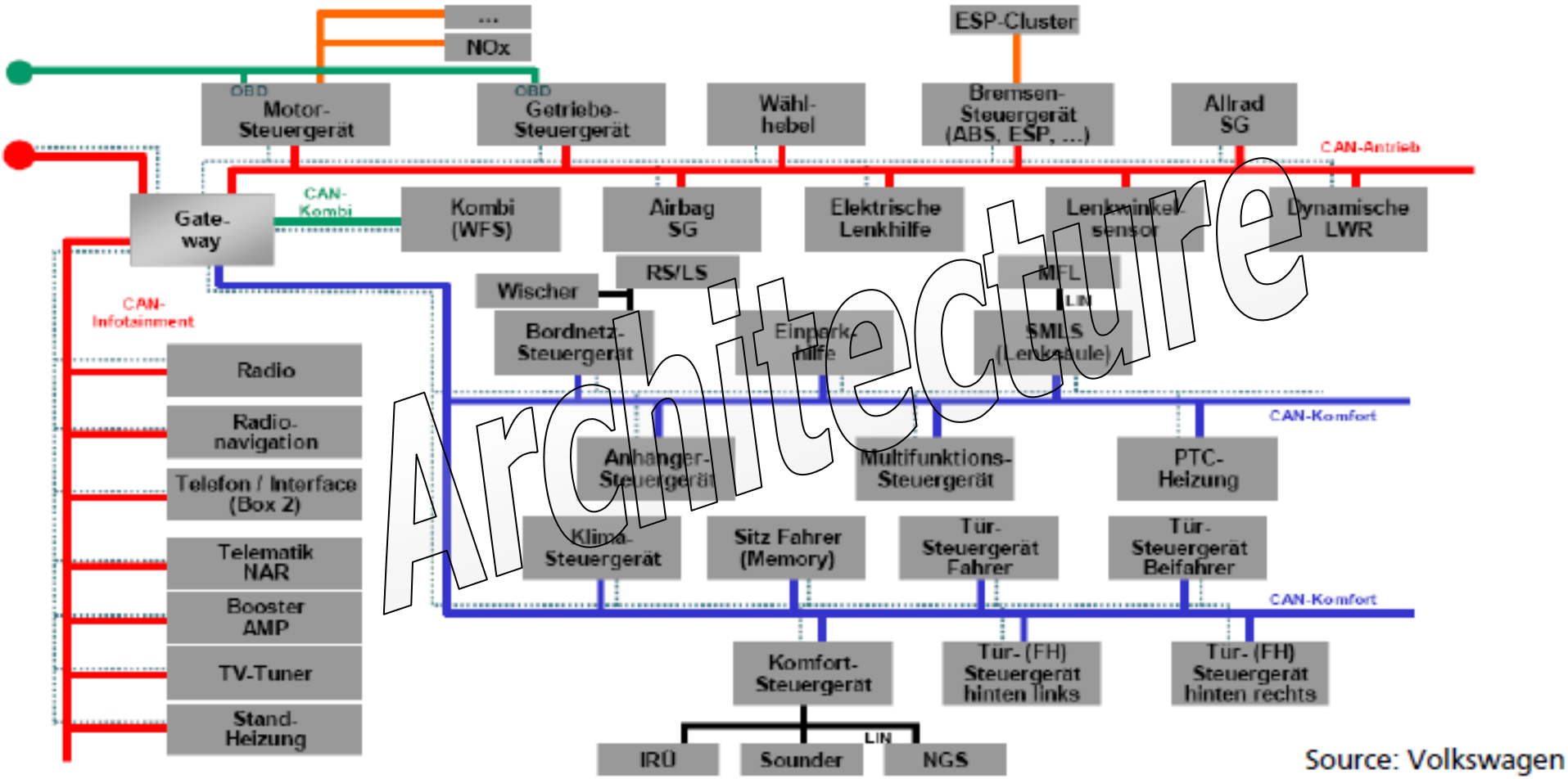
- < 1 Mb memory
- Engine Control

Systems and Software Engineering

Control Engineering

An increase in processing power is changing the nature of embedded automotive systems

