Experience with safe dynamic reconfigurations in component-based embedded systems

Juraj Polakovic - Sébastien Mazaré
Jean-Bernard Stefani - Pierre-Charles David

CBSE'07 - 9.7. - 11.7.2007 - Medford, MA, USA

FranceTelecom R&D
MAPS/AMS Lab Grenoble, France

SARDES Project, INRIA
INRIA Rhône-Alpes, Grenoble, France

Dynamic reconfiguration in emb.systems

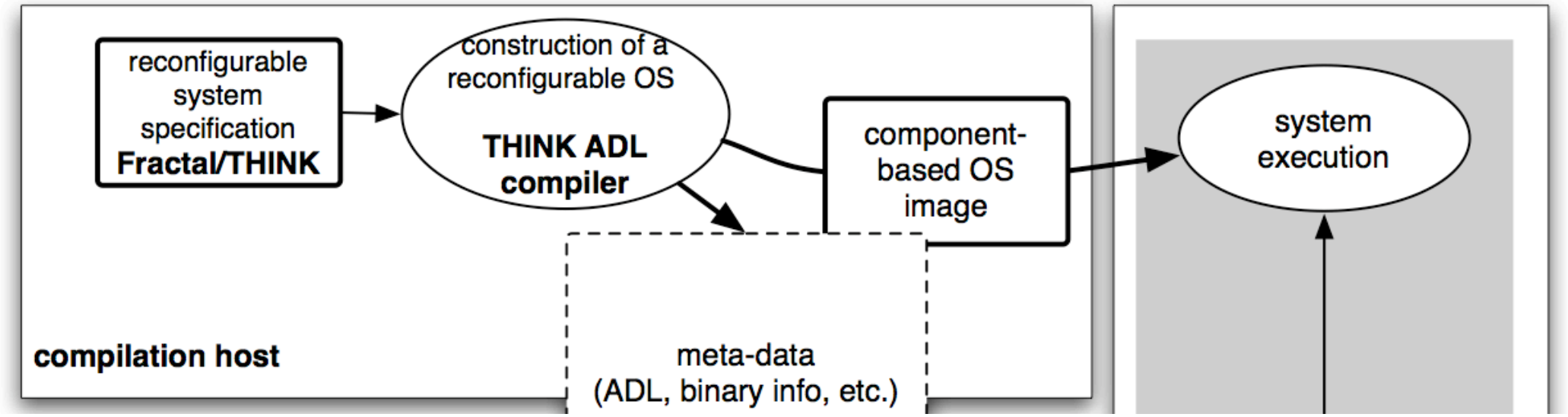
Dynamic reconfiguration - *"a system's capability to allow architecture (and behaviour) modifications during execution, without service interruption"*

- applications in OS
  - updates and bug-fixes
  - adaptive algorithms
  - extensions (3rd party modules, application-specific extensions, …)
  - monitoring, debugging (on-the-fly)
  - towards autonomic computing
  - …
- challenges
  - flexibility, efficiency, robustness, simplicity (Hicks)

Programming reconfigurations

- We build component-based OS …
- We have custom dynamic reconfiguration mechanisms …
  - reconfigurations as architectural change - add, remove, replace, (re)bind, attributes
- But …
  - reconfiguration programming in C - difficult and error-prone, too low-level
  - how to maintain a large device park with different configurations (mobiles)?
  - additionnal constraint - mechanisms scalable down to sensor networks
    - ATmega128/2561 (AVR) with 4kb (8kb) main memory and 128kb (256kb) program memory
- Idea: use a high-level reconfiguration language
  - combined with a component-based configuration representation (such as ADL)
  - build an appropriate support for resource-limited devices
- Alternatives
  - construct the OS and applications differently, i.e. VM (Mate on TinyOS)
  - use a native interpreter for the reconfiguration language (how in sensors?)

