Decision Making in DRM System Rights Exporting

Wenhui Lu

University of Tampere
Department of Computer Sciences
Computer Science
MSc thesis
June 2007

University of Tampere
Department of Computer Sciences
Wenhui Lu: Decision Making in DRM System Rights Exporting
MSc thesis, 64 pages
June 2007

Digital rights management (DRM) has been widely used in copyright protection. The nature of DRM leads to incompatibility between different DRM systems. Many authors have produced solutions for generating DRM interoperability. However, decisions making in rights exporting, identified as one of the most important aspects in DRM interoperability, still requires intensive research effort.

This thesis studies how the characteristics of rights influence decisions making in rights exporting, and how to mitigate their negative impacts. In order to study how rights can be transferred from one DRM system to another, we built a generic rights model to illustrate the basic elements within rights, i.e. permission and condition, and two types of linkage relationship between the basic elements, the valid-checking path and influential path. Furthermore, the direction and mode of rights flow have been analysed. On the basis of our rights model, we propose a process model for rights exporting and methods for rights decomposition and adaptation. These are further verified in a case study of interoperability between two major DRM standards: OMA DRM version 2 and Windows Media DRM version 10.

In summation, this study defines a generic rights model, upon which a process and methods for rights exporting between different DRM systems are proposed and verified in a case study. The rights model and the rights exporting methods can be further specialised for specific DRM systems. Together with the case study, this provides a good starting point for further study of the interoperability between DRM systems.

Key words and terms: DRM interoperability, rights exporting, decisions making, rights model.

Acknowledgements

Firstly, I would like to thank my advisor, Dr. Zheying Zhang, for her continuous support in my thesis work. She was always there to listen and give advice. Secondly, I would like to thank my supervisor at Nokia, Jukka Kainulainen, for his generous support from my office. I would have not been able to publish the thesis inside Nokia without his supportive help.

Special thanks go to my wife, Pan Pan, for her generous daily support throughout my thesis work. Most of the thesis was finished at weekends and during my spare time. Without her good understanding and encouragement, the thesis work would have been much more difficult to finish.

Table of Contents

1. Introduction 1
   1.1. Research questions 2
   1.2. Research methods 2
   1.3. Contributions and limitations 3
2. Digital Rights Management Systems 4
   2.1. Core entities 4
   2.2. Importance of DRM systems 6
   2.3. Rights Expression Language 7
3. DRM Interoperability 9
   3.1. Related research 10
   3.2. Our focus 12
4. Rights Modelling 13
   4.1. Key elements 13
      4.1.1. Permission 13
      4.1.2. Condition 14
      4.1.3. Linkage 14
      4.1.4. Right 16
   4.2. Right illustration 17
   4.3. Internal structure of right 18
      4.3.1. Multiple conditions 18
      4.3.2. Shared condition 19
   4.4. Right decomposition 19
5. Principles of Rights Exporting 22
   5.1. Modes of rights flow 22
   5.2. Directions of rights flow 23
   5.3. Results of rights flow 25
   5.4. Principles 26
6. Decisions Making in Rights Exporting 28
   6.1. Criteria 28
   6.2. Decisions in rights exporting 29
   6.3. Further equivalent decomposition 32
   6.4. Right adaptations 34
      6.4.1. Adaptation methods 34
      6.4.2. Deployment of rights adaptation 37
   6.5. Process of decision making 39

7. Case Study 41
7.1. Windows Media DRM 10 41
7.1.1. Terminology mapping 41
7.1.2. Right characteristics 43
7.2. OMA DRM 2 44
7.2.1. Terminology mapping 45
7.2.2. Right characteristics 46
7.3. Criteria of rights exporting 47
7.4. Analysis of the proposed process 50
8. Conclusions and Future Work 52
References 54
Appendix 1: Snapshot of XrML sample end user license chain 57
Appendix 2: VBScript example of issuing a license in a license chain 58
Appendix 3: Properties of a WMRMRights Object in WMDRM version 10 61
Appendix 4: Example rights expressed by ODRL 63
Appendix 5: Descriptions of permission and constraint in OMA DRM 2 64

1. Introduction

The demand for digital content services has been growing faster than expected. Various services have been implemented or scheduled on the digital content services domain, such as digital TV, e-newspaper subscriptions, real-time driving maps and gaming. Meanwhile, tremendous enhancements of data communication technologies allow service providers to distribute their content instantaneously to end users. Although these attractive services are technically feasible, there will not be more content providers until profits have been proven to be reachable and systematically secure. Moreover, end users are unlikely to pay for the service if they can access the contents freely and easily from the Internet. Well-defined business models are thus required to enable content marketing.

Digital Rights Management (DRM) [Liu et al., 2003], as the essential part of a business enabler, has been continuously evolving. It is a system to protect high-value digital assets and to control the distribution and usage of those digital. Many successful industry deployments are based on DRM technologies, such as Apple iTunes for music services and CinemaNow for video services. As companies implement their own DRM systems on the basis of a specific right's expression language, the DRM solutions differ from each other. Also, with the consideration of security, the implementation of DRM is strictly confidential for the company. The nature of DRM leads to interoperability problems between different DRM solutions. The fact that DRM systems do not interoperate with each other frustrates both content providers and end users. Content providers have to adapt their contents for different formats in order to fit into the business models based on different DRM systems. End users suffer from incompatibility problems for content usage between devices or software based on different DRM solutions. For example, an end user might purchase a music file for his mobile phone but he might not be able to play it on his PC or iPod, simply because these devices deploy different DRM systems. Incompatibility problems appear even on DRM systems with the same standard when new versions of the standard are released. In order to solve the problem, we either need a generic DRM standard to unify the DRM solutions, or we have to deal with incompatibility between various DRM solutions [Koenen et al., 2004]. A few mature and sophisticated DRM standards have been generated and well adopted by industry, such as Windows Media DRM [WMDRM_G, 2004], and Open Mobile Alliance (OMA) DRM version 2 [OMA_DRM_2, 2004]. However, the purpose of these is not to unify the DRM standards. DRM standardization is requiring a relatively long time to settle down. Therefore, the practical approach is to handle the system differences.

1.1. Research questions

The incompatibility problems between DRM systems have already received a lot of attention. How to ease the problems caused by the variations in DRM systems, identified as DRM interoperability issues [Koenen et al., 2004], becomes an interesting topic for general DRM solutions.

We have noticed the importance of DRM interoperability and reviewed the studies conducted by Koenen et al (2004), Schmidt et al (2004), and Safavi-Naini et al (2004). We discovered that within the domain many aspects still require more research efforts, which motivated us to further the discussion based on other authors' contributions. Accordingly, we have identified our research questions as follows:

- What are the main factors that affect a DRM system when exporting a right to another DRM system?
- How to mitigate the influences of these factors (answered by the first question)?

Answers to these questions define the problem domain and the solution domain of our research. By tackling the first question we analyse the general DRM system features that are involved in rights exporting, and identify the problem on which we centre our efforts. The second question is to provide the solution to the problem domain defined by the first question. We need to make a solution proposal and prove the feasibility of the proposal by deploying the proposal to the problem domain. Meanwhile, the research questions outline the logical steps of our study: raise the problem, analyse it, and solve it.

1.2. Research methods

In order to answer the research questions, we conducted a literature review and studied the relevant work on DRM, DRM interoperability, and rights exporting. Based on the output of the literature review, we established a rights model to assist the research. The model expresses rights in a visualised manner. Meanwhile, we outlined the boundary of our problem domain by clarifying the principles of rights exporting. Then, with the help of the established model, we identified the criteria for rights exporting and several methods that can be used to solve the issues we found in the problem domain. In order to utilise the methods in a logical manner, we also proposed a process of decision making in rights exporting. Lastly, we have deployed the proposed solution to a case study with two major DRM standards: Window Media DRM 10 and OMA DRM 2. The case study verified our proposed solution and provided valuable feedback. Based on the findings of case study, we conclude our study, list our contributions, and open discussions for further study.

1.3. Contributions and limitations

The main contribution of this thesis is the solution proposal for decision making upon rights exporting among distinctive DRM systems. The solution contains insights into rights exporting, a rights model, methods, and processes.

First, based on the characteristics of rights and the features of rights exporting, we identified many new aspects, for example, a valid-checking path and influential path, multiple conditions and shared conditions, rights decomposition, modes of rights flow, direction of rights flow, and results of rights flow. Second, we established a generic rights model to visualise the characteristics of rights, which is extensible and flexible when dealing with DRM interoperability issues in general. Third, we have identified methods for rights decomposition and rights adaptation to achieve better results from rights exporting. Fourth, we also proposed a process of decisions making in rights exporting. The process utilises our methods in a logical and efficient manner. Additionally, we have provided a case study based on real DRM specifications, which offers more practical findings for those who are interested in those DRM specifications.

Some aspects that are important for rights exporting are not discussed here. For example, how to secure the activities taking place during rights exporting between two DRM systems, and how to transform and store the right exported to another system. In this thesis, we have focused on the decisions making in rights exporting. Also, our proposed solution is generic to DRM systems. When dealing with specific DRM systems, adaptation to our solution might be necessary in order to achieve a more optimised performance.

2. Digital Rights Management Systems

As digital rights management (DRM) solutions are still in a state of development, no standard definition exists [Helberger et al., 2004]. The variety of DRM concepts is continuously broadening, which makes it difficult to gain a unified understanding of DRM systems. Therefore, before stepping into our focused area we set out our scope of DRM systems. We refer to the website of Europe's Information Society for a definition of generic DRM systems for the later discussion, "Digital Rights Management Systems are technologies that describe and identify digital content protected by intellectual property rights (IPR), and enforce usage rules set by right-holders or prescribed by law for digital content" [EEurope, 2005]. Digital content can be text, graphics, images, audio, video, software, or services in a digital format.

2.1. Core entities

There are three core entities in the concept of DRM systems: rights, content, and parties, as illustrated in Figure 1. The DRM system's purpose is to maintain the associations and manage the activities between them.

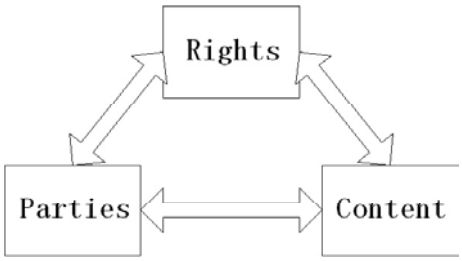

Figure 1 Core entities of DRM [Iannella, 2001-2]

The content is the proportion of a substance in digital format. Instances of content can include a song in MP3 format, a Java game for a smart phone, an online GPS driving map for a PDA, or an Excel spreadsheet containing confidential information. The content can be distributed through physical copies, like DVDs, or a digital network like the Internet. In order to protect the content, DRM systems usually encrypt the content.

The rights define the usage rules applied to the end user content. They are composed of permissions restricted by constraints, obligations, and any other agreements that affect the usage of the content for end users. A simple right can be, for example, to play an MP3 file 10 times, where the permission is to play and its associated constraint is 10 times. The end user uses rights to render content, while DRM systems grant rights to end users. The content can be rendered only if the constraints are satisfied for the intended permission in the rights. Therefore DRM systems check constraints before granting rights to end users. Besides the definition of content usage, the rights also contain (or point to) a content encryption key (CEK). The CEK can decrypt the encrypted content, thereby connecting a right with its corresponding content. Some

DRM systems use other terms for instances of right, for example “rights objects” in OMA DRM [OMA_REL_2, 2006] and “license” in Windows Media DRM [WMDRM_G, 2004].

The parties represent any involved entities that are associated with rights or content. They can be a content provider who originally created the content, a content issuer who packs and distributes the content, a rights issuer who generates and distributes rights for the content, an end user who purchases rights and renders the content via a DRM agent software on a target device, and a billing service provider who collects and confirms payments when an end user purchases rights.

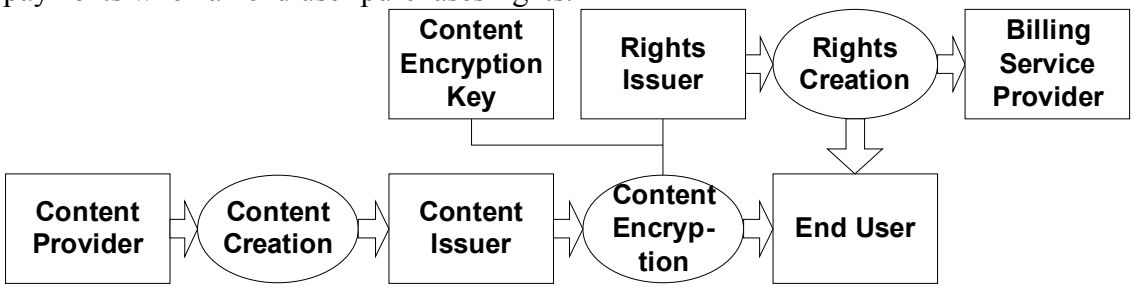

Figure 2 Functional architecture of DRM

Figure 2 illustrates how different parties are involved and interact in a DRM system. A content provider creates the original content and authorises both the content issuers and rights issuers to deliver the content and its associated rights. The content issuers encrypt the content with a CEK, while the rights issuers record the corresponding CEK and are able to create rights on demand. An end user can obtain the encrypted content from the content issuers, but he cannot render the content without rights as the content cannot be decrypted without a CEK. Therefore, content issuers can distribute the encrypted content as free content. If the end user wants to render the content, he must order the right from the rights issuers. The rights issuer will generate the right with a CEK according to the user’s query and deliver it to the end user. The billing service provider takes care of the payment based on the agreement between the rights issuer and the end user. Once a right arrives at the end user’s device, the DRM agent on the end user’s device must precisely enforce the usage rules defined by the right.

To demonstrate how different parties work in the DRM system, we use the following scenario: Jack London wants to buy a Michael Jackson (the content provider) song (the original content) as the ring tone for his smart phone (the target device with a DRM agent on it). He may use his phone to browse the web and find a web store (provided by the content issuer) that offers the song to him. He downloads the file (encrypted content) from the link on the web store and tries to open it. There is a note that appears on his phone, which tells Jack that he cannot open the song until he purchases rights for the ring tone. Jack may select the “purchase rights” item from the option’s menu (provided by the DRM agent) from the note on his phone. The phone will then launch the browser and open the webpage (provided by the rights issuer)

automatically for Jack to buy rights for the ring tone. Jack just wants to use the ring tone for a week so he chooses the rights that have the permission to play the ring tone for a week. During the procedure where Jack orders the rights, he has to offer the information from his credit card to a secure billing system (provided by the billing service provider). After confirmation of the transaction, Jack receives the rights for his new ring tone. Now he can open the ring tone and set it as the default ring tone for his smart phone. After one week, his rights have expired. Jack cannot open the ring tone anymore (enforced by the DRM agent on Jack's phone). He could choose to order new rights if he still wants to keep the ring tone for longer usage. This scenario provides a simple example of how DRM systems can be used by end users. Different DRM systems may offer diverse services to end users.

2.2. Importance of DRM systems

The definition of DRM systems implies two major tasks that DRM systems fulfil: rights expression and rights enforcement. As Edward Felten (2005) mentioned, "Copyright owners tend to see DRM as the last defence against rampant infringement and as an enabler of new business models". Rights enforcement is about protection of the digital content against illegal usage, whereas rights expression, the methods to express rights, enables new business models for content providers.

Renato Iannella (2001) noted that traditional rights management of physical materials benefited from the materials' physicality as this provided a barrier to unauthorised exploitation of content. The ease of distribution for digital content causes problems in the enforcement of copyrights. Books are an example of physical materials that are under the traditional rights management. To make a hardcopy of a book, a reader has to borrow it and copy it in a copy store. He has to spend a certain amount of time performing the copying task and the cost may even be higher than the price of the book. Meanwhile, the quality of the copy is definitely not as good as the original. Deficiencies derived from the features of the physical materials may hamper readers from breaking the law of copyright, and protect the copyright of the book. However, if the book is digitalised, the process of copying the book is different from the traditional one. First, the time and cost of copying decreases significantly. Thousands of copies can be produced within a minute. Second, because there is no information lost in the copy of the digital form, the quality of the copy can be exactly the same as the original one. Third, the format makes it easy to spread the content swiftly. Copies of the book can be widely distributed on the Internet through both authorised and unauthorised distribution channels. Consequently, piracy becomes a concern when security measures are not in place to protect the digital content. The user may spend more time on purchasing an original digital copy from an authorised web store than finding a pirated copy on the Internet. Because the digital contents can be easily copied and distributed, without any reduction in quality, content providers and distributors face serious problems in

protecting the rights over the digital contents. DRM systems provide content providers with a platform for secure distribution and access to digital content. They encrypt the original content with CEK. Without knowing the key, the content cannot be decrypted, which means end users cannot use the content. DRM systems ensure rights enforcement by restricting the CEK usage.

DRM systems not only provide content providers and distributors with an approach for protecting digital content, but also offer end users possibilities to purchase different usage rules for the content. In the traditional rights management of physical materials, end users usually purchase the content in the form of physical materials, which implies full rights upon the content. Rental services offer end users possibilities to pay for the usage of the content but only with limited options. We can take CD albums as an example of physical materials that are under the traditional rights management. In order to listen to a song, an end user needs to purchase a CD album that contains the song. After the purchase, he owns the CD and can play it without limitations. In fact, the total amount of usage may be only for a limited number of times but the end user has to pay the price for the usage of unlimited times. For digital content, the ownership of the content makes less sense to end users since the contents are not tangible but merely a piece of binary data. What end users want to purchase is explicitly the usage of content. DRM systems enable the business model for digital content by expressing usage rules with a machine-readable language, which is named Rights Expression Language (REL). Rights expression makes it possible for rights issuers to offer various rights to end users on demand. For example, some complex usage rules, like a right to play a song 10 times in 10 days, can be easily enforced by DRM systems. End users benefit from a low price since they just need to pay for the rights they want to consume.

2.3. Rights Expression Language

As previously mentioned, DRM systems express rights by using RELs. As RELs play an influential role in the DRM domain, RELs have shown signs of consolidation. Lots of efforts have been put into RELs and some major branches of standards have been built up. According to the historical review made by Schmidt et al (2004), the genealogy of RELs can be illustrated as in Figure 3.

