

# Static and Dynamic Weaving in System Software with AspectC++

---

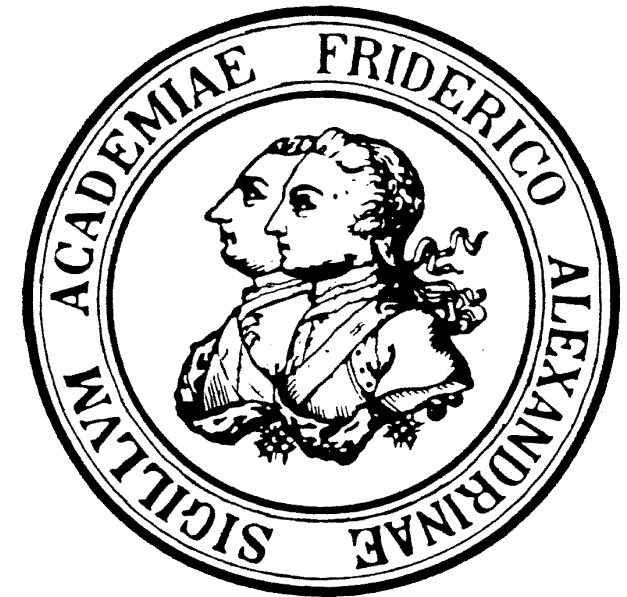
Wolfgang Schröder-Preikschat

***Daniel Lohmann***

Fabian Scheler

Wasif Gilani

Olaf Spincyk



Department of Computer Science IV  
Distributed Systems and Operating Systems  
Friedrich-Alexander-University Erlangen-Nuremberg

<http://www4.cs.fau.de/>



# Motivation

---

**Ambient  
Intelligence**

**Body Area  
Networks**

**Pervasive  
Computing**

**Mobile  
Computing**

**Ubiquitous  
Computing**

**Smart Dust**

**Smart  
Devices**

**Nomadic  
Computing**

**Sensor  
Networks**

**Wearable  
Computers**



# Motivation

---

- **Large networks of very small (mobile) devices**
  - broad variety of systems and platforms
  - still dominated by **8 bit  $\mu$ -Controllers**
  - strictly limited resources (memory, CPU, energy)
  - high demands on **functional adaptability**



# Motivation

---

- **Large networks of very small (mobile) devices**
  - broad variety of systems and platforms
  - still dominated by **8 bit  $\mu$ -Controllers**
  - strictly limited resources (memory, CPU, energy)
  - high demands on **functional adaptability**
- **This has to be reflected in the system software**
  - **configurable** and **application-tailorable** operating systems
  - operating system product line, based on feature modelling
  - **fine-grained features**
  - on-demand **reconfigurability**



# Motivation

---

- **Large networks of very small (mobile) devices**

- broad variety of systems and platforms
- still dominated by **8 bit  $\mu$ -Controllers**
- strictly limited resources
- high demand for energy efficiency

**Aspect-oriented techniques  
are particularly useful**

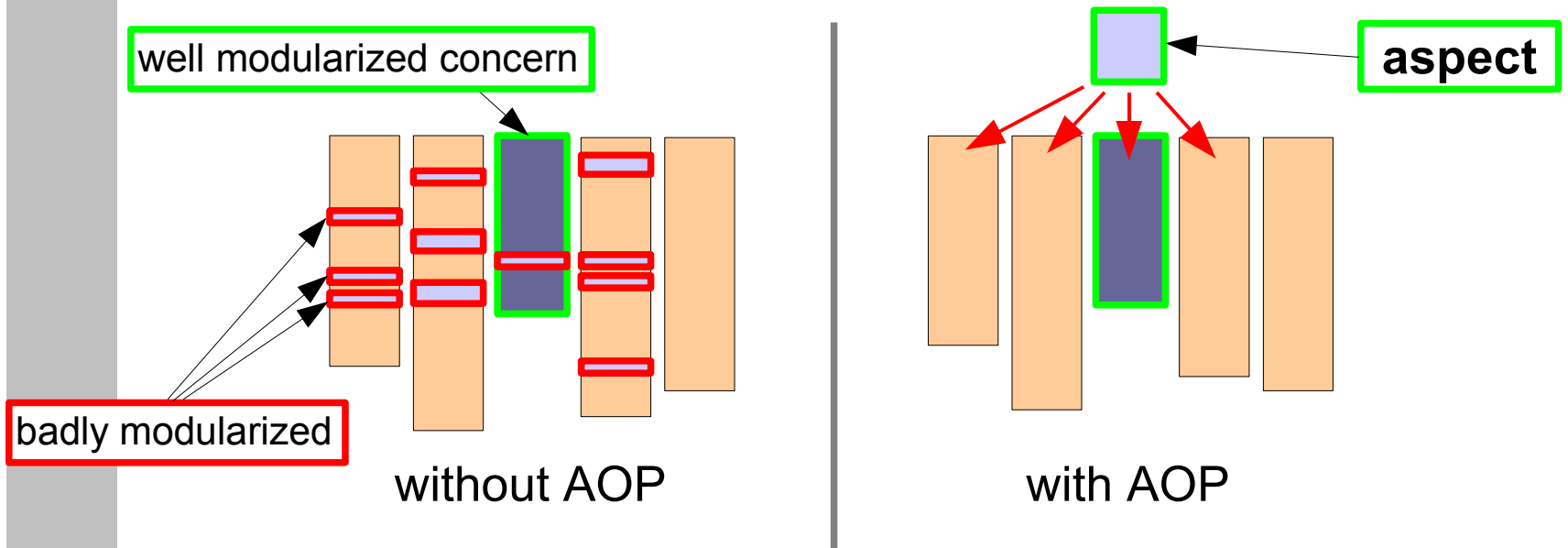
- **This has to be reflected in the system software**

- **configurable** and **application-tailorable** operating systems
- operating system product line, based on feature modelling
- **fine-grained features**
- on-demand **reconfigurability**



# Aspect-Oriented Programming

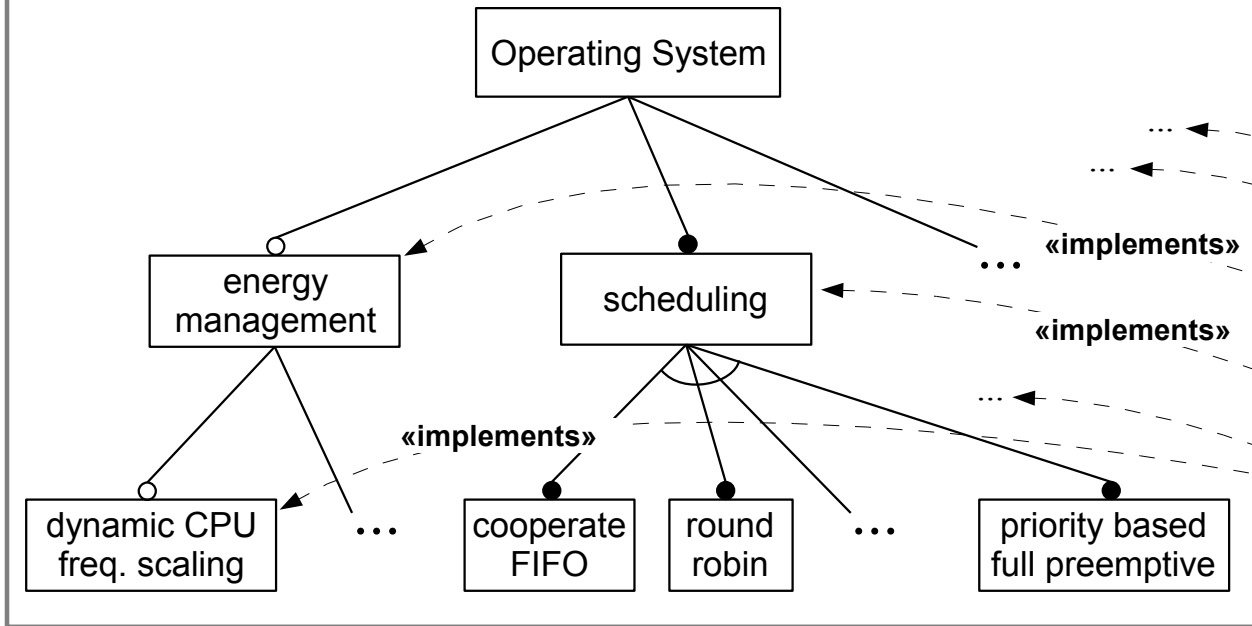
- AOP is about modularizing crosscutting concerns



