Challenges in Modeling Evolving Access Control Policies using Feature Modeling

K.Shantha Kumari
Research Scholar, Department of Banking Technology, Pondicherry University, India
Email: shanthajayakumar@gmail.com

T.Chithralekha
Associate Professor, Department of Computer Science, Pondicherry University, India
Email: tchitu@yahoo.com

***Abstract* — With the growth of Enterprises and organizations, the paper-based systems are replaced with software systems. These software systems are built to support a multitude of users with a variety of roles accessing the resources from anywhere and at any time. These operations are regulated through proper definition of Access control policies (Permissions); this plays a major role in protecting the system and its resources. Initially the software developers focused solely on the customer's requirements without concentrating on access control policies [1]. The later inclusion of them in the software system always created problems that resulted in financial loss, data loss and integrity loss of critical systems [2]. The significance of the Access control policies has made the researchers to recommend its adoption in the early phases of the software development. Unlike olden days, today's business processes are evolving day by day. The Access control policies also continually evolve to meet the organization's business needs and customer's interest. This issue is serious because if the evolving Access control policies are not handled properly, the system is continuously vulnerable to data loss, financial loss and integrity loss. The existing works in the literature rarely address the approaches for handling the evolving Access control policies [3]. New abstraction and approaches are needed to represent such policies specific during the software design.**

This paper discusses research directions that could result in approaches for handling the evolving access control policies in the design phase. This should also ensure the early inclusion of the access control policies at design phase.

***Index Terms*— Access control policies, Model based approaches, Evolutions, Feature modeling**

I. MOTIVATION

In the recent years, large scale software systems are progressively being placed as indispensable essentials of the government sector and industry. In tandem with this increase, there has been an overwhelmed awareness of security, in particular to the authorized access to the resources related to the systems. Hence the software systems of such organizations are developed with utmost care to handle the security issues in an efficient way. In multi-user information systems, certain resources are open to everyone and certain resources are for restricted usage.

These requirements are modeled using an Access control model like Role based access control model that defines the same as Access control policies [ACPs]. An Access control policy defines the (high-level) rules according to which access control to the resources must be regulated. An ACP may express conditions that must be satisfied before an access request can be granted.

Given the magnitude and complexity of the software systems, the design of the ACPs that protect the software systems and information resources also becomes an increasingly complex and difficult problem. From simple Access control lists [ACLs]; the ACPs of today's software systems are totally complicated and spread over the entire functionality of the system. The simple ACLs are added to the software system after the implementation phase. Though it is suitable for those simpler policies, it is not an advisable solution for today's complex software critical systems.

We observe that the ACPs are determined by the corresponding Functional requirements of the software system [4]. The ACPs may have an impact on multiple Functionalities of the system. There might be some scenario that may lead to inconsistency between the ACPs and the functionalities [5]. This issue should be considered as a serious issue as this makes the system vulnerable to the threats. The effect of the ACPs on the Functional requirements is not taken into account in the traditional software engineering approach as they always recommend the later inclusion of the ACPs. And this would lead to poor design and further security failures, violations of the access control rules, leakage of vital information etc. [2]. With respect to the complexity and pervasiveness of today's software systems, this kind of late & ad-hoc inclusion of ACPs might not be completely satisfactory.

Many researchers proposed that the ACPs should be considered during the early analysis and design phases of the software development process to increase the overall system security. From the design perspective, access control policies provide an insight into the various kinds of threats, violations which could be handled effectively

doi:10.4304/jsw.9.5.1089-1094

in further software development process. The overall system development process is fruitful when the design phase supports modeling of ACPs with the functional requirements.

Based on this viewpoint, one group of researchers [6], [7] and [8] analyzed the ACPs by externalizing them from the application domain. This approach provided an additional advantage – the changes to policies can be performed without the need to modify applications. Since the ACPs are separated from applications, it can be rebuilt, shared, and thus reused. But the independent specification of security policies presents a problem—how to integrate the policies in an application design.

