Aspects at the crossroads of SE?!

Mario Südholt

Équipe Ascola (EMNantes-INRIA, LINA)

Keynote FOAL 2011, 21 March 2011
Where are we?

- Crosscutting as a fundamental problem of SE
- AOP has its place within SE:
  Integrated use of languages/frameworks/implementations
Where are we?

- Crosscutting as a fundamental problem of SE
- AOP has its place within SE:
  - Integrated use of languages/frameworks/implementations

What about the foundations of AOP?

- Formal methods in SE: large domain, uses generally rare but sometimes critical domains
- Do formal methods for aspects connect?
- (Real) uses of formal methods for AO?
Aspects at the crossroads

- **Is formal AO at the center of formal SE?**
  - Importance of the techniques/results?
  - Interest in the field?
- **Do we go/crawl/stumble in the right direction?**
  - Connect and apply to non-AO problems, methods, techniques
Where do we go?

- (Positive) Hypothesis:
  Foundations of AOP have come a long way . . .
  and go (slowly) towards use and application

- Some progression
  - From the specific (semantics for individual mechanisms),
  - via the general (integrated models),
  - to applications (property enforcement and analysis)
1. The specific

2. The general
   - Modules, components, events
   - Aspects and objects
   - Distributed aspects

3. The connected and the applied
   - Aspects and security
   - Aspect interfaces
   - Distributed events and patterns

4. The crossroads!?
1. The specific

- Language and weaving mechanisms
- Aspect categorizations
- Aspects for concurrent and distributed languages
1. The specific

- Language and weaving mechanisms
- Aspect categorizations
- Aspects for concurrent and distributed languages

Influential and inspirational, building blocks, but few uses as such
Language mechanisms and properties

- Semantics for specific AO constructs
- First semantics for subsets of AspectJ
  [Wand et al.: TOPLAS’04]
- Data flow: $\text{dflow}[x, x'](p)$ bypassing $[x](p)$
  [Masuhara, Kiczales: ASPLAS’03]
- Context-free tracecuts
  [Walker, Viggers: FSE’04]
Aspect categorizations

- Observers, assistants [Clifton, Leavens: FOAL’02]
- Augmentation, replacement . . . advice [Rinard et al. FSE’04]
  Definition in syntactic terms
- Spectative, regulative aspects [Katz, TAOS’06]
  Defined using temporal Logic
- Observers, confiners, aborters, weak intruders, selectors, regulators
  [Djoko Djoko, PEPM’08]
  Defines corresponding language classes that enforce properties
Concurrent and distributed applications

- Distributed AOP ≠ sequential AOP on distributed infrastructures
- Zoo of proposed language mechanisms: synchronization sets, operators for concurrent composition, remote pointcuts, (a)synchronous advice, distributed aspects with distributed state
- Proposed approaches focus on a small set of features
  - Encoding of sequential aspects in a CSP-like calculus [Andrews, Reflection’01]
  - Composition of superimpositions [Sihman and Katz, AOSD’02]
  - Composition of concurrent aspects [Douence et al., GPCE’06]
Outline

1. The specific

2. The general
   - Modules, components, events
   - Aspects and objects
   - Distributed aspects

3. The connected and the applied
   - Aspects and security
   - Aspect interfaces
   - Distributed events and patterns

4. The crossroads!?
2. The general

More general models or usage ("aspects for SE")

- Modules, components and events
- Aspects and objects
- Distributed aspects
Modules, components and events

- **Modules**
  - Trade-off invasiveness and strong encapsulation
  - Modular property verification

- **Components**
  - Aspects for black, gray and white boxes
  - AO over interaction protocols

- **Events**
  - Explicit vs. implicit announcement
  - Integration with event-based approaches in SE
Modules, components and events

- **Modules**
  - Trade-off invasiveness and strong encapsulation
  - Modular property verification

- **Components**
  - Aspects for black, gray and white boxes
  - AO over interaction protocols

- **Events**
  - Explicit vs. implicit announcement
  - Integration with event-based approaches in SE

Wide range of complementary models, clearly relevant to SE
Modular aspect definitions

Large variety of formal and semi-formal models

- Applicability conditions [Douence et al.: AOSD’04]: restrict aspect application by means of regular pointcuts
- Open modules [Aldrich: ECOOP’05]: advice only on external and exported calls
- Demeter interfaces [Skotiniotis et al.: ECOOP’06]: constraints on call graphs
- Aspect-aware interfaces [Kiczales, Mezini: ICSE’05]: full access but “external” pointcut specifications

Range from limited to farreaching invasiveness
Aspects and objects

- Integration (partially) obvious: use OO features if possible
  - Advice similar to method calls
  - (Some) pointcuts realized by advanced dispatch mechanisms
- Keep remaining features of AOP
Aspects and objects

- Integration (partially) obvious: use OO features if possible
  - Advice similar to method calls
  - (Some) pointcuts realized by advanced dispatch mechanisms
- Keep remaining features of AOP

- Few formal approaches
- What’s essential to AOP?
Principles
- Essentiality criterion: relevance to type safety
  - Many mechanisms, e.g. pointcuts, are not
- Enable reuse using standard OO features
- Support large space of pointcut and advice mechanisms
The A calculus: integration of AO features

