Developing Embedded Software
Product Lines with AspectC++

AOSD 2005 Demonstration
Presenters

- Olaf Spinczyk
  os@aspectc.org
  University of Erlangen-Nuremberg, Germany

- Daniel Lohmann
  dl@aspectc.org
  University of Erlangen-Nuremberg, Germany
In this demo we present...

- **AOP in deeply embedded devices**
  - AOP is suitable for resource-thrifty domains!

- **AOP in software product line development**
  - Aspects provide great benefit here!

- **AspectC++ features**
  - Practical solutions for practical problems

- **A complete Eclipse-based tool chain**
  - AspectC++ Eclipse Plugin (ACDT)
  - pure::variants Eclipse Plugin
Demo Scenario

- **Embedded weather station product line**
  - sensors: wind, temperature, air pressure, ...
  - actors: display, alarm, PC connection, ...

- **Based on a small AVR ATmega µ-controller**
  - 8 Bit 4MHz RISC CPU
  - 2 – 128 kb Flash
  - 0.5 – 4 kb RAM
  - digital, analog, serial and I²C based I/O
Demo Scenario

- Embedded weather station product line
  - sensors: wind, temperature, air pressure, ...
  - actors: display, alarm, PC connection, ...

- Based on a small AVR ATmega µ-controller
  - 8 Bit 4MHz RISC CPU
  - 2 – 128 kb Flash
  - 0.5 – 4 kb RAM
  - digital, analog, serial and I²C based I/O

AOP on this platform?

“Hello World” in AspectJ takes around 20 MB RAM (on a PC)...

© 2005 Daniel Lohmann and Olaf Spinczyk
Demonstration Platform

Sensors

Wind
Pressure
Temp

µController (AVR)

I²C

USB
RS232

Display
Weather Station Variants

- Thermometer: LCD, Temperature
- Home: LCD, Temperature, Pressure
- Outdoor: LCD, Temp., Pressure, Wind
- Deluxe variants: + PC Connection
- PC-only variants: + PC Connection - LCD
- Serial PC Connection
- USB PC Connection
- ...

© 2005 Daniel Lohmann and Olaf Spinczyk
Weather Station Feature Model

WeatherMon

Actors
- Alarm
- Display
- PC Connection

Sensors
- Temperature
- Air Pressure
- Wind Speed

RS232Line
USBLi ne
Protocol
- SNGProto
- XMLProto
```cpp
int main() {
    Weather data;
    Sink    sink;

    while(true) {
        // aquire data
        data.measure();

        // process data
        sink.process(data);

        wait();
    }
}
```
Sensor Integration

Weather::measure() {
    _pressure.measure();
    _wind.measure();
    _temp.measure();
}

Sink::process(const Weather& w) {
    process_data(w._pressure);
    process_data(w._wind);
    process_data(w._temp);
}

Design

© 2005 Daniel Lohmann and Olaf Spinczyk
Sensor Integration

Weather::measure() {
    _pressure.measure();
    _wind.measure();
    _temp.measure();
}

Sink::process(const Weather& w) {
    process_data(w._pressure);
    process_data(w._wind);
    process_data(w._temp);
}

---

...crosscuts the modules

© 2005 Daniel Lohmann and Olaf Spinczyk
Weather::measure() {
}

Sink::process(const Weather& w) {
}

Sensor Integration with Aspects

Design

Weather
measure()

Pressure
Handling

Pressure
measure()

Wind
measure()

Temperature
measure()
Sensor Integration with Aspects

Weather::measure() {
}

Sink::process(const Weather& w) {
}

Sensor Integration with Aspects

Weather
measure()

Sink
process(Weather)

Pressure
measure()

Wind
measure()

Temperature
measure()
Sensor Integration with Aspects

```
Weather::measure() {
}

Sink::process(const Weather& w) {
}
```

![Diagram of sensor integration with aspects](image)

Design

© 2005 Daniel Lohmann and Olaf Spinczyk
Sensor Integration with Aspects

Weather::measure() {
    _pressure.measure();
    _wind.measure();
    _temp.measure();
}

Sink::process(const Weather& w) {
    process_data(w._pressure);
    process_data(w._wind);
    process_data(w._temp);
}

Design

loose coupling of sensor slices
AspectC++

...at work
Actor Integration

process(Weather)

process_data (Pressure)  process_data (Wind)  process_data (Temperature)

Display
Actor Integration

- process(Weather)
  - process_data (Pressure)
  - process_data (Wind)
  - process_data (Temperature)
- XMLConnection
- Display
AspectC++

...at work
AspectC++ Join-Point API

Compile-Time Joinpoint API

JoinPoint::Result
JoinPoint::Arg< i >::Type
JoinPoint::Arg< i >::ReferedType
JoinPoint::ARGS

... 

Runtime Joinpoint API

Result* result()
Arg< i >::ReferedType* arg< i >()

... 

Type of the function result
Type of the $i^{th}$ function argument (with $0 \leq i < \text{ARGS}$)
Number of arguments
result value
value of $i^{th}$ argument
advice execution("void Sink::process_data(%)") : before() {
    ...
    String<4> val_str;
    tjp->arg<0>()->string_val(val_str);
    ...
}
Sensors/Actors Connection with Generic Advice

Integrating a new sensor requires no changes to other components.

- process(Weather)
- process_data (WindDir)
- process_data (Pressure)
- process_data (Wind)
- process_data (Temperature)

Display
advice execution("void Sink::process_data(%)") : before() {
... String<4> val_str;
    tjp->arg<0>()->string_val(val_str);
... }

Pressure::string_val()
Wind::string_val()
Temperature::string_val()
WindDir::string_val()
Design Conclusions

By using aspects, we achieved...

- complete decoupling of components
  - component slices are merged in by advice
  - actors and sensors “integrate themselves”
  - not a single #ifdef

- Plug & Play of components

...without sacrificing efficiency

- minimal stack usage due to advice code inlining
- everything is resolved at compile-time
  - no dynamic data structures to manage sensors/actors
  - no virtual functions
Configuration

WeatherMon

Actors

PC Connection

RS232Line

USBLine

Protocol

SNGProto

XMLProto

Sensors

Temperature

TempHandling

Sink

Temperature

USB

Serial

Weather

XMLConnection

SNGConnection

...
pure::variants

- General-purpose tool for product-line engineering
  - based on program families and feature modeling
  - not restricted to AOP or AspectC++
  - but provides some special support for aspects

- Implemented as an Eclipse-plugin

- Commercial product from pure-systems GmbH
  - free “community edition” available
  - http://www.pure-systems.com
Variant Management

- Problem domain
- Feature model
- Variant description
- Single problem

- Solution domain
- Family model
- Product variant
- Single solution

pure::variants
pure::variants

...at work
Scalability of the Product Line

Weather-Monitor Image Sizes (Bytes)

- Barometer
- Barometer PC
- Outdoor Basic
- Thermometer
- Outdoor Deluxe
- Outdoor Deluxe PC

Results
Summary: This Demo showed..

- Aspects in embedded product lines
  - loose coupling of components
  - Plug&Play configurability
  - highly efficient code

AOP provides real benefits for product-line development!

- Complete, Eclipse-based tool chain
  - AspectC++ Eclipse plug-in (ACDT)
  - pure::variants

All required tools are available today!
Questions?

Download AspectC++ from

www.aspectc.org

More about pure\::\::variants at

www.pure-systems.com
PC Connection Integration

PCConnection

- USB
  - send(char*)
- Serial
  - send(char*)

«alias»
PCConnection

XMLConnection

SNGConnection