Automotive Systems Today

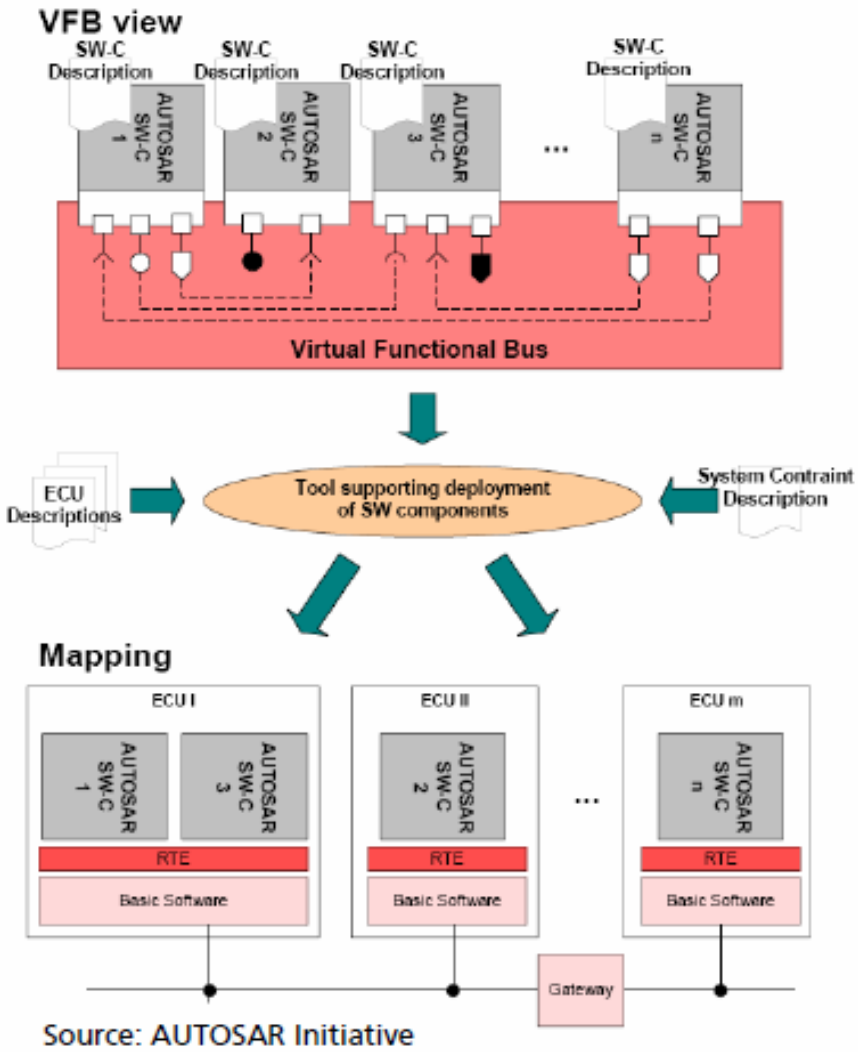


Up to 80 ECUs on a highly distributed system with different networks

Automotive Architecture

- AUTOSAR understands the nature of the emerging problems today: **architecture**
  - System, not ECU centric approach
  - Component-based, not monolithic approach
  - Well defined interfaces for distributed component development
  - Platform independent

- *Exactly the issues we listed earlier*



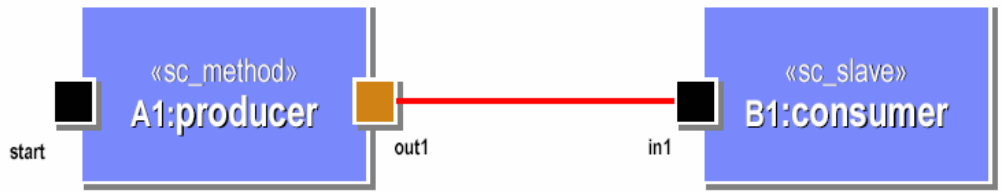
Models Capture Architecture

```
SC_MODULE(producer)
{
sc_outmaster<int> out1;
sc_in<bool> start; // kick-start
void generate_data ()
{
for(int i =0; i <10; i++) {
out1 =i ; //to invoke slave;}
}
SC_CTOR(producer)
{
SC_METHOD(generate_data);
sensitive << start;}};
SC_MODULE(consumer)
{
sc_inslave<int> in1;
int sum; // state variable
void accumulate (){
sum += in1;
cout << "Sum = " << sum << endl;}
```

Code doesn't capture architecture ...

```
SC_CTOR(consumer)
{
SC_SLAVE(accumulate, in1);
sum = 0; // initialize
};
SC_MODULE(top) // container
{
producer *A1;
consumer *B1;
sc_link_mp<int> link1;
SC_CTOR(top)
{
A1 = new producer("A1");
A1.out1(link1);
B1 = new consumer("B1");
B1.in1(link1);}};
```

