2011 Imperial College Computing Student Workshop

Applying Algebraic Specifications on (Mobile) Digital Right Management Systems

Nikolaos Triantafyllou, Katerina Ksystra, Petros Stefaneas and Panayiotis Frangos

National Technical University of Athens

29-30th September 2011

Birth of Mobile DRM

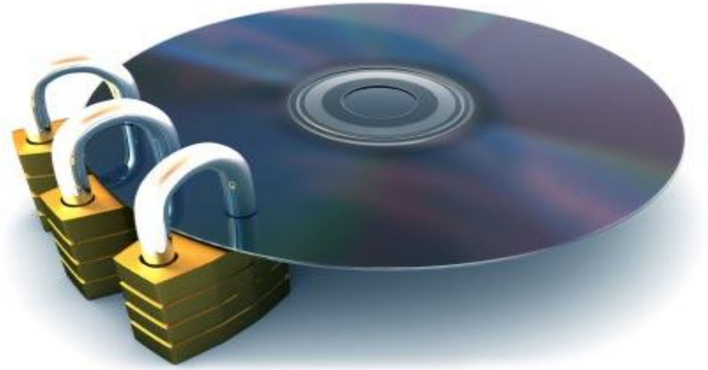
- Mobile phones have evolved
- No longer do they provide only voice services
- Technological advances like:
  - Wider color screens
  - More computational power
  - 3G networks
- Transformed mobile phones to mini personal computers

This lead to content available only to personal computers before to become available to the mobile end users

Birth of Mobile DRM

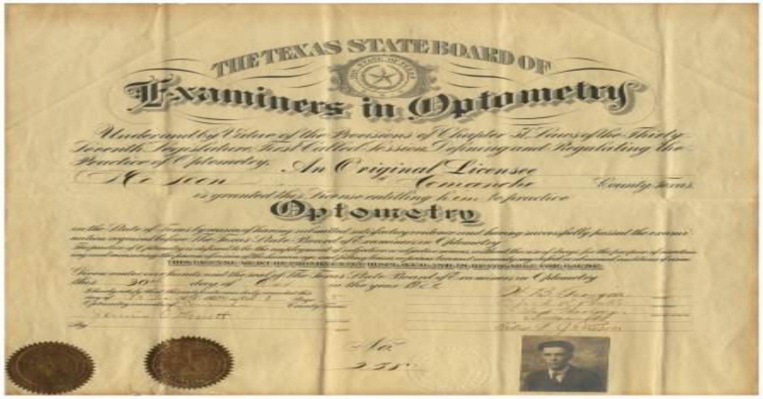
- The industry was afraid
- Piracy on internet
- Didn't want to have it duplicated over the mobile environment

- The solution came in the name of Mobile Digital Rights Management Systems (MDRMs)
- What are MDRMs really?
- More then a cryptographic protection of contents!!