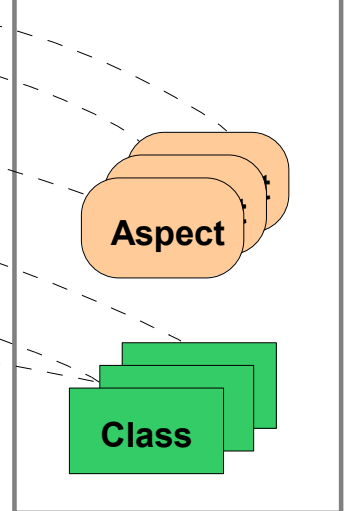
- AOP thereby provides loose coupling of components



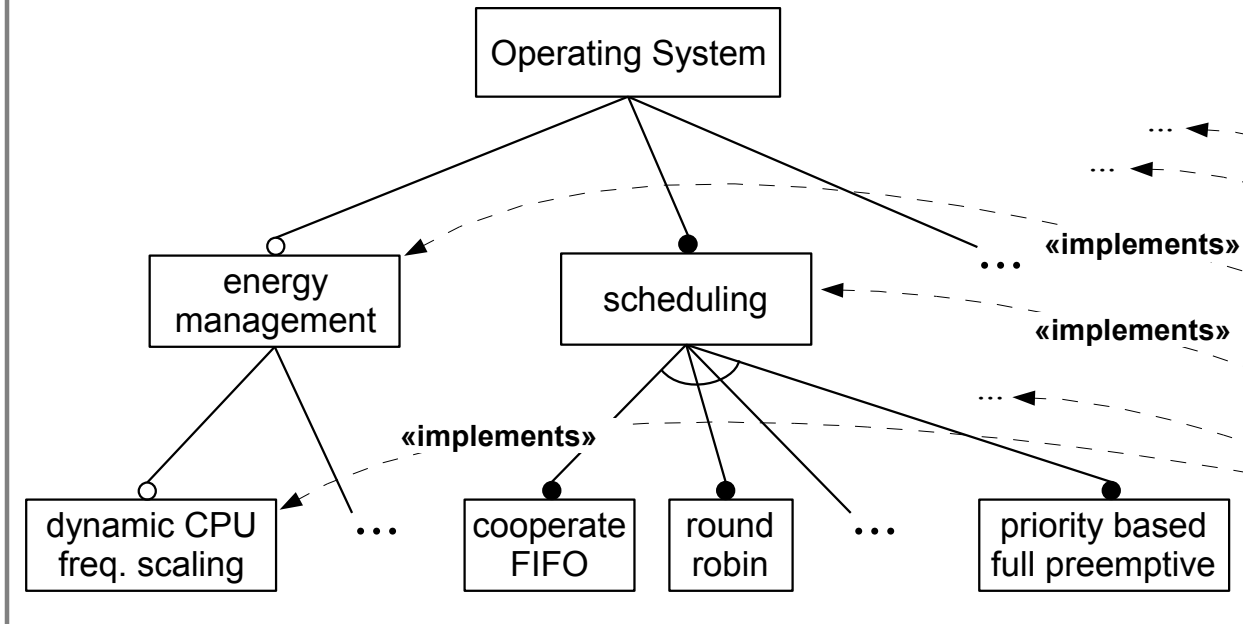
# Operating System Feature Modell



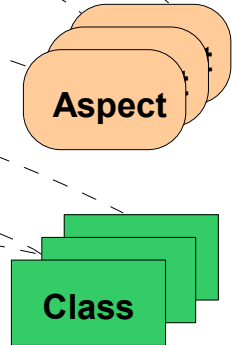
# Implementation Components



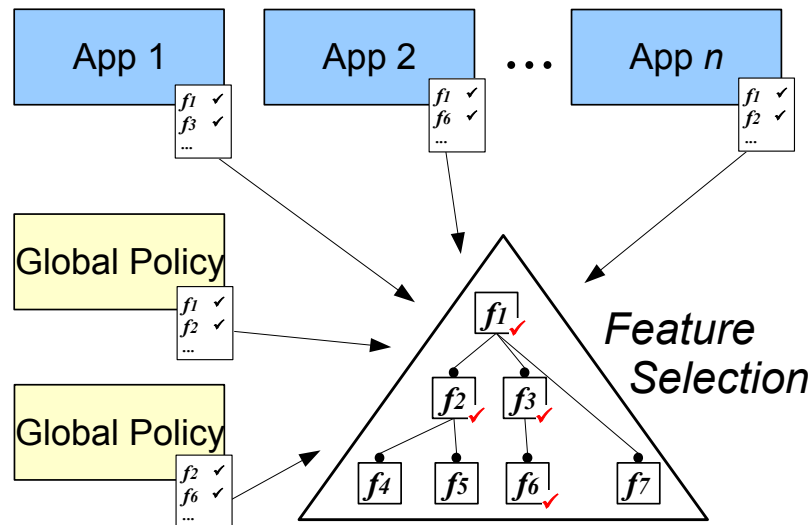
# Operating System Feature Modell



# Implementation Components

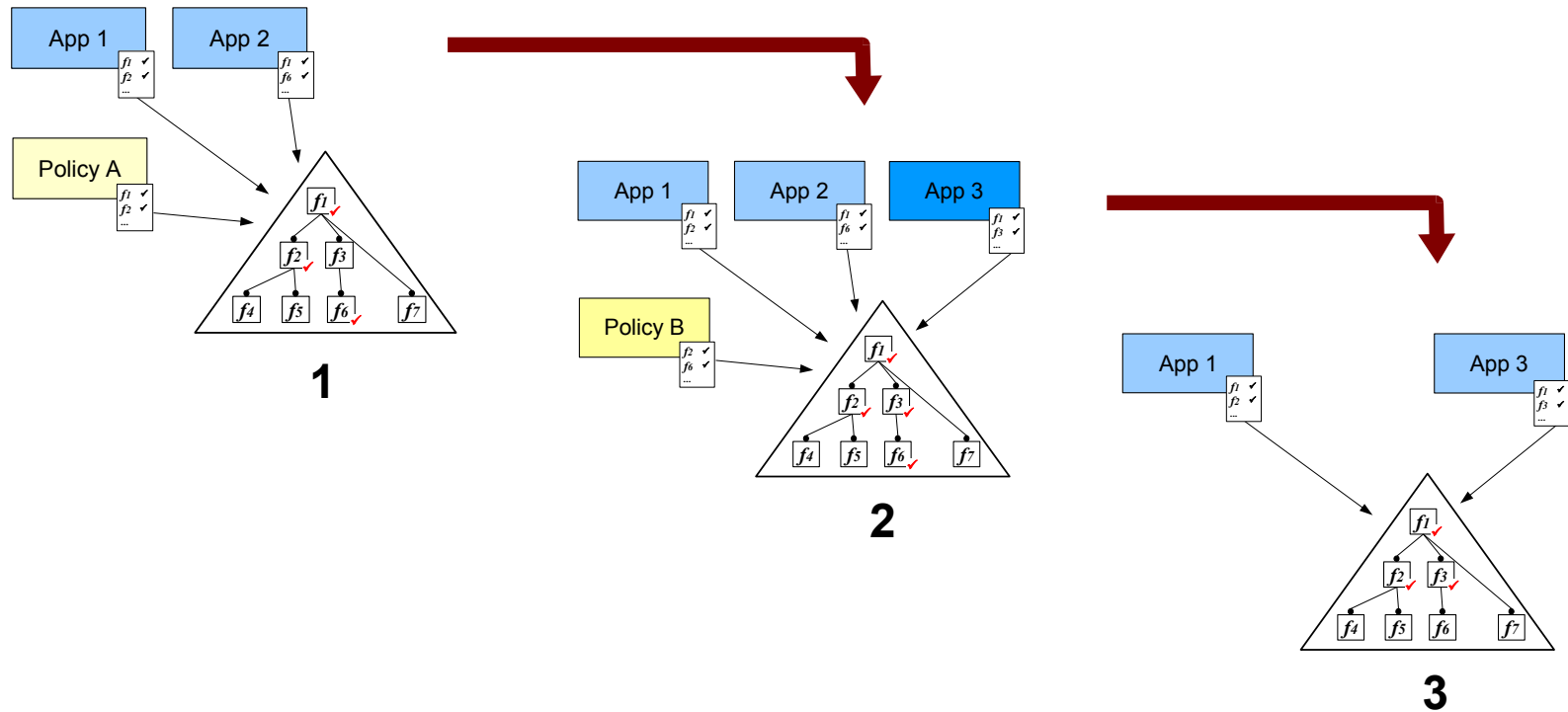