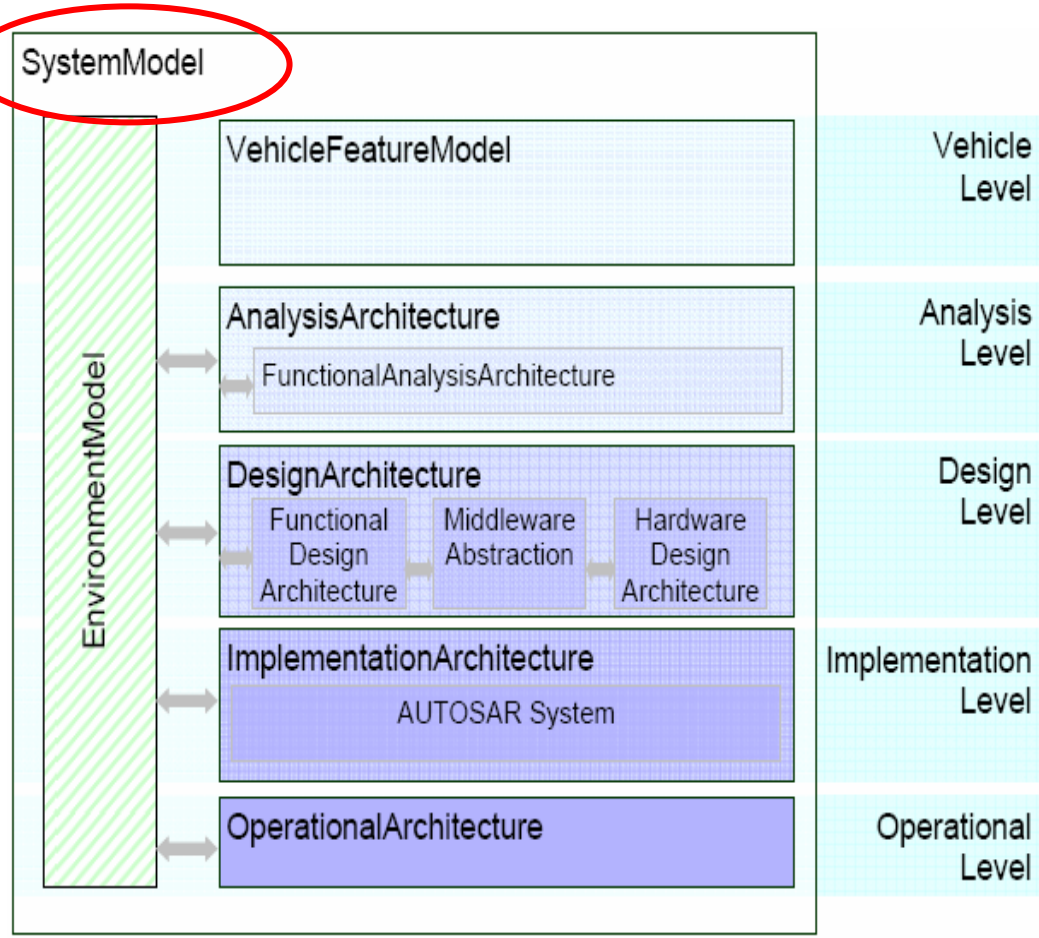
... models do



Source: B. Selic

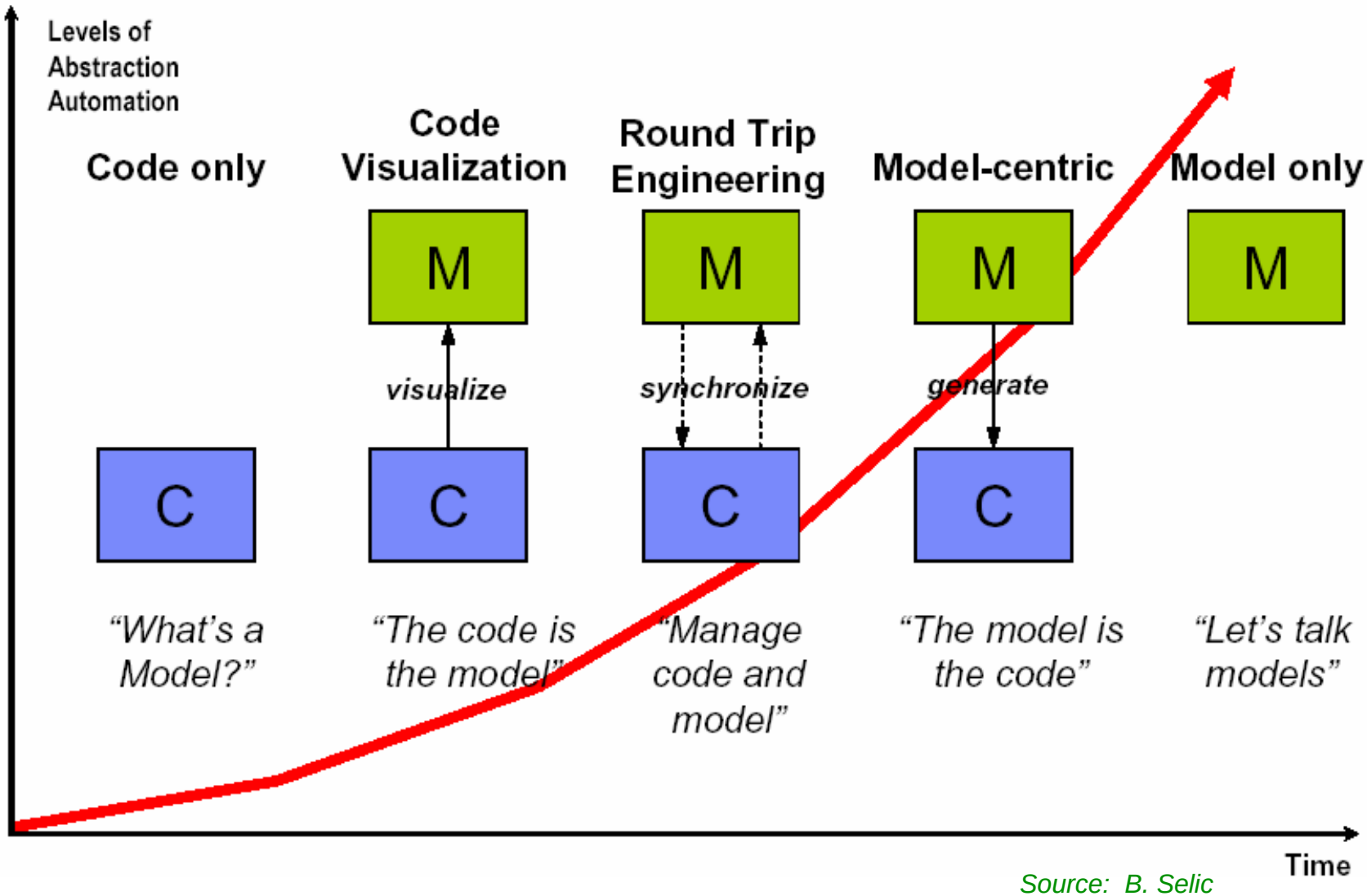
An Opportunity

- The automotive industry is uniquely positioned to leverage its experience with model-based development
  - Few industries have that advantage!
- Expanding the model based approach to the full automotive system and software architecture will give us the same advantages that Simulink has already given us in the monolithic, control-oriented domain