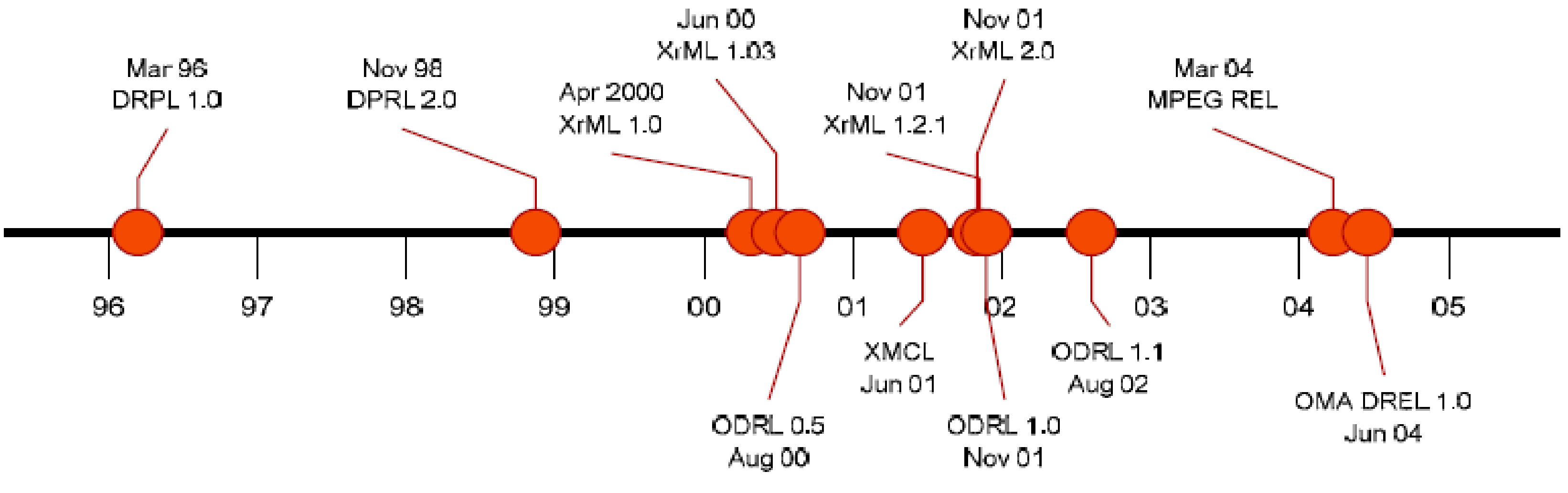
Birth of Mobile DRM

- A form of Digital Copyright
- With the use of Licenses they enforce the consumption of contents under a specific set of rules
- Giving birth to new commercial models and seller – buyer relations



DRM Languages

- On of the most important parts of a DRM system is the Language in which the Licenses are written in.
- Several have been proposed
  - Most are XML based
  - Some logically based



Open Mobile Alliance (OMA)

- An international organization with the purpose of creating standards for the mobile environment
- Members of it are most of the major mobile terminals producers and mobile services providers
- In 2001 they began the creation of a MDRM standard even before the birth of such a market
- In 2006 their second and more complete standard was presented.

Open Mobile Alliance (OMA)

- The standard is separated in three specifications:
  - OMA Rights Expression Language
  - On for the communication protocols and the basic architecture of the system
  - On that describes the required format of the contents

Need for verification

- Why do we need to have a verification for DRM systems?
  - DRM protected contents are a commodity
  - Their success depends on the acceptance of the market
  - Products advertised to behave in one manner and ending up behaving in another will lose fast the confidence of the consumers
  - Act in the best interest of the consumer as well as the creator.

Our Approach

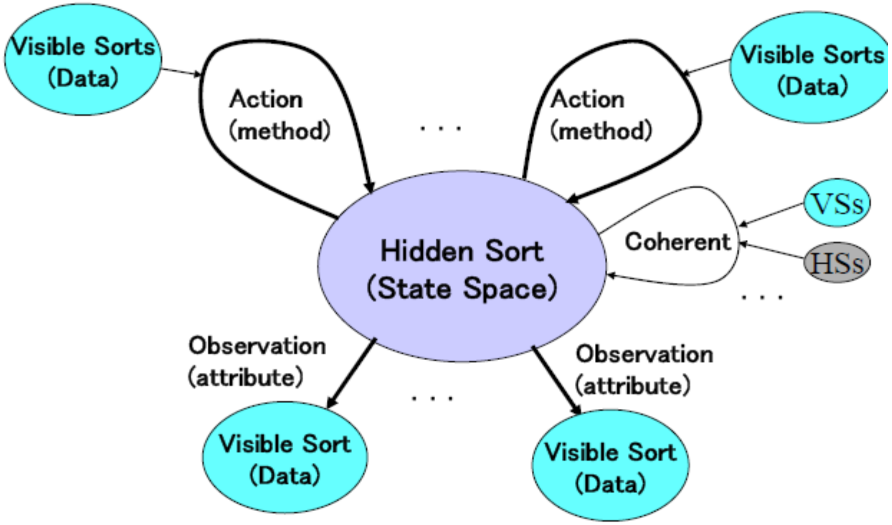
- Give OMA REL formal semantics
- Verify the algorithm involved in choosing what license to use
- Redesign the algorithm
- Verify the new algorithm
- Hints to addressing interoperability

Behavioral Specification

- We are interested in specifying the Behavior of a system
- With **initial algebras** we describe abstract data types while with **hidden,** the states of an abstract object
- There exist two kinds of data types:
  - Visible sorts
  - Hidden sorts
- There exist two kinds of operators for hidden sorts:
  - Action operators
  - Observation operators