application-specific  
feature selection  
(Tailoring)



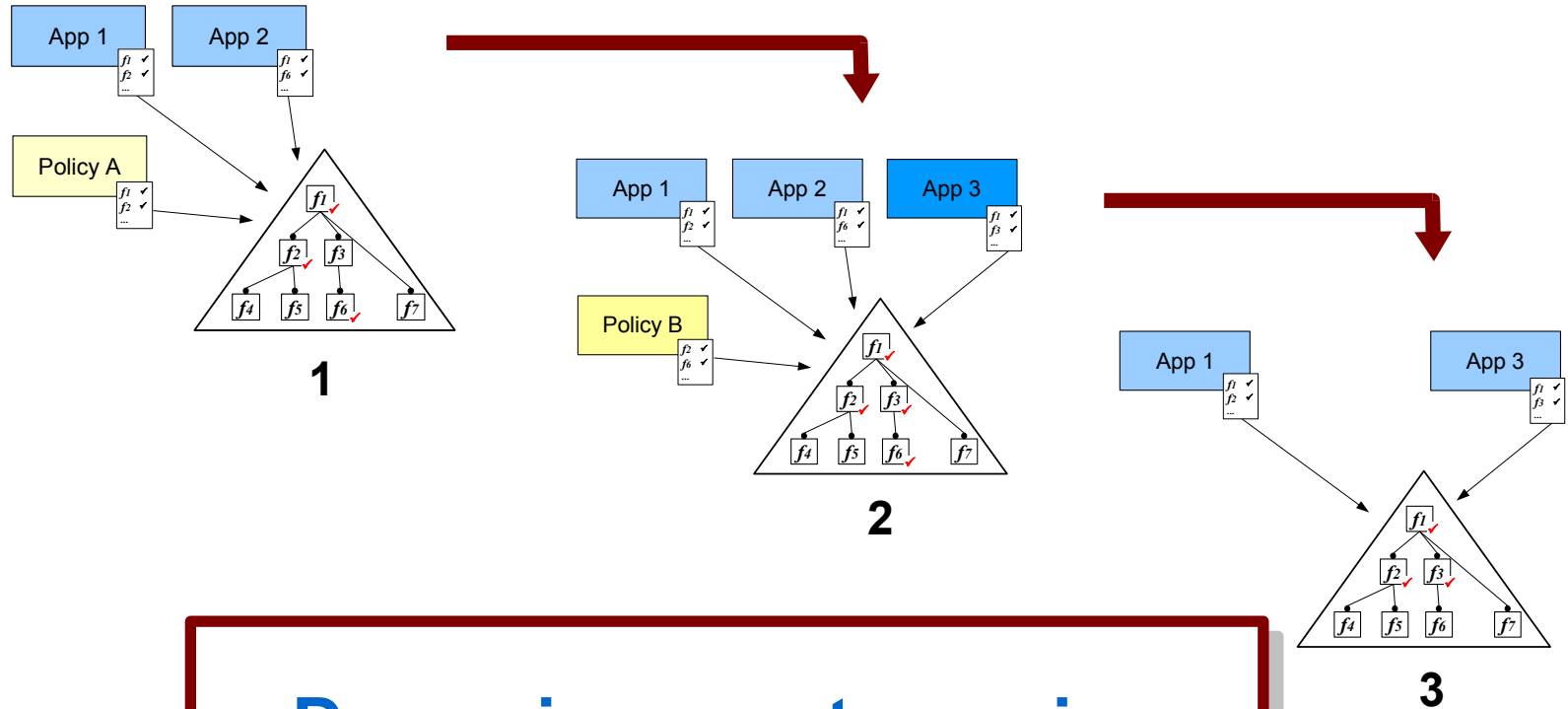
# Scenario: Adaptable System Software

Set of required features may change at runtime



# Scenario: Adaptable System Software

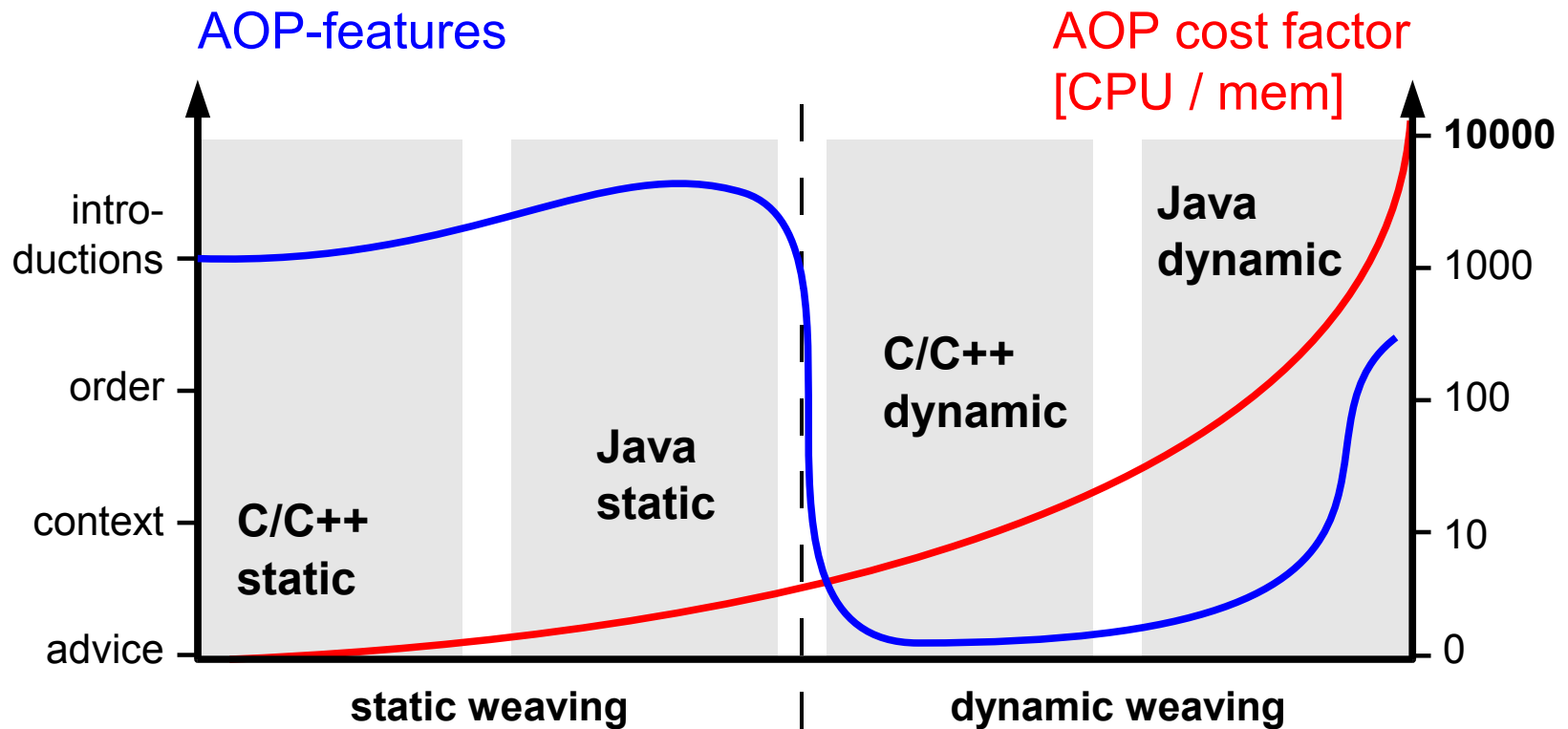
Set of required features may change at runtime