EAST Architectural Description Language

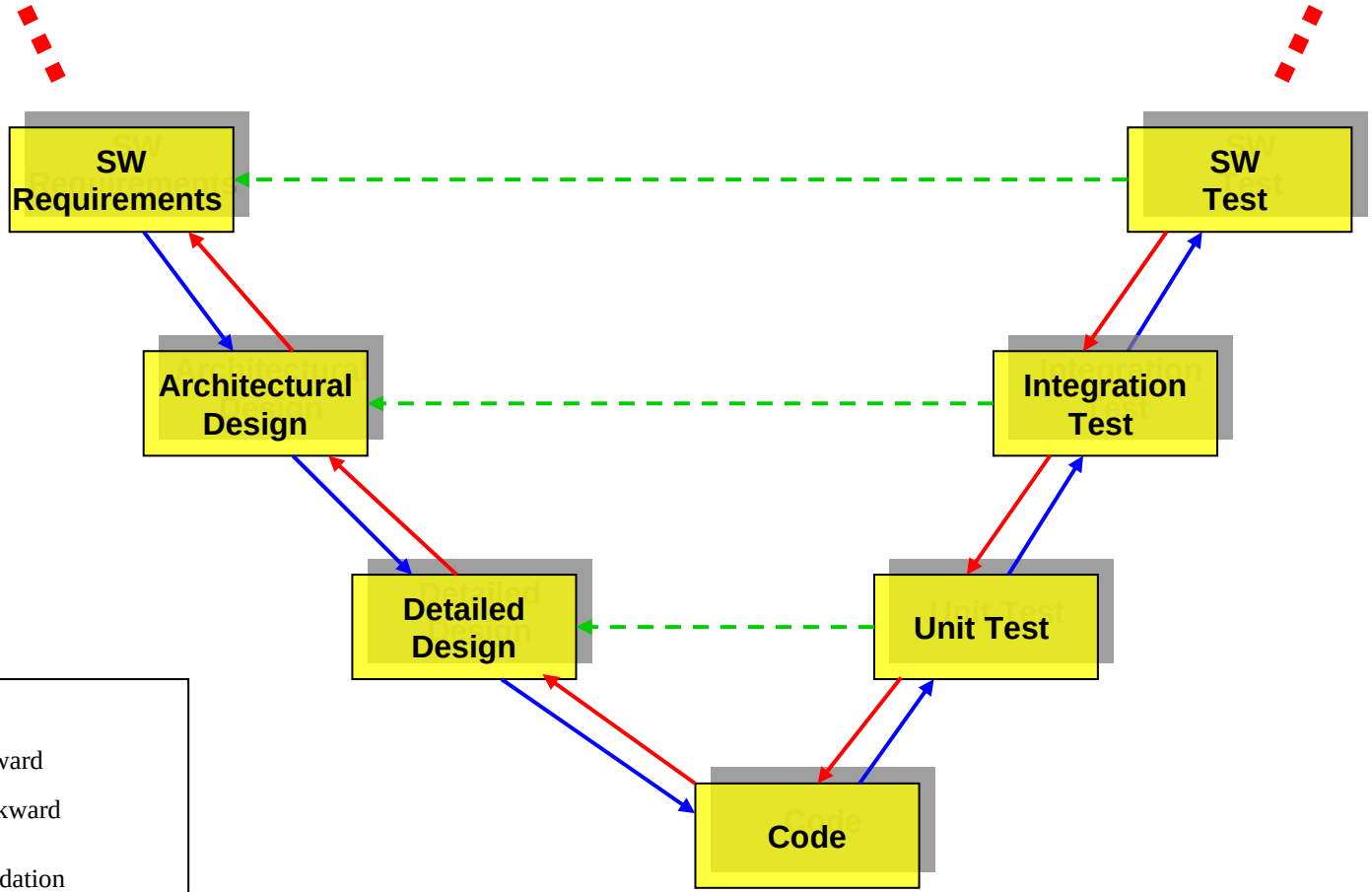
The MBD Maturity Model



Source: B. Selic