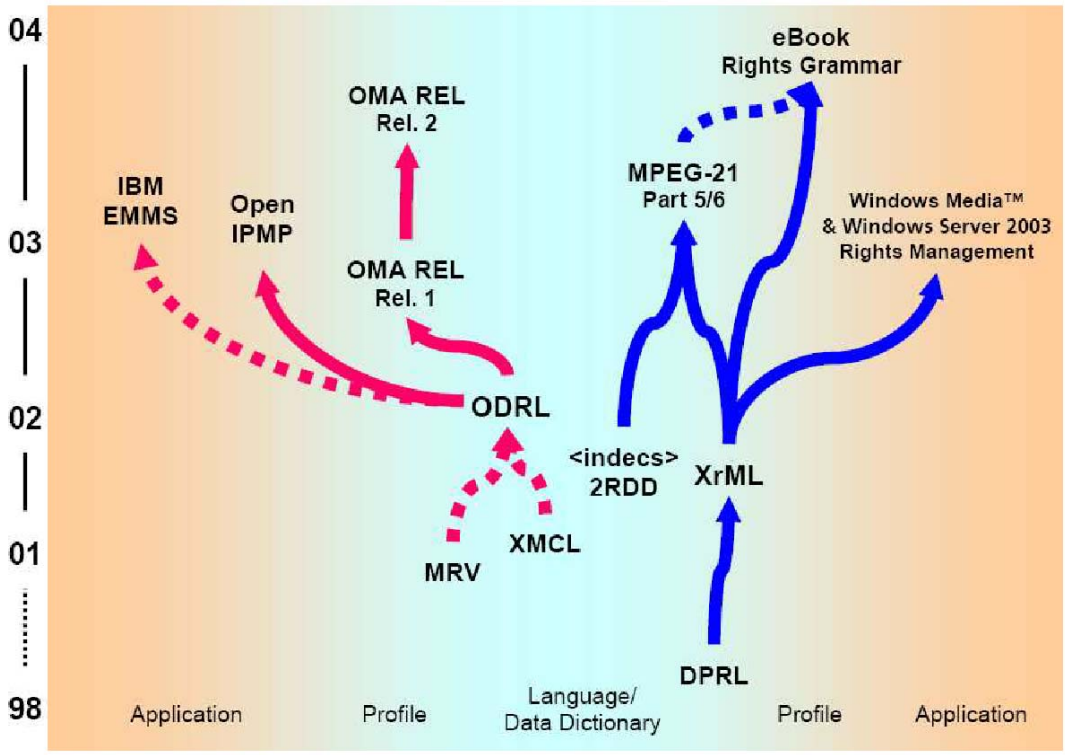

Figure 3 A genealogy of Rights Expression Languages [Schmidt et al., 2004]

In 1996, Digital Property Rights Language (DPRL) [DPRL, 1998] was first introduced by the Xerox Palo Alto research centre, which formed the roots of XrML [Wang et al., 2002]. It became XrML when its meta-language changed from lisp to XML in 1999. ContentGuard Inc. submitted XrML 2.0 in November 2001 to the Moving Picture Experts Group (MPEG) working group (ISO/IEC). XrML was then selected as the basis for MPEG-21 that aimed at an international standard. ContentGuard froze its releases of XrML at Release 2.0. Future releases and upgrades are issued by OASIS and MPEG consistent with the XrML architecture and design intent [XrML, 2001]. Compared with XrML, Open Digital Rights Language (ODRL) [ODRL, 2002] is an open standardization initiative aiming to develop a freely available REL for DRM. ODRL is influenced by RealNetworks' extensible media commerce language (XMCL), an earlier version of which merged with Nokia's Media Rights Voucher (MRV) to form the ODRL standards track in the fall of 2001. XMCL is still under independent development as an open standard [XMCL, 2001]. The Open Mobile Alliance (OMA), the leading industry forum for developing mobile service enablers, has chosen ODRL as a basis for the language profile defined in OMA DRM Releases 1 and 2 [OMA_REL_2, 2006].

RELs express rights in a machine-readable manner. If two DRM systems use the same REL, theoretically they are able to talk to each other. This is one of the reasons why RELs are regarded as the key to technical interoperability between proprietary DRM systems [Polo et al., 2004]. However, different RELs are not conforming to each other. Consolidation does not appear to be near completion, so we should not rely on the unification of REL to solve DRM interoperability in the near future.

3. DRM Interoperability

Copyright protects the interests of content providers, not the end users; and end users purchase the usage of the content, not the usage of the DRM system. Ideally, DRM systems should not disturb a user's legal usage of the protected content. However, in practice systems have their ranges of control. Once the usage crosses the border of a DRM system, the system is incapable of protecting the content. That is to say, the system can either deny access to the content, or interoperate with the outside. In order to grant legal access to the protected content outside of the domestic DRM system, the environment outside must also be a DRM system. Otherwise, once the domestic system passes the usage control out, the end user would obtain free access to the protected content, which breaks the usage rules defined in the purchased rights. For example, if a protected e-book can be saved as a plain text file, then the end user obtains free access to the content of the e-book. Therefore, the domestic DRM system is only allowed to interoperate with other DRM systems. Currently, there are several specifications for DRM systems that have already been adopted by the IT industry, for example Windows Media DRM [WMDRM_A, 2004], and OMA DRM [OMA_DRM_1, 2004]. The active deployment of DRM solutions exists within the on-line music industry such as Napster, Apple's iTunes, and Loudeye. For video downloads, Movielink, CinemaNow, and Starz welcomed their new competitors Amazon and Apple in 2006 [DRM-Enabled Content Services, 2007]. In general, DRM Systems based on different specifications are not mutually interoperable. The problem has been identified as an issue of DRM interoperability.

Although DRM interoperability has been addressed by many authors, there is still no common recognised definition for it. As Heileman and Jamkhedkar (2005) noted, the basic concept of DRM interoperability could be the ability of one DRM system to interact with another DRM system in order to implement some useful functionality. However, they also pointed out that the real issue is not interoperability *per se*, but rather the level of interoperability that allows better DRM solutions to be created. In this thesis, we only discuss the level of interoperability that allows the usage of protected content granted by one DRM system to be transferred to, and controlled by, another DRM system.

During the transfer of content usage between two DRM systems, there is always one system to export rights and one system to import rights. We name the system that exports rights the domestic system, and the system that imports rights the target system. Rights exporting can thus be defined as a set of activities to transfer content usage from the domestic system to the target system.

3.1. Related research

As various DRM systems are continuously emerging, the lack of DRM interoperability becomes increasingly problematic. Recently many authors have raised discussions against DRM interoperability in general.

Koenen et al., (2004), proposed three approaches to interoperability in DRM systems. They are:

- **full format interoperability**, which requires all participants (creators, distributors, manufactures, etc) to use the same data representation, encoding, protection scheme, trust management, key management, etc;
- **connected interoperability**, which translates content and rights from one DRM to another via online service offered by a third-party;
- **configuration-driven interoperability**, whereby tools can be downloaded to the importing device, and configured in real time in order to use protected content from other DRM systems.

Full format interoperability is regarded as the ideal approach. However, it is not practical in the short term as it requires consolidation among all existing standards and an effort to build a standard that covers all the requirements that come from various environments. Configuration-driven interoperability is especially suitable for solutions between two specific systems since it forces the target DRM systems to change by way of installing tools. When dealing with interoperability between many systems, this approach shows its limitation, in that installations are required on all the DRM systems concerned. Also, tool installation implies the potential risk of security compromise. The connected interoperability approach centralises the effort of interoperability among DRM systems and it is relatively secure as the service is provided remotely. However, it requires connection between the service providers of interoperability and the targeted DRM systems.

Schmidt et al., (2004) expanded the idea of connected interoperability and introduced the concept of intermediated DRM. Moreover, they also listed four general tasks for the intermediary DRM, as presented below:

- **content and rights reformatting,** which is regarded as the simplest possible task since the formats for both content and rights are usually well defined in DRM specifications;
- **data management**, which requires storage and access control for rights related information, and ensures the security and reliability of data flow;
- **condition evaluation**, which checks the feasibility of rights flow among different DRM systems;
- **dynamical state evaluation**, which evaluates rights on behalf of the end user's device.

These four tasks are generic to solutions of DRM interoperability. As Schmidt et al. (2004) mentioned, the reformatting of rights and content is a relatively simple task. Data management is not only a task for interoperability but also a mandatory task for rights purchasing on one DRM system. The technical aspects are well studied and mature compared with other tasks. Therefore, the condition evaluation and dynamical state evaluation are the core tasks for research on DRM interoperability.

Safavi-Naini et al., (2004) concreted two of these tasks, content and rights reformatting and condition evaluation, on the architectural level. They categorise the possible results of rights translation into:

- **isomorphism,** meaning that the translation of rights from one DRM system to another is exactly equivalent;
- **adaptation,** meaning that rights from one DRM system need to be modified in order to suit the requirement of another DRM system;
- **untranslatable expressions,** whereby some expressions in rights cannot be translated from one DRM system to another.

Isomorphism is the ideal situation for rights exporting since the end user will not lose any part of the original rights in the target DRM system. Adaptation indicates that the target system re-interprets rights in a different manner compared with the rights interpretation on the domestic DRM system, which usually implies that rights on the target system are more restrictive than on the domestic system, since granting extra benefit on the target DRM system for free is unlikely to be accepted by the domestic DRM system. Untranslatable expressions will not be exported to the target DRM system. Therefore, the end user cannot use this kind of rights on the target system at all.

Although Safavi-Naini et al., (2004) categorised the possible results, they regarded the results as the static attributes of rights that need to be exported. Moreover, they did not define the process to achieve the results.

Safavi-Naini et al. also presented three methods for adaptation:

- **reduction**, which deletes the expression that cannot be expressed in the target DRM system;
- **pre-enforcement**, which enforces the constraints that cannot be checked in the target DRM system, and removes the constraints before exporting rights;
- **storage**, whereby a third party stores expressions that cannot be expressed in the target system for possible retrieval.

The methods are well categorised and can cover most of the adaptation of rights. However, since there is no clear process defined for rights adaptation, how to use the methods, and with what principles, has not been fully instructed.

Besides the research presented above, other authors have studied DRM interoperability from different perspectives. For example, Heileman and Jamkhedkar (2005) proposed to utilise a layered framework to mitigate the huge amount of effort to realise DRM interoperability. Polo et al. (2004) analysed the possibility of translating rights based on different RELs (MPEG-21 REL and ODRL).

3.2. Our focus

According to previous studies made by other authors, full format interoperability was regarded as an approach unlikely to be achieved across the broader DRM market [Heileman and Jamkhedkar, 2005]. If we accept the differences between DRM systems, how to deal with the distinctions among DRM systems is the key question we need to answer for DRM interoperability. In the list of tasks proposed by Schmidt et al., (2004) we find that the condition evaluation is to handle the systems' differences. Although Safavi-Naini et al., (2004) have proposed some strategies for condition evaluation, we notice that many aspects still need to be discussed further. Since the task is making decisions on whether or not rights can be exported to a target DRM system, we regard the task as decisions making in rights exporting. According to our concerns and the nature of rights, we define a model to illustrate rights and emphasise our focus.

4. Rights Modelling

In DRM systems, rights are composed of permissions and constraints. Permissions define the type of usage allowed for content under protection. Constraints include obligations, conditions, or any other agreements that affect the usage of the content for the end users. RELs express rights in an XML format. However, different RELs are not compatible with each other. In order to study how rights can be exported from one DRM system to another, we specify the rights structure in a generic model.

4.1. Key elements

All kinds of permissions, constraints, obligations and any other agreements can be regarded as conditions that need to be satisfied in order to execute the corresponding permissions. Therefore, rights can be defined as permissions restricted by conditions. The key elements include right, permission, condition, and the linkage between the permission and condition. They form a generic model to represent rights. Each element represents one abstract concept. An instance of a concept represents the instantiation of the concept under the rights model. For example, an instance of right represents a specific right with corresponding instances of permission and condition.

4.1.1. Permission

Permission is defined as an abstract data class that contains all the information about the approach an end user could use to render protected content.

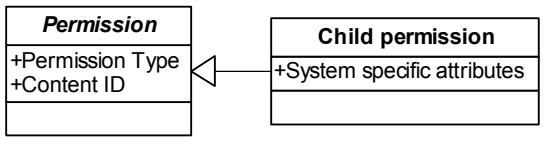

Figure 4 Class diagram of permission

Permission contains different attributes in different DRM systems. The common ones, permission type and content ID, are defined in the abstract class as shown in Figure 4. The permission type classifies the instance of permission, such as to play, to view or to print. The content ID is the identifier of the content for which the instance of permission is granted. DRM systems could have their own child permission class inherited from the abstract class. The child permission class could have system specific attributes.

In DRM systems, there might be many instances of permission with the same permission type and the same content ID. However, only one instance is used at a moment when an end user is solitarily rendering content with the same content ID. DRM systems have their own mechanisms to select which instance of permission should be

used first. The logic behind the mechanisms differs in different systems and do not concern us.

4.1.2. Condition

Condition is an abstract data class containing all the information about restrictions the permission must obey in order to render the content. DRM systems should be able to judge if the condition is true or false from the attributes of condition. As illustrated in Figure 5, common attributes of condition include condition type and condition detail.

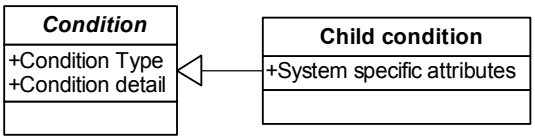

Figure 5 Class diagram of condition

The purpose of condition type is to classify the instance of condition, for example, the count, end time, or a specified period. Condition types can be separated into stateful conditions and stateless conditions. A stateful condition means the condition has a dynamic internal state to determine its validity. The internal state is changed based on the end user's action. For example, a count based condition is a stateful condition. Its internal state is the number of the count. When the instance of permission that links to the condition is granted once, the number of the count is decreased by 1. When the number of the count becomes zero, the condition becomes false from the DRM system's point of view. On the contrary, a stateless condition means the condition has no dynamic internal state, which can be changed by end user's action, to determine its validity. For example, an end time-based condition is a stateless condition. The validity of an end time-based condition is determined only by the time after which the end user's action should not have an influence.

Condition detail describes the instance of condition in details according to the condition type. For example, a number for a count based condition and a specific time for a time based condition belong to condition detail. Condition type determines the format of condition detail.

System specific attributes are not usually compatible in different DRM systems, which may affect the decisions making in rights exporting. However, we could reflect the difference by using a different condition type. Therefore, we do not discuss system specific attributes for condition and leave it for future work.

4.1.3. Linkage

The linkage between permission and condition can be expressed by two kinds of virtual paths: a validity-checking path and an influential path.

A validity-checking path means that the linkage is the virtual path which DRM systems follow when checking the validities of instances of condition in order to grant an

instance of permission. Before a DRM system grants an instance of permission to the end user, it checks the validity of all the instances of condition linked to the instance of permission. Only if all the instances of condition are satisfied can the DRM system then grant the permission. The linkages between permission and condition lead the DRM system on a virtual path to check the validities. Valid-checking paths represent how condition restricts permission.

Influential paths only exist between permission and a stateful condition. It is the virtual path that DRM systems follow when updating the internal state of a stateful condition. If the instance of permission links to some instances of a stateful condition, the internal states for all instances of the stateful condition should be updated by the DRM systems. The linkages show the virtual path for the DRM systems. Influential paths represent how permission influences condition.

The two kinds of path, together, can represent the interaction between permission and condition. However, a linkage does not necessarily have two paths. If the instance of condition is a stateless condition, only the validity-checking path exists in the linkage between the instance of condition and instance of permission. If the instance of condition is a stateful condition, there are three possibilities for a linkage between an instance of permission and the instance of condition: validity-checking path only, influential path only, and both paths. We use an example to explain the differences between the four types. Suppose an end user has an instance of permission to play a game and an instance of condition “10 times”. If the linkage between the permission and condition is a validity-checking path, then the count will not decrease no matter how many times the end user plays the game. However, if the count is decreased to zero by other permission, the end user cannot play the game anymore. If the linkage is an influential path only, then the count will decrease by 1 if the end user plays the game once. However, if the count is decreased to zero, then the end user can still play the game. If the linkage is both paths, then the count will decrease when the end user plays once. When the count is decreased to zero, the end user cannot play anymore.

In most of the usage cases described by the DRM standards [WMDRM_SDK, 2004], [OMA_DRM_2, 2006], both paths exist for linkages between a stateful condition and permission. Validity-checking path only and influential path only appear in a few cases. A validity-checking only case can be used to realise the following subscription use case. An end user is allowed to perform certain actions when the subscription is still valid. For example, if Jack has subscribed to an e-news service for 10 times. As long as the subscription is still valid, he is allowed to either print or view all the news items. In this case, there is a validity-checking path between permission to print and the condition 10 times. An influential path only case can be used to realise this metering use case. The metering mechanism allows a DRM system to count the number of times that protected content is used. For example, a service provider wants to know how many times a music

file is played by an end user. The linkage between the instance of condition “times” and permission to play the music file is an influential path. Also, a bonus system can be implanted based on the metering usage case. If the usage of a protected content exceeds a certain amount, then the end user is granted the right to consume other content as a bonus.

The differences between the influential path and validity-checking path are not mentioned in the specifications of most RELs. Since only a few cases need to distinguish the differences between them, they are handled as system specific cases in DRM standards. Therefore, we leave the discussion of the differences for future work. In this paper, we only discuss the case whereby the linkage between condition and permission is both paths. However, we leave room in our model to allow for future extension.

Figure 6 Class diagram of linkage

As illustrated in Figure 6, common attributes for linkage include an instance of permission, an instance of a condition, and a type of linkage. The type of linkage is to allow for future extension of other types of linkage.

An instance of permission could link to more than one instance of condition, while an instance of condition could also link to more than one instance of permission. However, if there is no instance of condition linked to the instance of permission, it means that the instance of permission is granted without any restriction from the DRM system. On the contrary, an instance of condition must link to at least one instance of permission. An independent instance of condition is meaningless and should not be presented in the instance of right.