**Dynamic aspect weaving**



# Problems of Dynamic Weaving



- **Fewer** AOP features at **higher** costs
- Costs **not affordable** for small  $\mu$ -Controllers











# Reason

- Incredibly high number of *potential* join-points
- Code has to be prepared / interpreted / attributed / ...

execution join point 

call join point 

```
—  void getUser( char* user ) {  
—  scanf( "%s", user );  
— }  
  
—  void printHello( const char* user ) {  
—  printf( "Hello, %s. ", user );  
—  printf( "How do you do?" );  
— }  
  
—  int main() {  
—     char user[ 80 ];  
—  getUser( user );  
—  printHello( user );  
— }
```



# Our Approach

---

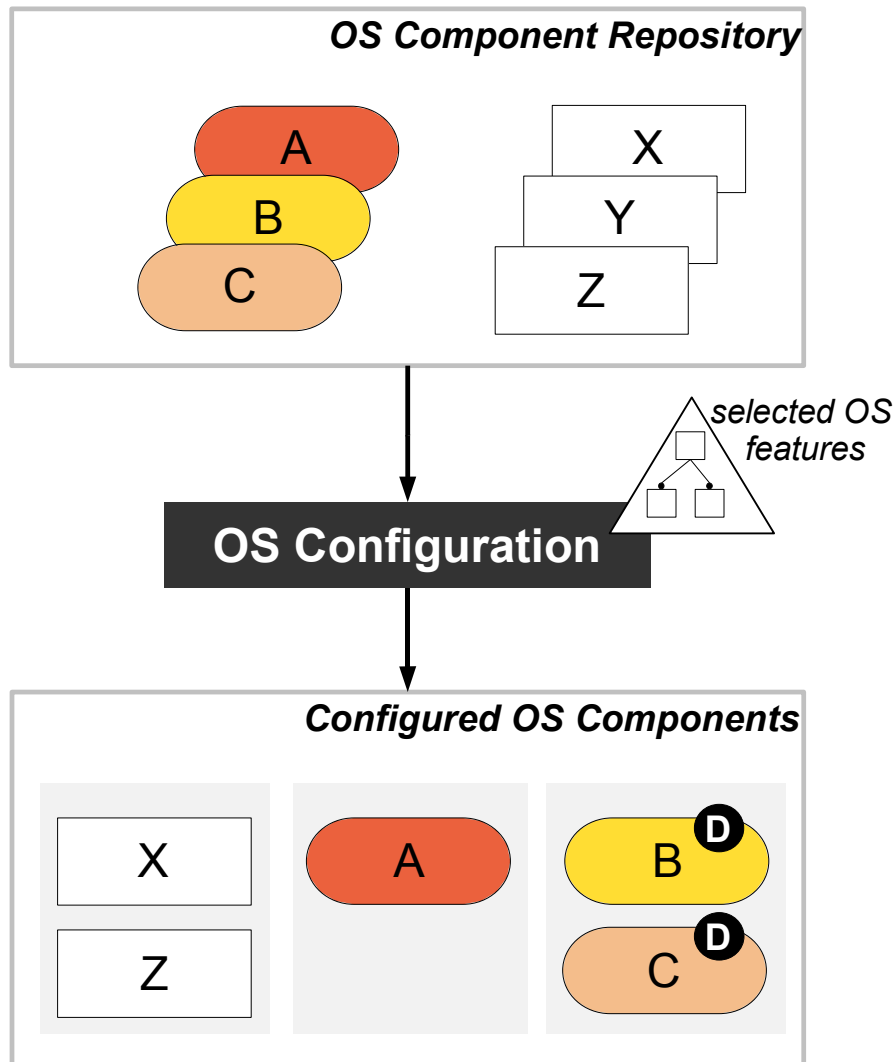
- **static *and* dynamic weaving**
  - many aspects do not need to be changed at runtime
  - combine the **flexibility** of dynamic weaving with the **resource-thriftiness** of static weaving
- **single language approach: AspectC++**
  - one language for both, static and dynamic aspects
  - binding of features becomes **configurable**
- **application-specific tailoring** of the dynamic weaver
  - incorporate ***a priori knowledge*** about the system
  - restrict the set of supported AOP-Features
  - restrict the set of dynamic join-points



