mbeddr
C the Difference.
C the Future.

mbeddr – Extensible C, DSLs and Formal Verification

mbeddr supports embedded software development based on an extensible version of the C programming language. New types, statements, operators, expressions or top-level constructs can be added, and those C constructs that are considered „dangerous“ can be removed. mbeddr C allows the direct integration of domain-specific abstractions into C. A library with reusable extensions is available. mbeddr C supports program verification, traceability of requirements and management of product line variability. mbeddr is open source software (EPL) and is based on the open-source language workbench JetBrains MPS. Using mbeddr leads to higher productivity, better code quality, as well as more readable, testable and maintainable software.

Context and Solution Approach

The development of embedded software is becoming increasingly complex: systems become larger, expected time-to-market becomes shorter and product line variability is common-place. Languages such as C address these challenges only to a limited degree, which is why modeling tools are used more and more. However, these are typically not adaptable or extensible to specific domains and often do not integrate well with other tools and those parts of the system written in C. mbeddr addresses this problem by means of language engineering, supporting the incremental and modular extension of C. The advantages of C (portability, runtime efficiency) and the advantages of domain specific languages (conciseness, verifiability, code generation) are combined seamlessly.

mbeddr C Extensions

New data types and literals with physical units are supported. The type checker verifies unit compatibility.

Interfaces with pre- and post conditions as well as components allow the black box specification of behavior. They hide implementation details and support exchanging parts of the system. In contrast to classes in C++, mbeddr components do not use polymorphism, avoiding runtime overhead. Mocks can be used to write expressive test cases for component-based systems.

State machines allow the concise description of state-based behavior.

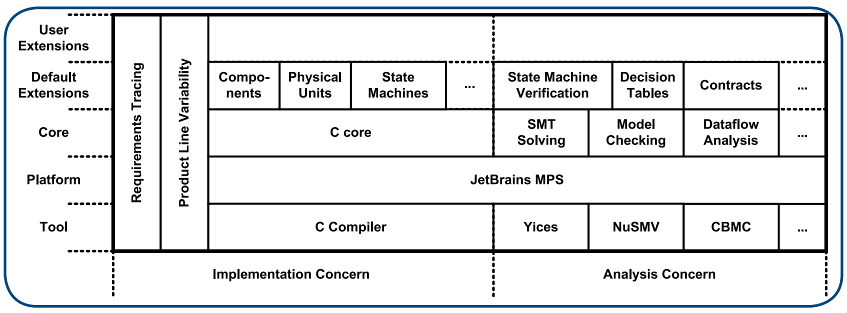
Requirements tracing and product line variability is directly supported.

Formal Verification

Based on the default extensions, mbeddr directly integrates formal verification tools. State machines can be model checked, decision tables can be analyzed for completeness and consistency, feature models can be checked for consistency and violations of interface contracts and protocols can be detected statically.

State-of-the-Art IDE

mbeddr comes with a state-of-the-art IDE that supports syntax highlighting, code completion, go-to-definition, data flow analysis, find usages, quick fixes and refactorings. A debugger for C and its extensions (on the level of the extension) is also included in the IDE.



High-Level overview over the mbeddr stack.

itemis

Domain-Specific Languages

mbeddr includes an SDK that allows users to develop their own languages. These might be modular extensions to C or to the default extensions. They might also be completely separate languages that may or may not integrate with or generate to C. Even for separate DSL, it is possible to integrate parts of C, for example, the expression language.

Technological Basis

mbeddr is based on MPS, an open-source language workbench developed by JetBrains, a leading vendor of developer tools. itemis has entered into a partnership with JetBrains to jointly develop and evolve MPS in the context of mbeddr. MPS is a tool to build modular languages. It is based on a projectional editor: There is no parser to process source code. Instead, a user's changes to the code directly lead to changes of the abstract representation of the program (AST). This avoids syntactic ambiguities when independent language modules are composed. Furthermore, non-textual syntax such as a decision table embedded in C code can be used. Graphical notations will be supported later in 2013. Also, an integration with Eclipse will be available later in 2013.

