5th 4DIAC Users' Workshop

Dynamically loadable Function Block types to reduce application development time

Matthias Plasch

ETFA 2014, Barcelona, 12 September 2014

LEADING
INNOVATIONS

WWW.PROFACTOR.AT

Contents

Introduction and Motivation

- Requirements of flexible production systems
  - Modular and distributed architecture
  - Fast adaptation, scalability, extensibility
- Challenge: Integration of heterogeneous modules / components

- IEC 61499 provides an open basis for component interoperability
- Function Block (FB) development for component interaction
  - Time consuming runtime compilation required in many cases
  - Interpretation of FBs during runtime scarcely supported

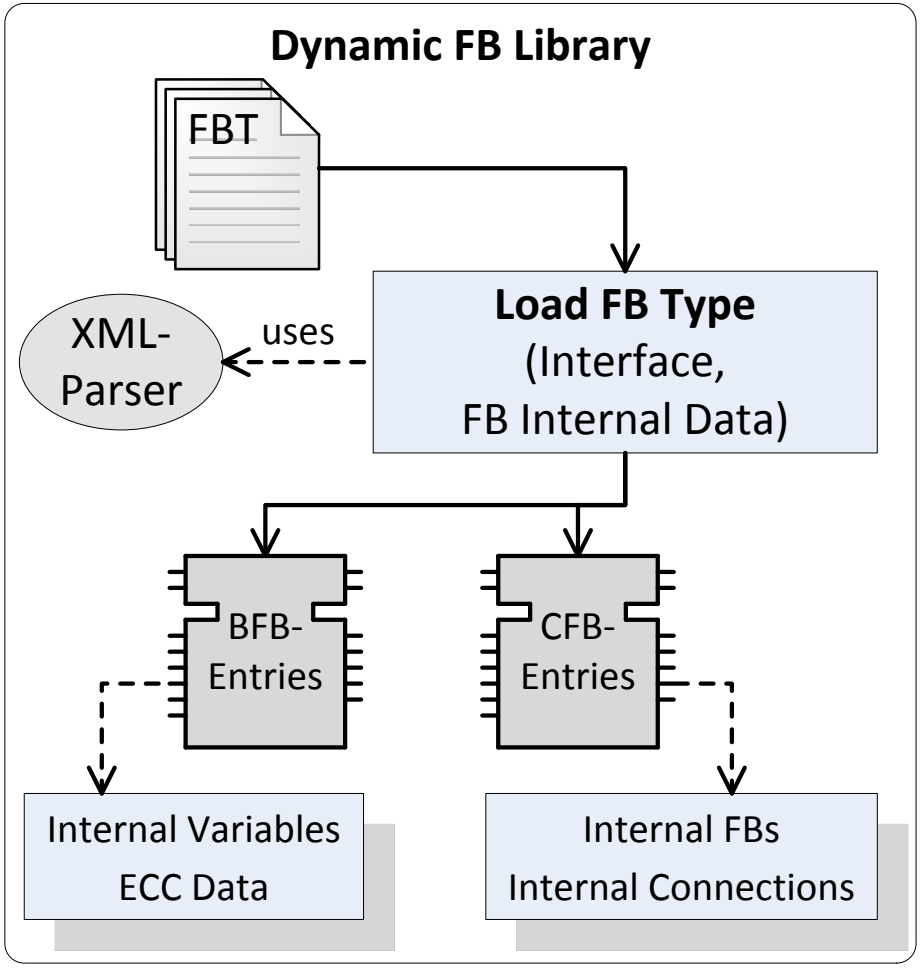
- **Target:** Speed-up development, system integration and ramp-up

Approach to reduce FB development time

- **Goal 1:** Automatic interpretation of FB type functionality
  - Based on FB definition files
  - Interpretation and execution of algorithms based on scripting languages
  - Fast reconfiguration of FB behaviour
- **Goal 2:** Dynamically configured FB type library
  - Enabling lightweight runtime configurations
  - Type library extension does not require runtime re-compilation
- 4DIAC – Framework for Distributed Industrial Automation and Control
  - 4DIAC-IDE – Development Environment
  - FORTE – 4DIAC Runtime Environment