***“Do as much as possible statically”***



# Combining Static and Dynamic Weaving – 1



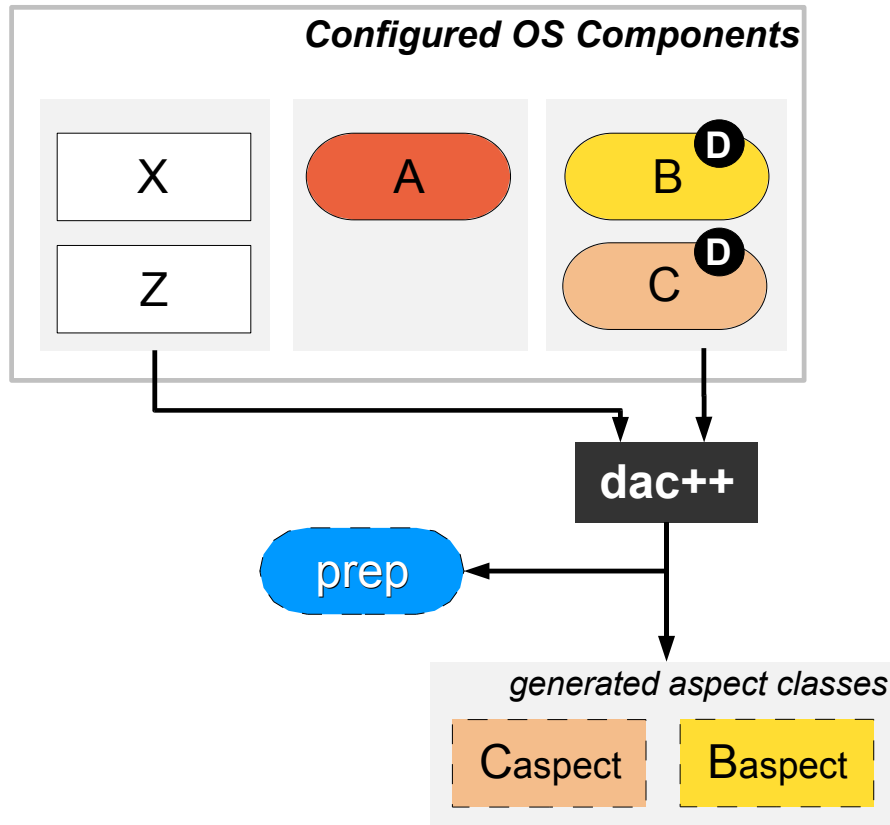
*single language approach*

binding-time independent  
aspect implementations

aspects with configured  
binding time



# Combining Static and Dynamic Weaving – 2

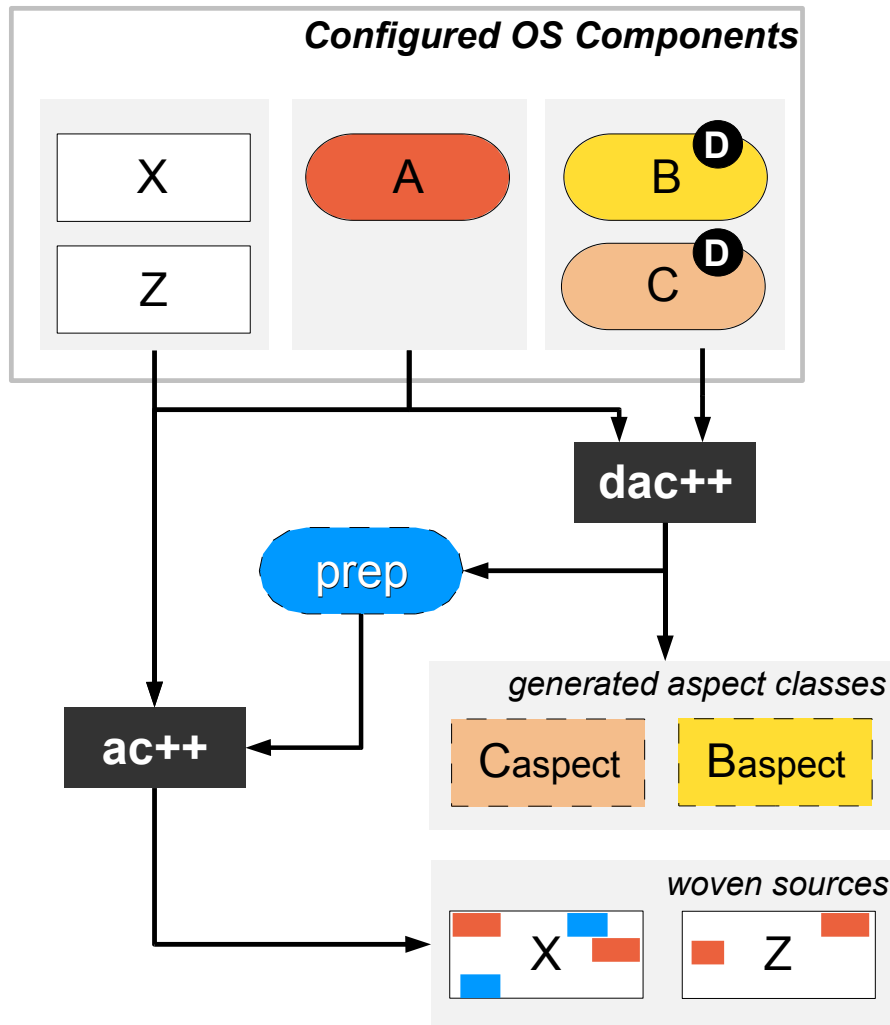


aspects with configured binding time

instrumentation aspect



# Combining Static and Dynamic Weaving – 3

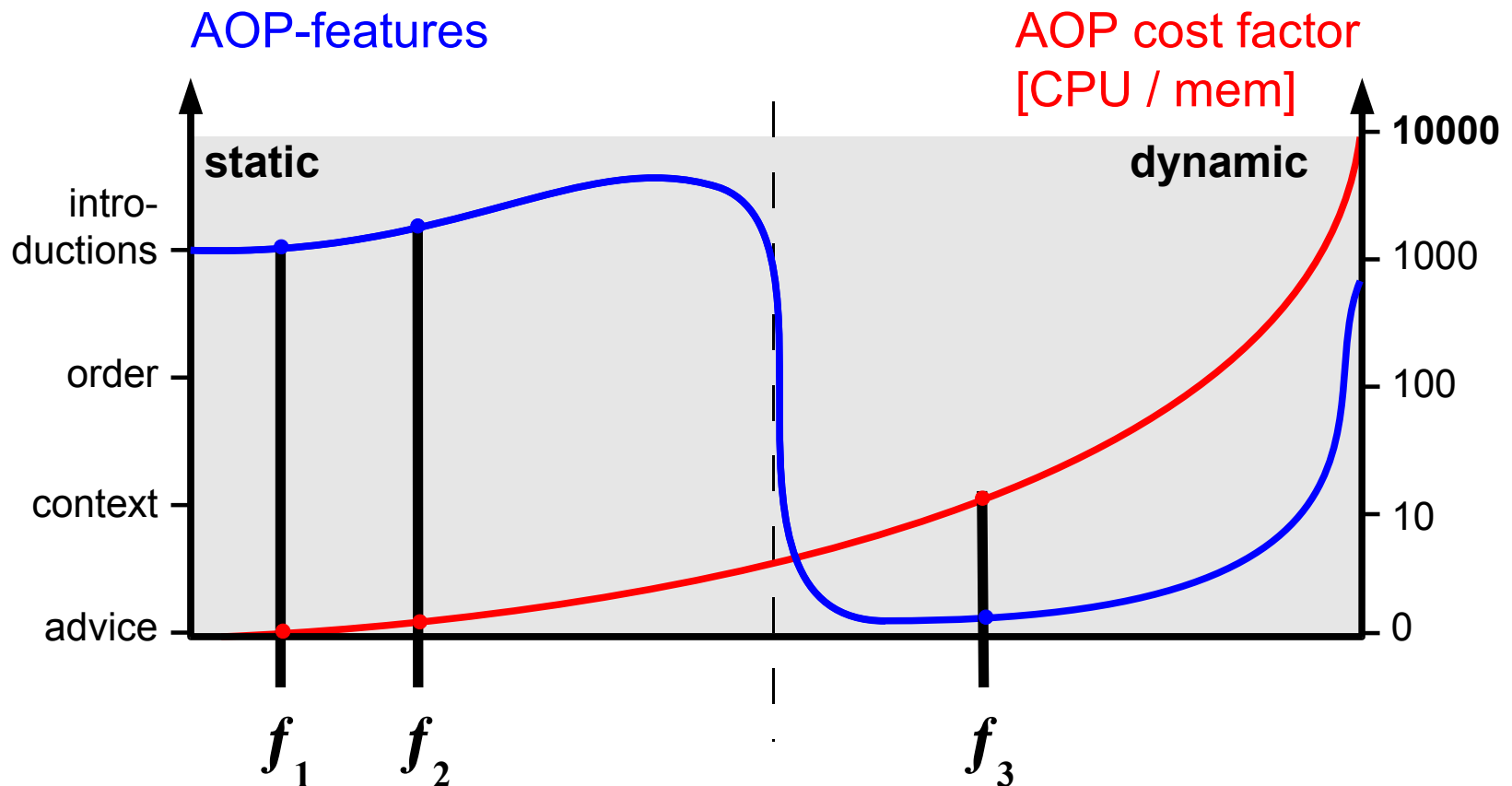