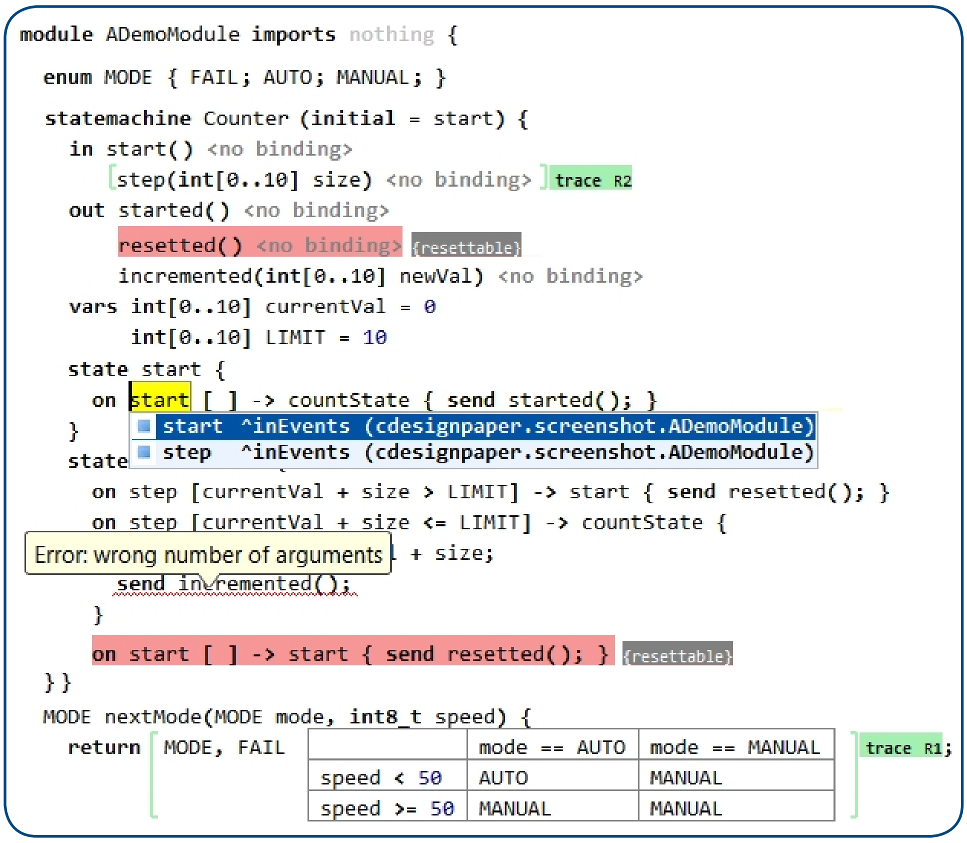
MPS is available at:
http://jetbrains.com/mps

Learn more

Website, Download and User Guide:
http://mbeddr.com

Paper: *mbeddr: Instantiating a Language Workbench in the Embedded Software Domain*. Detailed overview over mbeddr from a tool customization perspective.
http://bit.ly/XPUN9V

Paper: *mbeddr: an Extensible C-based Programming Language and IDE for Embedded Systems*. General overview emphasizing the language engineering perspective.
http://bit.ly/WN0Uwu



```
module ADemoModule imports nothing {

  enum MODE { FAIL; AUTO; MANUAL; }

  statemachine Counter (initial = start) {
    in start() <no binding>
       step(int[0..10] size) <no binding>  trace R2
    out started() <no binding>
        resetted() <no binding> {resettable}
        incremented(int[0..10] newVal) <no binding>
    vars int[0..10] currentVal = 0
         int[0..10] LIMIT = 10
    state start {
      on start [ ] -> countState { send started(); }
    }      start  ^inEvents (cdesignpaper.screenshot.ADemoModule)
    state  step   ^inEvents (cdesignpaper.screenshot.ADemoModule)
      on step [currentVal + size > LIMIT] -> start { send resetted(); }
      on step [currentVal + size <= LIMIT] -> countState {
Error: wrong number of arguments  + size;
        send incremented();
      }
      on start [ ] -> start { send resetted(); } {resettable}
  }}

  MODE nextMode(MODE mode, int8_t speed) {
    return MODE, FAIL |              | mode == AUTO | mode == MANUAL |  trace R1;
                      | speed < 50   | AUTO         | MANUAL         |
                      | speed >= 50  | MANUAL       | MANUAL         |
```

Example program that shows the combination of different aspects in a single program: a statemachine, a decistion table, requirements traces and product line variability.

Status and Availability

mbeddr is now being used in the the first real-world projects, for example, to build a smart meter. Initial experience shows that mbeddr scales to real-world systems and delivers on its promises in terms of developer productivity, code quality and software maintainability, while incurring very limited runtime overhead.

The LWES Research Project

mbeddr is developed by itemis fortiss, BMW Car IT and Sick as part of the KMU Innovativ research project Language Workbenches for Embedded Systems. The goal is to create an integrated environment for embedded development based on extensible C and the integration of formal methods.

OFFICES

- **Lünen near Dortmund (Central Office)**
- **Kiel**
- **Hamburg**
- **Leipzig**
- **Dortmund**
- **Bonn**
- **Frankfurt**
- **Stuttgart**
- **International Offices: France, Switzerland**

CONTACT

itemis AG

Am Brambusch 15-24
44536 Lünen | Germany
Tel. +49 231 98 60-606
Fax +49 231 98 60-211
info@itemis.com | www.itemis.com