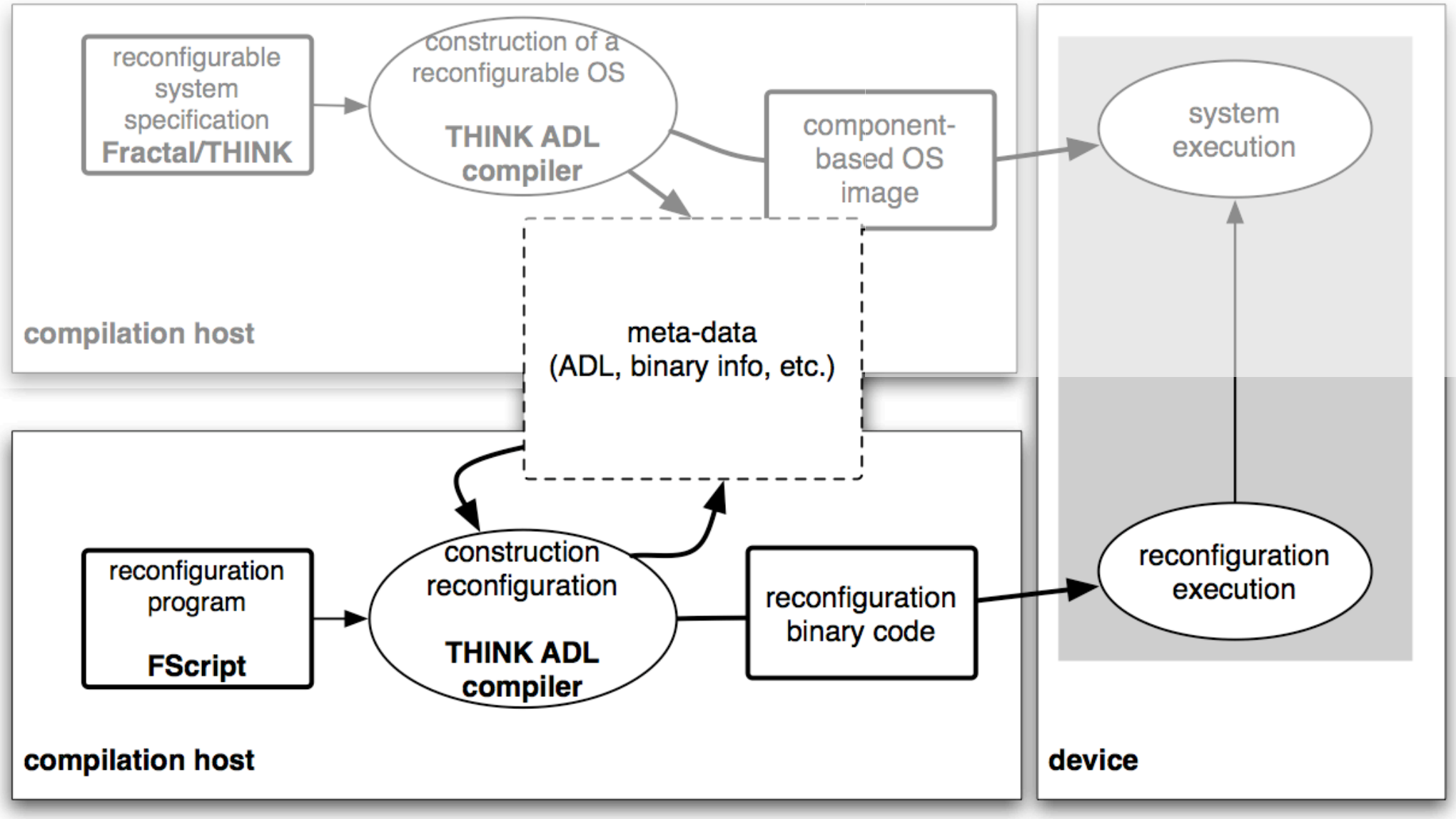
Extensive use of the Fractal comp. model



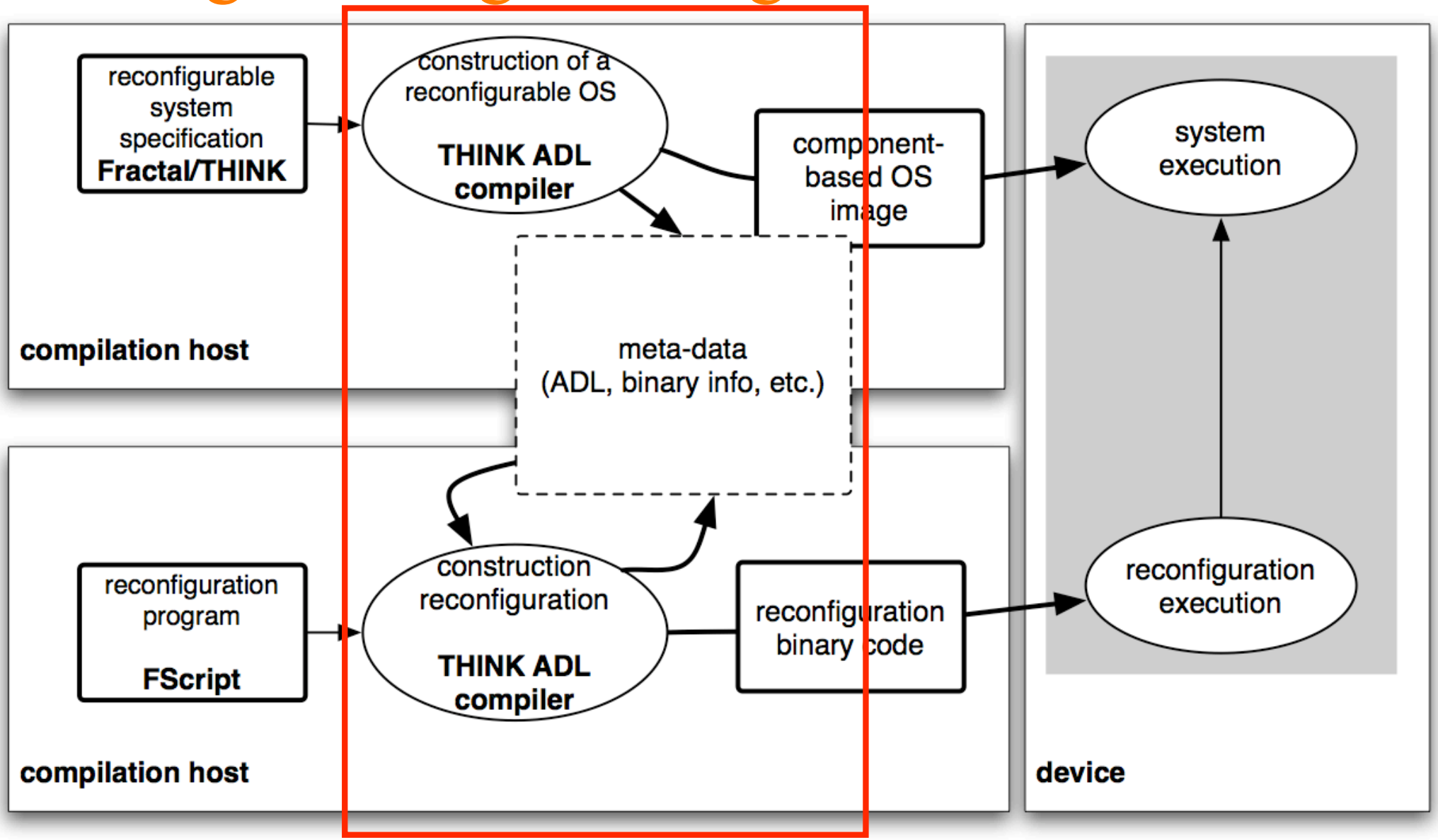
- build custom reconfigurable OS with the Fractal/Think framework
  - Think - a C implementation of the Fractal Component Model (native components)
  - systems ‘à la carte’ - everything as a component - optimal runtime performances
  - no predefined system philosophy (micro-kernel, exokernel etc.)
  - no predefined core functionnality (scheduler, memory management etc.)

