Statemachines…
…for embedded

Daniel Siegl
Daniel.Siegl@lieberlieber.com
www.lieberlieber.us

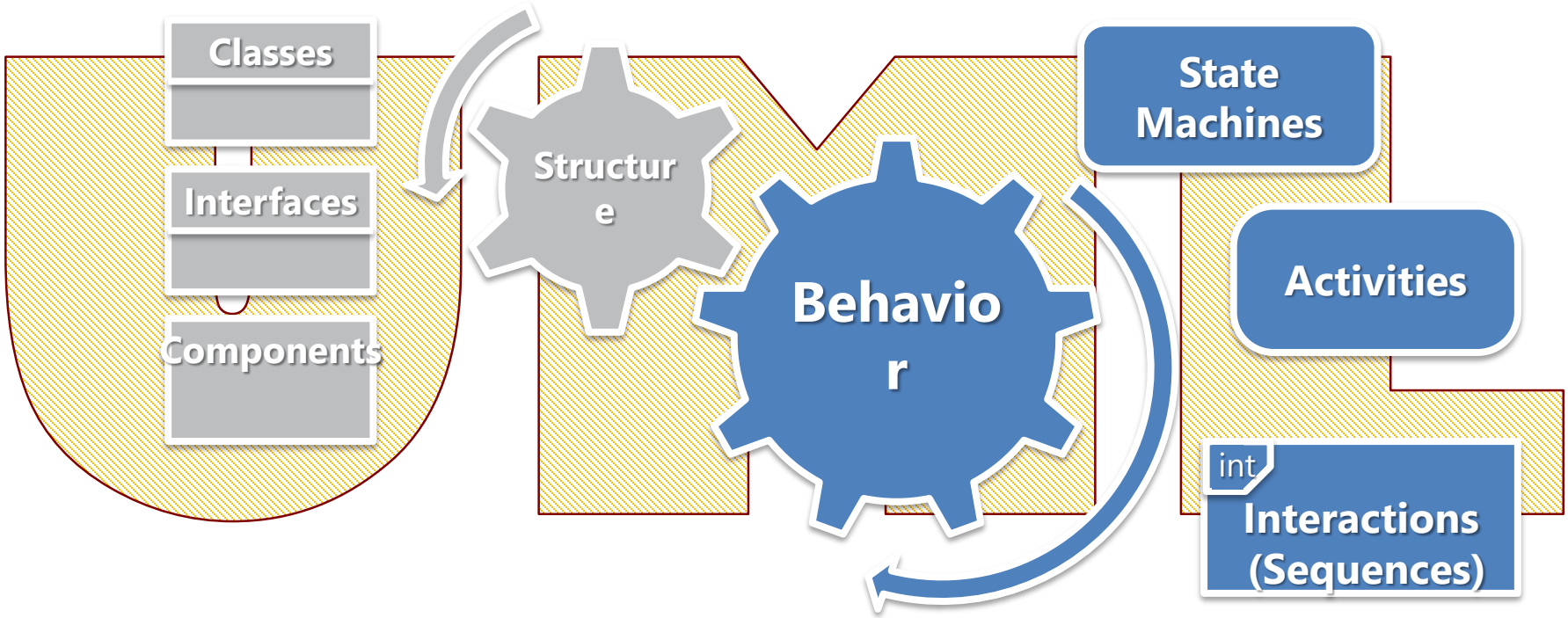
Generate Code from behaviour Models

Lot's of new and old challenges ahead!

- Functional Safety (ISO 26262,…)
- UI complexity
- Multi and many core hardware
- Traceability
- Documentation requirements

