CHROMOSOME

A Modular Middleware for Cyber-physical Systems

Michael Geisinger

fortiss GmbH
An-Institut Technische Universität München
Observation

- Technical systems consist of more and more individual controllers
- *Example:* Modern cars
  - Up to 70 electronic control units (ECU) in a premium segment car
- Challenges:
  1. How to reduce the complexity of such systems w.r.t. hardware?
  2. How to design software for such systems independent of the concrete hardware?
Facing the Challenges

- **Hardware**: Modular hardware platform
  - *Not* focus of this talk
  - For more information, see for example RACE project: [http://www.projeckt-race.de/](http://www.projeckt-race.de/)

- **Software**: Modular middleware platform
  - Focus of this talk
  - Middleware must satisfy various requirements
  - For more information, see CHROMOSOME project: [http://chromosome.fortiss.org/](http://chromosome.fortiss.org/)
Software Design Requirements (1/4)

1. Individual parts should be able to be exchanged without touching the other parts of the system (e.g., software update, addition of new functionality) → Component-based design

2. Sending data to a component on the other end of the system should “feel” the same than sending data to a “nearby” component (i.e., developers don’t want to care about the concrete topology) → Abstraction of communication
Software Design Requirements (2/4)

3. It must be possible to state the maximum latency of data transmission between components (e.g., airbag triggering) → *Real-time capability*

4. Critical components (e.g., ABS in a car) must execute next to uncritical components (e.g., infotainment) without influencing each other → *Mixed criticality support, isolation*
5. It must be possible to execute the components on many different target platforms with different operating systems (OS) (e.g., microcontrollers / PCs with real-time OS / embedded OS)
   → Target platform abstraction
   → Support for cyber-physical systems (CPS)*

*From Wikipedia [1]: “A cyber-physical system (CPS) is a system of collaborating computational elements controlling physical entities. […] Unlike more traditional embedded systems, a full-fledged CPS is typically designed as a network of interacting elements with physical input and output instead of as standalone devices.”

6. For maximum use, the system should be applicable to different domains such that devices of these domains can cooperate (e.g., automotive, automation, multimedia, …)

→ Multi-domain support

(automotive domain is just an example here)
Software Stack for Realization of Requirements

Application(s)
- Component-based design
- Platform-independent implementation based on middleware abstraction

Middleware (XME runtime system)
- Abstraction of communication
- Real-time capability
- Mixed criticality support
- Multi-domain support
- Target platform abstraction

Target platform(s)
- Operating system
- Target hardware
- Platform-specific functionality and features
What is a Middleware?

• From Wikipedia [2]:
  “Middleware is computer software that provides services to software applications beyond those available from the operating system. […] The term is most commonly used for software that enables communication and management of data in distributed applications.”

→ Basically an “add-on” to a system to provide a higher level of (programming) abstraction for software developers
→ Located in between the application and the operating system (hence the name “middleware”)

CHROMOSOME Architecture

- Application and Quality of Service Model
- Service and Platform Repository
- Model Transformation and Code Generation
- Generated Application
- Generated Middleware
- Operating System
- Hardware Abstraction Layer (HAL)
- Target Platform

CHROMOSOME: Cross-domain open modular operating system or middleware
Architecture of a CHROMOSOME Node

Node:
Communication participant with calculation and storage capabilities (e.g., software process, firmware on embedded controller)
Basic Functions

- *Data centric communication*: Decoupling of sender and receiver
- *Modeling*: Model-driven design of the application
- *Platform support*: Customizable support for various platforms
- *Plug & Play*: Dynamic reconfiguration at runtime
- *Real-time capability*: Event or time driven model of execution
- *Open source*: Free licensing model (Apache License Version 2.0)

🌐 [http://chromosome.fortiss.org/](http://chromosome.fortiss.org/)
Data Centric Communication

Goal:
• Make all data available to all components

Rationale:
• Avoid costs by unneeded redundancy

Realization:
• Decoupling of sender and receiver
  → Data centric communication

Concepts:
• **Topic:** Data type with assigned semantics and fixed structure
  (e.g., temperature, velocity, voltage, pressure)
• **Publication:** Intent to send data of a given topic
• **Subscription:** Request to receive data of a given topic
Data Centric Communication