4.1.4. Right

Right is an abstract class that packs instances of permission and instances of condition, and stores the linkages between them.

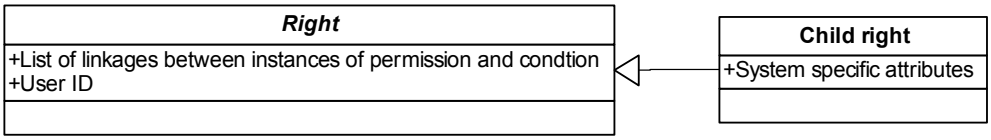

Figure 7 Class diagram of right

As illustrated in Figure 7, common attributes for rights are a list of linkages between instances of permission and condition, and user ID. User ID is used to identify the end user to whom the instance of right is granted.

4.2. Right illustration

Right can be expressed based on the concepts of the key elements. Moreover we have developed an illustration template in order to benefit our later discussions and to ease the effort of illustration.

The template of right is illustrated in Figure 8. A rectangle represents an instance of permission, a diamond represents an instance of condition, and a line with an arrow pointing from a rectangle to a diamond represents the linkage between an instance of permission and an instance condition. Figure 8 demonstrates an instance of right, which consists of one instance of permission (Permission 1) and one instance of condition (Condition 1). Permission 1 is restricted by Condition 1, which means that only if Condition 1 is satisfied can Permission 1 can be granted to the end user who owns the instance of right.

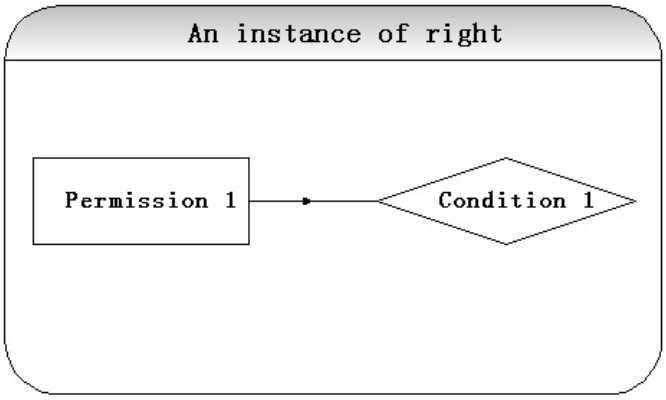

Figure 8 An instance of right

Illustrated in Figure 9 is an instance of right for Jack that grants him permission to play a file of music 3 times.

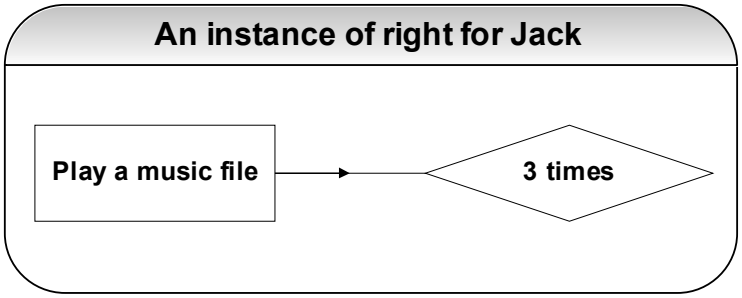

Figure 9 An example of an instance of right

4.3. Internal structure of right

Right consists of permission and condition. The linkages between permission and condition determine the internal structure of right. We recognise and categorise the basic structure elements from the point of view of permission, and name them for later discussions.

4.3.1. Multiple conditions

If an instance of permission links to more than one instance of condition, then we say that the instance of permission has multiple conditions.

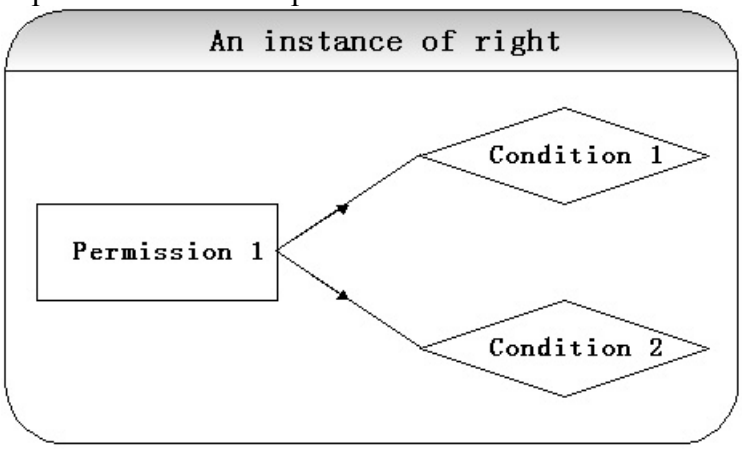

Figure 10 Multiple conditions

Figure 10 shows Permission 1 linked to both Condition 1 and Condition 2, which indicates that Permission 1 has multiple conditions. Permission 1 can be granted only if both Condition 1 and Condition 2 are satisfied. Once Permission 1 is granted, the DRM system needs to check if there is an internal state to update in both instances of condition. Therefore, Condition 1 and Condition 2 are linked to each other indirectly by Permission 1. Figure 11 illustrates an example of multiple conditions. It is an instance of right for Jack that grants him permission to play a movie file 10 times within a month.

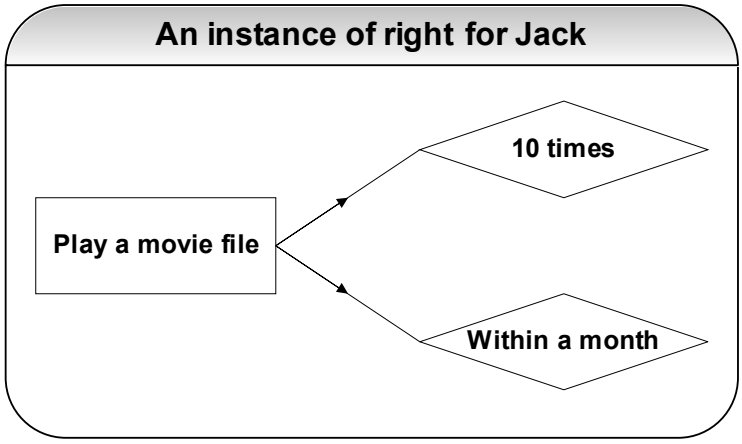

Figure 11 An example of multiple conditions

4.3.2. Shared condition

If more than one instance of permission links to the same instance of condition, we say that all of the instances of permission have a shared condition.

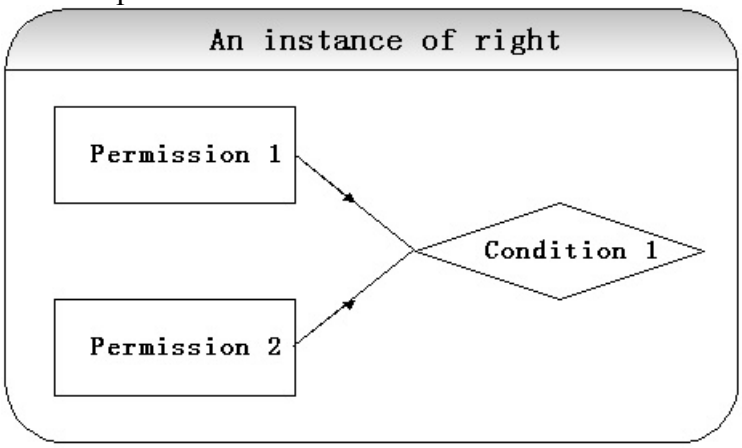

Figure 12 Shared condition

In Figure 12, both Permission 1 and Permission 2 link to Condition 1, which indicates that both instances of permission have a shared condition. If Condition 1 is not satisfied, neither Permission 1 nor Permission 2 can be granted by the DRM system. Once either of the instances of permission is granted, the DRM system needs to check if the condition needs to update its internal state. Therefore, Permission 1 and Permission 2 are linked to each other indirectly by Condition 1. Figure 13 illustrates an example of a shared condition. It is an instance of right for Jack that grants him permission to either view or print an image file 10 times. Jack can allocate the amount of times to view or to print the image as long as the total amount of times for both actions is not more than 10 times. In this case, permission to view and permission to print share the condition 10 times.

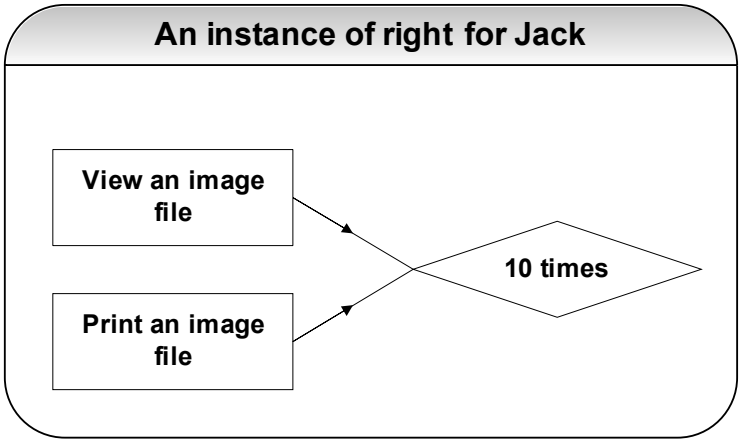

Figure 13 An example of shared condition

4.4. Right decomposition

Linkage associates permission with condition and the internal structure represents complex relationships among instances of permission and condition. However,

components inside an instance of right do not always associate with each other either directly or indirectly, which means that the instance of right can be decomposed into two instance of right without any difference from the right's point of view. We call it rights decomposition.

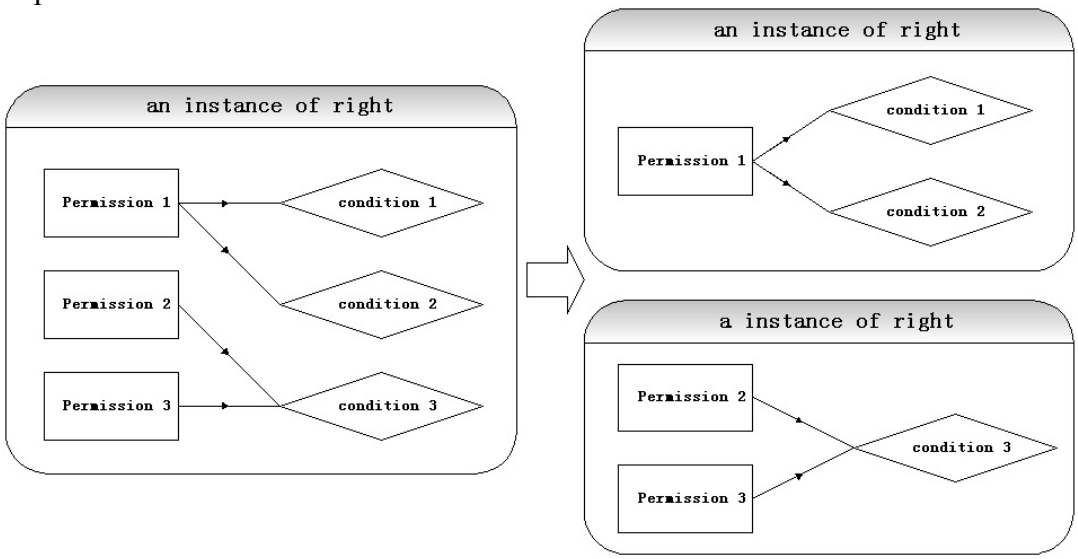

Figure 14 Rights decomposition

Figure 14 illustrates an example of rights decomposition. There is an instance of right that contains three instances of permission and three instances of condition. Permission 1 has multiple conditions: Condition 1 and Condition 2. Permission 2 and Permission 3 have a shared condition: Condition 3. In this case, Permission 2 and Permission 3 are linked to each other indirectly by Condition 3, while Condition 1 and Condition 2 are linked to each other indirectly by Permission 1. So in this case, there are two groups of components in the instance of right. Two groups are not linked to each other either directly or indirectly, which can be decomposed into two instances of right, as shown in Figure 14. After decomposition, other than the list of linkages, both instances of right should have the same attributes that the original instance of right has. There is no change of attributes in instances of permission and condition.

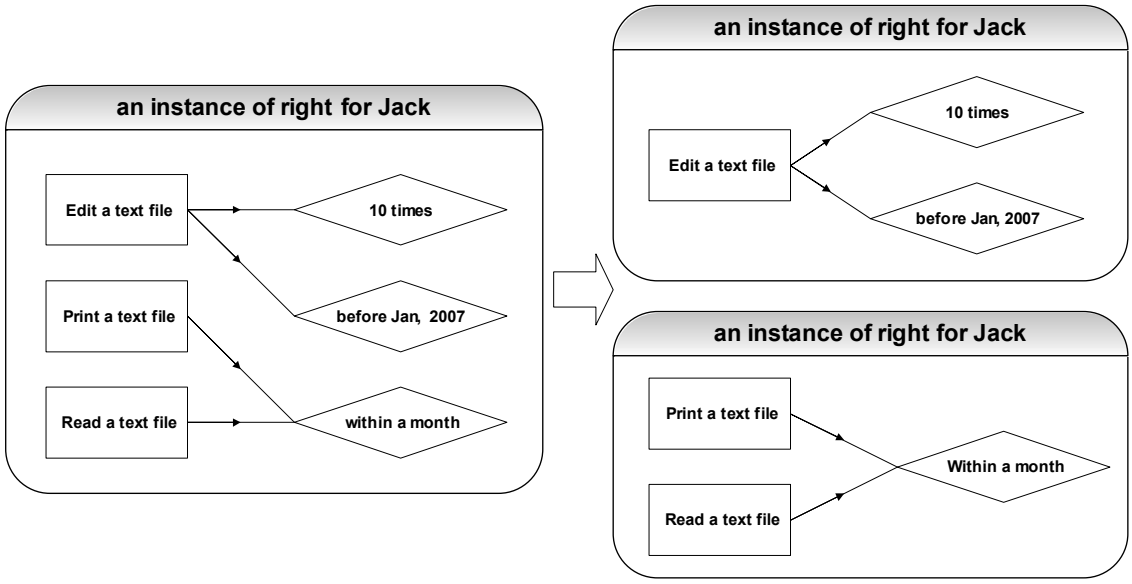

Figure 15 An example of rights decomposition

Figure 15 illustrates an example of rights decomposition. It is an instance of right for Jack that grants him permission to edit a text file 10 times before January 2007, and to print and to read the file within a month. The instance can be decomposed into two instances of right for Jack. One instance allows him to edit the file 10 times before January 2007, while another instance allows him to print and to read the file within a month.

5. Principles of Rights Exporting

Our focus is on the decisions making in rights exporting. However, before delving into decisions making, we need to clarify the principles of rights exporting. The principles are the requirements we need to meet in decisions making. Rights exporting results in rights flow from the domestic system to the target system. The following factors of rights flow have influences on the principles of rights exporting: the modes of rights flow, the directions of rights flow, and the results of rights flow.

5.1. Modes of rights flow

OMA DRM 2 has defined two modes for rights exporting: copy mode and move mode [OMA_REL_2, 2006]. In copy mode, the rights to export stay unchanged on the domestic system after rights exporting. In move mode, the rights to export will be removed from the domestic system after rights exporting. We have extended the discussion in order to cover all the possible rights flow. Our study results in three modes of rights flow as illustrated in Figure 16:

- Copy mode, which retains the original instance of right on the domestic system and makes a copied instance of right on the target system;
- Move mode, which makes a copied instance of right on the target system and removes the original instance of right from the domestic system;
- Share mode, which retains the original instance of right on the domestic system and shares the access to the instance of right with the target system.

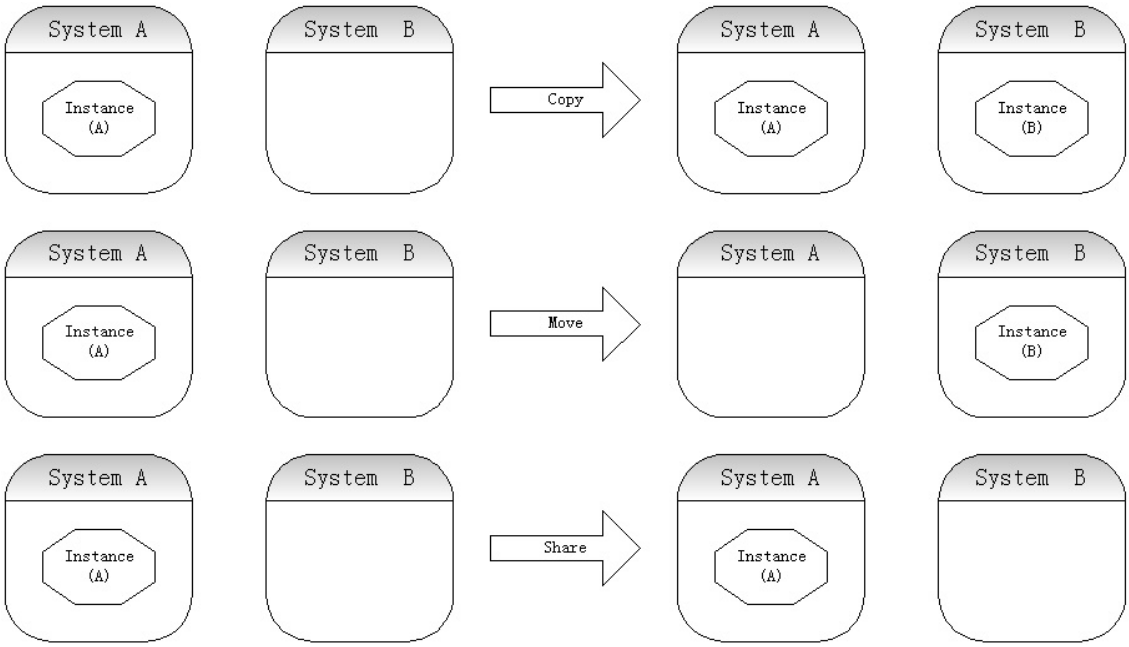

Figure 16 Types of rights flow

Copy mode and move mode do not require continuous data transfer between the concerned systems. Once the transaction is done, the copied instance of right is valid for the target system to use. On the contrary, share mode needs to maintain the connection between the concerned systems in order to grant rights to the target system since the instance of right is still under the control of the domestic system.