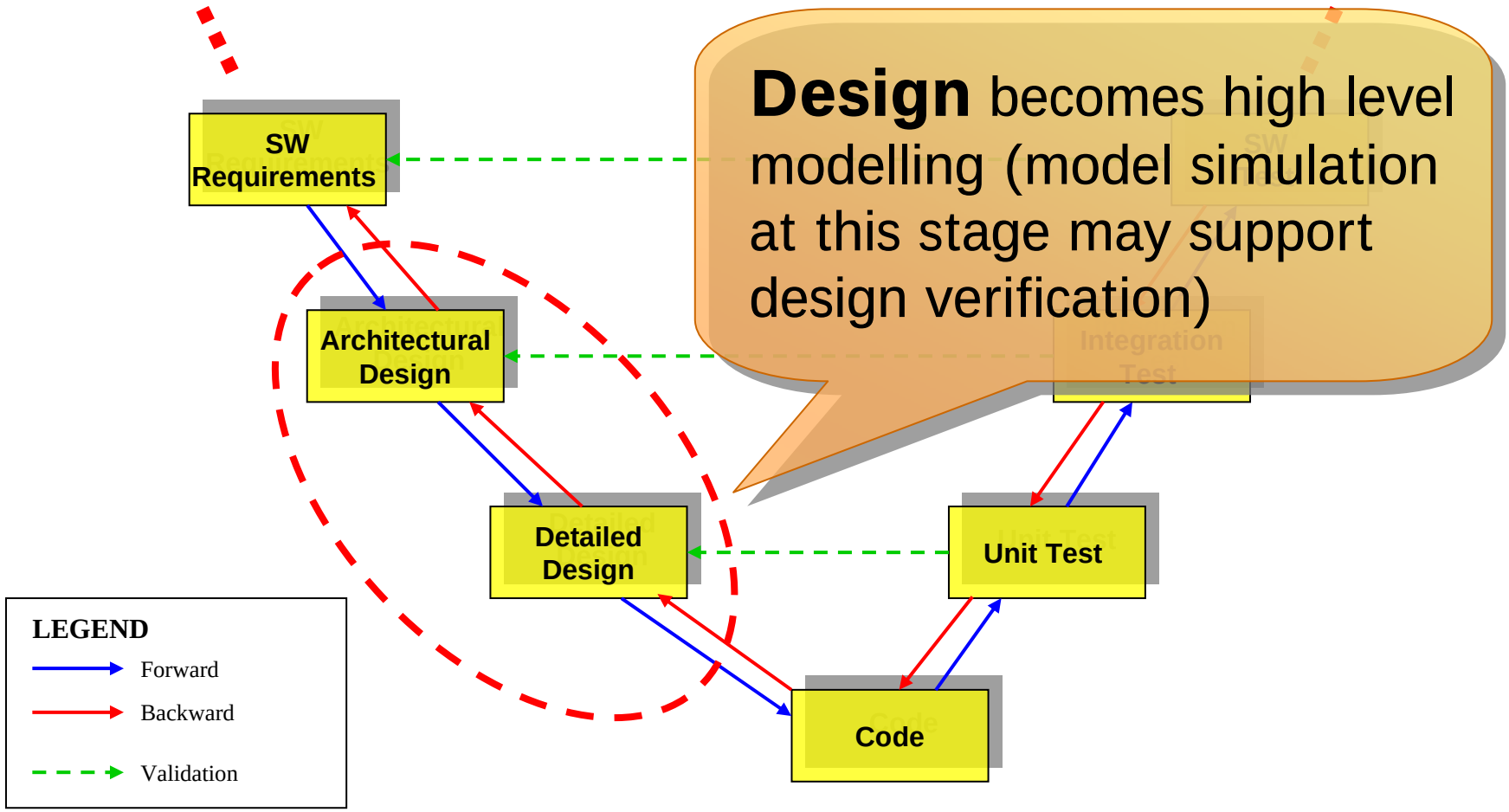
Traditional “V” Life Cycle

How will Model Based Development change the development lifecycle?