Following this, the research community proposed for integrated modelling of the ACPs along with the functional requirements from the initial phase of system development. This helped to deal with complex ACPs and also avoid many inconsistency issues. Addressing the ACPs in the earlier phases is one of today's challenges in software and requirements engineering research since they cannot be blindly inserted into the system design. The ACPs have to be conceptualized / modeled with suitable abstractions and then have to be validated prior to their inclusion.

Multitudes of research works for this purpose are available in the literature. The researchers propose many solutions to address the ACPs in the design phase using various abstraction mechanisms. The following diagram presents the classification of the most prominent approaches in the area of research-Modeling ACPs. This classification covers almost all the approaches that are available in the literature.

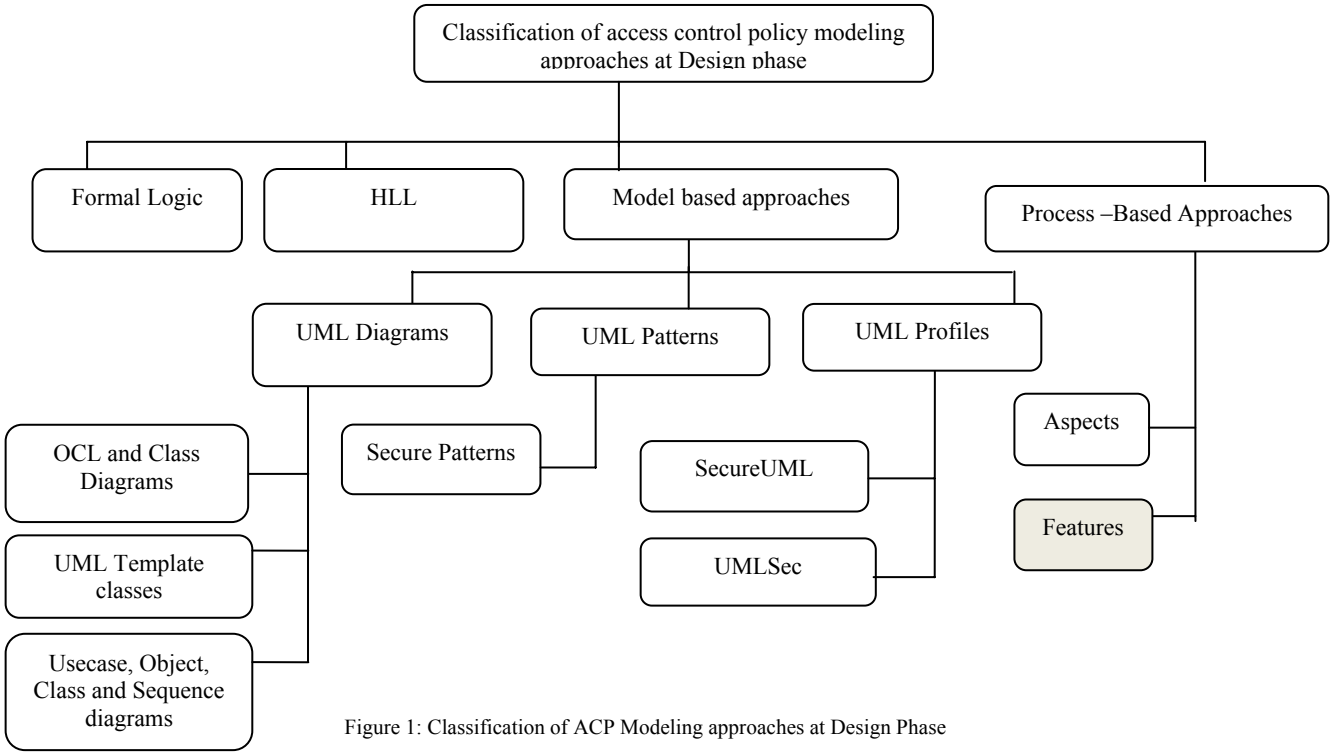

Figure 1: Classification of ACP Modeling approaches at Design Phase

These approaches provide effective modeling abstractions for ACPs in the design phase. Also certain approaches specify systematic approach for the composition of the ACPs with the Functional requirements [9]. Still there is a need for enhancements in these approaches to handle the modeling of evolving access control policies at the design phase.