Behavioral Specifications

- In general a behavioral specification looks like:



The OTS method

- *Observational transition systems, or OTSs are* mathematical models of (distributed) systems.
- A mathematical definition of the above concept
- A transition system written in terms of equations

The OTS method

- Assuming there exists a universal state space, say Y
- An OTS S is a triplet <O, I, T>;
  - O : A set of observation functions

    bop o : Sys V_{o1} ... -> V_o

    For two states s_1,s_2:Sys, s_1 is equal to s_2 wrt S iff
    $o(s_1, x_{o1}, \ldots) = o(s_2, x_{o1}, \ldots)$ for all $o \in O$, x_{o1}: V_{o1}, …
  - $I \subseteq Sys$
  - T : A set of transition functions.

    bop t : Sys V_{t1} ... -> Sys

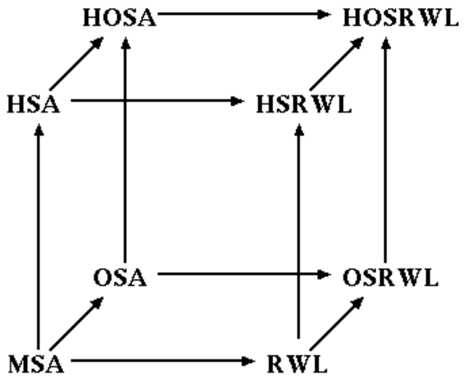
    Each t has a condition called the effective condition:

    bop c-t : Sys V_{t1} ... -> Bool

    If c-$t(s, y_{t1}, \ldots)$ does not hold, $t(s, y_{t1}, \ldots)$ is equal to s wrt S.

CafeOBJ

- CafeOBJ algebraic specification language
- writing a formal model and reasoning about the model
- Not a programming language, is executable however
- Developed by the Japan Advance Institute of Science
- A part of the OBJ family
  - Started by Joseph Goguen
  - Other similar languages OBJ3, Maude, etc.



OTS in CafeOBJ

- They transferred in a natural way
- The system states are defined by a hidden sort module
- Observers are denoted by observation operators
- Transitions by action operators
- We need to declare what the observers observe after each transition is applied on an arbitrary state
- What do they observe in the initial state

Abstract Syntax for OMA REL (need)

- DRM RELs on the run
- Cause lack of formal semantics
- Licenses used this moment may not implement what the creator intended

Abstract Syntax for OMA REL

- Our answer;
  - Create an abstract syntax for it
  - Transfer it into CafeOBJ so as e-validation on licenses can occur
- Example of syntax;

```
<o-ex:asset o-ex:id="Asset-1">
<o-ex:context>
<o-dd:uid>ContentID1</o-dd:uid>
</o-ex:context>
</o-ex:asset>
<o-ex:asset o-ex:id="Asset-2">
<o-ex:context>
<o-dd:uid>ContentID2</o-dd:uid>
</o-ex:context>
<o-ex:permission>
<o-ex:asset o-ex:idref="Asset-1"/>
<o-ex:asset o-ex:idref="Asset-2"/>
<o-dd:display/>
</o-ex:permission>
<o-ex:permission>
<o-ex:asset o-ex:idref="Asset-2"/>
<o-dd:print/>
</o-ex:permission>
```

$$
\begin{aligned}
agr := \ & \text{agreement} \\
& \text{about } \{\text{ContentID1 ,ContentID2}\} \\
& \text{with True} \longrightarrow \text{or[P1 ; P2 ; P3]} \\
& \textit{where} \\
& \text{P1} := \text{True} \xrightarrow{\text{ContentID1}} \textit{display} \\
& P2 := \text{True} \xrightarrow{\text{ContnentID2}} \textit{display} \\
& P3 := \text{True} \xrightarrow{\text{ContentID2}} \textit{pr}\,\text{int}
\end{aligned}
$$

Abstract Syntax for OMA REL (3)

