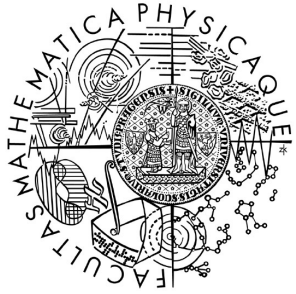
An Overview of the Koala Component Model

DISTRIBUTED SYSTEMS RESEARCH GROUP
http://nenya.ms.mff.cuni.cz

CHARLES UNIVERSITY PRAGUE
Faculty of Mathematics and Physics



Introduction

- Component model for embedded devices
  - TV set-top-boxes,...
- Developed by Philips
- Papers
  - van Ommering, R.: Koala, a Component Model for Consumer Electronics Product Software, In Development and Evolution of Software Architectures for Product Families, LNCS 1429, February 1998
  - van Ommering, R., van der Linden, F., Kramer, J., Magee, J.: The Koala Component Model for Consumer Electronics Software, In IEEE Computer, Vol. 33, No. 3, March 2000

Picture is taken from the Koala model website

Introduction

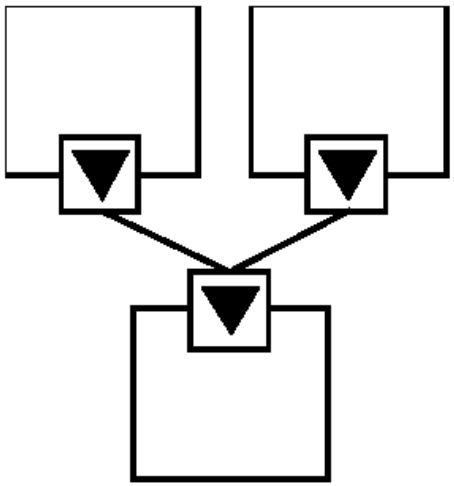
- Primary goals
  - manage increasing complexity of SW
    - components
    - explicit architecture
      - (previously – extracting architecture from code)
  - manage diversity
    - reuse of components
    - support for product lines
      - the same set of components ("primitive" components)
      - different configurations ("composite" components)
    - parametrization of components

Koala components

- Inspired by Darwin
- Components
  - defined in ADL
  - set of provided and required interfaces
- Interfaces
  - defined in IDL
- Configurations
  - set of connected components
  - required to provided interfaces
    - no explicit connectors
    - multiple required interfaces to one provided

```
interface Ituner {
  void SetFrequency(int f);
  int GetFrequency(void);
}

component CtunerDriver {
  provides ITuner ptun;
           IInit pini;
  requires II2c ri2c;
}
```

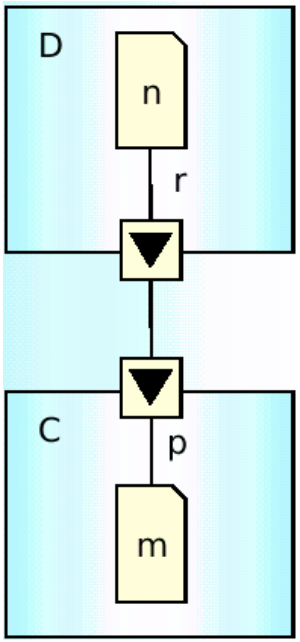


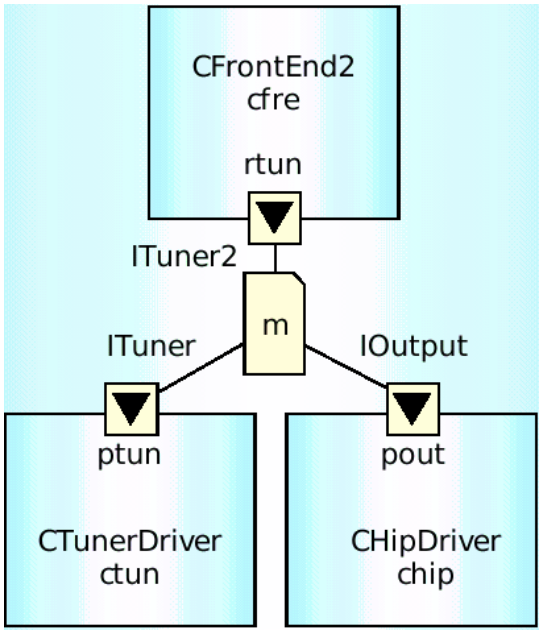
Compound components

```
component CtvPlatform {
  provides IProgram pprg;
  requires II2c slow, fast;
  contains
    component CFrontEnd cfre;
    component CTunerDriver ctun;
  connects
    pprg = cfre.pprg;
    cfre.rtun = ctun.ptun;
    ctun.ri2c = fast;
}
```