Design

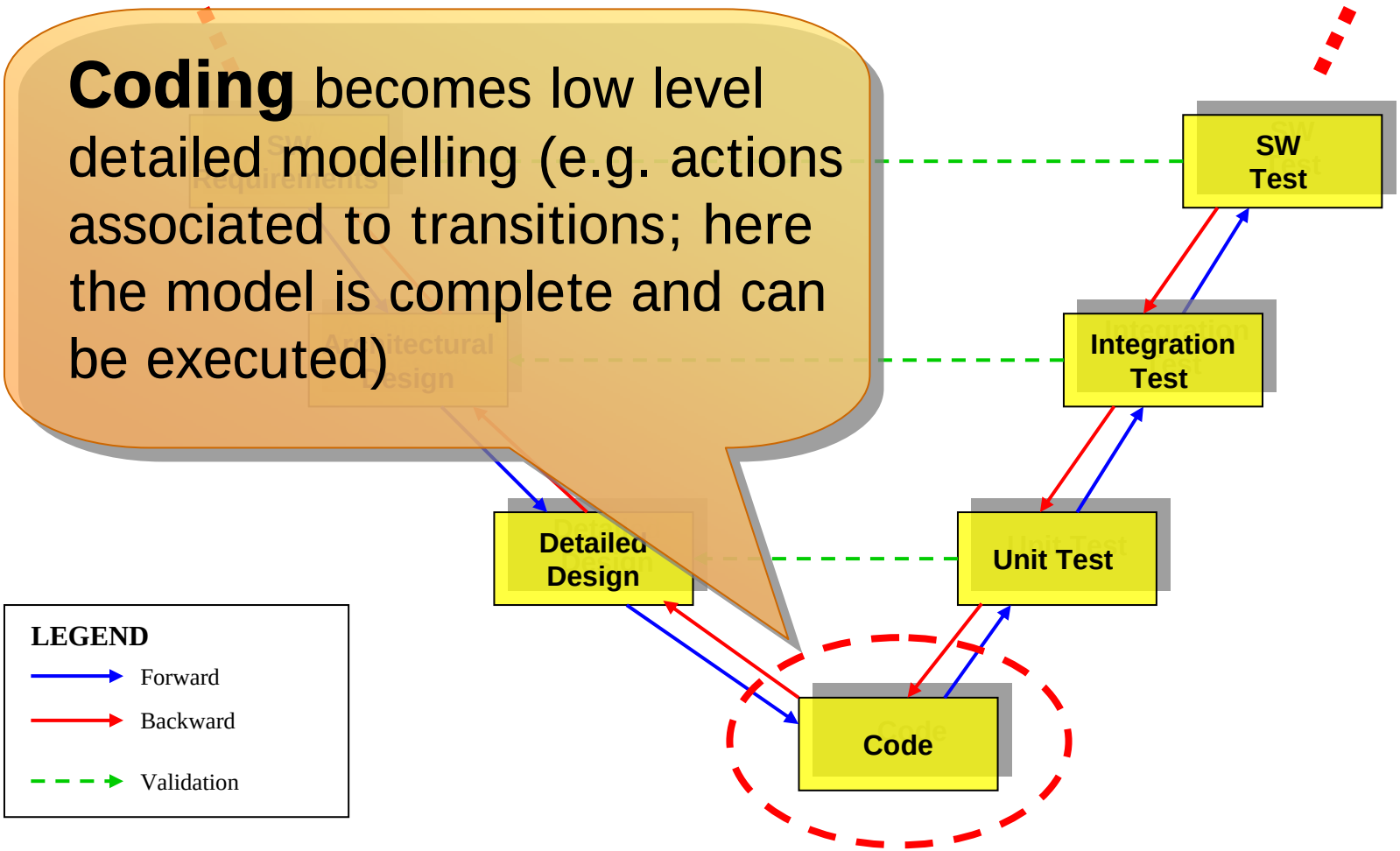
How will Model Based Development change the development lifecycle?



Coding

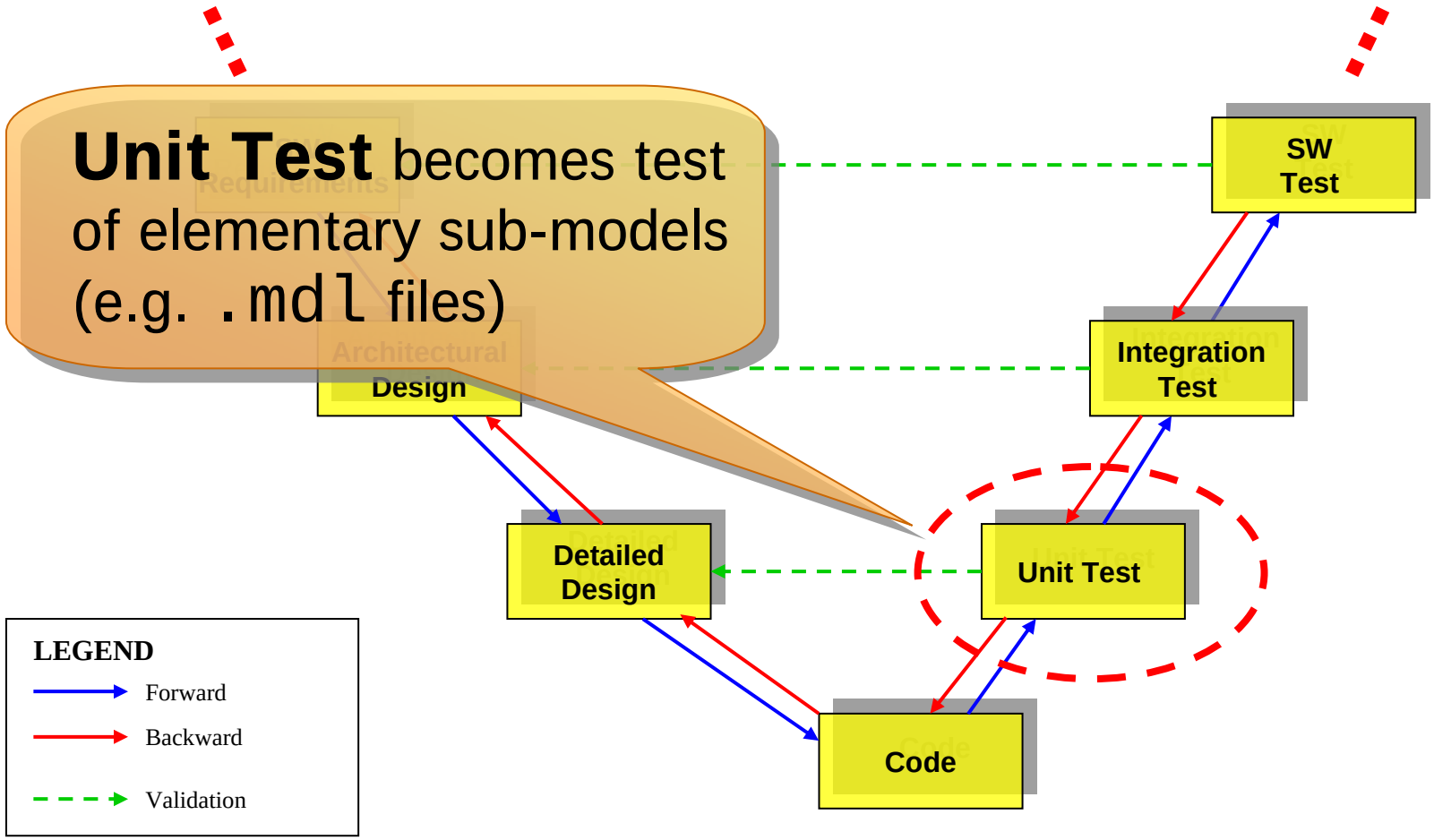
How will Model Based Development change the development lifecycle?