Example

Temperature sensor 1
publish(Temperature)

Temperature sensor 2
publish(Temperature)

Display
subscribe(Temperature)
Attribute Support for Topics (1)

Goal:
• Allow selective filtering of data at the transport layer

Realization:
• Annotation of data with additional properties → Attributes (a.k.a. meta data)
• Definition of a simple ontology for attributes

Concepts:
• **Attributes**: Key/value pairs for exact specification of data properties (e.g., timestamp, accuracy, confidence, source)
• **Annotation**: Specification of meta data for a *publication*
• **Filtering**: Specification of data acceptance criteria for *subscriptions*
Attribute Support for Topics (2)

Example:

- *deviation* is an attribute of topic *Position*

  ![Diagram showing the relationship between GPS decoder, Cell phone cell decoder, Galileo decoder, and Navigation system.]
Static and Dynamic Systems

• On resource constrained systems, the middleware can be statically configured and only contains predefined functions
  ➔ Suitable for “small” target systems

• On less resource constrained systems, aspects like plug & play allow dynamic reconfiguration at runtime
  ➔ Flexibility

• Model-driven tooling allows to select the appropriate variant:
  ➔ CHROMOSOME Modeling Tool
CHROMOSOME Modeling Tool

Goal:
- Describe applications via models
- Reusable models for component types, data types (topics), …
- Generate stub code
- Configure application
- Generate build system

Realization:
- Eclipse Modeling Framework based rich client application

Live Demo
Complete Workflow

- Models
- Edit
- Code generation
- Node 1 (static)
- Node 2 (dynamic)
- Dynamic reconfiguration at runtime
- Nodes (e.g., process, firmware)
- Communication infrastructure

© fortiss

München, 2013-06-24
Platform Support

- Linux (PC 32bit/64bit): primary, released
- Windows (PC 32bit): secondary, released
- FreeRTOS (ARM 16/32bit): not yet released
- Contiki (AVR 8bit): experimental
- PikeOS: internal only
Embedded Target Support

- **Platforms:**
  - Texas Instruments ARM Cortex M3 (Stellaris LM3S8962)
  - ST Microelectronics ARM Cortex M3 (STM32F10xxx)
- **Operating system:** FreeRTOS
Plug & Play

Goal:
- Dynamic addition/removal of components at runtime

Rationale:
- Applications need to be updated or extended while in use

Realization:
- Plug & Play Manager and related components in the middleware

1. Plug & play request
2. Logical route calculation
3. Physical route calculation
4. Deployment over network (to all affected nodes)
5. Local reconfiguration
Real Time Capability

Goal (initial implementation):
• Realization of specified end-to-end latency by real-time scheduling on a real-time capable platform (e.g., PikeOS or FreeRTOS)

Rationale:
• Prototype for real-time critical implementations

Realization:
• Specification of real time requirements via attributes

WCET = 5ms
Throttle
publish(Acceleration | timestamp = now)

WCET = 20ms
Speed controller
subscribe(Acceleration)
subscribe(…)
publish(Rotation-Speed)

WCET = 10ms
Motor
subscribe(Rotation-Speed | age ≤ 50ms)

out.timestamp = in.timestamp
age := now - timestamp

Requirement for total latency: \( \leq 50ms \)
Availability of CHROMOSOME

CHROMOSOME is open source software

- Download at http://chromosome.fortiss.org/

Current release (v0.3) as of May 29, 2013

- Model-driven static configuration of runtime system
- Platform abstraction for Linux and Windows
- 40-page introduction and tutorial included
- Installation instructions: see tutorial
- Next release planned for end of June
Evolution of CHROMOSOME

- CHROMOSOME is still in development; new features are added over time.
- By providing an open platform, we expect community contributions.
- Community extension can deliver additional benefits to existing applications when combined with each other (compare Smartphone apps).
- If you are interested, you are welcome to try it out!
Contact information // Michael Geisinger

fortiss GmbH
An-Institut Technische Universität München
Guerickestraße 25 • 80805 München • Germany
tel: +49 89 3603522 513 mail: michael.geisinger@fortiss.org
fax: +49 89 3603522 50 web: www.fortiss.org