Modules

- Interfaceless components
- Modules
  - implements all functions of all interfaces
    - i.e. implementation of "primitive" components
  - can be used to glue components
    - "connectors"





Figures taken from the paper: The Koala Component Model for Consumer Electronics Software

Implementation

- Components implemented in **C**
  - resource constraints
- Koala compiler
  - generates C header files

Handling diversity

- Koala features for handling diversity
  - interface compatibility
  - function binding
  - partial evaluation
  - diversity interfaces
  - diversity spreadsheets
  - switches
  - optional interfaces
  - connected interfaces

Handling diversity

- Interface compatibility
  - structural compatibility of interfaces
    - an interface can be bound to one of a different type if the provided interface supports at least all the functions of the required interface
- Function binding
  - normally – functions in bounded interfaces connected on the basis of their names
  - can be connected explicitly
    - using glue module
      - implemented using macros

```
within m {
  cfre.rtun.SetFrequency(x) =
    ctun.ptun.SetFrequency(x);
  cfre.rtun.GetFrequency() =
    ctun.ptun.GetFrequency();
  cfre.rtun.EnableOutput(x) =
    chip.pout.EnableOutput(x);
}
```

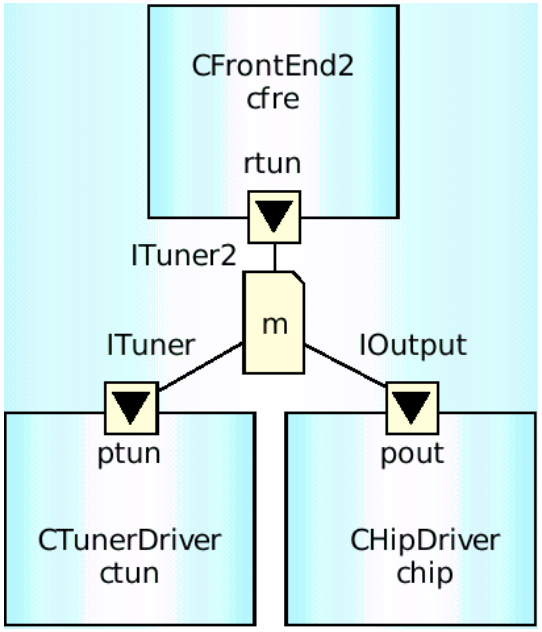


Figure taken from the paper: The Koala Component Model for Consumer Electronics Software

Handling diversity

- Partial evaluation
  - support of subset of C expressions
  - `1+1 -> 2`
  - `1 ? f(x) : g(x) -> f(x)`
- Diversity interfaces
  - configuration – usually get and set methods of a provided interfaces
  - but works well just with few parameters
    - tens or hundreds params expected
  - Koala reverses roles -> *required* interface with params
    - called diversity interfaces
  - params implemented with functions
    - Koala can optimize them
      - partial evaluation
      - removing parts of unnecessary code

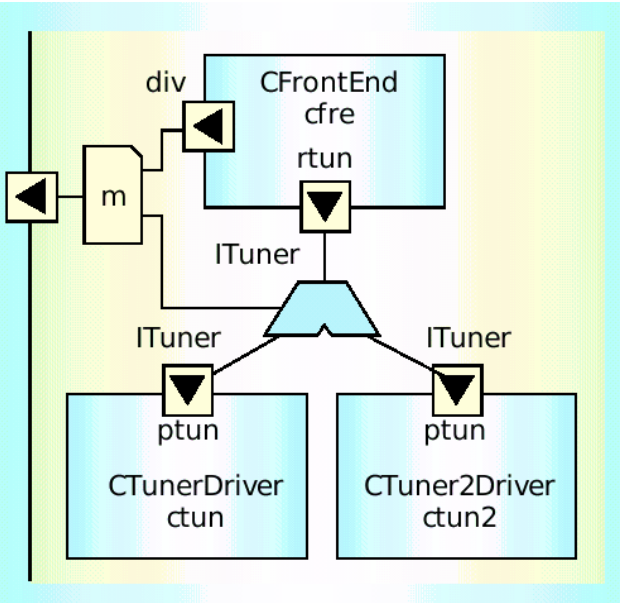


Figure taken from the paper: The Koala Component Model for Consumer Electronics Software

