

DSG

Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



PARMA

A Pervasive Application Rights Management Architecture based on ODRL

Dominik Dahlem
Ivana Dusparic
Jim Dowling

DSG

Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



PARMA

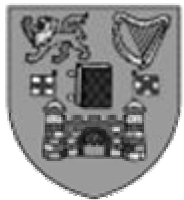
- Pervasive
 - Computing devices embedded in various objects, communicating with each other
 - Occasionally connected devices
 - Mobile phones
- Application rights management
 - traditionally software licensing
 - DRM concerned with content rather than applications



Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



Motivation

- Current DRM models for mobile applications not flexible enough
 - no pay-per-use, feature-based
 - differences in passive vs active content/apps
- Current desktop applications rights management not applicable to mobile environment
 - differences in fixed vs mobile environment, not always connected, cost of connection-use of free short range protocols



Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



Traditional App Rights Management

- Some Standards Exist
 - XSLM, No Adoption
- Rights Management API Calls Embedded in Source Code
 - Proprietary APIs
 - Calls Scattered in Different Files
- Inflexible Usage Models
 - Where are feature-based, pay-per-use models?
- Assumes Connection-Oriented Transport Layer
 - Session-Based
 - Mobility Issues Not Addressed
 - Connectivity - can't always establish session
 - Cost of Connection



Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



PARMA Design Decisions

- Use a Rights Expression Language
 - Instead of embedding Rights Management API calls in source code
 - Based on existing Standards - **ODRL**
- Translate REL into Rights Management Code
 - Centralise and Encapsulate Rights Management Code for Easier Maintenance and Modification
 - Use Aspect-Oriented Programming (AOP)
- No Assumption of 24/7 connectivity
 - Audit-based model
 - Connect when connection available and send usage data
 - Use different protocols – GPRS, Bluetooth



Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



Rights expression languages

- OMA DRM 1.0/2.0 currently used to specify rights on content/applications on mobile devices
 - too narrow to implement pay-per-use and feature-limited models
 - defines if given user has a right to run it or not at a given time on a given device
 - restrictions on time, user, device, count etc
- ODRL as a superset of OMA DRM 1.0
 - has more features PARMA requires
 - parties involved, integration with payment
 - still not enough for PARMA feature-limited model



Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



PARMA – ODRL extension

- RightsType – added as contextElement
 - defines type of rights on the application: node-locked, unlimited, pay-per-use real-time, pay-per-use audit-based, time limited, feature limited, named user
- ServerType
 - type (and URI in dLocation) of application rights server (EDS/VDS)
- disabledFeatures- constraint
- pointcut/package/class/method constraint
- auditLogData
 - for audit-based model (stored in RMS)
- userID
 - for named-user (phone number, registration ID)



Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



Additional PARMA validations

- Not all PARMA requirements are possible to validate against PARMA schema
 - Design decision to remain compatible with ODRL
- Additional validation after schema validation to make sure PARMA requirements are satisfied
 - RightsType is mandatory
 - Depending on RightsType certain constraints need to be present (NodeLocked needs IMEI constraint, feature-limited needs pointcut constraint)
 - dLocation and ServerType for licensing server mandatory (as contextElement)



Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



Mapping of REL into code

- AOP adds rights management code to the application without modifying original source code
- Original source code doesn't need to be available – compiled classes are enough
- Licensing code completely separate from application code
- Defines *pointcuts* – fully-qualified names of methods in the original code where rights management code is to be executed
- *Before, after, around pointcuts*