- Closures to replace advice incl. `proceed`: enables reuse
  
  ```java
  class C { int m1(int i, int j) { return i+j; }}
  class D { void m2(int x, String s, int y) { System.out.println(x*y); }}
  class A {
      int m((int,int)->int proceed, int a, int b) { return proceed(a+1,b-1); }
      around1: execution(int C.m1(int a, int b)) { return m(proceed,a,b); }
      around2: execution(void D.m2(int a, String s, int b)) {
          m((int a, int b => proceed(a,s,b); return 0),a,b); }
  }
  ```

- Call/execution advice: static/dynamic closures
  - Type safety determines ordering of call/execution advice
The A calculus: support for mechanisms

- Rich pointcut languages through transformation and advice selection strategies
- Calculus parametrization support advice selection strategies
  - Ex.: flat login sessions

\[
[f]_{\_\text{login}} = \text{if } !f \text{ then } f = \text{true}
\]

\[
\text{getCAdvice}(f, \_, \_, \_v\text{.login}, \_\_) = \text{if } f \text{ then } \epsilon \text{ else } \bullet
\]

\[
\text{getCAdvice}(f, \_, \_, \_, \text{.login}, \_\_) = \bullet
\]

\[
\text{getEAdvice}(\_, \_, \_, \_, \_, \_) = \bullet
\]
A general basis for distributed aspects

Aspect Join Calculus [Tabareau, AOSD’10]
- Objects, Concurrency, Distribution
- Remote pointcuts, distributed advice and aspects, migration
- Accommodates features of many proposed languages

Ex.: cache replication

\[ \models \varphi \quad \text{aspect} \quad buffer\textit{Repl} = \]
- \( \text{intercept} \) : \( \text{rule}(buffer.(put(n) & empty())) \land \lnot \text{host}(\varphi) \)
- \{ obj \( b = buffer\textit{init} b.textit{empty}() \) in(\( b.put(n) \) & proceed\( n \)) \}

- Translation into the standard join calculus: correctness proof of weaving
Outline

1. The specific
2. The general
   - Modules, components, events
   - Aspects and objects
   - Distributed aspects
3. The connected and the applied
   - Aspects and security
   - Aspect interfaces
   - Distributed events and patterns
4. The crossroads!?
The connected and the applied

Connect and apply to non-AO problems

- Aspects and security
- Property-aware aspect interfaces
- Event-based aspects patterns for distribution
Aspects and security

- Security: paradigmatic crosscutting functionality
  - Formalization critical
  - Many different properties
    - High-level: authorization, authentication, confidentiality, ...
    - Low-level: information-flow, control-flow, ...

- Formal models needed for base program, aspect/aspect weaving and security properties
Ex.: aspects for secure service compositions

- Context: horizontal and vertical service compositions (choreography/orchestration and service implementation)

- Ex.: regulatory changes entail changes to both composition types (use case: SAP)
Secure service compositions: base, aspect models

- Base program
  - Collaboration model for choreography
  - $\pi$-based processes for vertical implementation
- Aspects: need to represent multiple features
  - Horizontal comp.: distribution features
  - Vertical comp.: sequential model
Secure service compositions: secure interactions

Security properties defined based on session types [Honda, Vasconcales et al.]

- Expressive model of interaction
  - Multiparty
  - Asynchronous and synchronous communication
  - Event-based interactions
  - Dynamic (multi)roles

- Global protocol for system understanding
- Projection: per-site protocols used for implementation and type-based verification
- Type safety, refinement and progress properties
Property-aware aspect interfaces

- Restrict aspects by properties on external and internal events
  - Structural conditions
  - History-based pointcuts
  - Data-flow or possibly even information flow
  - Other more expressive properties

- Generalization of existing approaches to aspects and modules
  - Flexible model of black box to (guarded) white box compositions
  - Corresponding notions of refinement?
Distributed events and patterns

- Relevant for numerous distributed applications
  - Service compositions, Cloud (virtualization, map-reduce) . . .
- Distributed event models are tricky
  - Complex event definitions
  - Grouping, scope and lifetime of events
  - Ordering causal relationships
  - Efficient implementation
Distributed aspects

- Many crosscutting uses of events
- Low-level definition in terms of event groups, scopes, casual relationships
- High-level abstraction: distribution, interaction patterns
- High-level properties?
  - Exclusion of race conditions in pattern compositions
  - Interactions between patterns that involve the same sites or even computations
Outline

1. The specific

2. The general
   - Modules, components, events
   - Aspects and objects
   - Distributed aspects

3. The connected and the applied
   - Aspects and security
   - Aspect interfaces
   - Distributed events and patterns

4. The crossroads!?
Initial questions revisited

- Formal AO at the center of SE problems?
  - Not yet! Close?
- Right direction?
  - Yes! More work on connection with and applications to other fields.
  - Pace of progress?
Conclusion

- More work on connection and application
- Important means: general models and properties
- But work on the foundations for aspects (only) is still worthwhile ... especially to look for holy grails (e.g., “The theory of crosscutting”) :)