Handling diversity

- Diversity spreadsheets
  - delegating diversity interfaces to higher level components
- Switches
  - handling structural diversity
  - module gluing components controlled by diversity interface
    - can be done even without switches
    - but a very common pattern -> introducing switches
  - Koala can remove unreachable components
    - thanks to partial evaluation
    - optimization of an architecture

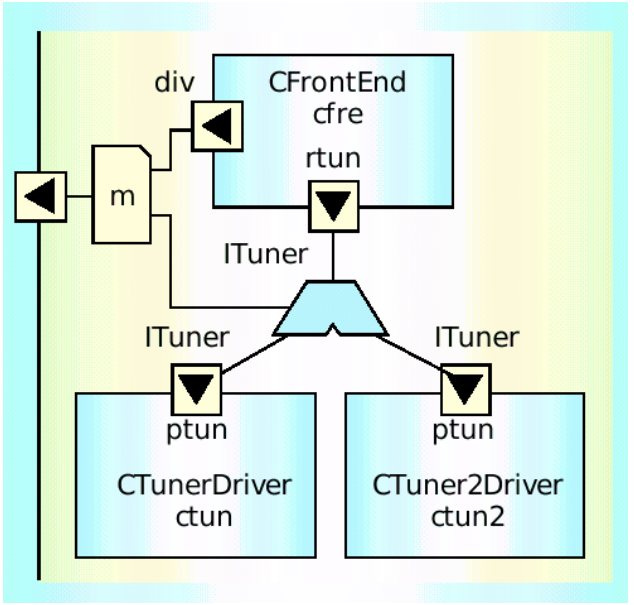


Figure taken from the paper: The Koala Component Model for Consumer Electronics Software

Handling diversity

- Optional interfaces
  - required interface "r" -> function `bool r_iPresent()` in the implementation
  - provided interface
    - interface automatically extended by the `iPresent()` function
      - component must provide an implementation of the function
    - e.g. availability of the interface depends on present hardware
- Connected interfaces
  - evaluation of availability and reachability of components and interface
  - removing unreachable

Coping with evolution

- Interface definitions stored in the *Interface repository*
  - global
  - globally unique name of each interface
  - interface definition
    - IDL
    - text document with a description of the semantics
  - repository is web-based
  - existing interfaces *cannot* be changed
- Components definitions stored in the *Component repository*
  - global
  - each component has
    - globally unique long name – used in component definitions
    - globally unique short name – used as a prefix in C functions

Coping with evolution

- Component repository (cont.)
  - component definition
    - CDL
    - textual description
    - implementation (.c and .h files)
  - allowed changes into the repository
    - adding new components
    - adding new provided interfaces to an existing component
    - adding new optional required interfaces to an existing component
    - making existing optional interface optional
- Repositories
  - manages history of components
  - allows temporary branches

Availability

- Koala web page
  - http://www.extra.research.philips.com/SAE/koala/
    - papers,grammar, tools (binary)
- Koala compiler
  - http://www.program-transformation.org/Tools/KoalaCompiler
  - open source
    - GPL
  - based on Stratego

Examples

Conclusion

- For embedded devices
  => strong focus on optimization
- Comparison with SOFA
  - common features
    - components ;-)
    - development – repositories
  - differences
    - Koala tied with C – SOFA multiplatform (theoretically)
    - managing diversity
      - SOFA – separated frame and implementation
      - Koala – switches, diversity interfaces,...
    - component content
      - SOFA – strictly primitive (just code) vs. composite (just subcomponents) components
      - Koala – mix of code (modules), subcomponents, internal interfaces