Distributed Systems Group

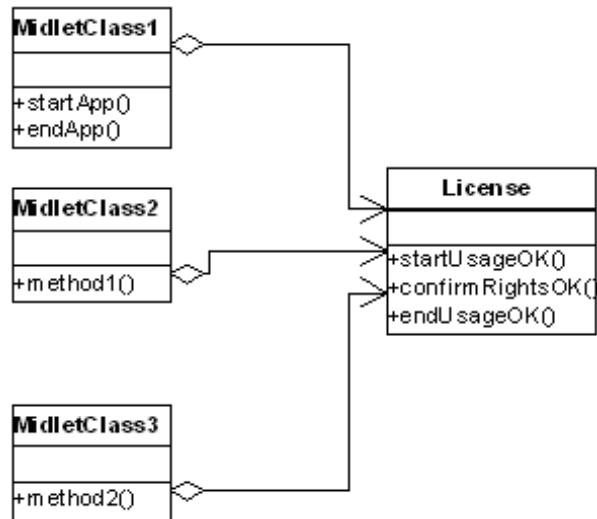
Department of Computer Science

Trinity College, Dublin



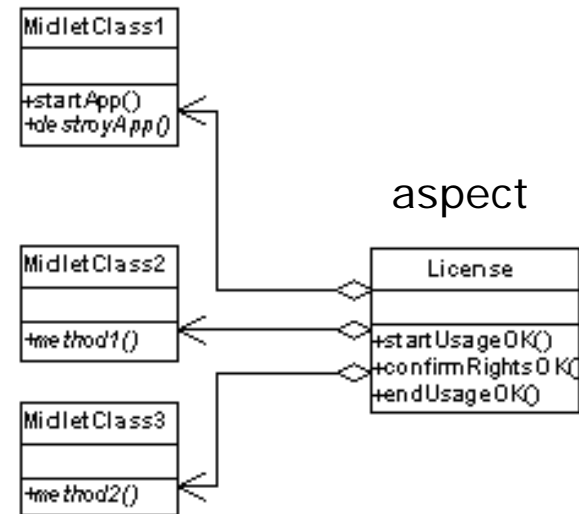
Mapping of REL into code

OOP



```
startApp(){  
  License.startUsageOK();  
}
```

AOP



```
startApp(){  
}
```



Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



Mapping of REL into code

```
= <o-ex:constraint>  
=   <parma:pointcut>  
=     <parma:adviceType>before</parma:adviceType>  
=     <parma:adviceType>after</parma:adviceType>  
=     <parma:package>  
=       <parma:packageName>mypackage</parma:packageName>  
=       <parma:class>  
=         <parma:className>mymname</parma:className>  
=         <parma:method>  
=           <parma:methodName>mymethod</parma:methodName>  
=           <parma:returnType>myReturnType</parma:returnType>  
=           <parma:argType>String</parma:argType>  
=           <parma:argType>int</parma:argType>  
=         </parma:method>  
=       </parma:class>  
=     </parma:package>  
=   </parma:pointcut>  
= </o-ex:constraint>
```

```
pointcut validateLicense (String a, int b) :  
  execution(myReturnType mypackage.mymname.mymethod(String, int))  
  && args(a, b);
```

- pointcut code passes context information to container that checks the permissions



Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



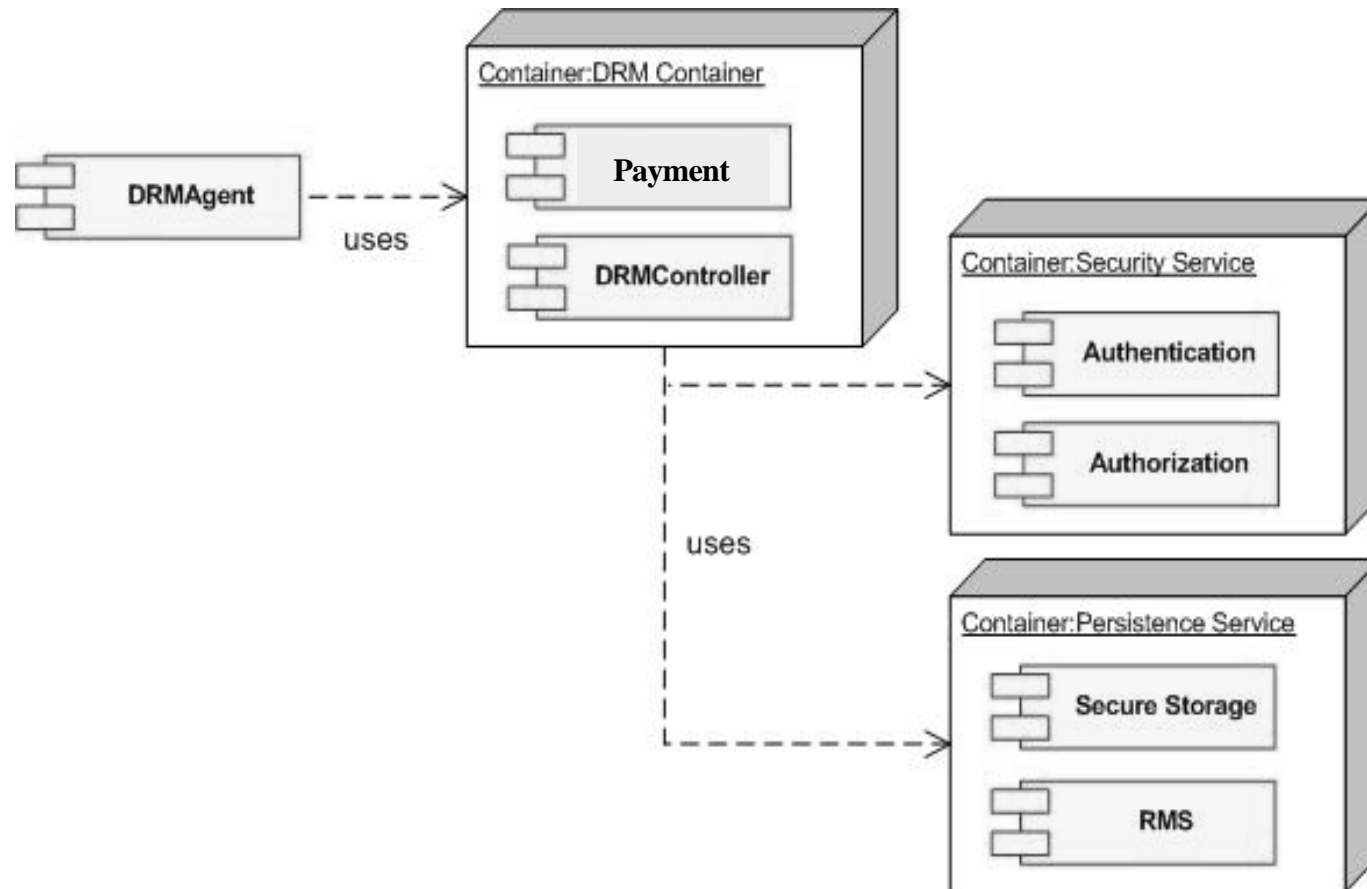
Enforcement Architecture

- The enforcement architecture is based on a lightweight container (J2ME/MIDP 2.0).
- The two most important patterns are Inversion of Control and Separation of Concerns.
- When the application starts up it initializes the EA based on the ODRL file.
- The container instantiates the services, e.g. authentication/authorization and payment and exposes them to the EA.



Enforcement Architecture

- Container Overview





Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



Server side

- Enterprise DRM server (EDS)
 - Hosted by the enterprise that obtained usage rights from vendor and distributes them within the enterprise
- Vendor DRM server (VDS)
 - Distributes applications and usage rights to individual customers
- WSDL interfaces
 - exposed to clients for registration and usage rights management