LieberLieber

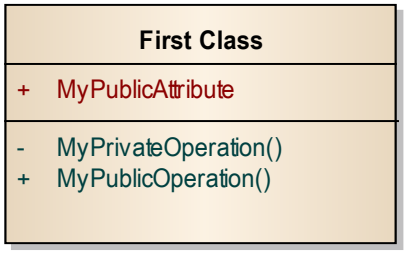
Use full power of UML



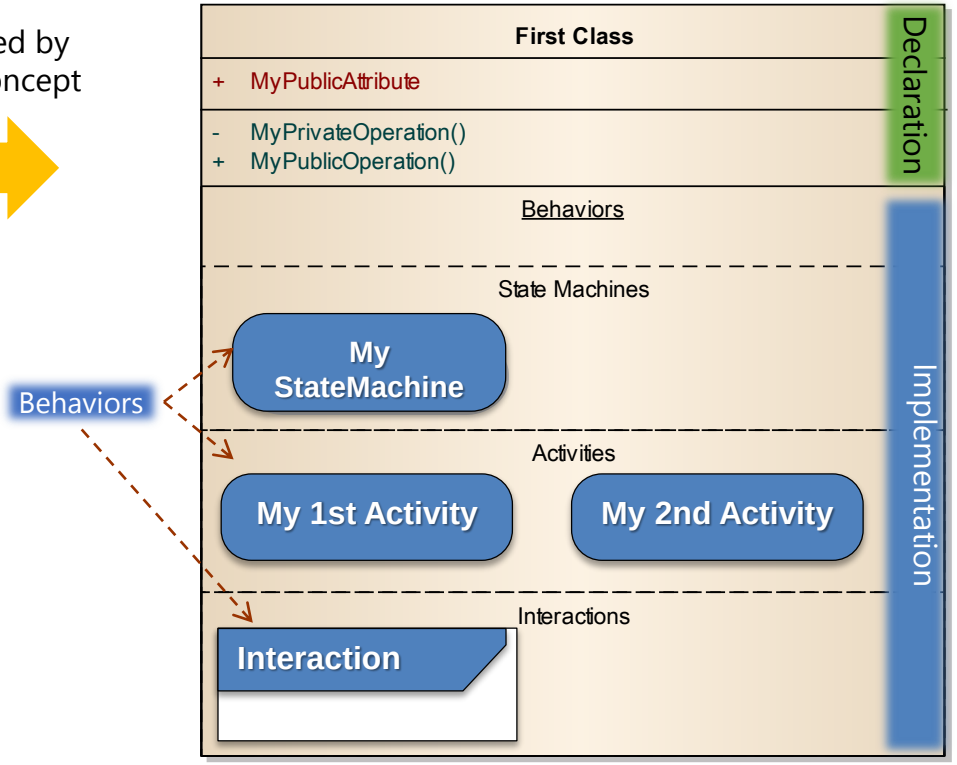
LieberLieber

What it is about – UML Behavior?

Classic Object Oriented



OO extended by Behavior concept



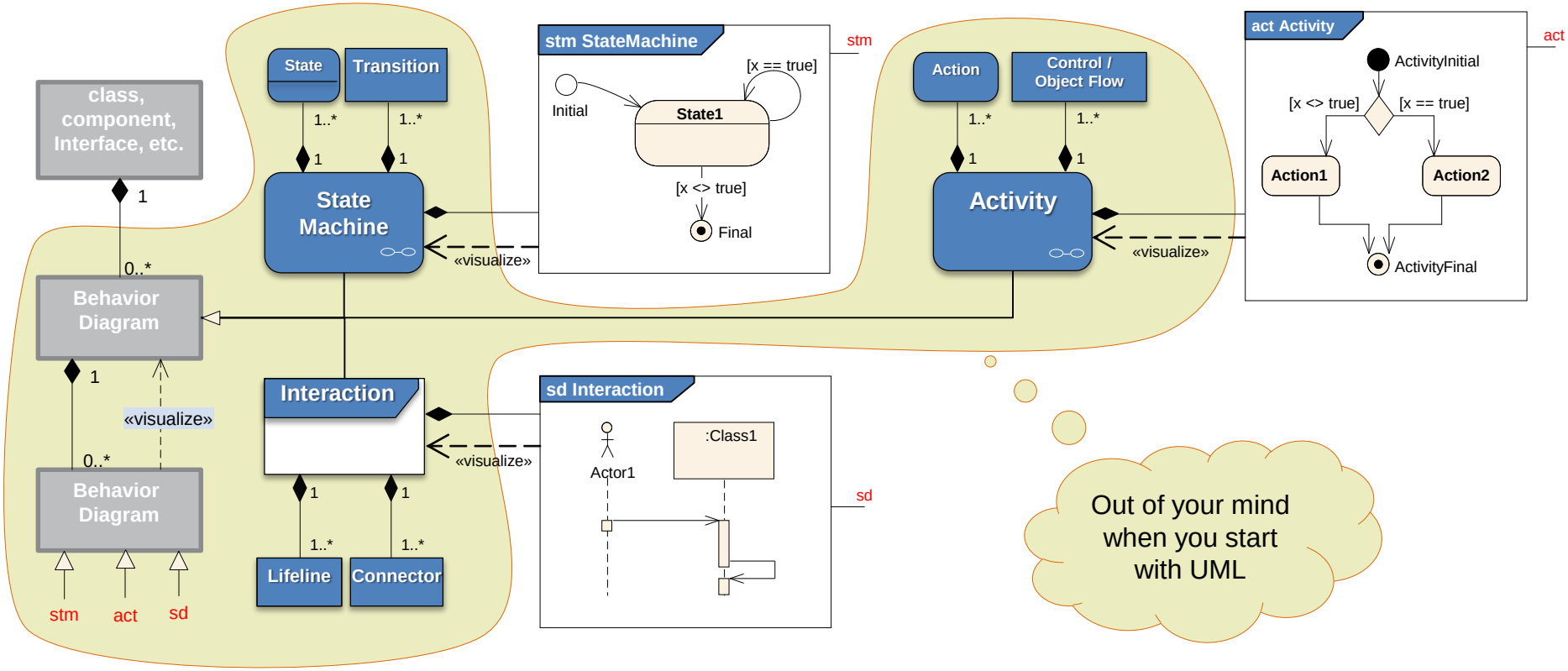
- Attributes and Operations are declaration only
  ⇨ Same as HEADER files in C or C++
- And now - how to implement using UML??