- Translation to CafeOBJ notation

```
<o-ex:asset o-ex:id="Asset-1">
<o-ex:context>
<o-dd:uid>ContentID1</o-dd:uid>
</o-ex:context>
</o-ex:asset>
<o-ex:asset o-ex:id="Asset-2">
<o-ex:context>
<o-dd:uid>ContentID2</o-dd:uid>
</o-ex:context>
<o-ex:permission>
<o-ex:asset o-ex:idref="Asset-1"/>
<o-ex:asset o-ex:idref="Asset-2"/>
<o-dd:display/>
</o-ex:permission>
<o-ex:permission>
<o-ex:asset o-ex:idref="Asset-2"/>
<o-dd:print/>
</o-ex:permission>
```

eq aboutset = add (contentID2 , add (contentID1 , emuidset)) .

eq ps1 = add(True ==> contentID2 print , add(True ==> contentID2 display , add(True ==> contentID1 display , em-permset))).

eq TPS1 = add (True ~> ps1 , emtoppermset) .

eq agr1 = agreement-about (ebook) with (TPS1) .

- *Validation;*

eq permissionSET = add (F (agr1, aboutset, emreset).

red Permitted(print , ebook , contentID2) in permissionSET .

Proof Score Method

- Using a CafeOBJ/OTS specification
- Prove properties;
  - Invariant
  - liveness

Proof Score Method

- In order to prove such a property several steps need to be made :
  1. Express the property in a formal way as a predicate, say *invariant pred(p,**x**)*,
  2. In a module, usually called INV, *pred(p,**x**)* is expressed in CafeOBJ
  3. In a proof score we show that our predicate holds at any initial state, say *init*.
  4. We write a module, usually ISTEP, where the predicate to prove in each inductive case is expressed in CafeOBJ

Proof Score Method

5. For each transition we write the appropriate proof score
6. If istep(x) is reduced to true, it is shown that the transition preserves pred(p, x) in this case.
   - Otherwise, we may have to ***split the case***, may need some invariants that will be used as **lemmas** (*lemma discovery*),
   - or we may show that the predicate is not invariant to the system.

Order Rights Object Evaluation

- **Only rights** that are valid at the given time should be taken into consideration
- Rights that are **unconstrained** should be preferred over others
- Rights that contain a **date-time** constraint should be preferred over other constrained rights
- In the case where multiple date-time constraints are present the one with the nearest to the present, **<end>** tag should be preferred
- If no date-time constraint is present the **interval** constrained rights should be preferred over other constraint rights
- **Timed count** constraint rights should be preferred over **count** constraint rights

Unconstrained > Date time > Interval > Timed Count > Count

Verification of the Algorithm

- Following the above method the Specification for the Algorithm as an OTS in CafeOBJ was created
- The desired property to prove was :

Whenever a license is chosen for a given content, then the license is valid at that specific time.

Verification of the Algorithm

No.	Informal definition of Properties to be proven
1	Whenever a license is chosen for a given content, then the license is valid at that specific time.
2	*If a license L is the chosen license by the OMA Choice Algorithm for a given set S and that license exits, i.e. is not nil then L belongs to the set S.*
3	*If the choice made by the OMA choice algorithm for the set R union S, where R is an arbitrary license containing one usage right and S is a set of Licenses, is not R nor is it a choice made solely on S then the chosen license is nil, i.e. not valid license is available*
4	*If the set of licenses contains only a single license, say L and the choice made by the OMA Choice Algorithm is not nil, i.e. there exists a valid license, then the choice is this license L*
5	*If the choice made by OMA Choice Algorithm when the license set contains two licenses L and L' is not nil, and if the choice made is not that made based on the second license L' then the chosen license is L*

Verification of the Algorithm

- Using the above lemmas all transitions where reduced to true
- That concludes the verification for the initial property
- For the verification to be sound we need to show first that all the lemmas used are invariant as well of course
- Those verifications where created in a similar manner

A bug

- On the above algorithm consider we have the two license;
  - *License 1; the owner can listen to songs A or B* ***ten times***
  - *License 2; the owner can listen to songs A or C* ***one time before the end of the month***
- Request to listen to song A

  - Loose the ability to ever listen to song C!!!!!