Coding becomes low level detailed modelling (e.g. actions associated to transitions; here the model is complete and can be executed)



Unit Test

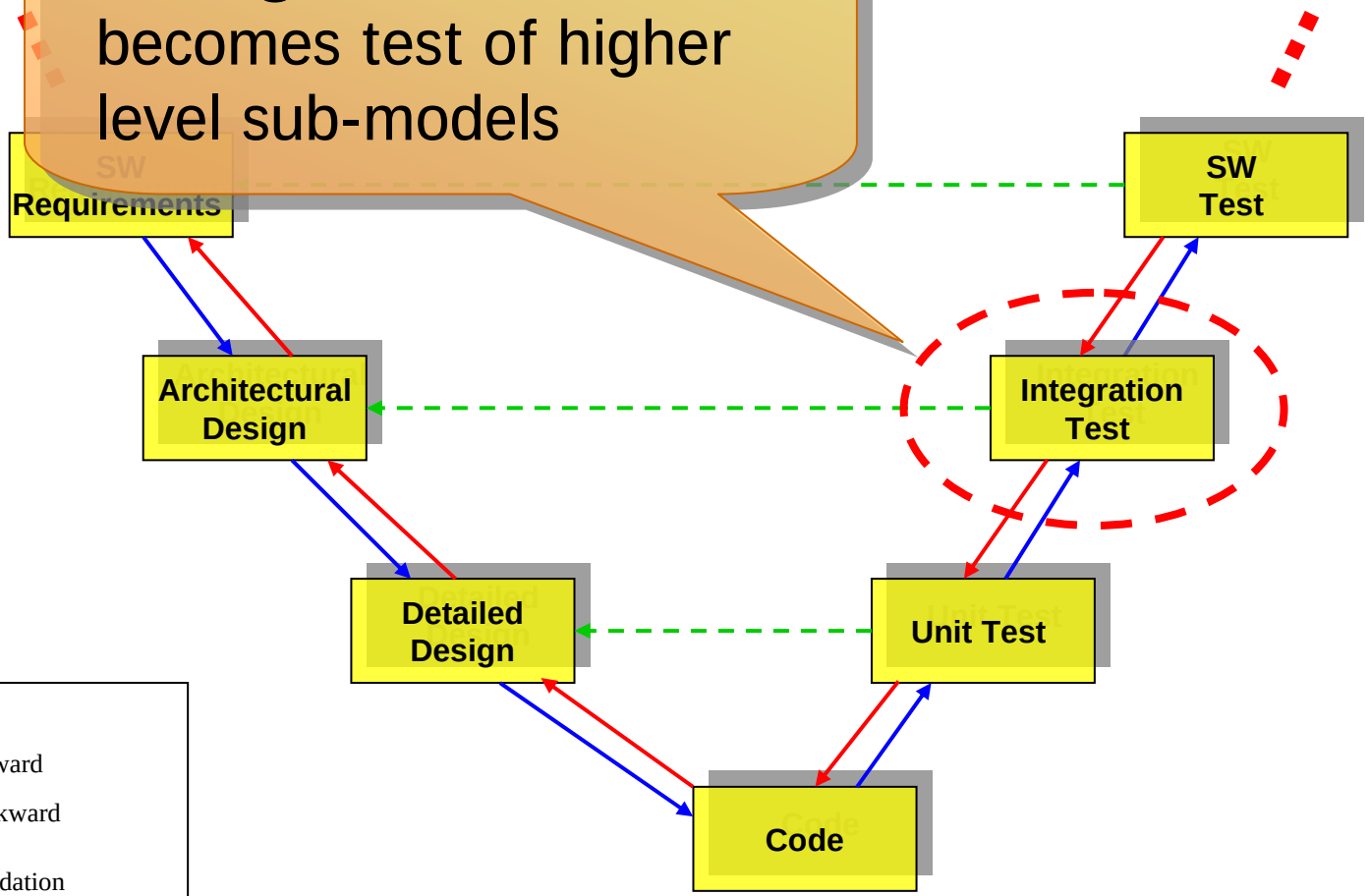
How will Model Based Development change the development lifecycle?