- custom dynamic reconfiguration mechanisms (Euromicro CBSE’06)
  - various mechanisms available, different trade-offs performance-flexibility
  - separation of concerns (functionnal vs. control) - using Fractal control interfaces
  - custom granularity of reconfigurations (Fractal’s hierarchic composition)
  - mechanisms added by the compiler, based on user specification

Programming reconfigurations - overview



Programming reconfigurations - overview



FScript language

- FScript (P.-C. David, PhD)
  - reconfiguration DSL for Fractal (access to all Fractal API's)
  - simple statements, actions, control structures
  - FPath expressions to navigate/select specific elements - sets (itf, comps etc.)

ADL

```
component SmallNetKernel {
  provides Net as net

  contains net = net.lib.stack
  contains alloc = memory.malloc
  contains eth = net.lib.tulip

  binds this.net to net.net
  binds net.eth to eth.driver
  binds net.alloc to alloc.alloc
  binds eth.alloc to alloc.alloc
}
```

FScript

```
k = $root/child::kernel
n = new('new_alloc');
o = $root/child::alloc;
k_alloc = $k/child::*/interface::alloc;

add($k, $n);
suspend($k, $k/child::alloc);
stop($o);
unbind($k_alloc);
bind($k_alloc, $n/interface::alloc);
start($n);
resume($k);
remove($o);
```

FScript support for embedded OS

- FScript « offline » compilation
  - **in** - system actual configuration (ADL), FScript reconfiguration program
  - **out** - binary reconfiguration comonent (with new instances)
  - verifications (using simulation, itf types, etc.)
- Target device - Think-based component OS
  - communication link - FScript compilation host
  - dynamic loader
  - minimal FScript run-time

Problems

- Consistency of architecture views
  - offline reconfiguration compiler vs. device
- FScript conditionnals and FPath expressions compilation
  - or, how to compile the reconfig. program in an intelligent way
  - run-time evaluation needs memory allocation
  - current solution - simulation
- Size of the resulting binary component !!
  - final binary component - relocatable object
  - using ELF format
  - overhead due to the Think glue code - cross-referenced C variables
  - overhead proportionnal to # of new components, not the reconfig. code
  - not yet suitable for sensors (ex. simple reconfig: 4,5kb…)

Conclusion

- FScript reconfiguration programs compilation seems promising
  - still ongoing
  - some ideas for offline pre-linking the reconfig. component - no ELF bloat
- Fractal/Think and FScript meet challenges of reconfigurable OS construction
  - flexibility
  - efficiency
  - robustness
  - simplicity
- Think framework
  - http://think.objectweb.org

Future work

- Event-based execution and dynamic reconfiguration
- Optimisations, or "*selective architectural optimisations*"
  - customized glue code generation
  - eliminate **selectively all** overhead due to the use of components
  - combine optimisations and reconfiguration specifications
  - impact on the FScript compilation - probably eliminate the ELF bloat
- Custom Java Virtual Machines on resource-limited devices
  - customized kernels for JVM + OSGi
  - Atmel AT926 (+Jazelle)
  - dynamic reconfiguration ?