aspects with configured binding time

instrumentation aspect



# Combining Static and Dynamic Weaving – Result



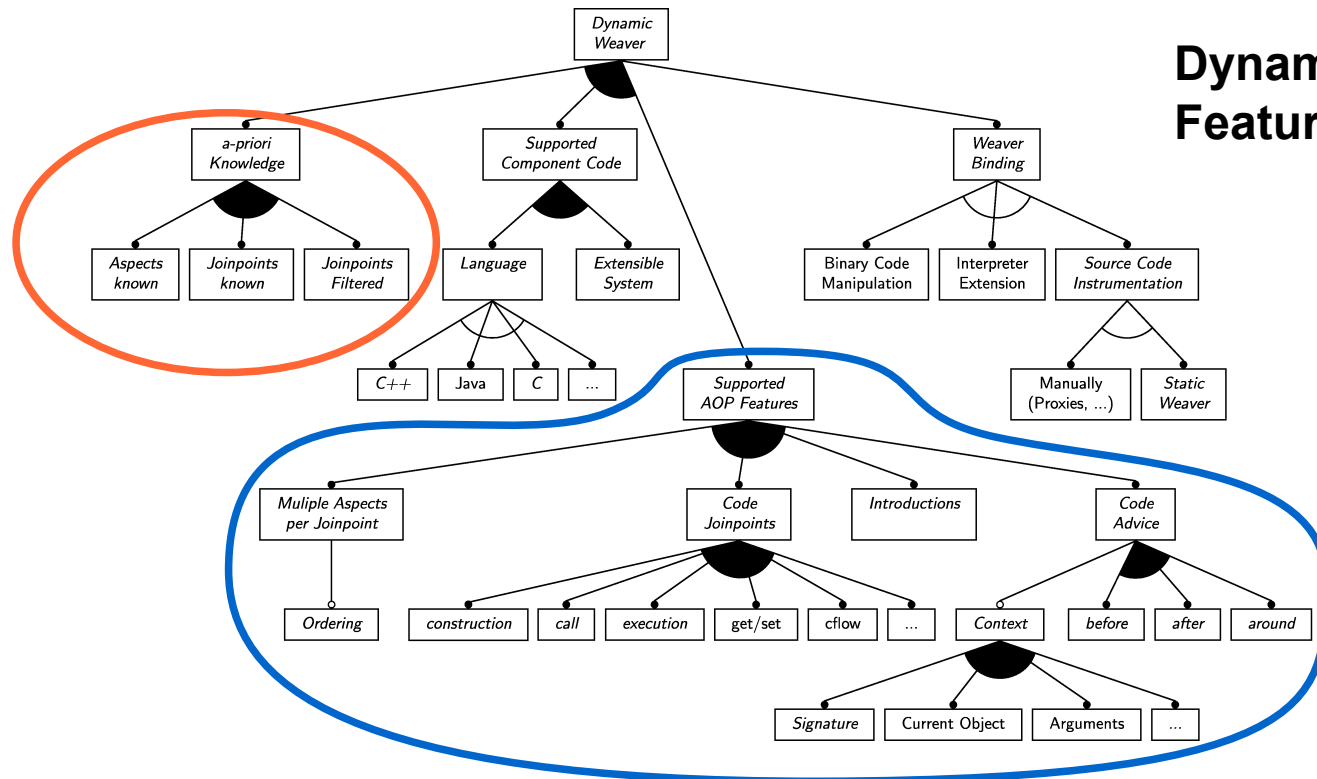
**per OS-feature configurability of the costs – dynamism trade off**



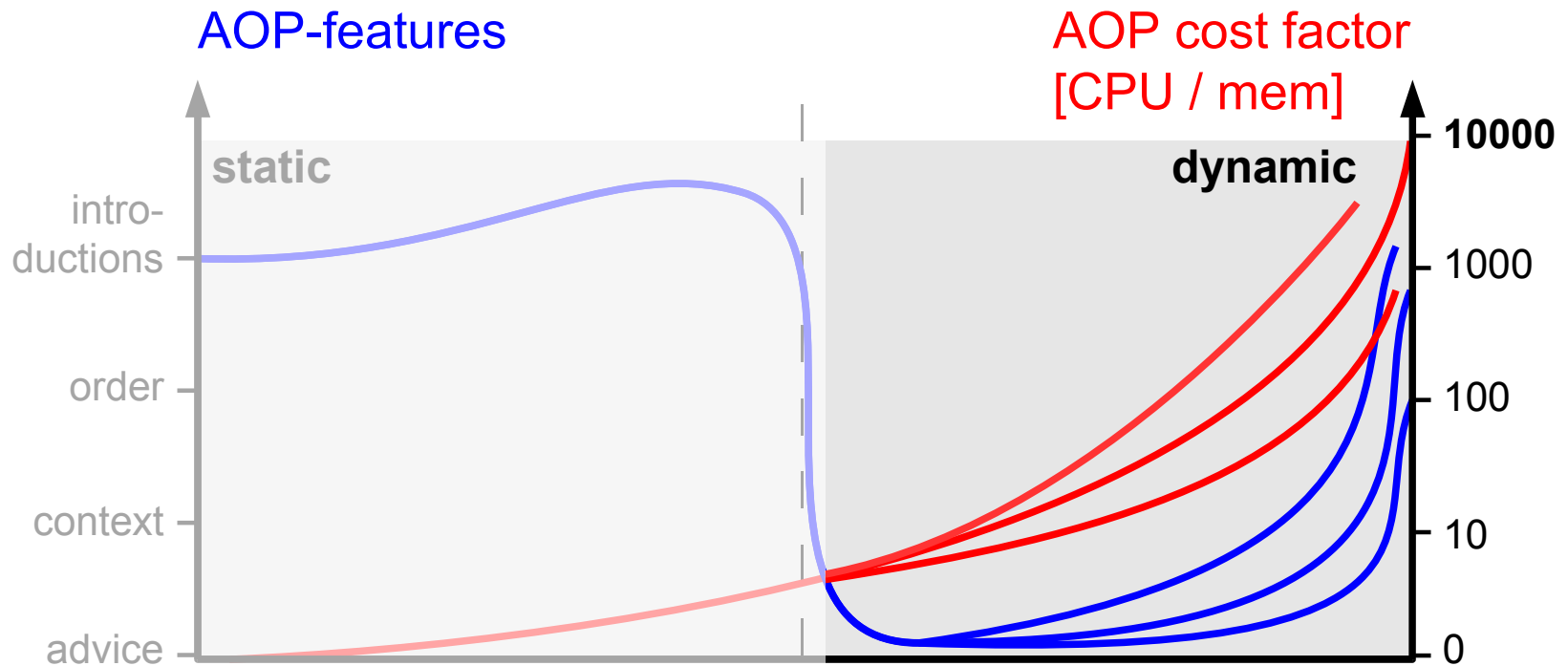
# Tailoring of the Dynamic Weaver – 1

Understand a Dynamic Weaver as a **Product Line**

- fine-grained **tailoring of the amount of join-points**
- fine-grained **tailoring of AOP features**