Different modes meet distinctive needs. Copy mode allows an end user to use an instance of right on different systems at the same time. However, it is only suitable for rights with a stateless condition. When an instance of right with a stateful condition is exported from system A to system B with copy mode, the internal state in the instance of right on system A can not be updated according to the usage on system B. That is to say, the end user receives two instances of right with the same stateful condition, and the usage right is expanded. It implies that the end user may retrieve the right freely by exporting the instance of right between system A and system B. Move mode allows an end user to use an instance of right on different systems but the right can be assigned on only one system each time. That is to say when an end user exports an instance of right from system A to system B, if he later wants to render the content on system A, the end user has to export the right from system B back to system A, or purchase another instance of right for system A, which obviously causes inconvenience to the end user. Share mode allows an end user to use an instance of right on different systems at the same time, and it supports all kinds of rights. However, as we mentioned earlier, share mode requires continuous data transfer among concerned systems. Once the connection is interrupted, only the domestic system on which the instance of right is located can use it and the others cannot share the access anymore. The mode heavily relies on the connection, so systems on portable devices may not be qualified to adopt it due to their lack of connection capacity. Selection of a mode is a not only a technical issue, but also a decision with respect to business strategy. Usually, systems using different DRM technologies are rivals from a market share point of view.

5.2. Directions of rights flow

When talking about rights exporting, the only direction of rights flow is from the domestic system to the target system. However, a specific DRM system can be the domestic system in one case of rights exporting, and be the target system in another case. Therefore, when discussing the direction of rights flow between two specific DRM systems, system A and system B, there are several possible cases, as illustrated in Figure 17.

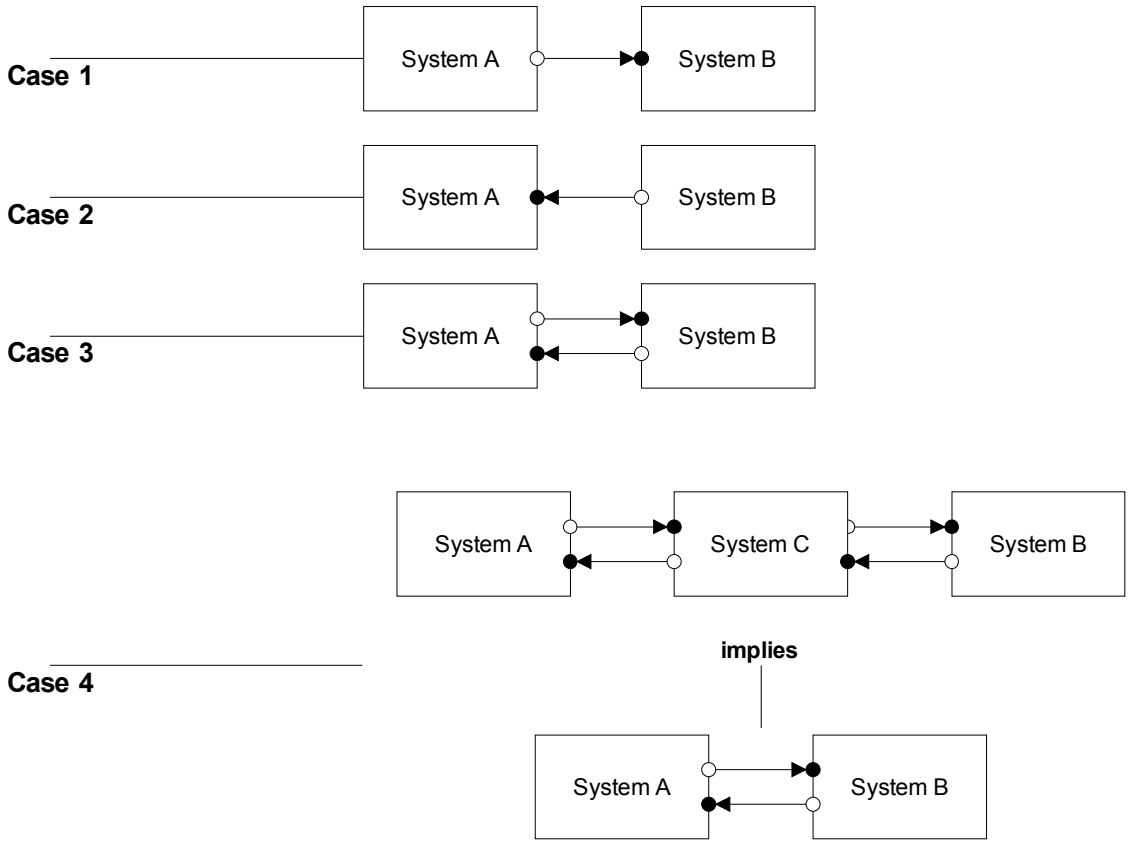

Figure 17 Directions of rights flow

In Case 1, an instance of right is only allowed to export from system A to system B. In Case 2, an instance of right is only allowed to export from system B to system A. In Case 3, an instance of right is allowed to export in both directions. Indirect rights flow between systems also needs to be considered when talking about the directions of rights flow between two systems. In Case 4, both directions are supported between system A and system B as well as between system B and system C. But there is no support for rights flow between system A and system B directly. In this case, an instance of right can still be transferred between system A and system B via system C, although there might be rights reinterpretation during the process of rights flow. From a rights flow point of view, both Case 1 and Case 2 have a one-directional rights flow between system A and system B, while both Case 3 and Case 4 have a mutual rights flow between system A and system B.

The direction of rights flow between two systems may cause potential risk to system integrity. If the rights flow between system A and system B is a mutual rights flow, and some right may be reinterpreted to grant looser usage rules than the original does, then by proceeding via the rights flow back and forth between the two systems, the

end user could obtain a free right other than the right originally granted to him. The selection for the direction of rights flow between two systems is also sensitive to business competition. The capability to export rights from system A to system B definitely benefits end users of system A more than it does end users of system B.

5.3. Results of rights flow

In theory, the level of DRM interoperability varies from access denial to free access - if measured by the rights upon protected content outside of the domestic system. Ideally the rights should not be reinterpreted differently than they are in the domestic system. Theoretically if system A accepts rights exporting from A to B, system B should grant the same right to the end user as system A does. This is the goal of DRM interoperability in theory and the ideal result of rights exporting. However, in practice system B may not be fully compatible with system A from the right's point of view and achieving the theoretical goal might not be realistic, if for example system B doesn't support certain types of permission that system A does. Under this circumstance, rights exporting needs a practical solution between the two systems.

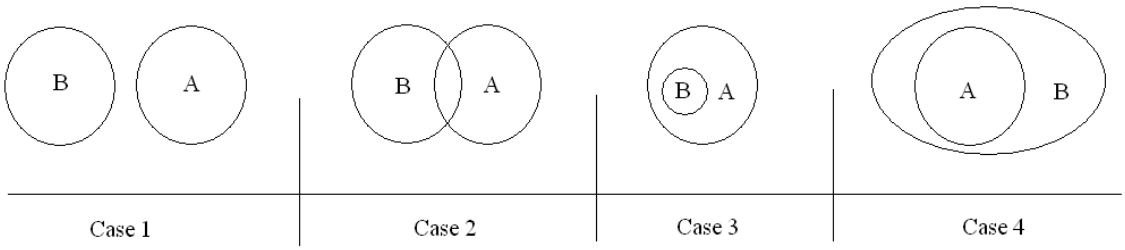

Figure 18 Results of rights flow

Besides the ideal result, there are four types of results when an instance of right is exported from system A to system B. They are illustrated in Figure 18, where rights are represented as circles inside which are the system names. Area A represents the rights allowed by the domestic DRM system, system A, while area B represents the rights in the target DRM, system B. The overlapping area represents the rights available on both systems.

Case 1 describes an extreme situation where totally different rights are granted to target system B when the end user exports the instance of right from system A to system B. Mapping the case to the rights model, it can be the case that the permission type or the content ID is changed after rights exporting. For example, Jack has a right to play an MP3 file on his smart phone. If he transfers the song to his PC, the DRM system on his PC grants him a right to burn the song into a CD instead of the right to play. The permission type changes from "play a song" to "burn a song onto a CD". Since the right is reinterpreted, Case 1 can hardly be accepted in rights exporting. However, under certain circumstances, the case can be used to realise special features. For example, it

may allow end users to exchange some instances of right for content on system A into instances of right for other content on system B.

Case 2 indicates that after rights exporting from system A to system B part of the usage rules will be reinterpreted differently for the end user on system B. It can be a change of condition types, linkage, permission types, or content IDs. For example, our imaginary actor Jack London has a right to edit a document file and to send out the text within the document via a short message service on his phone. Then, after he transfers the right to his PC, the DRM system on his PC allows him to edit and to print the document. Compared with Case 1, this case at least exports some part of the right precisely from system A to system B, although it still reinterprets differently some part of the right from system A. The case may be used, for example, to handle differences between devices, like the right to transfer data through infrared or the right to transfer data through Bluetooth.

Case 3 represents when system B makes the usage rules more restrictive when rights are exported from system A to system B. Removing some instances of permission or adding some instances of condition can result in this case. For example, Jane has the right to print and view a picture on her PC. If she exports the right to her phone, then she could only view the picture on her phone. The case is quite common when types of permission or constraints supported by system A are not supported by system B.

Case 4 indicates that system B not only grants the rights that system A does to the end user, but also offers some extra rights that system A does not. It could be adding instances of right or permission, or removing instances of condition. For example, Jack has a right to play a music file. If he exports the right to his PC, he could then play the file and burn it onto a CD as well. The case could be caused by differences of interpretation for permissions or rights between DRM systems. For example, the interpretation of full rights for content on a PC grants end users more permission than the interpretation of full rights on a mobile phone does.

Not all the cases can be accepted as a practical result of rights flow. Except for Case 3, all other cases lead to extra right in the target system. If some extra rights can be directly or indirectly transferred back to the original system or other systems, then it means that some DRM systems could obtain extra rights without payment or with less payment, which implies that the system integrity might be jeopardised. Therefore, we choose Case 3 as a practical goal for DRM interoperability, if the theoretical goal cannot be achieved, which means we aim at enforcing that the exported rights on the target device are no less restrictive than the exporting rights on the domestic device.

5.4. Principles

By analysing the factors of rights flow, we noticed that in some cases rights exporting could lead to extra rights than those an end user pays for. For example, in copy mode if a stateful condition is exported to the target system; or the result of rights flow brings

extra rights. If extra rights are generated in one-direction rights flow between two systems, the consequences are limited since it cannot be repeated for an instance of right. However, if a mutual rights flow could bring extra rights, then the end user could gain endless extra rights by simply repeating rights exporting between the two systems. Therefore, creating extra rights should be strictly forbidden during rights exporting.

On the other hand, although the end users may accept the fact that not all rights from the domestic system can be exported to the target system, they still want to export as many rights as possible.

So the principles of rights exporting can be summarised as below:

- to maximise the amount of right to be exported;
- to prevent generating extra rights

6. Decisions Making in Rights Exporting

The reason why an instance of right cannot be exported from the domestic system to the target system is that the target system cannot express the instance exactly equivalent to how the domestic system does. In other words, if an instance of right does not meet the criteria of rights exporting between the two systems, the domestic system should not allow the instance to be exported to the target system. Therefore, we need to first define the criteria of rights exporting, then based on these we can make decisions for each instance of right.

6.1. Criteria

According to the rights model we established, there are five aspects that the domestic system needs to check in order to decide whether an instance of right can be exported to the target system: right, permission, condition, linkage, and the internal structure of right.

For the aspect of right, the domestic system needs to check whether the user ID is available on the target system or not. If the target system does not have the user ID, the target system does not serve the end user. For example, Jack has an instance of right to play a MP3 file on his mobile phone, and his girlfriend Jane wants to have the same file on her phone by exporting the instance to her phone. In this case, the DRM system on Jane's phone does not support the user ID that can identify Jack. So the system on Jack's phone must not allow the instance of right to export to Jane's system.

For the aspect of permission, the domestic system needs to check whether the permission type is supported in the target system or not. If the target system does not support the permission type, the corresponding right cannot be used on the target system. Therefore, a right containing such a permission type should not be exported to the target system. For example, Jack has an instance of right to print a document file onto paper from his PC. Since Jack's phone does not support printing the system on his PC does not support the permission "printing". Therefore the system on Jack's PC should not export the permission "printing" to his phone. The content ID is not checked because the protected content could be exported after the rights exporting takes place. Consequently, the content ID may not be supported at the moment on the target system when the right is being exported, which should not stop the right from exporting to the target system without adaptation.

For the aspect of condition, the domestic system needs to check whether or not the condition type is supported by the target system. Ideally, once the condition type is set, the format of the content detail is set as well. However, in practice there might be different formats of content detail. Different formats should be able to convert to each other; otherwise the content detail also needs to be checked. If the condition type is not

supported on the target system, it means that the target system cannot enforce this kind of condition. The right that contains a condition with this condition type should not, therefore, be exported to the target system. For example, Jack has an instance of right to use a GPS map for one month on his PDA. However, his phone does not support time-based conditions. Therefore the system on his PDA should not allow the instance of right to be exported to his phone without adaptation.

For the aspect of linkage, the domestic system needs to check whether the linkage type is supported on the target system or not. If the target system does not support the linkage type, it means the target system cannot enforce separately either the influential path or the validity-checking path. For example, suppose that the system on Jack's phone supports a bonus system. If Jack plays a game more than 10 times, one extra bonus time will be granted. If the bonus system does not work on his PC, Jack either abandons the bonus part of the right or gives up exporting the right to his PC.

For the aspect of internal structure, the domestic system needs to check whether multiple conditions are supported or not; whether a shared condition is supported or not; and whether the combination of permission type and condition type is supported. For example, Jack has an instance of right to access one bank account 10 times within a month on his PC. However, his mobile phone does not support the combination of a time-based condition and permission to access the bank. So the system on his PC should not allow the instance of right that contains this internal structure to be exported to his phone. We also need to take the capacity into consideration. The capacity for multiple conditions refers to how many instances of condition an instance of permission is allowed to have, while the capacity for a shared condition is how many instances of permission are permitted to have an instance of shared condition. For example, Jack has an instance of right to play an MP3 music file 10 times in one day starting from June 1st 2006. Three instances of constraint restrict the permission to play: 10 times, in one day, and starting from June 1st 2006. Jack's PC does not support the internal structure whereby more than two instances of constraint restrict one instance of permission. Therefore, Jack is not allowed to export the instance of right to his PC without rights adaptation.

The criteria decide whether an instance of right can be exported to the target DRM system. If an instance of right does not satisfy one of the criteria, the instance simply cannot be expressed precisely by the target system. As a consequence, the instance should not be exported to the target system. The criteria are the main factors that affect a DRM system exporting rights to another DRM system. Therefore, we have answered the first research question by defining the common criteria of rights exporting.

6.2. Decisions in rights exporting

With defined criteria for rights exporting, we can make decisions based on them. The decisions define what we need to do further for the concerned instance of right.

Safavi-Naini, et al. (2005) categorised the possible decisions for condition evaluation as:

- Isomorphism, which means that the instance of right can be expressed by the target system in an exactly equivalent manner as it is on the domestic system;
- Adaptation, which means that the instance of right needs to be adapted in order to suit the requirements from the target system;
- Untranslatable expressions, whereby the instance of right cannot be expressed by the target system.

Three decisions cover all the possible results of rights exporting. However, Safavi-Naini, et al. did not discuss decisions making from a process point of view. Therefore, the decisions they defined were the static and final decisions for rights exporting. In order to describe the process of decisions making in rights exporting, we need to regard decisions as states of rights in the process. Therefore, we define three possible states:

- Accepted, which means that the instance of right meets the criteria and therefore can be expressed by the target system exactly the same as it is on the domestic system without adaptation.
- Rejected, which means that the instance of right does not meet the criteria and therefore cannot be expressed by the target system at all.
- Adapted, which means that the instance of right is adapted.

An adapted right implies that the original right is rejected. After rights adaptation, the original right becomes the adapted one. The domestic system then needs to decide if this adapted right can be accepted for rights exporting. So the state “adapted” can be regarded as a temporary state for an instance of right. The relationship between the three states can be expressed as a process view as illustrated in Figure 19. If an instance of right needs to be exported to a target system, the domestic system needs to decide whether the right can be accepted by the target system by using the criteria. If the answer is yes, then the right can be exported to the target system. If the answer is no, then the domestic system needs to check the feasibility of rights adaptation for the instance of right. If the instance cannot be adapted further, then the right is rejected. If it can be adapted, then the domestic system performs rights adaptation upon the instance of right. After rights adaptation, the instance of right is adapted into a new instance. The domestic system needs to make decision for the new instance again. At the end of the process, all instances of right are either accepted or rejected for rights exporting. Rejected instances will remain unchanged on the domestic system, while accepted instances will be exported to the target system. Adapted instances that have been accepted will be exported to the target system in the adapted manner; while those that have been rejected will be stored back in the domestic system in the original manner.