- UML introduced behaviors to extend classic object oriented concept

- Behavior is UML concept for "implementation"
  ⇨ Same as SOURCE files in C or C++

LieberLieber

Class, Behavior and Diagrams



LieberLieber

Generate Code from behaviour Models

Inspiration:

- Higher level of abstraction than the generated code – especially State Charts are very powerful
- Render requirement and hazard information into the code automatically!
- Documentation = Product

LieberLieber

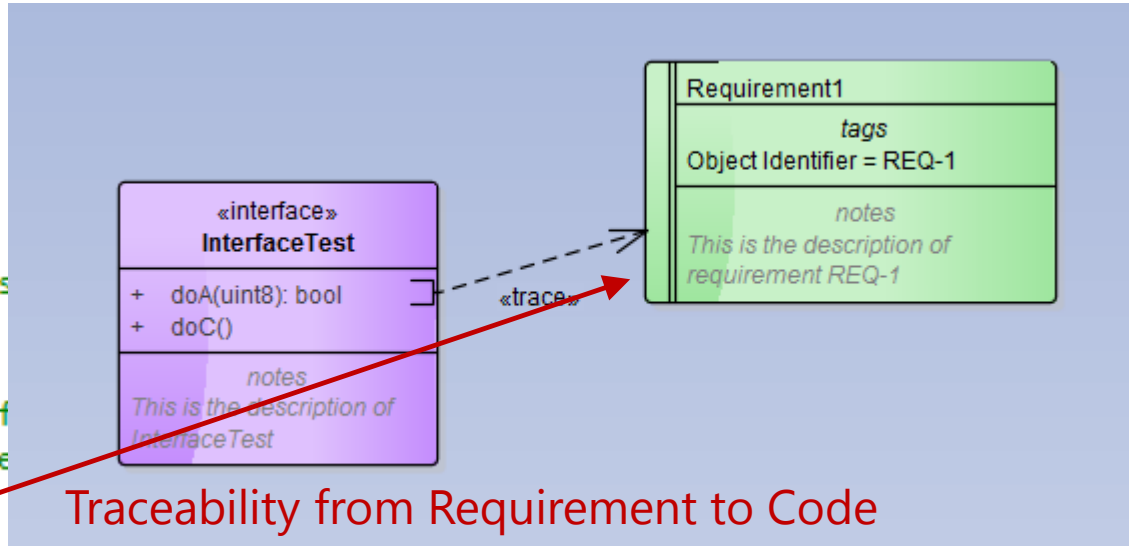
```
class InterfaceTest
{
public:

    /// auto generated virtual des
    virtual ~InterfaceTest() {}

    /// This is the description of
    /// @param param1: this is the
    ///
    /// @covers REQ-1
    virtual bool doA(uint8 param1) = 0;

    /// this is the description of method doC()
    virtual void doC() = 0;

};
```