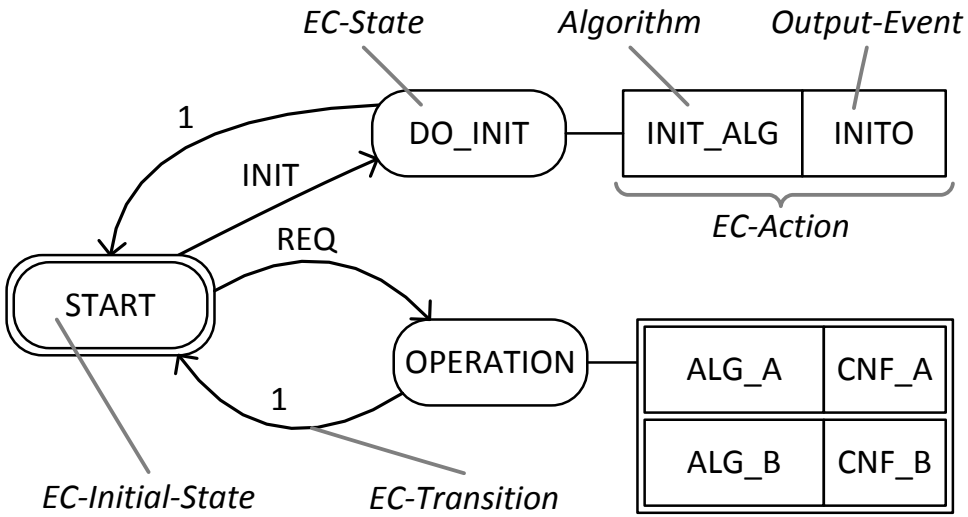
Creation of a Dynamic Type Library

- Definition files are queried on runtime startup
- XML parser to extract information
  - Interface (event and data ports)
  - Type specific data
- Basic FB-Types
  - Internal Variable definitions
  - Execution Control Chart (ECC)
- Composite FB-Types
  - Internal FB instances, parameterization
  - Internal connections



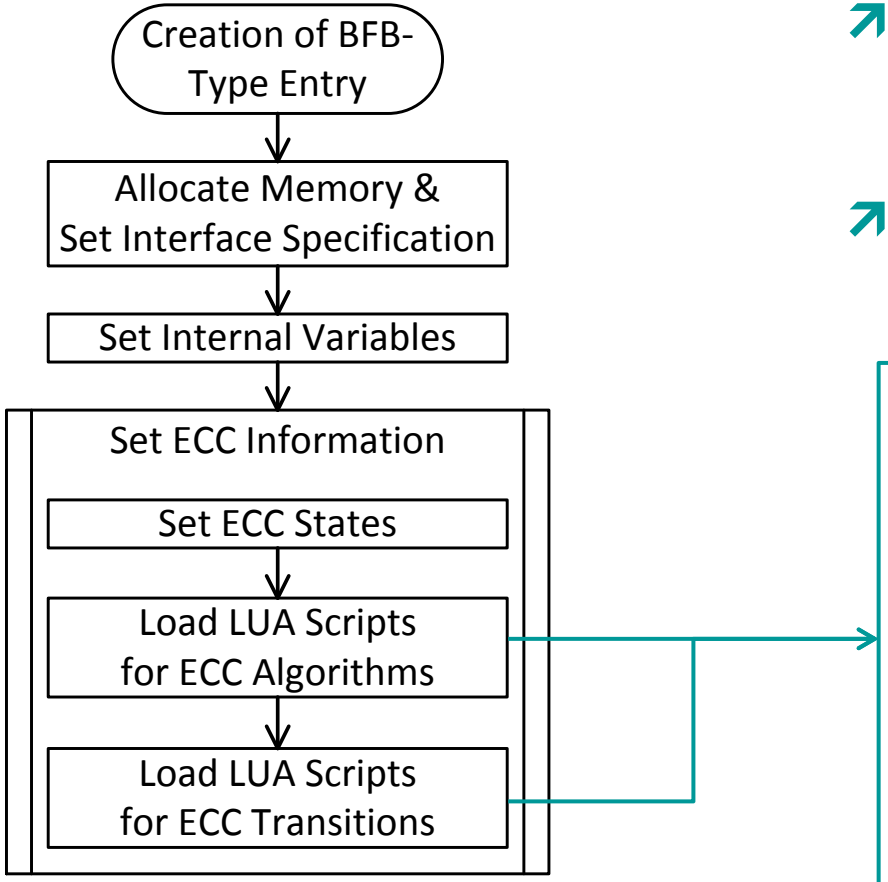
Dynamic Configuration of Execution Control Charts (ECCs)