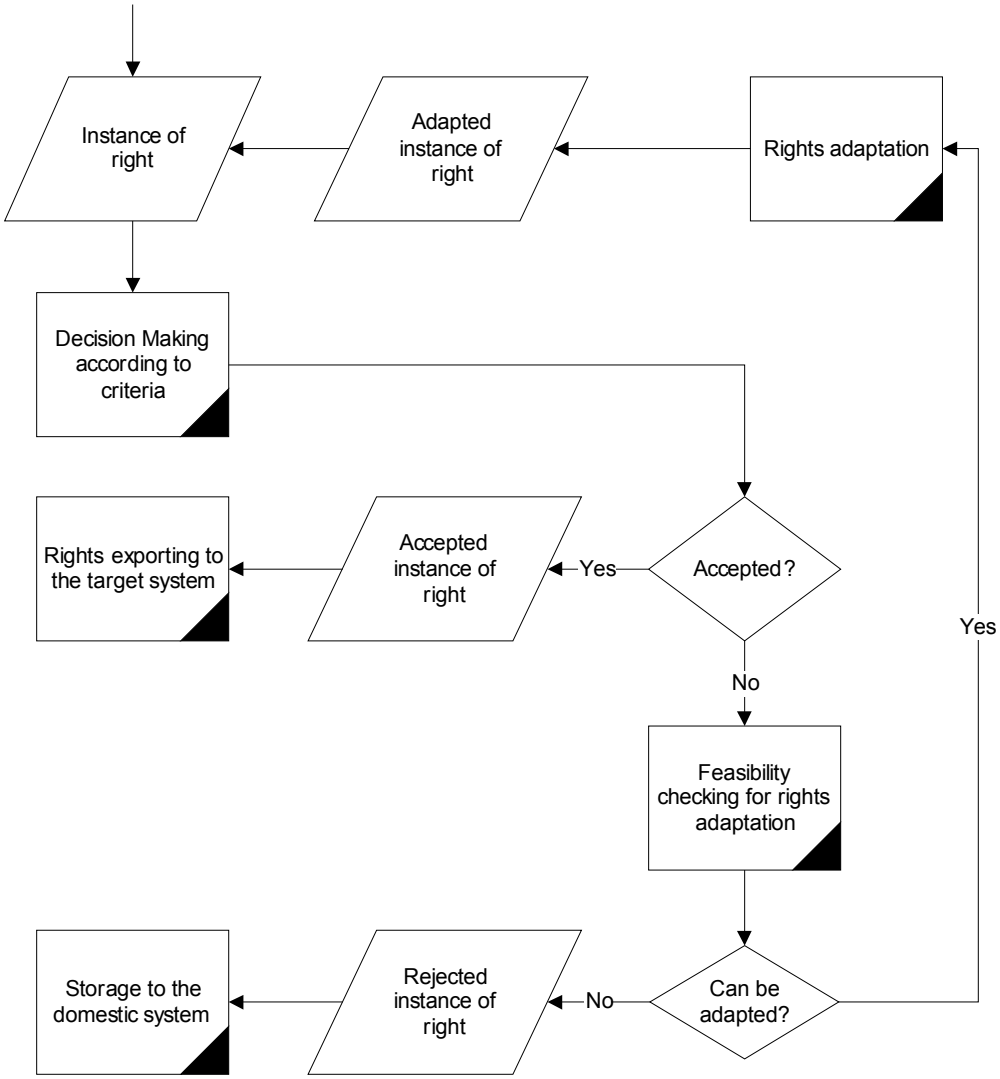

Figure 19 Decisions in rights exporting

6.3. Further equivalent decomposition

According to the principles of rights exporting, we need to maximise the rights that are exported. For one instance of right, if some criteria are not satisfied, the whole instance of right may not be exported. Therefore, we need to decompose the right into its smallest instances. By doing this the instance should have the simplest structure of permission and condition.

According to the rights model, an instance of right can be decomposed to several sub-instances if some groups of components inside the instance are neither directly nor indirect linked to each other, as mentioned in section 4.4. After rights decomposition, components in the sub-instances are linked to each other. We questioned if it is possible to further decompose equivalently the instance of right. We then recognised that under certain circumstances equivalent decomposition is feasible even though the components inside an instance of right are linked to each other.

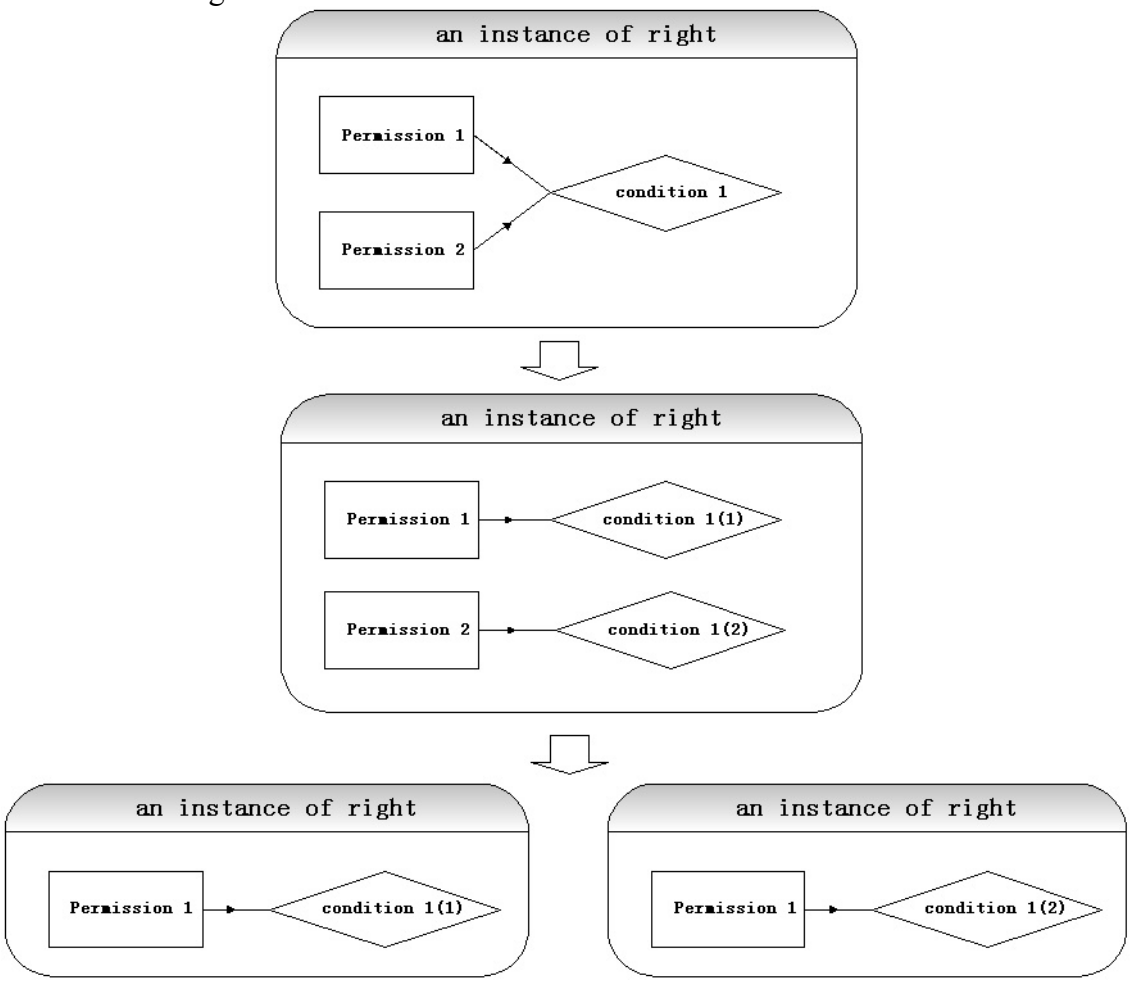

Figure 20 Further equivalent decomposition

Once the type of condition instance is a stateless condition, the influential path does not exist between this instance of condition and its linked instances of permission, and only the validity-checking path exists. All permissions linked to the same instance of condition cannot influence each other's permissions by any means. Therefore, it is

equivalent to the case whereby those instances of permission are linked to different instances of condition, where those instances of condition have the same condition details and condition type. Therefore, we make a copy of an instance of condition for each instance of permission, in order to further decompose the right with a shared condition. Since there is no internal state for a stateless condition, the copied instances are always equivalent to the original instance. By doing so, the shared condition is not shared anymore. The rights decomposition can then be extended so that each further decomposed instance of right has definitely less components than the original decomposed instance of right. As Figure 20 illustrates, the condition type of Condition 1 is a stateless condition and it is a shared condition for both Permission 1 and Permission 2. The instance of right is equal to the case where Permission 1 links to Condition 1's copied instance Condition 1(1), while Permission 2 links to another copied instance, Condition 1(2). Both Condition 1(1) and Condition 1(2) have the same attributes as Condition 1. The instance of right can then be further decomposed into two instances of right. Figure 21 illustrates an example. It is an instance of right for Jack that grants him permission to either play or view an image during May 2007. The instance can be further decomposed into two instances of right: permission to print the image in May 2007, and permission to view the image in May 2007.

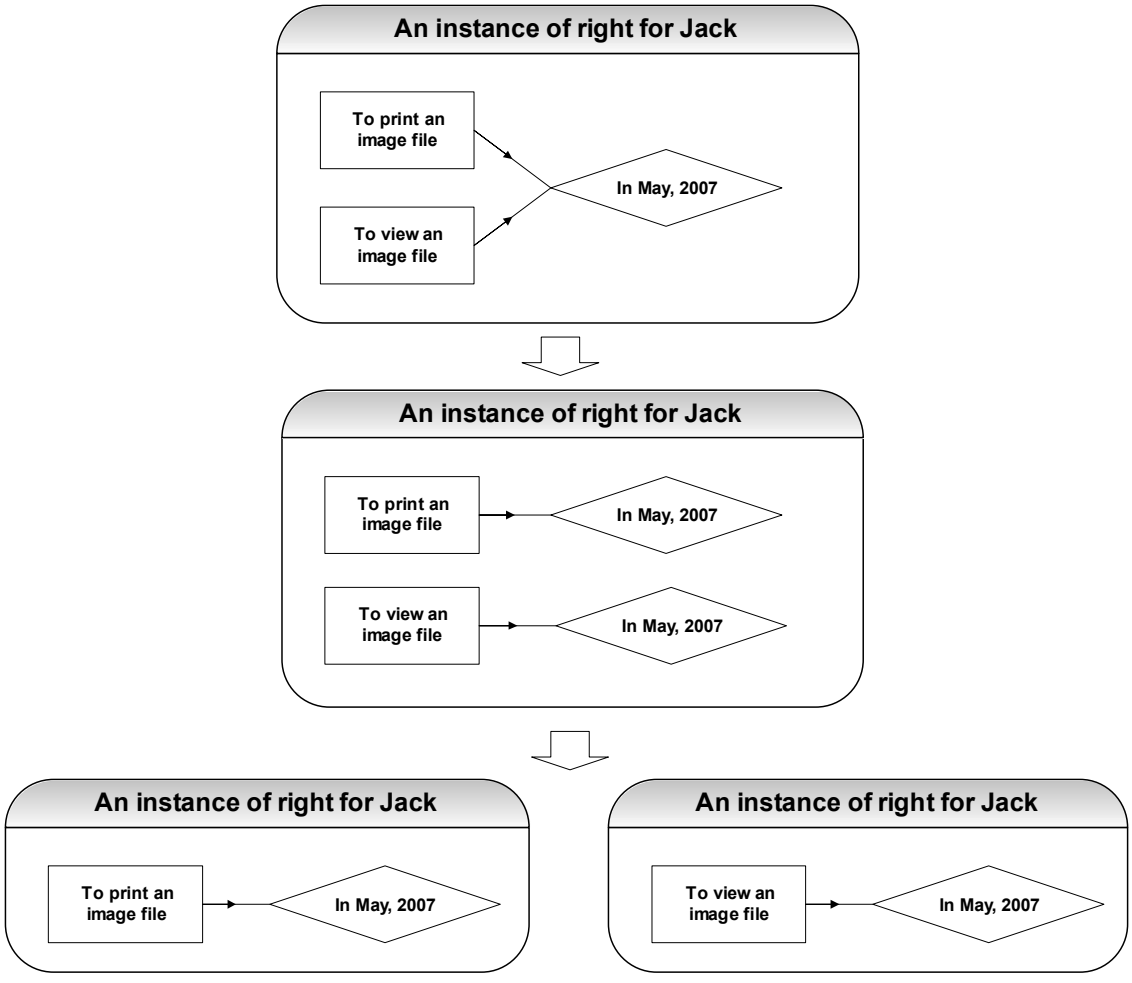

Figure 21 An example of further equivalent decomposition

Further equivalent decomposition is only feasible for a shared stateless condition. Also, in exceptional cases, a right cannot be decomposed with the mechanism. As shown in Figure 22, Permission 1 and Permission 2 both link to Condition 1 and Condition 2, while Condition 1 is stateless and Condition 2 is stateful. In this case, we cannot break the influential path linked to Condition 2 by adding a copy of the instance of Condition 1.

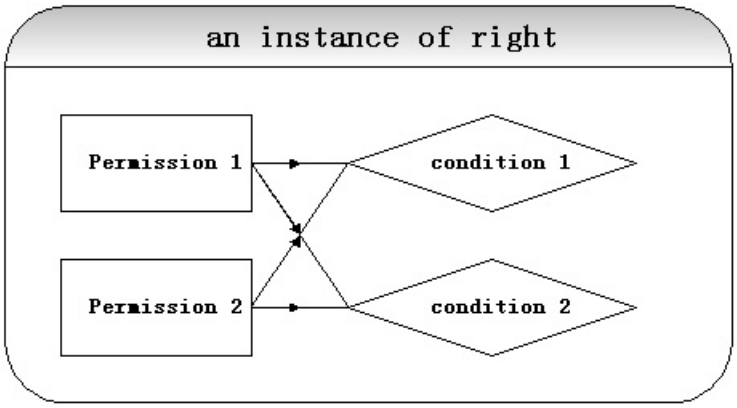

Figure 22 Infeasible for further decomposition

Once rights are decomposed, the original instance of right could be changed to a few decomposed instances of right. Therefore, we can regard the decomposed rights as a temporary state of the instance of right.

6.4. Right adaptations

After the instance of right is decomposed, the decision for each decomposed instance can be made according to the criteria we defined. If the criteria are not satisfied, the domestic system shall either reject to export, or adapt the instance in order to reach the criteria.

6.4.1. Adaptation methods

As we mentioned earlier, Safavi-Naini et al. (2004) defined three methods for adaptation. We adapt those with respect to the rights model and define our own methods as listed below:

- Condition pre-enforcement
- Permission reduction
- Condition division
- Condition merge

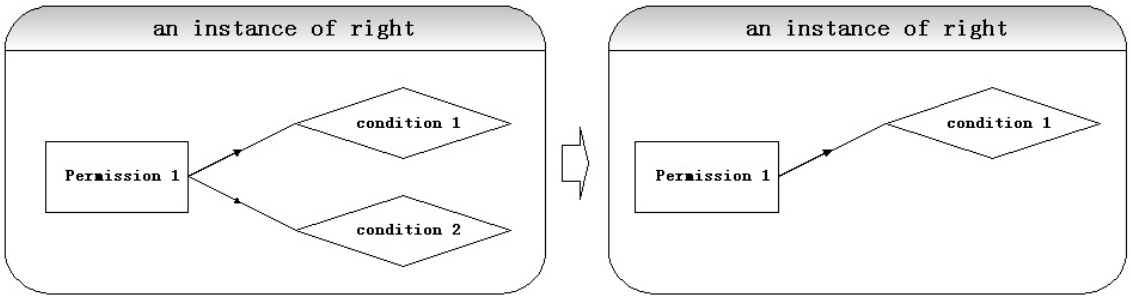

Figure 23 Condition pre-enforcement

With condition pre-enforcement, instances of condition can be enforced on the domestic system before the instance of right that contains it is exported to the target system. Once the instance of condition is enforced, it becomes true from then onwards. It does not then have any impact on the validity checking for the instance of right. Therefore the instance of right can remove this instance of condition as well as its linkage to instances of permission. Condition pre-enforcement can be used to handle the case where the target system does not support the type of condition that the domestic system does. Figure 23 illustrates this: Permission 1 has multiple constraints, Condition 1 and Condition 2. The type of Condition 2 is not supported by the target system. Therefore the domestic system tries to enforce Condition 2. If Condition 2 can be enforced, then Condition 2 and its linkage with Permission 1 can be removed. Condition pre-enforcement deprives the end user of the right to enforce the condition after rights exporting, which to some extent makes the usage rule stricter than it is on the domestic system before rights exporting. For example, Jack has an instance of right to chat via a piece of software on his PC after he registers on a website. In this case, the registration is the instance of condition. Now Jack wants to export the right to his mobile. However, the system on his phone does not support registration to this website. Therefore, Jack has to first register on the website on his PC before the system on his PC can allow the instance of right to be exported to his phone.

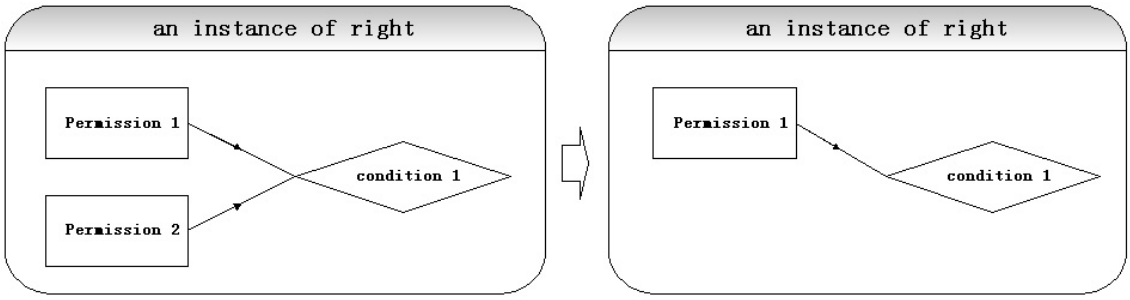

Figure 24 Permission reduction

Permission reduction means that some instances of permission and their linkage to condition can be reduced before exporting an instance of right to a target system. Once the instance of permission is reduced, the end user who purchases the instance of right will lose the permission on the target system. As illustrated in Figure 24, the type

Permission 2 is not supported by the target system. Therefore Permission 2 is removed from the instance of right as well as its linkage to Condition 1. Permission reduction can be used to handle a case where the target system does not support the same type of permission that the domestic system does. For example, Jack purchases an instance of right to view or print an image file within a month on his PC and he also wants to set the image as the wallpaper on his phone. He exports the content and its right to his phone. However, the phone does not support printing as a type of permission. Therefore, the permission to print will be removed from the right to export. After rights exporting, Jack can view the image on his phone for one month.