Distributed Systems Group

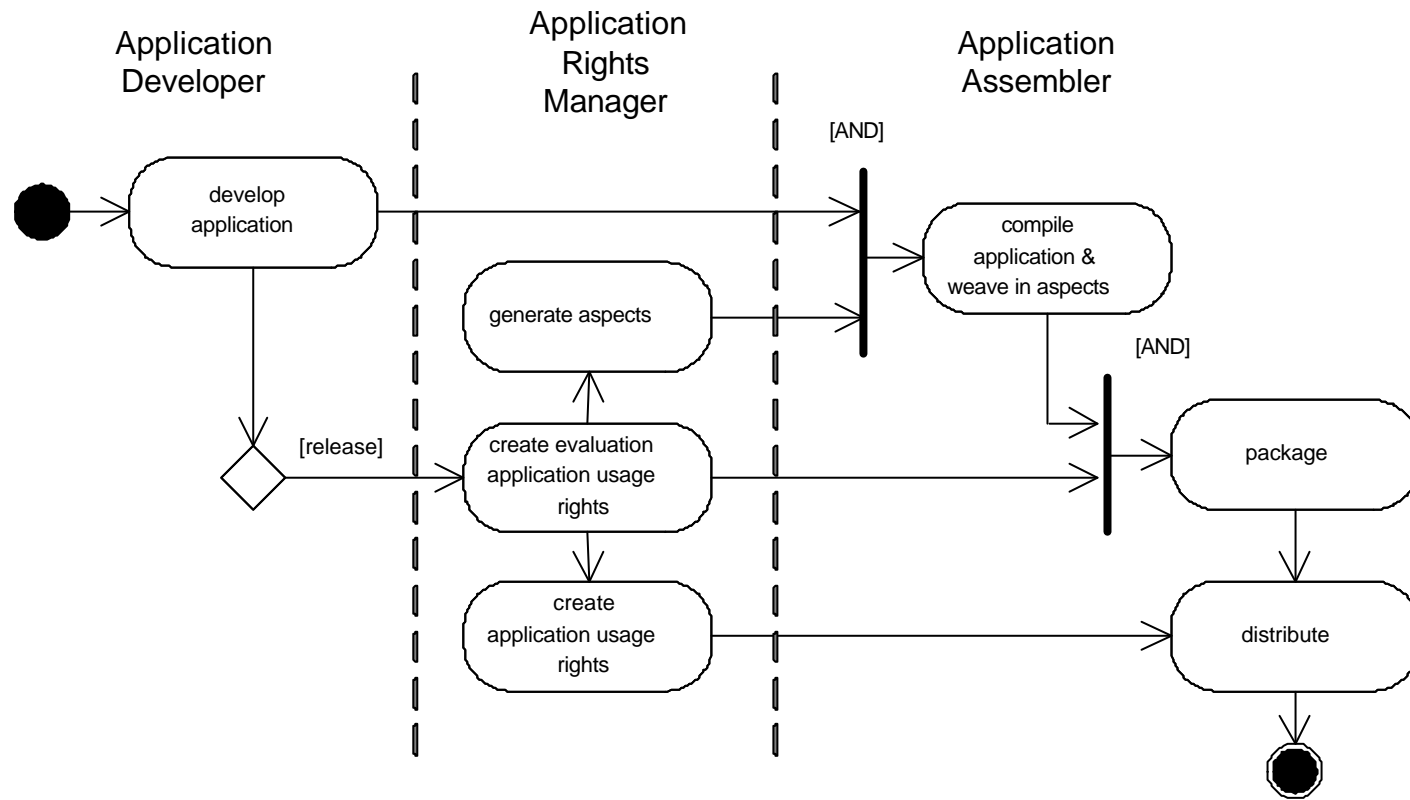
Department of Computer Science

Trinity College, Dublin



Summary

PARMA development lifecycle





Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



Conclusions

- Use of REL and AOP in application usage rights management
 - Reduces level of programming skill needed for addition of rights usage code
 - Reduces time for development and maintenance of rights usage code
 - Separates app code from usage rights code
 - Enables fine-grained usage rights models – feature-based
- Extension of ODRL to support fine-grained application rights usage models on mobile platforms
 - Pay-per-use (real-time and audit-based)
 - Feature-based
- Additional level of validation
- In current MIDP 2.0 version some rights usage code needs to be added manually – not all can be automatically generated from PARMA XML rights object

DSG

Distributed
Systems Group

Department of
Computer
Science

Trinity College,
Dublin



Questions?