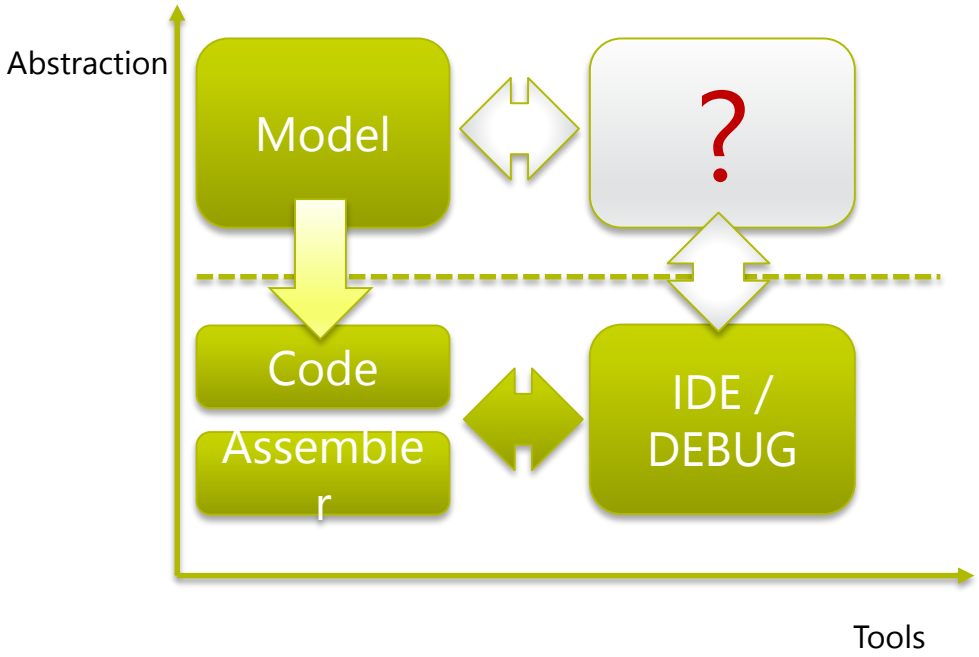
LieberLieber

Generate Code from behaviour Models

- Full (Behavior) round trip is a myth
- 2015 forward only
- Reverse for legacy
- Optional: synchronization of method/function content

LieberLieber

The missing Link – Debugging for Models



LieberLieber

The missing Link – Debugging for Models

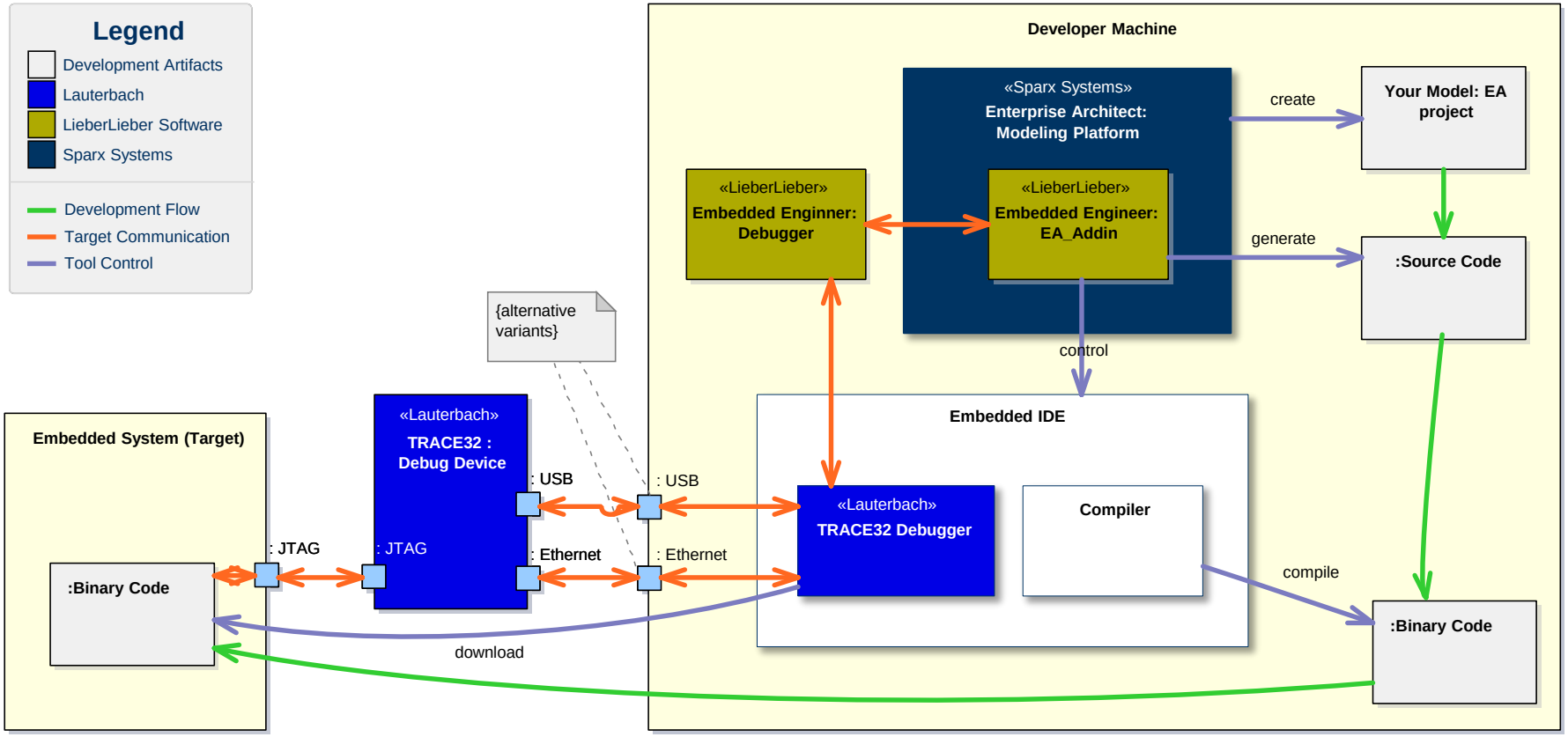
- Engineers need feedback – Feedback means debugging
- We need them to debug with the model
- Ability to understand and fix issues in the model/generator and not in the “code”.
- Pure UML “Simulation” is not the best solution for embedded