- Behaviour of a Basic FB is determined through its ECC
  - States, including Actions
  - Transitions between States
- Dynamically configured ECCs imply
  - Interpretation of guard conditions and algorithms
  - Configurable execution based on extracted ECC information

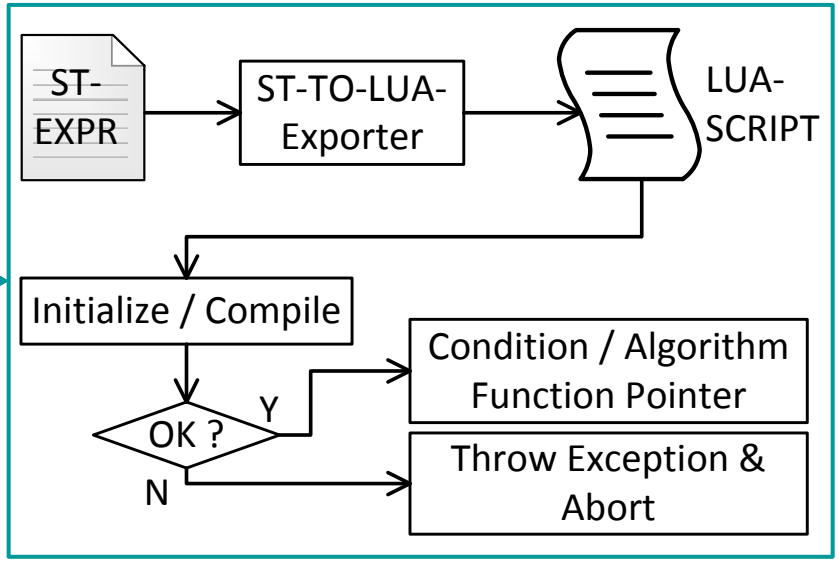


- Online interpretation of algorithms and conditions
  - Integration of the scripting language LUA into the FORTE
  - Requires Structured Text (ST) expressions to be transformed into LUA code

Dynamic Configuration of Execution Control Charts (ECCs)