Integration Test

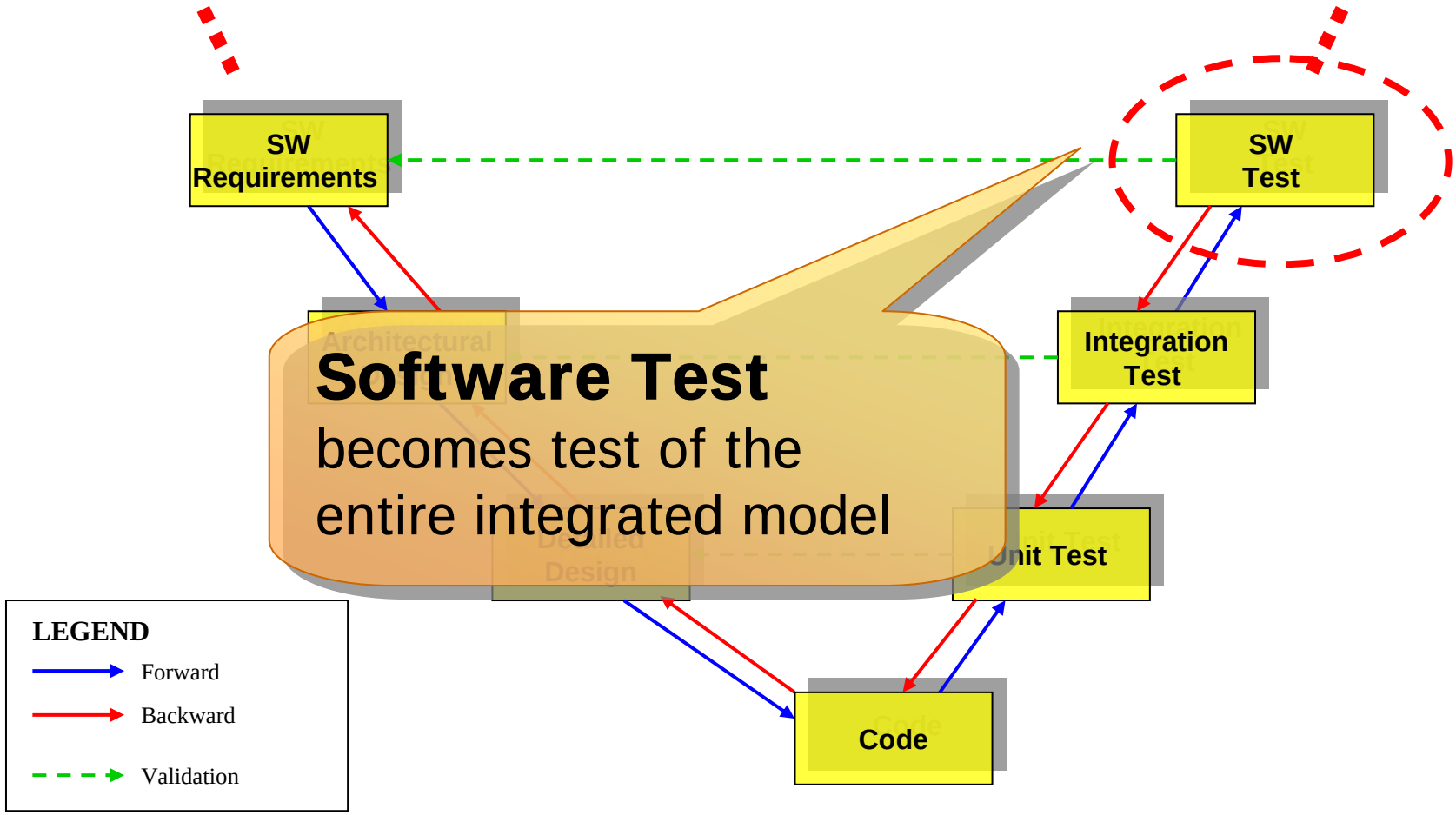
How will Model Based Development change the development lifecycle?

Integration Test
becomes test of higher level sub-models



Software Test

How will Model Based Development change the development lifecycle?



No Revolution

- In summary: *No introduction of life cycle phases, no disappearance of life cycle phases!*
  - … just a different, intelligent interpretation of them
- The **design** of systems from high-level architecture down to detailed design through to construction (coding) are concepts that must remain also for model-based development
- Analogously, the **test** of elementary units up to aggregates (sub-systems) and through to the entire system are also concepts that must remain for model-based development
- **No excuses**: a monolithic, “big-bang” approach is unacceptable both for the construction and for the test of systems, whether model-based or otherwise
- **No revolution**: the model-based lifecycle remains firmly grounded in the principles of ISO 26262

Brave New World?

- Some current topics in Model Based Development
  - Proper use of hierarchical top-down decomposition to handle complex models (e.g. sub-models as separate models "referenced" by parent models)
  - Separate configuration management for sub-models
  - Unit Test of sub-models
  - Integration test of aggregate of sub-models
  - Reuse of submodels
  - Diff of models
  - Model standards (analogous to coding standards)
  - “Pair modeling”? (analogous to pair programming)
  - Model metrics for size, complexity, defect density, productivity, etc.
  - Model coverage criteria (e.g. statuses, transitions)
  - Model level debugging (e.g. set a breakpoint when a state is reached, or when a transition is activated)

A Tools Community

- A fair question: “But how much of this is possible TODAY?”

“Little models, little tools”

eclipse

gmf GRAPHICAL MODELING FRAMEWORK

emf ECLIPSE MODELING FRAMEWORK

A large and active community has grown up around model based development – nearly all of it open source technologies

ATL

UML2 plug-in

MOFscript

A Change of Mindset

From

Control engineering *artefacts*
with a software dimension

to

Systems and Software engineering *artefacts*
with a control dimension