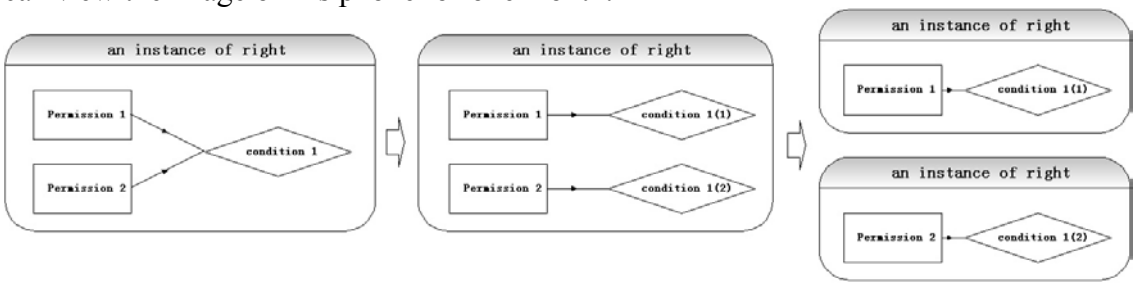

Figure 25 Condition division

Condition division means that some instances of condition can be divided into several instances of condition so that the divided instances of condition are a subset of the original instance of condition. Condition division can be used to handle the case when in move mode the target system does not support the shared condition. After the condition is divided, different sets of instance of permission that are linked to the original instance of condition link to different divided instances of condition. The condition division is only suitable for types of stateful conditions that have a dividable internal state, for example, count based, end time based and time interval based. Once the instance of condition is divided, each divided instance has its own influential path, which means that the divided internal states are updated without cross-instance influences. From the right's point of view, once the instance of condition is divided, the instance of condition for each instance of permission is fixed, and the end user cannot re-arrange the allocation of an instance of condition to an instance of permission. It makes the right more restrictive than the original right. As illustrated in Figure 25, Permission 1 and Permission 2 are sharing Condition 1. The target system does not support a shared condition. In order to export the instance of right, Condition 1 should be divided into Condition 1(1) for Permission 1 and Condition 1(2) for Permission 2. As a side effect of condition division, the instance of right can be further decomposed into two instances of right. For example, Jack has the right to play an MP3 file, or to share playing it with a friend, for 20 times on his PC. Jack wants to use the file on his phone as well. The system on his phone does not support shared conditions. Therefore, the system on his PC asks Jack how many times he wants to play the file on his phone, and Jack inputs 10 times. Then, after rights exporting, Jack has two instances right on his phone: the right

to play the MP3 file for 10 times, and the right to share playing with a friend for 10 times.

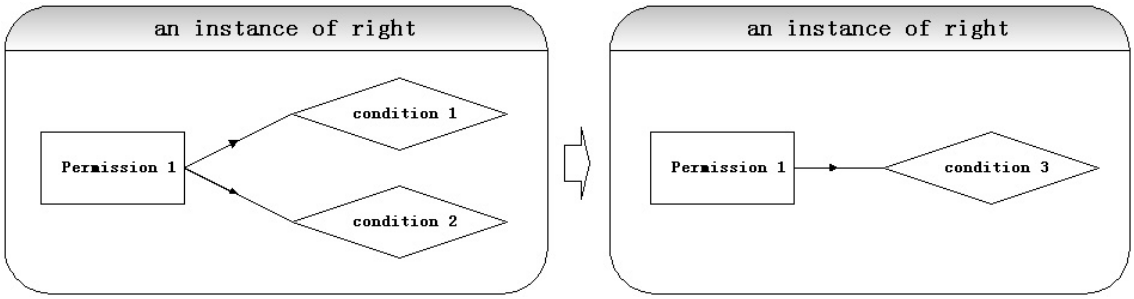

Figure 26 Condition merge

Condition merge takes place when more than one instance of condition link to the same set of instances of permission, and these instances of condition have the same condition type. After merge, the internal state should be the intersection of those instances of condition that are merged. For example, permission to play a game with two instances of condition: 3 times, and 5 times. After the merge, it becomes permission to play a game 3 times. As illustrated in Figure 26, Permission 1 has multiple conditions, Condition 1 and Condition 2. Both conditions have the same condition type and the internal states can be merged. The two conditions are then merged into Condition 3. After merging, Permission 1 links only to Condition 3. Condition division can be used to handle a case where the target system does not support multiple conditions. For example, Jack subscribes to an e-newspaper service for a month on his PC, and today he receives a special offer to read an e-book for a year. However, if his subscription is out of date then he will also lose the right to read the e-book as well. Now he decides to export the right to read the e-book to his phone. However, his phone does not support multiple conditions as an internal structure of right, so the system on his PC suggests to him to merge the right. After condition merge and rights exporting, Jack can read the e-book on his phone for one month.

6.4.2. Deployment of rights adaptation

Different adaptation methods shall be applied in different cases when an instance of right does not meet the criteria to export to a target system.

The direct effects of the methods on the criteria are mentioned in the previous section that introduces each method. There are also indirect effects for each method, which are listed in Table 1. For example, condition merge has indirect effects on both the capacity of multiple conditions and shared conditions since it reduces the number of linkages. In Table 1, N/A means the method is not applicable to the criteria. For example, condition division is not applicable if the target system does not support the permission. According to Table 1, DRM systems could implement the strategy to utilise the correct methods when adaptation is needed for rights exporting. For example, an instance of right has a shared condition structure that the target system does not support.

When performing rights adaptation we know that condition division has a direct impact on a shared condition, while condition pre-enforcement and permission reduction have an indirect impact on it. Hence, we know that we should first try to use condition division and then try to use the other two methods. We also know that we don't have to try condition merge in this case since condition merge does not have influence on the problem.

Table 1 Effects of adaptation methods

Methods / Criteria	Condition pre-enforcement	Permission reduction	Condition division	Condition merge
Types of permission	N/A	Direct	N/A	N/A
Types of condition	Direct	Indirect	N/A	N/A
Types of linkage	Indirect	Indirect	N/A	N/A
Multiple conditions	Indirect	Indirect	N/A	Direct
Shared condition	Indirect	Indirect	Direct	N/A
Capacity of multiple conditions	Indirect	Indirect	N/A	Indirect
Capacity of shared condition	Indirect	Indirect	N/A	Indirect

According to the principles of rights exporting, maximizing the portion of rights to be exported is what end users expect. Therefore, when several adaptation methods can be applied to rights exporting, they shall be prioritised in order to ensure that the best results are achieved. For example, both condition division and permission reduction may solve a case where the target system does not support shared conditions. The condition division should be used first since it reduces the right less than permission reduction. Similarly, both condition merge and condition pre-enforcement may solve a case where the target system does not support multiple conditions. The condition merge should be used first. Table 2 shows the priorities of methods when adaptation is needed in order to reach each criterion. A value of zero means that the method should be considered first; as the values increase the priority of consideration of the method decreases. For example, an instance of right has a shared instance of condition with three instances of permission; but the target system only supports a maximum of two instances of permission sharing for one instance of condition. Therefore, rights adaptation is needed before rights exporting. From Table 2, we know we first need to try condition merge, then condition pre-enforcement, and lastly permission reduction.

Table 2 Priorities of Adaptation methods

Methods / Criteria	Condition pre-enforcement	Permission reduction	Condition division	Condition merge
Types of Permission	N/A	0	N/A	N/A
Types of condition	0	1	N/A	N/A
Types of linkage	0	1	N/A	N/A
Multiple conditions	1	2	N/A	0
Shared condition	1	2	0	N/A
Capacity of multiple conditions	1	2	N/A	0
Capacity of shared condition	1	2	N/A	0

6.5. Process of decision making

Right adaptation is an incremental process. When an adaptation method is used, the adapted instance of right needs to be decomposed again. As shown in Figure 25, after one round of rights adaptation, the original right may become several adapted instances of right. Moreover, the adapted instances of right may be further decomposed. Therefore we need to extend the simple process illustrated in Figure 19 into a sophisticated solution.

As shown in Figure 27, an instance of right first needs to be decomposed. Then decisions need to be made for each decomposed instance of right based on the criteria to export right from the domestic system to the target system. Accepted instances of right can be later exported to the target system. If an instance of right is not accepted then we need to check if the instance can be further adapted in order to pass the criteria. If the instance cannot take any further rights adaptation, then we need to reject the instance. If the instance needs further adaptation, then after rights adaptation the adapted instance of right needs to go through the whole process again. At the end of the process, there are only accepted instances of right and rejected instances of right. If there is no accepted instance of right, it means the original instance of right cannot be exported to the target system. Then all rejected instances of right can be discarded, and the original instance should not be changed on the target system. If there are accepted instances of right, those accepted instances can be exported to the target system, while those rejected instances need to be stored back on the domestic system.

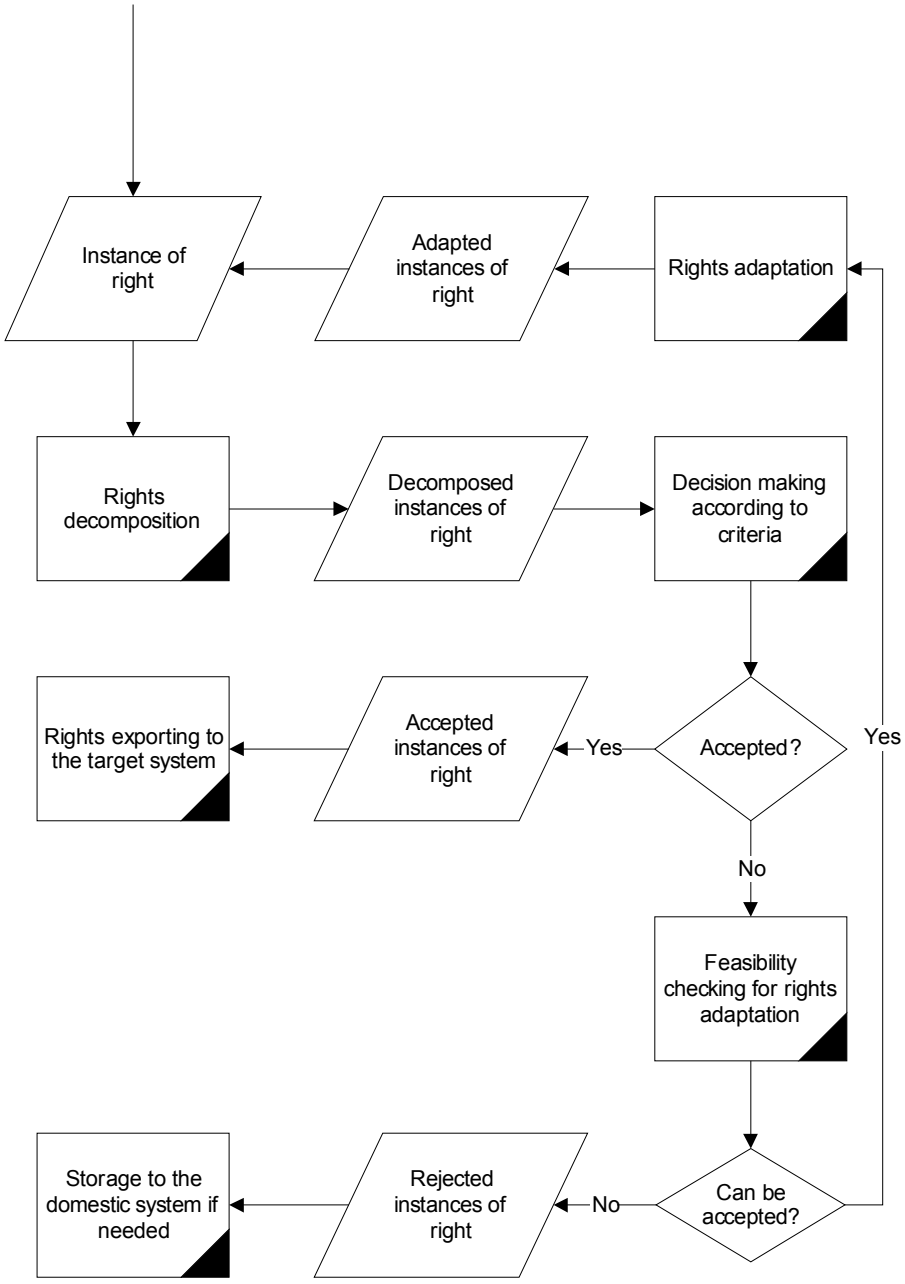

Figure 27 Process of solution to decisions making

The process aims at maximizing the right to export, given a certain amount of right and a set of criteria for rights exporting. In other words, the process intends to mitigate the influences caused by the difference between the domestic DRM system and target DRM system when exporting right between the two systems. Therefore, the process we proposed here is our answer to the second research question.

7. Case Study

In the earlier chapters, we have established a rights model. Based on the model we analysed the requirements for rights exporting and developed a set of solutions to meet the requirements. In order to verify the feasibility of the model and the solutions, we attempt to adopt them in a case study of rights exporting between existing DRM systems. We select two representative groups of systems: Windows Media DRM 10 [WMDRM, 2006] compliant, and OMA DRM version 2 [OMA_DRM_2, 2004] compliant. Both are based on well-defined standards, and have a large amount of industry implementation.

7.1. Windows Media DRM 10

"Windows Media Digital Rights Management (WMDRM) is a platform to protect and securely deliver content for playback on computers, portable devices, and network devices."[WMDRM, 2006] Microsoft initialised WMDRM in April 1999. After several years' evolution, the latest version, version 10, came out in 2004 [WMDRM_G, 2004]. WMDRM utilises XrML based REL, as an example illustrates in Appendix 1. XrML is one of the two major branches of REL standards as mentioned earlier. Microsoft has licensed WMDRM and partnered with numerous industry-leading content providers, service providers, solution providers, application developers, hardware manufacturers, and processor companies. The complete list is available on the Microsoft's official website [WMDRM_G, 2004]. As WMDRM is a huge concept to describe, we scale down the scope to our concerns and summarise the characteristics of rights in WMDRM.

7.1.1. Terminology mapping

WMDRM uses a different set of terms to describe DRM concepts. In order to avoid confusion, this section maps concepts that are most relevant to our concerns into the concepts defined in our rights model.

In WMDRM, a license is a logical unit for right. Appendix 2 provides an example of issuing a license by using a VB script and it presents the structure of a license. A license consists of a group of instances of permission and their linked condition. Each license has a key ID that is equivalent to content ID. The key ID indicates that under a license all instances of permission apply to the same content ID. The definition of a license is similar to the definition of an instance of right, except that WMDRM allows dependencies between licenses. This dependency is called license chaining. A simple license chain contains two parts: root license and leaf license. As shown in Figure 28, the leaf license is the license that has an uplink ID. The uplink ID indicates the key ID of the root license. When an end user wants to render content whose right is granted by a leaf license, both the leaf license and its root license must be valid.

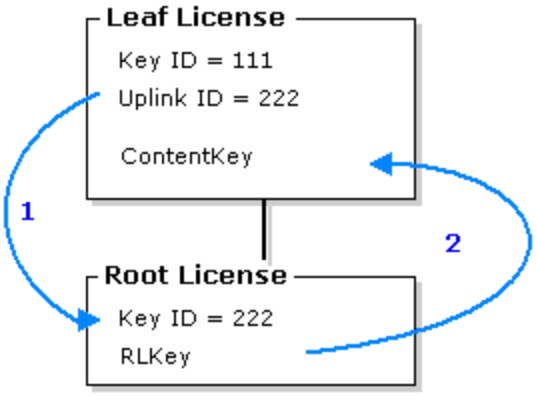

Figure 28 License chain in WMDRM [WMDRM_SDK, 2004]

The instances of permission and condition are in the form of properties of the license. Some conditions are specific to a certain type of permission, for example “play count” is a type of condition only for permission to play. Meanwhile some conditions are common to all instances of permission in a license, such as “begin date”. These common conditions can be expressed by a shared condition in our rights model, while those conditions specific to permission can be expressed by an instance of condition to the specific permission. Also, the instance of condition in a root license can be expressed as a shared condition between the root license and leaf license.

In WMDRM, the type of most linkages is both paths. However, one exception is the metering mechanism, in which linkage is a pure influential path. Metering is a technology that allows digital-media content providers to count the number of times protected content is used (for example, played or copied).

For the internal structure, if the number of common conditions within a license is more than one, then those conditions are not only shared conditions for all instances of permission within the license, but they are also multiple conditions for each instance of permission within the license. Moreover, if a license chain exists, the shared condition is used to express the relationship between the root license and leaf license. Figure 29 shows a leaf license consisting of Permission 2 and Condition 3, while its root license consists of Permission 1, Condition 1, and Condition 2. Permission 1 and Permission 2 have different content IDs since they are from different licenses. When Permission 2 is granted, the system needs to check all three instances of condition and update their internal states when needed. However, if Permission 1 is granted, which means the root license is used, only Condition 1 and Condition 2 are checked.

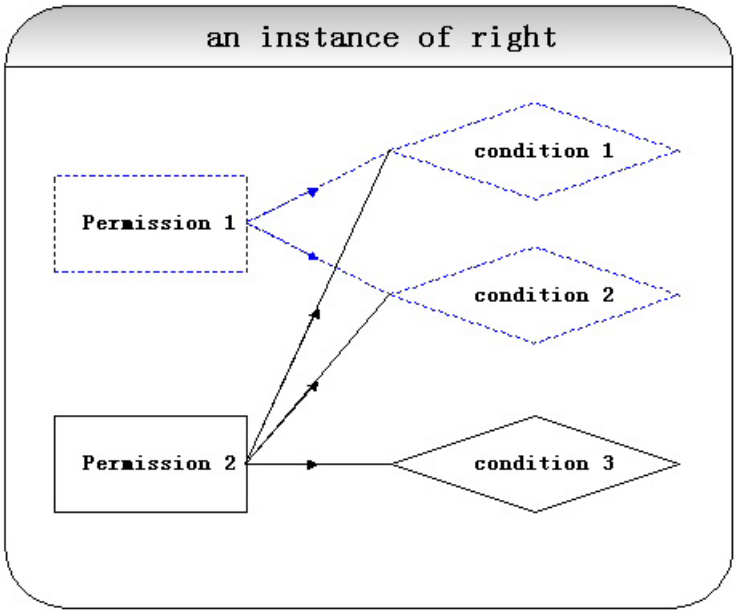

Figure 29 Structure of a root-leaf license in the rights model

7.1.2. Right characteristics

WMDRM has defined right in the specification [WMDRM_SDK, 2004]. The features of right are summarised in this sub-section.