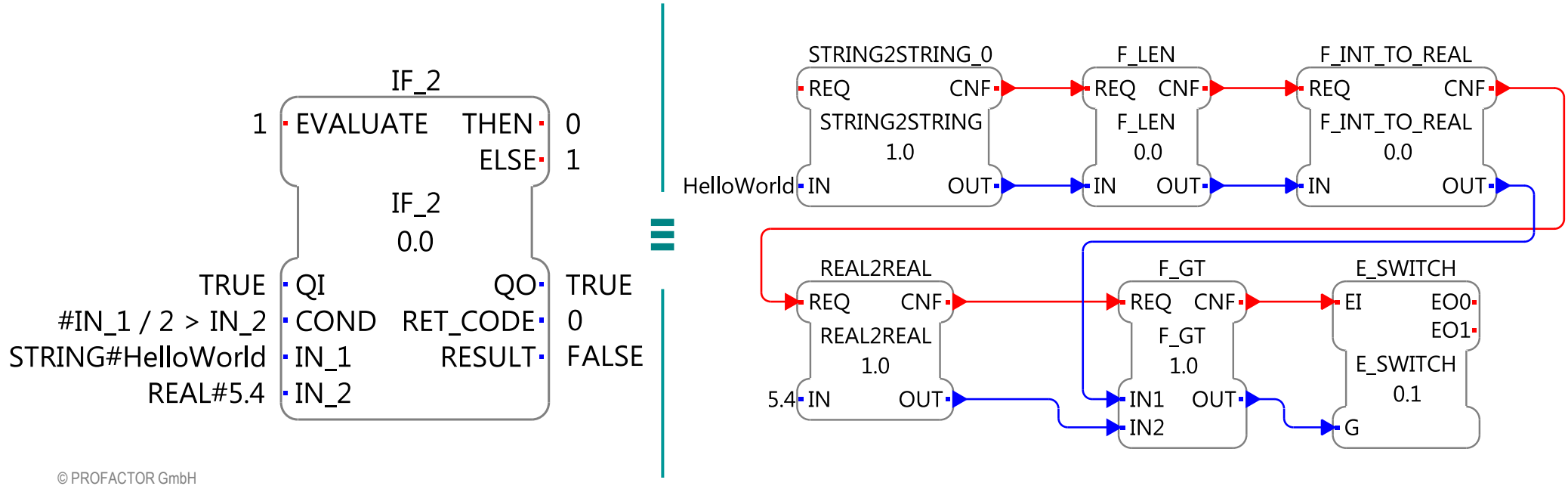
- Compilation of scripts for each transition condition and algorithm expression
- Function pointers refer to calling functions of compiled scripts



Benefits of using LUA

- Fast integration through C-API (Application Programming Interface)
- Lightweight, causing low memory consumption
- LuaJIT – Just-In-Time interpreter to speed up execution

- LUA can be used to force flexible usage of logic FBs



Performance Evaluation

- Execution of LUA-based FBs is significantly slower
  - LuaJIT increases performance
- Small differences in memory consumption (range of 1 MB)
- Size of FORTE executable is increased
  - Factor 2.4 for LUA
  - Factor 3.4 for LuaJIT