New algorithm to solve the problem

- This bug can occur when;
  - *“A license contains more than one permission elements and after the execution it becomes depleted”*

- We redesign the algorithm by adding labels to license that state;
  - The License becomes depleted after the execution of a right
  - The License contains more than one permission elements
  - The characterizing constraint based on the OMA constraint ordering

The new algorithm

1. Check the licenses installed on the mobile DRM device for the ones matching the request of the user
2. *See if any of these licenses falls into the special case.*
3. If all the matching licenses fit into that category use the OMA Algorithm
4. If there exists a set of licenses that does not fall on this special category use the OMA Algorithm on them
5. Update the labels

Verification of the algorithm

- In order to prove that no loss occurs
- Introduce a coloring on permission
- Initially all permissions are white
- A permission gets colored black when;
  - If it is the users request
  - It is not the users request BUT it belongs to the ONLY license containing it, and that license gets depleted

Verification of the Algorithm

- Liveness property;
  - *If a right belongs to the installed licenses and is colored white* ***leads to*** *it being colored black.*
- Proof procedure different then invariant properties
  - In a module write the deduction rules for, *leads-to, ensures, unless*
  - Using those rules break it into *unless* and *ensure* predicates

Verification of the Algorithm

- Prove the ensure predicate (p ensure q)
  - Unless case
    - For all transitions (p(s) and ┐q(s)) → (p(s') or q(s')).
  - Eventual case
    - There exists a transition where; (p(s) and ┐q(s)) → q(s')
- Prove the unless predicates
    - Same as above

Verification

- Using the above the property;
  eq lto(S, P) = ((color(S,P) = white) /\ (P /in allowed(S))) |--> (color(S, P) = black) .
- Broken into two ensure properties

First ensure property

eq ens1(S , P) = (((makeReq(P) = useReq(S)) \/ (belong3?(makeReq(P) , find3(useReq(S) , best(S))) /\ (type3?(labelCP?(find3(useReq(S), best(S)))) = once) /\ (not(type3?(label?(find4(useReq(S) , best(S)))) = once)) /\((# build2(useReq(S) , licIns(S),license(S)) == 1) \/ (possLic(S) = emptyLic) \/ (finalLic(S) = emptyLic))) \/ (belong3?(makeReq(P) , skolem(P)) /\ (skolem(P) /inCP2 best(S) /\ ~(best(S) = emptyLic)) /\ (type3?(label?(find4(useReq(S) , best(S)))) = once) /\ ((# build2(useReq(S) , licIns(S), license(S))== 1) \/ (possLic(S) = emptyLic) \/ (finalLic(S) = emptyLic))) /\ (P /in allowed(S))) ensures (color(S , P) = black) .

Second ensure property

eq ens2(S ,P) = (((color(S , P) = white) /\ (P /in allowed(S)))) ensures (((makeReq(P) = useReq(S)) \/ (belong3?(makeReq(P) , find3(useReq(S) , best(S))) /\ (type3?(labelCP?(find3(useReq(S), best(S)))) = once) /\ (not(type3?(label?(find4(useReq(S) , best(S)))) = once)) /\ ((# build2(useReq(S) , licIns(S),license(S)) == 1) \/ (possLic(S) = emptyLic) \/ (finalLic(S) = emptyLic)))\/ (belong3?(makeReq(P) , skolem(P)) /\ (skolem(P) /inCP2 best(S) /\ ~(best(S) = emptyLic)) /\ (type3?(label?(find4(useReq(S) , best(S)))) = once) /\ ((# build2(useReq(S) , licIns(S), license(S))== 1) \/ (possLic(S) = emptyLic) \/ (finalLic(S) = emptyLic)))) /\ (P /in allowed(S))) .

Interoperability problems

- Outside the mobile environment;
  - Many different standards
  - Many different RELs
- Result
  - The do not work together;
    - You buy one license on your mobile phone
    - Cannot use it on your pc

Start of a project

- Using *Theory of Institutions*
  - tries to capture the essence of the concept of "logical system"
  - An Institution I is defined as;
    - A category Sign, the signature (names of sorts)
    - A functor that takes us from Sign to the category of Sets, and represents the sentences of our institution
    - A functor that takes us from Sign to the Cat^{op} the models of the institution
    - And a satisfaction relation such that for each signature morphism the satisfiability relation is preserved between the models and sentences

The idea

- Using the above we could
  - Define institutions for all RELs
  - And the translation would be automated from license to license (sentences) through institution morphism
  - While preserving the meaning of a license

  - We have begun this work by defining an institution for OMA REL

THANK YOU !

Questions??