Table 3 shows that WMDRM supports eight types of permission and twelve types of condition. Not all types of condition are applicable to all types of permission. N/A indicates that the type of condition is not applicable to the type of permission. A cross (X) indicates that the type of condition is applicable to the type of permission. The detailed description of each type of permission and condition can be found in Appendix 3.

Table 3 Type of permission and condition in WMDRM systems [WMDRM_SDK, 2004]

Type of permission / Type of condition	Backup and restore	Collaborative play	Copy	Play	Play list burn	Transfer to non-SDMI	Transfer to SDMI	Play list burn track
Begin date	X	X	X	X	X	X	X	X
Count	N/A	N/A	X	X	X	X	X	X
Restrictions	N/A	N/A	X	X	N/A	N/A	N/A	N/A
Expiration after first use	X	X	X	X	X	X	X	X
Expiration date	X	X	X	X	X	X	X	X
Expiration on store	X	X	X	X	X	X	X	X
Grace period	N/A	N/A	N/A	X	N/A	N/A	N/A	N/A
Delete on clock rollback	X	X	X	X	X	X	X	X
Disable on clock rollback	X	X	X	X	X	X	X	X
Exclude application	X	X	X	X	X	X	X	X
Minimum client SDK security	X	X	X	X	X	X	X	X
Minimum security level	X	X	X	X	X	X	X	X

WMDRM also has some system specific features for permission and condition, which might not be supported by other DRM systems. Therefore we need to take them into consideration when making decisions in rights exporting. First, some types of condition need some type of precondition. For example, delete on clock rollback and disable on clock rollback are only applicable if there is a time-based condition being used. Second, some types of condition are exclusive to some other types of condition, e.g., delete on clock rollback is exclusive to disable on clock rollback, and expiration after first use is exclusive to expiration on store. Third, some types of condition or permission under certain circumstances have their own restrictions specific to WMDRM compliant systems. For example, the count condition for the permission to burn a play list and the permission to burn a play list track cannot be a root license, count condition for the permission to copy cannot exceed 249, and collaborative play does not support more than 10 listeners. Fourth, instances of permission could have various priorities and when granting right upon protected content, the instance that could grant the right and has highest priority would be consumed first.

7.2. OMA DRM 2

The Open mobile alliance (OMA) is a leading industry forum for developing market driven, interoperable mobile service enablers. It was formed in June 2002 by nearly 200 companies, including the world's leading mobile operators, device and network suppliers, information technology companies, and content and service providers [OMA,

2002]. The mission of the OMA is to facilitate global user adoption of mobile data services by specifying market driven mobile service enablers that ensure service interoperability across devices, geographic locations, service providers, operators, and networks, while allowing businesses to compete through innovation and differentiation. In September 2004, OMA approved its DRM version 1.0 specification. The latest version, version 2, was released during the summer of 2004. Different from the REL used by WMDRM, OMA DRM version 2 uses the ODRL based REL, which is another major branch of REL standards. Currently, as the major competitor to WMDRM, OMA DRM has the leading position in the mobile world within Europe [DRM, 2006-1]. OMA DRM version 2 is a well-defined open standard that contains many specifications for different aspects of DRM. We concentrate on the right's aspects.

7.2.1. Terminology mapping

As the REL of OMA DRM 2 is based on ODRL version 1.1, its terminology is also partially inherited from ODRL. This section maps the most relevant concepts to our rights model. An example of rights expressed by ODRL can be found in Appendix 4.

In OMA DRM 2, a rights object (RO) represents the basic unit for right. It is a collection of permissions, constraints, and other attributes that define under what circumstances access is granted to, and what usages are defined for, DRM protected content [OMA_REL_2, 2006]. An RO is not equivalent to an instance of right in our model, since OMA DRM 2 supports an inheritance model for ROs. The RO that inherits instances of permission is called a child rights object (C-RO), while the RO that contains the instances of permission that are inherited is called a parent rights object (P-RO). In the illustration in Figure 30, DCF stands for 'DRM content format' and two C-ROs have the same P-RO. In the case on the left, P-RO does not have the condition to play while the two C-ROs define their own count condition for permission to play. The C-RO for DCF-1 can play 5 times, and the C-RO for DCF-2 can play 2 times. In the case on the right, the P-RO defines the condition to play 7 times while no condition is defined for the C-ROs for permission to play. In this case, DCF-1 and DCF-2 can be played 7 times all together, like a shared condition in our rights model.

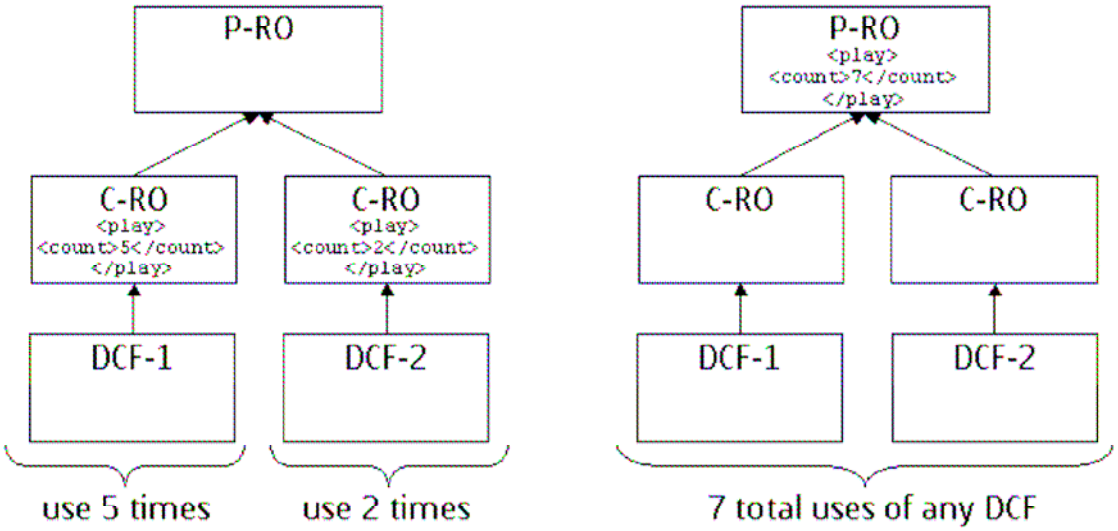

Figure 30 Parent RO and associated semantics [OMA_REL_2, 2006]

OMA DRM 2 also has the permission concept within an RO. The permission in OMA DRM 2 consists of a set of permission instances that share the same content ID. Each instance of permission has a set of constraints. The set of constraints is similar to the concept of multiple conditions in our model, except that OMA DRM 2 supports nested constraints called top level constraint. A top level constraint works as a shared condition for a set of permissions within one rights object. For example, a rights object for an image file defines its top level constraint as “8 times”. Under the top level constraint, the constraint for permission to view is “8 times”, while the constraint for permission to print is “10 times”. In this case, the top level count will decrease together with the count for the corresponding permission when the end user either views or prints the image file.

There is only one type of linkage, “both paths”, in OMA DRM 2. OMA DRM 2 supports a subscription use case in a different manner. Therefore, the validity-checking path does not appear in this case. However, some paths are not explicit. The group ID mechanism makes paths more difficult to follow since it adds an extra content ID to the protected content.

7.2.2. Right characteristics

OMA DRM 2 REL specifies all its features in its specification [OMA_REL_2, 2006]. We summarise all the relevant features that concern us.

Table 4 shows that OMA DRM 2 supports five types of permission and seven types of condition. Not all types of condition are applicable to all types of permission. N/A indicates that the type of condition is not applicable to the type of permission. A cross (X) indicates that the type of condition is applicable to the type of permission. The detailed descriptions of each type of permission and condition can be found in Appendix 5.

Table 4 Types of permission and condition in OMA DRM 2 [OMA_REL_2, 2006]

	Play	Display	Print	Execute	Export
Count	X	X	X	X	X
Timed-count	X	X	N/A	X	N/A
Date time	X	X	X	X	X
Time interval	X	X	X	X	X
Accumulated time	X	X	N/A	X	X
Individual	X	X	X	X	X
System	X	X	N/A	N/A	X

OMA DRM 2 intends to solve interoperability issues with other DRM systems through the permission type “export” and the constraint type “system”. However, the standard leaves the details open.

7.3. Criteria of rights exporting

According to the definition of criteria in section 6.1, we can sort out the criteria of decisions making in rights exporting between OMA DRM 2 and WMDRM 10.

From the right’s point of view, as long as both of the involved DRM systems can recognise the authenticated user ID from each other, both systems would accept the request for rights exporting.

From the permission and condition point of views, Table 5 and

Table 6 show the decisions when exporting rights from WMDRM 10 to OMA DRM 2, while Table 7 and Table 8 show the decisions when exporting rights from OMA DRM 2 to WMDRM 10.

Table 5 Decisions based on permission type from WMDRM 10 to OMA DRM 2

WMDRM 10	**OMA DRM 2**
Backup and restore	Rejected
Collaborative play	Rejected
Copy	Rejected
Play	Accepted -> Play
Play list burn	Rejected
Play list burn track	Rejected
Transfer to non-SDMI	Rejected
Transfer to SDMI	Rejected

Table 6 Decisions based on condition type from WMDRM 10 to OMA DRM 2

WMDRM 10	OMA DRM 2
Begin date	Accepted -> Date time
Count	Accepted -> Count
Restrictions	Rejected
Expiration after first use	Accepted -> Time interval
Expiration date	Accepted -> Date time
Expiration on store	Accepted -> Date time
Grace period	Rejected, or pre-enforced
Delete on clock rollback	Rejected
Disable on clock rollback	Rejected
Exclude application	Rejected, or pre-enforced
Minimum client SDK security	Rejected, or pre-enforced
Minimum security level	Rejected, or pre-enforced

Table 7 Decisions based on permission type from OMA DRM 2 to WMDRM 10

OMA DRM 2	WMDRM 10
Play	Accepted -> Play
Display	Rejected
Execute	Rejected
Print	Rejected
Export	Rejected

Table 8 Decisions based on condition type from OMA DRM 2 to WMDRM 10

OMA DRM 2	WMDRM 10
Count	Accepted -> Count
Timed-count	Rejected
Date time	Accepted -> Begin date and Expiration date
Time interval	Expiration after first use
Accumulated time	Rejected
Individual	Rejected
System	Rejected

From the linkage point of view, since WMDRM uses an influential path to realise the metering mechanism, which cannot be supported by OMA DRM 2, those rights requiring a metering mechanism coming from systems based on WMDRM 10, cannot be exported to systems based on OMA DRM 2.

From an internal structure point of view, both WMDRM 10 and OMA DRM 2 support both multiple conditions and shared conditions. However, OMA DRM 2

supports group ID by adding an extra content ID to content, which cannot be supported by WMDRM 10. Therefore, those group rights from systems based on OMA DRM 2 cannot be exported to systems based on WMDRM 10.

7.4. Analysis of the proposed process

Terminology mapping helps us to adapt a right from both WMDRM and OMA DRM 2 into the right of our rights model. With defined criteria for rights exporting between the two systems, we can deploy the process in Figure 27. We now need to analyse the process we proposed based upon deployment on real systems.

Right decomposition is necessary when dealing with rights exporting between WMDRM and OMA DRM 2. In WMDRM, if a license sets permission for "Allow Copy" to true and permission "Allow Play" to true without any common condition, then the license can be decomposed into two instances of right. One grants permission to copy with unlimited condition, while another grants permission to play with unlimited condition [WMDRM_SDK, 2004]. In OMA DRM2, if a rights object does not have top level constraint, the rights object can be decomposed into five instances of right. Each decomposed instance grants permission to one of the permissions to "play", to "execute", to "display", and to "print" [OMA_REL_2, 2006].

Decisions making is based on the criteria proposed in the previous section. "To play" is the only permission type that is allowed to be exported between the two systems. Therefore, any shared condition will not appear in the group of accepted instances of right. This simplifies the implementation of decisions making since we can reject the instance if it has an instance of permission with a permission type other than "to play". Since WMDRM 10 does not support the permission "to export", WMDRM cannot enforce rights exporting based on a certain condition in the same way as OMA DRM 2. Therefore, the OMA DRM 2 system should not export a right to WMDRM if WMDRM 10 allows exporting rights to other systems without proper control. This cannot be expressed by our model. A separate agreement is needed between systems based on OMA DRM 2 and WMDRM 10.

Right adaptation is also influenced by the fact that only "to play" is accepted for rights exporting. Condition division does not need to be considered, since only one type of permission is allowed to be exported. Permission reduction will be used in most cases. We can optimise the process by setting the method as priority 0 in Table 2. Therefore, the priorities for each method in Table 2 should be re-filled with consideration of the system specific features between the two concerned systems.

In summation, we noticed that the process we defined provides a logical method to solve the decision-making issues for rights exporting from one system to another. By using the process we defined, the decision making issues for rights exporting can be modulised, visualised, and simplified. It centralises the effort in exporting rights between DRM systems. However, we also noticed shortcomings in the proposal. The proposal

stays at a high architectural level that does not offer details at the design level for implementation. A certain amount of analysing effort is needed when the terminology in different DRM systems does not conform. Also, the concept of the model we established attempts to solve the problem in general. When dealing with system specific features, the model lacks control and needs to be optimised.

8. Conclusions and Future Work

We will review what we have done during this study. First we recognised the problem domain and analysed the requirement. Then, based upon the problem domain, we established a model to assist the study. We subsequently proposed a solution by using the model. Finally we verified our proposal by performing a case study.

The contribution we have achieved includes new findings in rights and rights exporting, rights modelling, methods in rights decomposition and rights adaptation, and in the process of decisions making in rights exporting.

First, based on the features of the problem domain, we identified many new aspects of rights and rights exporting. We noted two kinds of paths existing in the linkage between permission and conditions and defined them as the valid-checking path and the influential path. We also noticed the basic structure elements within rights and defined them as multiple conditions and shared conditions. With respect to rights exporting, we identified three aspects of rights flow in rights exporting and defined them as modes of rights flow, direction of rights flow, and results of rights flow. In order to achieve better results from rights exporting, we introduced the concept of rights decomposition.

Second, we established a model to illustrate right, which is extensible and flexible when dealing with DRM interoperability issues in general. The model itself is original, visualised, and simplified compared with RELs as common rights expression methods. The model can be re-used for further studies on DRM interoperability.

Third, we have identified methods that can be used in order to achieve better results from rights exporting. These are methods for rights decomposition and rights adaptation. Rights decomposition includes two-level decomposition. Normal decomposition is suitable for all situations, while further equivalent decomposition requires a stateless condition at the critical spot on the valid-checking path. Methods for rights adaptation include condition pre-enforcement, permission reduction, condition division, and condition merge.

Fourth, we also proposed a process of decisions making in rights exporting. The process contains modulised steps and their logical relationship. By following the process, the desired results can be expected from rights exporting. The process is the framework of the whole solution we proposed and it combines our contributions together. We did not find any similar processes provided by other authors during the literature review.

The study offers a starting point to further the research in the area of decisions making in rights exporting. We have identified the following areas that require further consideration and study:

- How to deal with system specific attributes

- How to handle other types of linkage: validity-checking path only, influential path only
- How to prioritise permission granted to one user for the same content

From the case study we noticed that the system specific attributes may require extra efforts in terminology mapping, criteria definition, and implementation. Also, the current amount of optimization to our proposed process can be expected from the system specific features between two concerned systems. It implies that there could be room to extend our process in order to meet the requirement for system specific features.

The validity-checking path and influential path are not well discussed in this thesis. However, as it is one of the attributes for linkage between permission and condition, further discussion on them would be valuable in order to utilise our rights model for further study, even though there are just a few use cases of them at the moment.

Since DRM systems have their own logics to choose which instance of permission to use when multiple instances of permission are granted to the end user for the same content, the order in which instances of permission are used might be changed when rights are exported to the target system. The change influences the usage of the content to some extent. Therefore, in order to control the order in which instances of permission are used, permissions need to be prioritised. In this thesis, we did not discuss this and have left it for further work.

Recently, some standards of DRM interoperability have been initialised, such as Coral, IDP-2 [DRM, 2006-12]. We may expect a unified interoperability solution in this feature. However, as long as DRM systems are not conforming to each other, studies on decisions making in rights exporting will continue.

References

[DPRL, 1998] Digital Property Rights Language, Manual and Tutorial - XML Edition, Version 2.00, November 13, 1998, http://xml.coverpages.org/DPRLmanual-XML2.html. (Checked 15.5.2005)

[DRM, 2006-1] 2005 Year in Review: DRM Standards, January 2, 2006, http://www.drmwatch.com/standards/article.php/3574511. (Checked 11.6.2006)

[DRM, 2006-12] 2006 Year in Review: DRM Standards, December 27, 2006, http://www.drmwatch.com/standards/article.php/3651126. (Checked 24.1.2007)

[DRM-Enabled Content Services, 2007] 2006 Year in Review: DRM-Enabled Content Services, December 27, 2006, http://www.drmwatch.com/ocr/article.php/3651116 (Checked 24.1.2007)

[EEurope, 2005] Europe's Information Society, Digital Rights Management http://europa.eu.int/information_society/eeurope/2005/all_about/digital_rights_man/index_en.htm (Checked 28.3.2006)

[Felten, 2005] Felten, E.W. DRM and public policy, *Communications of the ACM archive,* 48(7): July 2005.