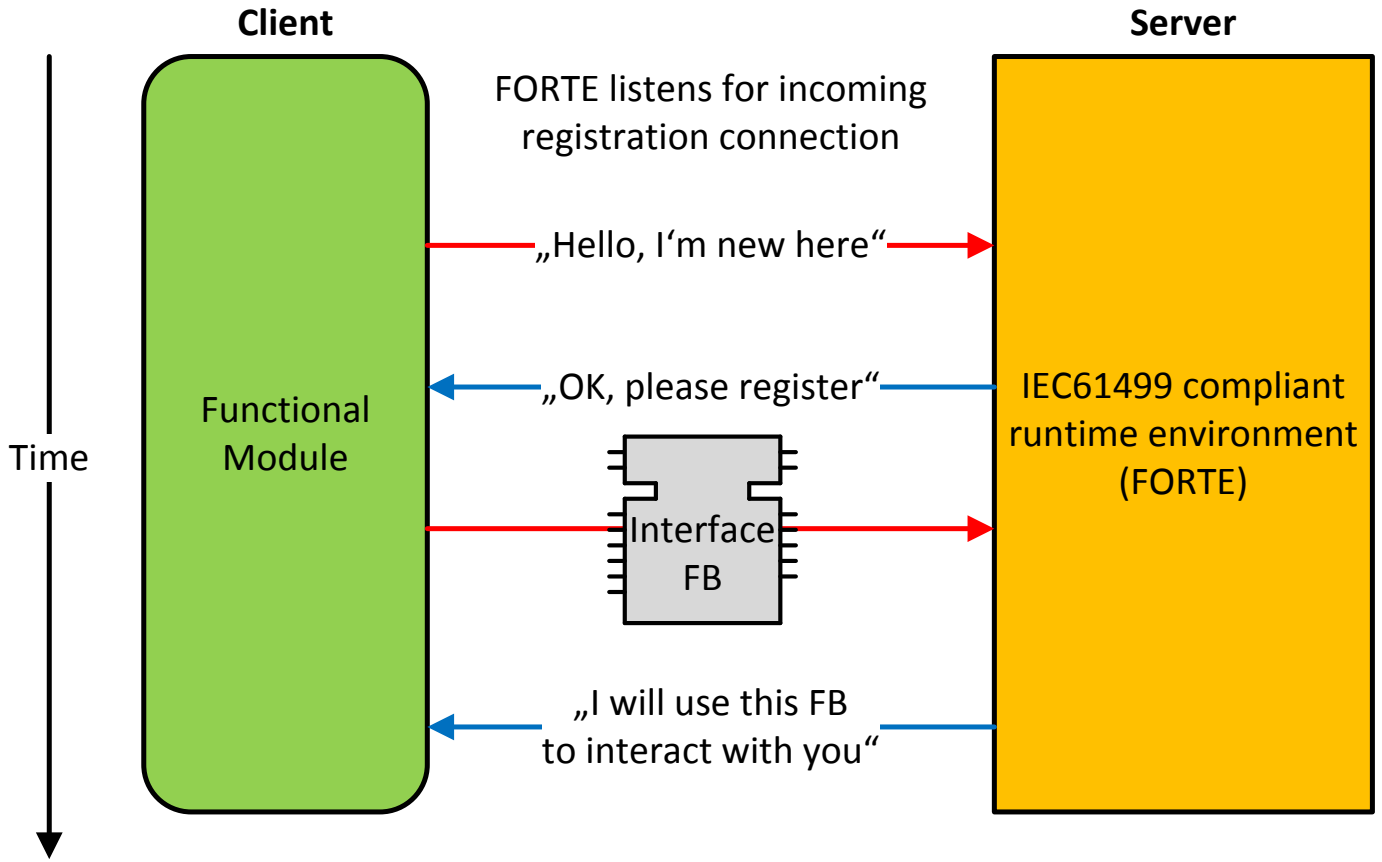
- Use of LUA-based FB types reasonable during development
  - In debugging and test phase, no re-compilation of FORTE required
  - FB type definitions can be exported in C++ code, after development
  - Suitable for testing complex basic FBs, instead of bulky networks

Related work

- Function Block Development KIT (FBDK)
  - Compilation of single FB types possible
  - Generated Java source and class files are linked accordingly
  - No full re-compilation of the runtime environment necessary
- nxtSTUDIO by NXT-Control
  - Intermediate code is generated for newly developed or changed FBs
  - Runtime environment nxtRT61499F interprets the intermediate code during runtime
- Comparison to our solution
  - Dynamic type library builds entirely on FB definition files
  - No pre-compilation of FB types necessary
  - Runtime compilation after FB development, to speed up execution

Future Work

- Development of an approach to dynamically register a functional module within a distributed control architecture, based on IEC 61499



Summary

- Dynamic FB type library
  - Constructed during runtime start-up
  - Extraction of FB type data from definition files
- Dynamic configuration of Basic FB types
  - LUA scripting language to interpret condition and algorithm expressions
  - State chart logic is configured dynamically
- Suitable for development and debugging to reduce time effort
- Future work item

- This work is supported by the European funded research project SMARTLAM (www.smartlam.eu)

Thanks for your attention!

Contact Speaker
Matthias Plasch
PROFACTOR GmbH
Im Stadtgut A2
4407 Steyr-Gleink, AUSTRIA
matthias.plasch@profactor.at
www.profactor.at