# Tailoring of the Dynamic Weaver – Result



**configurability of the costs – AOP features trade off**



# Case Study

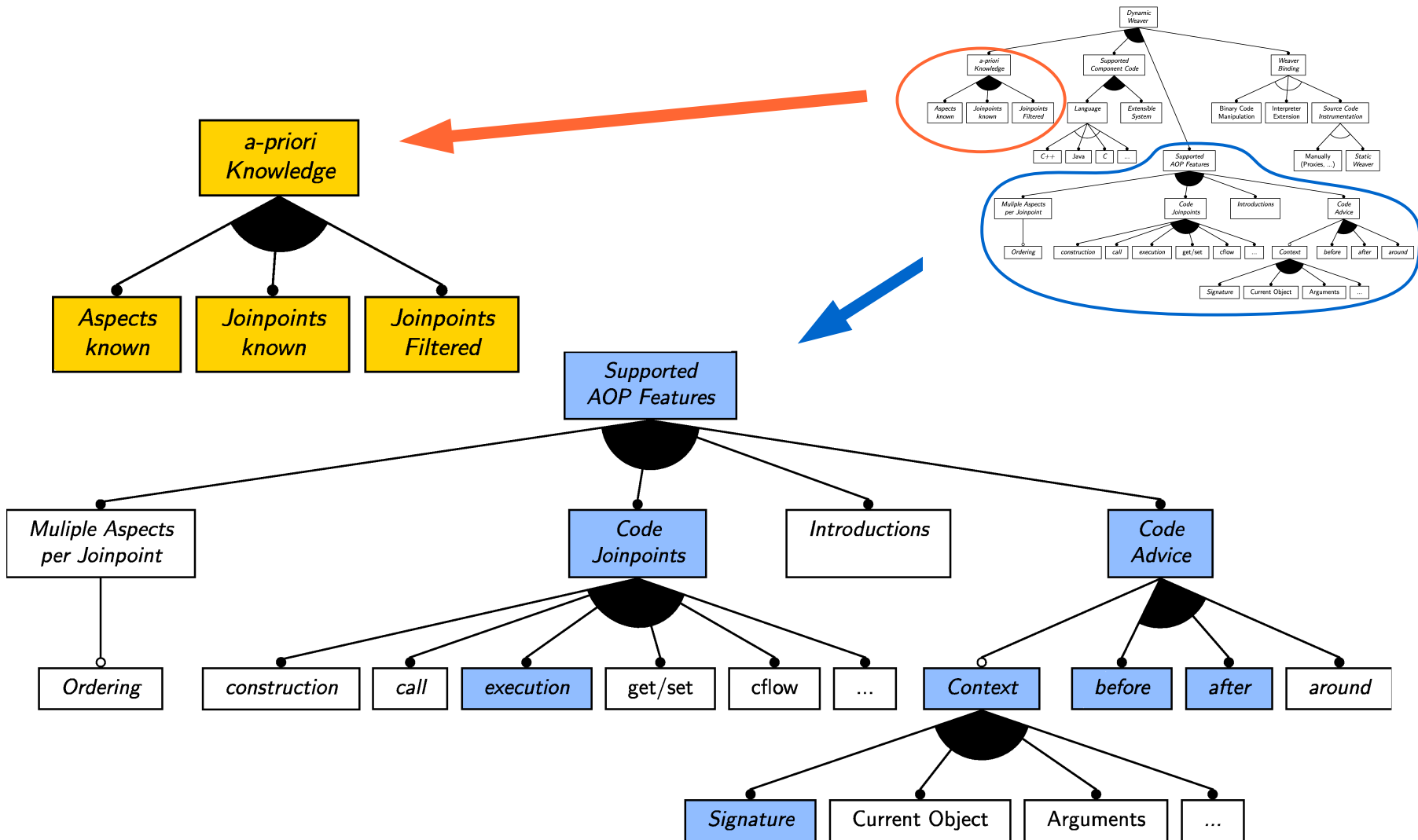
---

## Static and Dynamic Weaving in the eCos Kernel

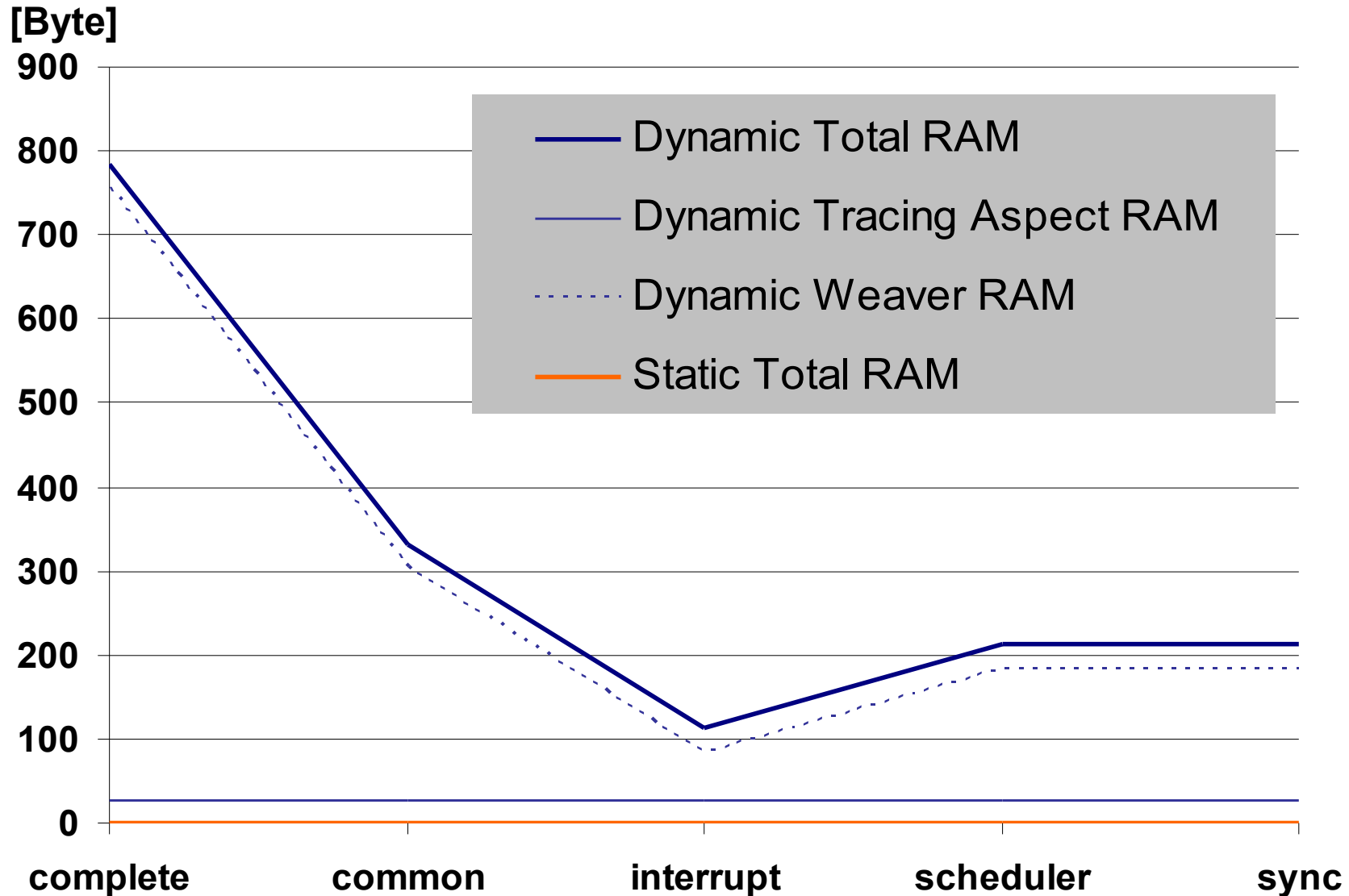
- eCos: highly configurable OS for embedded systems
  - C++ Kernel (5200 LOC)
- analysed and refactored CCCs in the eCos kernel
  - as a larger case study
  - overall 17 concerns refactored into static AspectC++ aspects
- experimental platform for our dynamic weaver family
  - Tracing
  - Counting