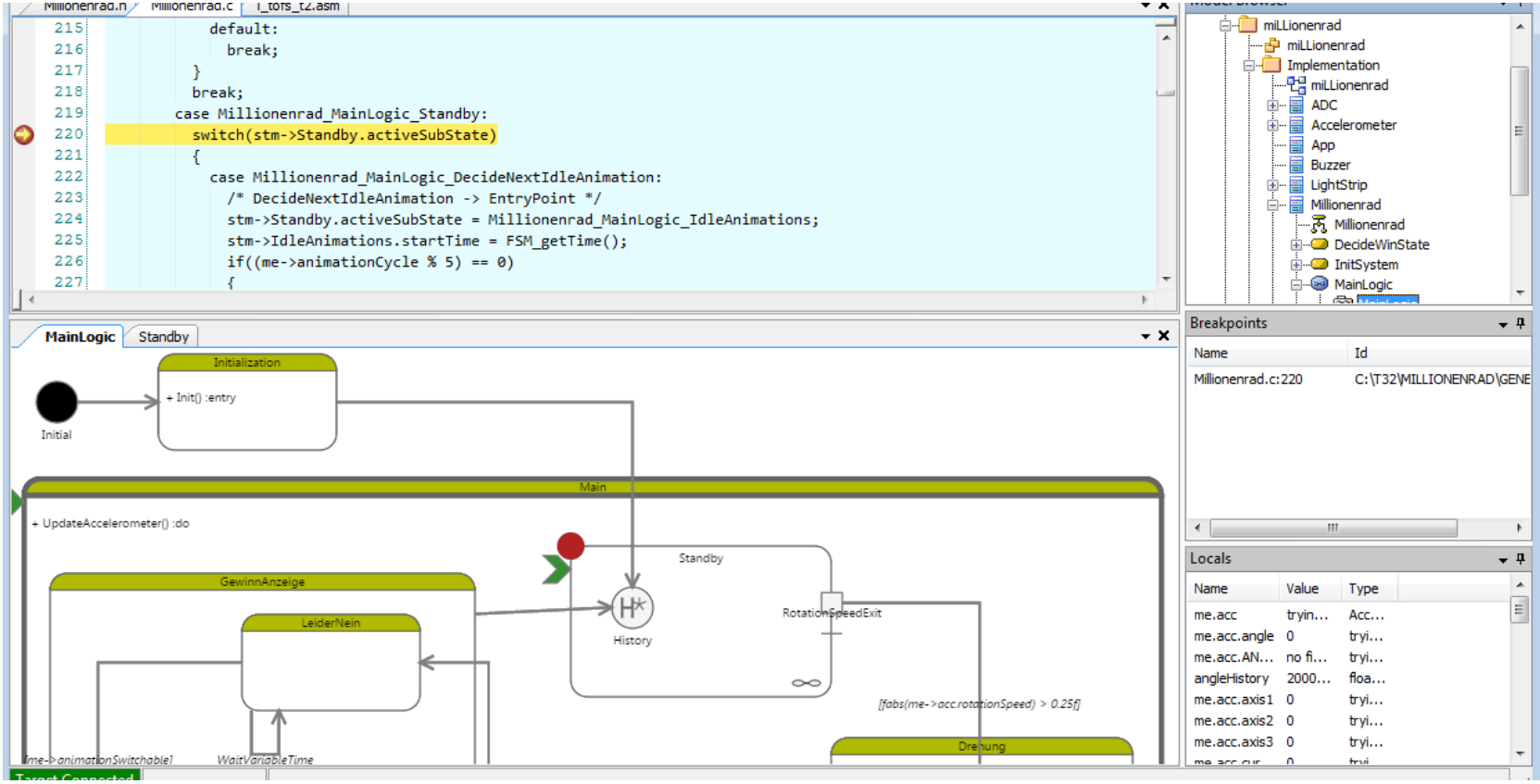
LieberLieber

Our Approach/Demo

- Enterprise Architect from Sparx Systems to build the models
- Programmed and debug able code generator (C#)
- C or C++ source code
- Compile and ceploy done with Code Composer
- LieberLieber UML Debugger linked to TRACE32 from Lauterbach

LieberLieber


Legend
Development Artifacts
Lauterbach
LieberLieber Software
Sparx Systems
Development Flow
Target Communication
Tool Control
Developer Machine
«Sparx Systems»
Enterprise Architect:
Modeling Platform
«LieberLieber»
Embedded Enginner:
Debugger
«LieberLieber»
Embedded Engineer:
EA_Addin
create
Your Model: EA
project
generate
:Source Code
control
{alternative
variants}
Embedded IDE
«Lauterbach»
TRACE32 :
Debug Device
: USB
: USB
«Lauterbach»
TRACE32 Debugger
Compiler
Embedded System (Target)
: JTAG
: JTAG
: Ethernet
: Ethernet
:Binary Code
compile
:Binary Code
download



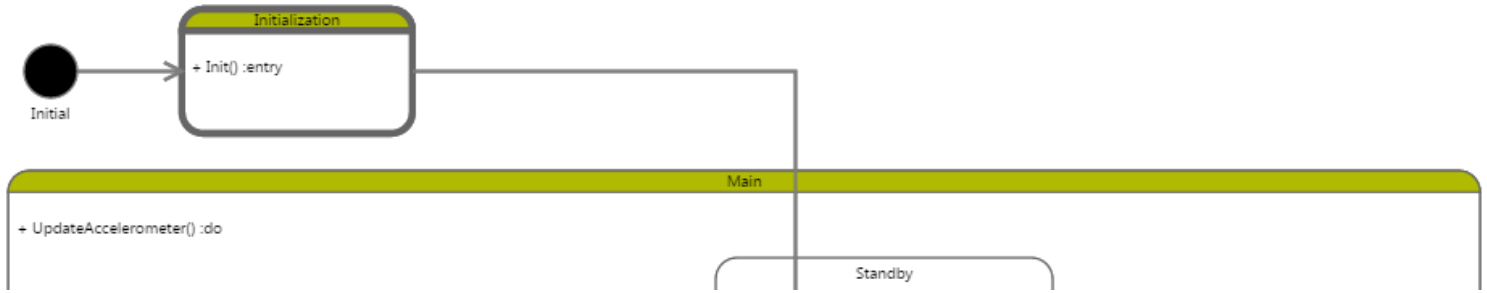
LieberLieber

Millionenrad.h | **Millionenrad.c**

```
57  bool Millionenrad_MainLogic(Millionenrad* const me, Millionenrad_MainLogic_STM* stm, Signals msg)
58  {
59    bool evConsumed = 0;
60    switch(stm->mainState.activeSubState)
61    {
62      case Millionenrad_MainLogic_Initialization:
63        evConsumed = 1;
64        /* Initialization -> History */
65        stm->mainState.activeSubState = Millionenrad_MainLogic_Main;
66        Millionenrad_MainLogic_EnterDeepHistory(me, stm, &(stm->Standby));
67        break;
68      case Millionenrad_MainLogic_Main:
69        /* do actio   for state Main */
70        Accelerom      pdateAxis(&me->acc, &me->adc);
71        /* end      ns for state Main */
72        switch(st      n.activeSubState)
73        {
```

MainLogic

Selected state also visualized in C



Conclusion

Can you afford and “survive” not to generate Code?

Start TODAY!

ENTERPRISE ARCHITECT USER GROUP

THANK YOU

DANIEL SIEGL
Daniel.Siegl@lieberlieber.com
sales@lieberlieber.com
www.lieberlieber.us

LieberLieber