[Heileman and Jamkhedkar, 2005] Heileman, G.L. and Jamkhedar, P.A. DRM interoperability analysis from the perspective of a layered framework. *Proceedings of the 5th ACM workshop on digital rights management,* Architectures: 17-26 (2005)

[Helberger et al., 2004] Helberger N. (ed.), Dufft, N., van Gompel, S., Kerényi, K., Krings, B., Lambers, R., Orwat C., Riehm, U. *Digital Rights Management and Consumer Acceptability: A Multi-Disciplinary Discussion of Consumer Concerns and Expectations*, State-of-the-Art Report, INDICARE, December 2004.

[Iannella, 2001-2] Iannella, R. Digital Rights Management. *Presented at: SOCCI Mini Conference*, Adelaide, Australia, 7-8 February 2001.

[Iannella, 2001-6] Iannella, R. Digital Rights Management (DRM) Architectures. *D-lib Magazine,* 7(6): June 2001.

[Koenen et al., 2004] Koenen, R.H., Lacy, J., MacKay, M., and Mitchell S. The long march to interoperable digital rights management. *Proceedings of the IEEE*, 92:883-897, 2004.

[Liu et al., 2003] Liu,Q., Safavi-Naini, R., and Sheppard, N.P. Digital rights management for content distribution. *Proceedings of the Australasian information security workshop conference on ACSW frontiers*, 21:49-58, 2003

[ODRL, 2002] Open Digital Rights Language 1.1 Specification, August 8, 2002, http://www.odrl.net/1.1/ODRL-11.pdf. (Checked 15.5.2005)

[OMA, 2002] Open Mobile Alliance website, http://www.openmobilealliance.org/about_OMA/index.html. (Checked 11.6.2006)

[OMA_DRM_1, 2004] OMA DRM 1.0, June 11, 2004, http://www.openmobilealliance.org/release_program/docs/DRM/V1_0-20040625-A/OMA-Download-DRM-V1_0-20040615-A.pdf. (Checked 15.5.2005)

[OMA_DRM_2, 2006] OMA Digital Rights Management V2.0 Approved Enabler, March 3, 2006, http://www.openmobilealliance.org/release_program/drm_v2_0.html. (Checked 11.6.2006)

[OMA_REL_2, 2006] OMA DRM Rights Expression Language approved version 2, March 3, 2006, http://www.openmobilealliance.org/release_program/docs/DRM/V2_0-20060303-A/OMA-TS-DRM-REL-V2_0-20060303-A.pdf. (Checked 11.6.2006)

[Polo et al., 2004] Polo, J., Prados, J., and Delgado, J. Interoperability between ODRL and MPEG-21 REL. *First International ODRL Workshop,* Vienna (Austria), April 2004.

[Safavi-Naini et al., 2004] Safavi-Naini, R., Sheppard, N.P., and Uehara, P. Import/export in digital rights management, *Proceedings of the 4th ACM workshop on Digital rights management*, 99-110, 2004.

[Schmidt et al., 2004] A. U. Schmidt, O. Tafreschi, and R. Wolf. Interoperability challenges for DRM systems, Presented at: *International Workshop for Technology, Economy, Social and Legal Aspects of Virtual Goods,* Ilmenau, Germany, 2004.

[Wang et al., 2002] Wang, X., Lao, G., DeMartini, T., Reddy, H., Nguyen, M. and Valenzuela, E. XrML -- eXtensible rights Markup Language. *Proceedings of the 2002 ACM workshop on XML security,* 3, 71-79, 2002,.

[WMDRM, 2006] Windows Media Digital Rights Management, 2006, http://www.microsoft.com/windows/windowsmedia/forpros/drm/default.mspx (Checked 01.06.2006)

[WMDRM_A, 2004] Architecture of Windows Media Rights Manager, May 2004, http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx (Checked 11.11.2005)

[WMDRM_G, 2004] Microsoft Windows Media DRM 10 in general, May 2004, http://www.microsoft.com/windows/windowsmedia/forpros/drm/default.aspx (Checked 01.4.2005)

[WMDRM_SDK, 2004] Windows Media Rights Manager 10.1 SDK, May 2004, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmrm10/htm/windowsmediarightsmanagersdk.asp (Checked 01.4.2005)

[XMCL, 2001] eXtensible Media Commerce Language, June 19, 2001, http://www.xmcl.org. (Checked 15.5.2005)

[XML, 2001] W3C XML Schema, May 2, 2001, http://www.w3c.org/XML/Schema. (Checked 15.5.2005)

[XrML, 2001] eXtensible rights Markup Language (XrML) 2.0 Specification, November 20, 2001, http://www.xrml.org. (Checked 15.5.2005)

Appendix 1: Snapshot of XrML sample end user license chain

[WMDRM_SDK, 2004]

```
<WORK>
   <OBJECT type="contentType">
      <ID type="contentIdType">VideoGUID</ID>
      <NAME>Training Video</NAME>
   </OBJECT>
   <METADATA>
      <OWNER>
         <OBJECT>
            <ID type="Windows" />
            <NAME>webmaster@contosoUSA.com</NAME>
         </OBJECT>
      </OWNER>
      <SKU type="asciitag">skuId</SKU>
   </METADATA>
   <RIGHTSGROUP name="MainRights">
      <RIGHTSLIST>
         <VIEW>
            <CONDITIONLIST>
               <ACCESS>
                  <PRINCIPAL internalid="1">
                     <ENABLINGBITS type="sealedkey">
                     </ENABLINGBITS>
                  </PRINCIPAL>
               </ACCESS>
               <TIME>
                  <RANGETIME>
                     <FROM>20030122T21:59</FROM>
                     <UNTIL>20040122T21:59</UNTIL>
                  </RANGETIME>
               </TIME>
            </CONDITIONLIST>
         </VIEW>
         <RIGHT name="OWNER">
            <CONDITIONLIST>
               <ACCESS>
                  <PRINCIPAL internalid="1">
                     <ENABLINGBITS type="sealedkey">
                     </ENABLINGBITS>
                  </PRINCIPAL>
               </ACCESS>
               <TIME>
                  <RANGETIME>
                     <FROM>20030122T21:59</FROM>
                     <UNTIL>20040122T21:59</UNTIL>
                  </RANGETIME>
               </TIME>
            </CONDITIONLIST>
         </RIGHT>
      </RIGHTSLIST>
   </RIGHTSGROUP>
</WORK>
```

Appendix 2: VBScript example of issuing a license in a license chain

[WMDRM_SDK, 2004]

```
<%
Response.Buffer = True
Response.Expires = 0

' Declare variables and objects.
Dim seed, contentowner_publickey, silent
Dim strLicenseRequested, varHeader
Dim kid, lResult, varKey, sRights
Dim varLicense, LicenseResponse
Dim strRevinfo, ContainsRevinfo, strClientCRLs
Dim ChallengeObj, HeaderObj, KeysObj
Dim RightsObj, LicGenObj, ResponseObj

do

' Replace XXX with your own values. In real practice, you would
' retrieve these values from a database.
seed = "XXX"  ' License key seed used by the packaging server.
contentowner_publickey = "XXX"  ' Public signing key for the
                         ' packaging server.

' Create objects.
Set ChallengeObj = Server.CreateObject("WMRMobjs.WMRMChallenge")
Set HeaderObj = Server.CreateObject("WMRMobjs.WMRMHeader")
Set KeysObj = Server.CreateObject("WMRMobjs.WMRMKeys")
Set RightsObj = Server.CreateObject("WMRMobjs.WMRMRights")
Set LicGenObj = Server.CreateObject("WMRMobjs.WMRMLicGen")
Set ResponseObj = Server.CreateObject("WMRMobjs.WMRMResponse")

' Find out whether the request is for silent or non-silent delivery.
silent = true
if (request.Form("nonsilent") <> "") then
   silent = false
end if

' Put the license request (challenge) into the Challenge object, and then
' extract the content header and client information from it.
strLicenseRequested = Request.Form("challenge")
ChallengeObj.Challenge = strLicenseRequested
varHeader = ChallengeObj.Header

' Check for revocation information.
strRevinfo = ChallengeObj.RevInfo
ContainsRevinfo = ChallengeObj.RevInfoPresent

' Put the content header into the Header object. Using the public key,
' verify that the content header has not been tampered with. The header
' is valid if the result equals 0.
HeaderObj.Header = varHeader
lResult = HeaderObj.Verify(contentowner_publickey)
```

```
if (lResult = 0) then
   ' TODO: Process for a corrupted or modified header.
end if

' Put the required individualization version from the content header
' into the WMRMLicGen object.
indiversion = HeaderObj.IndividualizedVersion
LicGenObj.IndividualizedVersion = indiversion

' Extract the key ID from the content header. Put the key ID and
' license key seed into the Keys object, and then generate the key.
kid = HeaderObj.KeyID
KeysObj.KeyID = kid
KeysObj.Seed = seed
varKey = KeysObj.GenerateKey()

' Get the certificate revocation lists that are supported by the client.
strClientCRLs = LicGenObj.SupportedCRLS

' Set the rights.
RightsObj.MinimumSecurityLevel = 1000
RightsObj.BeginDate = "#20050101Z #"
RightsObj.ExpirationDate =  "#20051231Z #"
RightsObj.AllowBackupRestore = true
RightsObj.AllowCopy = false
RightsObj.AllowTransferToSDMI = false
RightsObj.AllowTransferToNonSDMI = false
RightsObj.DeleteOnClockRollback = false
RightsObj.DisableOnClockRollback = true
SRights = RightsObj.GetAllRights

' Put the license information into the License Generator object.
' Including the following attributes is recommended.
LicGenObj.KeyID = kid
LicGenObj.SetKey "", varKey
LicGenObj.Rights = sRights
LicGenObj.Priority = 10
LicGenObj.Attribute("Copyright") = "copyright statement"
LicGenObj.Attribute("ContentType") = "audio or video"
LicGenObj.Attribute("Author") = "artist name"
LicGenObj.Attribute("ArtistURL") = "http://artist_web_site"
LicGenObj.Attribute("Title") = "title"
LicGenObj.Attribute("LicenseDistributor") = "license issuer"
LicGenObj.Attribute("LicenseDistributorURL") = "http://license_issuer_web_site"
LicGenObj.Attribute("ContentDistributor") = "content distributor"
LicGenObj.Attribute("Rating") = "rating"
LicGenObj.Attribute("Description") = "description"

' Bind the license to the public key, and then generate the license.
LicGenObj.BindToPubKey = contentowner_publickey
varLicense = LicGenObj.GetLicenseToDeliver()

' Use the Response object to deliver the license. If the client does
' not allow silent license delivery, display a page (Silent_ns.asp)
' saying that a license has been delivered.
```

```
call ResponseObj.AddLicense("2.0.0.0", varLicense)
call ResponseObj.AddRevocationData(strRevinfo, strClientCRLs, ContainsRevinfo)

if (silent = true) then
   LicenseResponse = ResponseObj.GetLicenseResponse()
   Response.Write LicenseResponse
else
'  ResponseObj.ReplaceQuotesWith = """""""    ' For VBScript
  ResponseObj.ReplaceQuotesWith = "\"""     ' For JavaScript
  LicenseResponse = ResponseObj.GetLicenseResponse()
%>
<!-- #include file="Silent_ns.asp" -->
<%
end if%>
```

Appendix 3: Properties of a WMRMRights Object in WMDRM version 10

[WMDRM_SDK, 2004]

Property	Description
AllowBackupRestore	Specifies and retrieves a Boolean value that indicates whether the license permits backup and restoration.
AllowBurnToCD	Specifies and retrieves a Boolean value that indicates whether the license permits content to be copied to a CD in the Red Book audio format. This right has been deprecated and replaced by AllowPlaylistBurn.
AllowCollaborativePlay	Specifies and retrieves a Boolean value that indicates whether the license permits consumers to play protected content in a collaborative peer-to-peer session.
AllowCopy	Specifies and retrieves a Boolean value that indicates whether the license permits content to be copied.
AllowPlay	Specifies and retrieves a Boolean value that indicates whether the license permits content to be played on a client computer or device. This right has been deprecated. Now, this right is always enabled and cannot be disabled.
AllowPlaylistBurn	Specifies and retrieves a Boolean value that indicates whether the license permits content to be copied to CD as part of a playlist.
AllowPlayOnPC	Specifies and retrieves a Boolean value that indicates whether the license permits content to be played on a client computer. This right has been deprecated and the AllowPlay right is always enabled.
AllowSaveStreamProtected	Not currently supported.
AllowTransferToNonSDMI	Specifies and retrieves a Boolean value that indicates whether the license permits content to be transferred to portable devices or portable media that support Portable Device DRM version 1 or Windows Media DRM 10 for Portable Devices, and are not SDMI-compliant.
AllowTransferToSDMI	Specifies and retrieves a Boolean value that indicates whether the license permits content to be transferred to SDMI-compliant portable devices or portable media that support Portable Device DRM version 1 or Windows Media DRM 10 for Portable Devices.
BeginDate	Specifies and retrieves the date before which the license is not valid.
BurnToCDCount	Specifies and retrieves the number of times that content can be copied to a CD. This right is no longer supported. It has been replaced by replaced by MaxPlaylistBurnCount and PlaylistBurnTrackCount.
CopyCount	Specifies and retrieves the number of times that content can be copied.
CopyRestrictions	Specifies and retrieves a string that indicates which technologies can or cannot be used to copy protected content.
DeleteOnClockRollback	Specifies and retrieves a Boolean value that indicates whether a license must be

	deleted if the clock is set to an earlier time.
DisableOnClockRollback	Specifies and retrieves a Boolean value that indicates whether a license must be disabled if the clock is set to an earlier time.
ExcludeApplication	Specifies the application ID of a player that you want to prevent from accessing your protected files.
ExpirationAfterFirstUse	Specifies and retrieves the license expiration in number of hours, beginning when the license is first used.
ExpirationDate	Specifies and retrieves the date after which the license is no longer valid.
ExpirationOnStore	Specifies and retrieves the number of hours that a license is valid, beginning when the license is first stored on the consumer's computer.
GracePeriod	Specifies and retrieves the number of hours during which protected content can be played on a device after its clock becomes unset.
MaxPlaylistBurnCount	Specifies and retrieves the number of times that content can be copied to a CD as part of a particular playlist.
MinimumAppSecurity	Specifies and retrieves the minimum security level that a player must have to use the content. This right has been deprecated and replaced by MinimumSecurityLevel.
MinimumClientSDKSecurity	Specifies and retrieves the minimum security level of the Windows Media Format SDK on which the client application is based.
MinimumSecurityLevel	Specifies and retrieves the minimum security level that a player must have to use the content.
Playcount	Specifies and retrieves the number of times the license permits the content to be played.
PlaylistBurnTrackCount	Specifies and retrieves the total number of times that content can be copied to a CD as part of any playlist.
PlayRestrictions	Specifies and retrieves a string that indicates which technologies can or cannot be used to play protected content.
PMAppSecurity	Specifies and retrieves the security level for content that is being transferred to portable devices or portable media. This right has been deprecated and replaced by MinimumSecurityLevel and MinimumClientSDKSecurity.
PMExpirationDate	Specifies and retrieves the expiration date for a portable license. This right is no longer supported.
PMRights	Specifies and retrieves the rights that govern content use with a portable license. This right is no longer supported.
TransferCount	Specifies and retrieves the number of times the content can be transferred to portable devices or portable media.

Appendix 4: Example rights expressed by ODRL

[OMA_REL_2, 2006]

```
<o-ex:rights
xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
xmlns:oma-dd="http://www.openmobilealliance.com/oma-dd"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
<o-ex:context>
<o-dd:version>2.0</o-dd:version>
<o-dd:uid>RightsObjectID</o-dd:uid>
</o-ex:context>
<o-ex:agreement>
<o-ex:asset o-ex:id="Asset-1">
<o-ex:context>
<o-dd:uid>ContentID1</o-dd:uid>
</o-ex:context>
<o-ex:digest>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>DCFHash</ds:DigestValue>
</o-ex:digest>
<ds:KeyInfo>
<xenc:EncryptedKey>
<ds:KeyInfo>
<ds:RetrievalMethod URI="REKReference"/>
</ds:KeyInfo>
<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
<xenc:CipherData>
<xenc:CipherValue>EncryptedCEK</xenc:CipherValue>
</xenc:CipherData>
</xenc:EncryptedKey>
</ds:KeyInfo>
</o-ex:asset>
<o-ex:permission>
<o-ex:constraint>
<oma-dd:timed-count oma-dd:timer=30>10<oma-dd:timed-count/>
<o-dd:datetime>
<o-dd:start>2004-01-01T00:00:00Z</o-dd:start>
<o-dd:end>2004-04-30T23:59:59Z</o-dd:end>
</o-dd:datetime>
</o-ex:constraint>
<o-dd:play>
<o-ex:constraint>
<oma-dd:timed-count oma-dd:timer=1800>2<oma-dd:timed-count/>
<o-dd:datetime>
<o-dd:start>2004-03-01T00:00:00Z</o-dd:start>
<o-dd:end>2004-06-01T23:59:59Z</o-dd:end>
</o-dd:datetime>
</o-ex:constraint>
<o-dd:play/>
</o-ex:permission>
</o-ex:agreement>
</o-ex:rights>
```

Appendix 5: Descriptions of permission and constraint in OMA DRM 2

[OMA_REL_2, 2006]

Property	Description
Play	The element grants the permission to create a transient representation of audio or video Content.
Display	The element grants the permission to make a transient visible rendering of the Content.
Execute	The element grants permissions over the primitive computing element execute.
Print	The element grants the permission to create a fixed (i.e., static), directly perceivable representation of Content.
Export	The element grants export rights over DRM Content and corresponding Rights Objects.
Count	The element specifies the number of times permission may be granted over an asset.
Timed-count	The semantics of the element are as for the count element with the addition of an optional timer attribute. The timer attribute contains a positive integer value which specifies the number of seconds after which the count state specified by the value of the timed-count element is reduced starting from beginning to render the Content.
Datetime	The element specifies the time range, respectively the time limit, for a containing permission.
Time interval	The element specifies a period of time during which the permissions can be exercised over the DRM Content.
Accumulated time	The element specifies the maximum period of metered usage time during which the rights can be exercised over the DRM Content.
Individual	The element specifies the individual to which content is bound.
System	The element specifies the target systems to which DRM Content and Rights Objects can be exported.