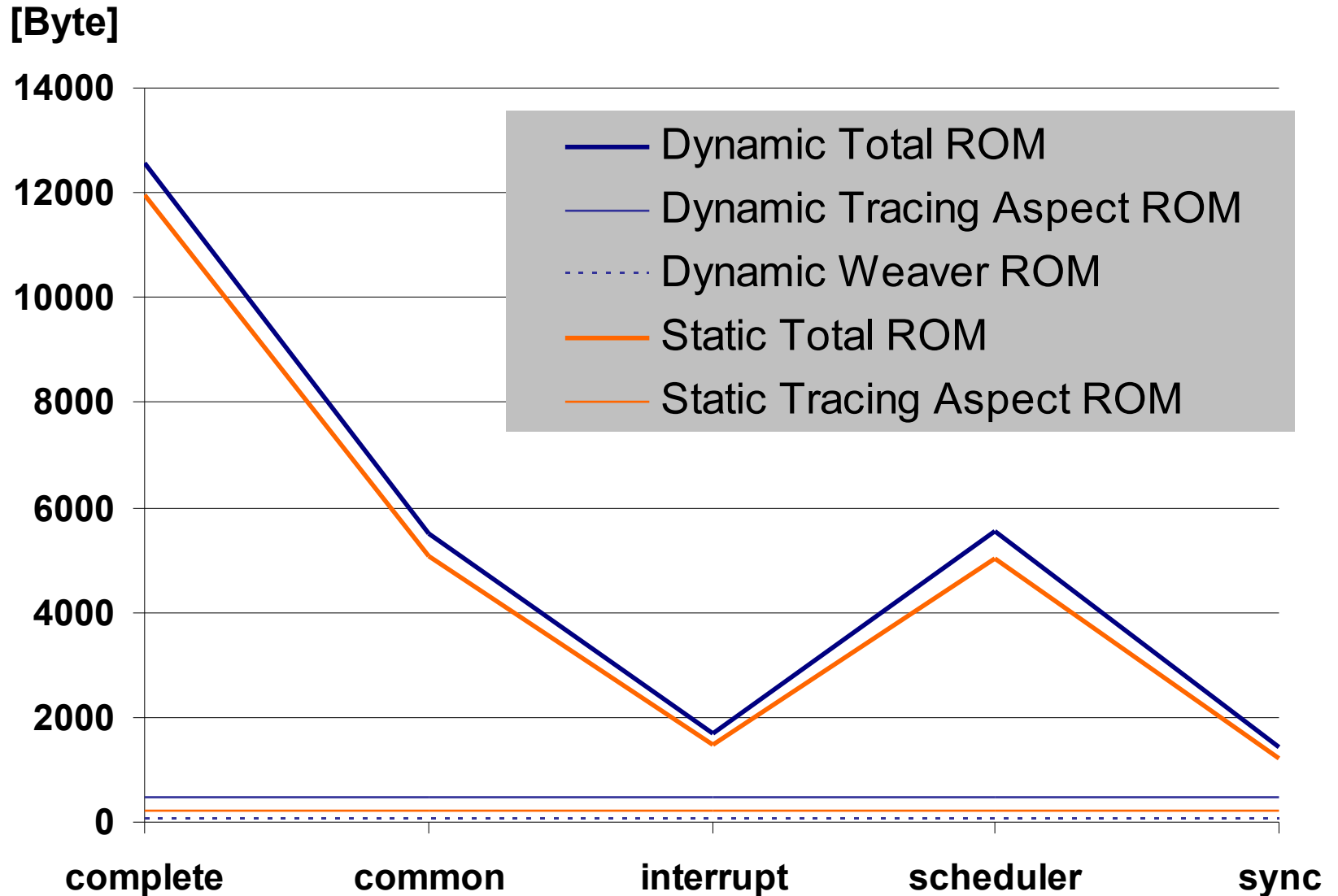
# Dynamic Weaver Configuration



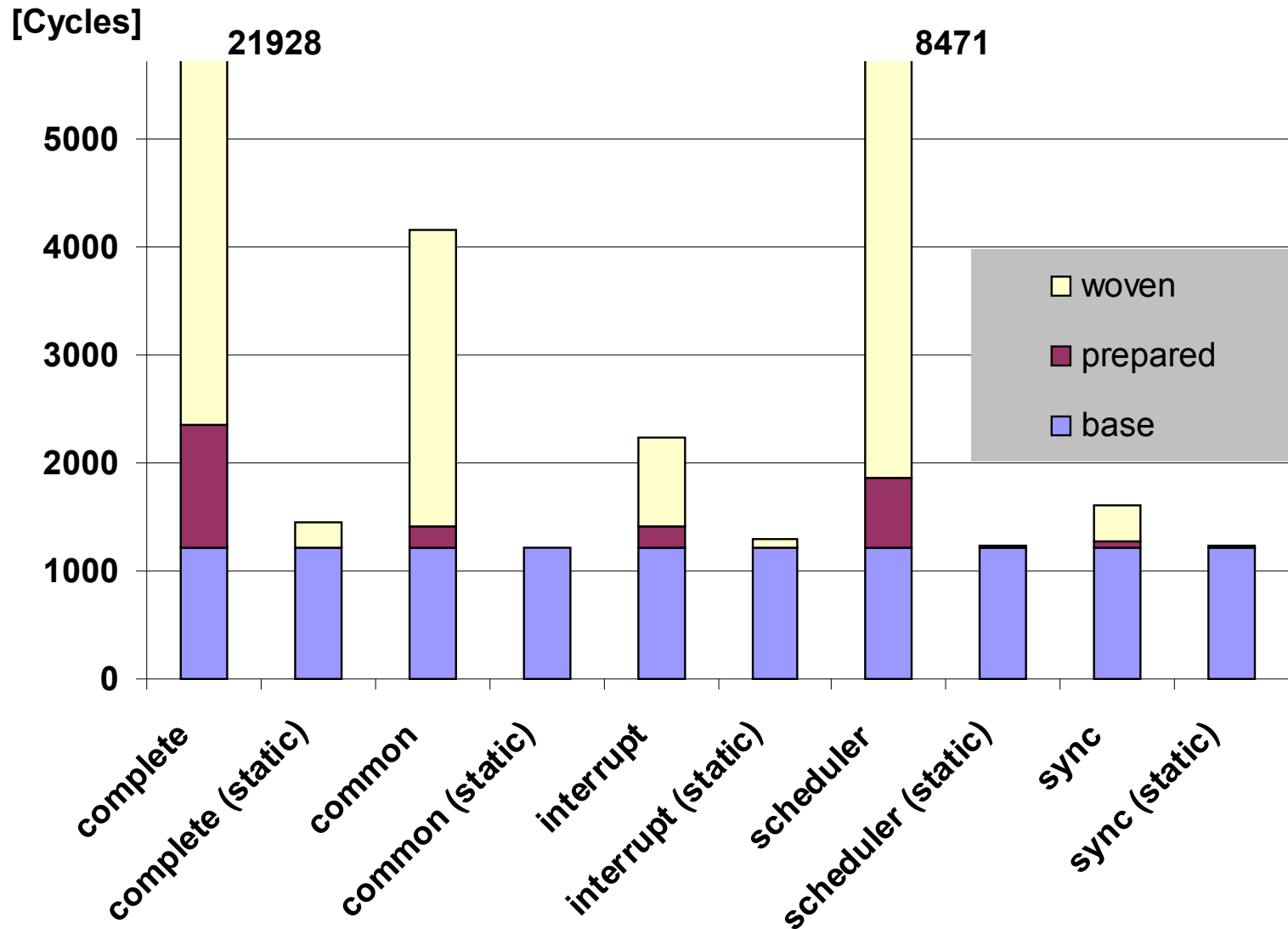
# Cost Analysis: RAM Overhead (Tracing)



# Cost Analysis: ROM Overhead (Tracing)



# Cost Analysis: Performance Overhead (Counting)



# Summary and Conclusions

---

- small and adaptable embedded devices
  - system software as product line
  - dynamic weaving
- current approaches for dynamic weaving not suitable
  - fewer AOP features at higher costs
  - not affordable for typical  $\mu$ -Controllers
- **idea 1:** combination of static and dynamic weaving
  - single language approach
  - aspect binding time becomes a matter of configuration
- **idea 2:** tailoring of the dynamic weaver
  - dynamic weaver as product line
  - use of *a-priori* knowledge to tailor join-points and AOP features



# Summary and Conclusions

---

- small and adaptable embedded devices

## Results:

- **fine-grained configurability of “dynamism”**
  - **affordable even for 8bit  $\mu$ -controllers**
- 
- **idea 1:** combination of static and dynamic weaving
    - single language approach
    - aspect binding time becomes a matter of configuration
  - **idea 2:** tailoring of the dynamic weaver
    - dynamic weaver as product line
    - use of *a-priori* knowledge to tailor join-points and AOP features



# Future Work

---

- improve implementation
  - dac++ still requires much manual intervention
- more case studies and applications
  - currently working on: MyServer, Squid, ACE
  - planned: Bochs
- server-side weaving
  - approach for extremely small systems
  - rebuild complete configuration on a remote server
  - transmit and apply only image deltas to the device

**Thank you four your attention**