From the initial literature review [3], we were able to find the lacking factors of the existing methodologies towards addressing the evolution in ACPs. From that we identified certain requirements that should be fulfilled by a modeling approach in order to handle the evolving ACPs at design phase. The following table lists the identified requirements to be fulfilled by a modeling approach, so that it is suitable for handling the evolving ACPs. We analyzed the abilities & limitations of the existing approaches based on the identified requirements.

The shortcomings of the approaches give the motivation for further research that has to be performed for handling the evolving issues in the ACPs.

TABLE 1
REQUIREMENTS FOR AN ACP MODELING APPROACH TO SUPPORT EVOLUTION- COMPARATIVE ANALYSIS

Requirements to be fulfilled by an ACP modeling approach	*Formal Methods* [10][11] [12][13] [14] [15] [16]	*High Level Languages* [17] [18] [19]	*UML Diagram & Profiles* [20] [21] [22][23] [24]	*Aspects* [25] [26]	*Features* [27] [28] [29]
Simplicity	×	✓	✓	∗	✓
Amenable for analysis	✓	×	✓	✓	✓
Usability	×	✓	✓	✓	✓
Understandability	∗	✓	✓	✓	✓
Clarity in syntax & Semantics	∗	∗	∗	∗	∗
Ability to model the crosscutting ACPs	×	×	×	✓	✓
Ability to foresee the likely evolutionary changes in the ACPs	×	×	×	×	∗
Ability to model the multiple occurrences of an ACP in the design.	×	×	×	×	✓
Ability to support the scalability issues in the ACPs	×	×	×	×	∗
Ability to present the commonalities and differences between multiple & related ACPs	×	×	×	×	✓
Ability to represent the real-time constraints & dependencies between ACPs	×	×	∗	×	∗
✓ - Yes ; × - No ; ∗ - Partial					

The above table presented the abilities of the existing ACP modeling approaches during the design phase and also highlighted their limitations with certain issues like ACP Evolution, Scalability and Consistent performance during ACP Evolution. In the following section, these issues are detailed to be used as pointers for future directions of research.

II DIRECTIONS FOR FUTURE RESEARCH

Our preliminary research and studies show that the ACPs evolution or the change occurs either periodically or irregularly. We need an approach to model the evolution suitably at the design phase. We begin by defining the ACP evolution and identifying a suitable abstraction. Later we present the kind of improvements needed to be attributed to the identified abstraction so that it satisfies the requirements listed in Table 1.

A. ACP Evolution

The Research Institute in Software Evolution (RISE) formally defines *software evolution as the set of activities, both technical and managerial, that ensures that software continues to meet organizational and business objectives in a cost effective way.*

When the software continues to evolve, the associated ACPs also evolve. The Evolution in ACPs may be involved with the addition or removal of roles, rules, operations or objects with respect to the system. E.g. Policy 1 – *Manager can read only the customer's accounts* is now changed into Policy 2 – *Manager can perform both read & write access to the customer's account.* Here 'Manager' is the role and the 'Customer's account' is the object. The access operations are 'read' & 'write'. By providing new permissions to the role "Manager", we need to check for conflicts in the existing setup and also assure the enforcement of the new policy in the model.

B. Abstraction for Modeling ACP and its Evolution

An abstraction that models the complex ACPs and also its evolution effectively without affecting the consistency has to be identified. An evolution can be an incremental & planned change or a sudden event. When an evolution occurs, the evolved policy can be treated as a variant of the previous version of the same policy. The evolution component acts as the variance between the two policies. So the abstraction should be able to represent the variability of an ACP in terms of its entities involved viz. Role, resource / object and access permission of the role of that object/resource is required.

From our literature survey [3], we observed that compared to other abstractions, "Feature" is quite suitable for representing the ACPs as it treats ACP as an externally visible and significant characteristic added to the system. Also its significance as a variability modelling tool makes it to be more suitable for our purpose. By representing the ACP as an external visible Feature and arranging them in a Feature Model, we can get twofold advantages.

- Feature model is regarded as an efficient domain analysis tool. It helps in the analysis of the entire set

of ACPs governing the software system of a business through common features and variable features. By this virtue, whenever a new ACP arises, it can be analyzed with the existing Features. There by the evolutionary process can be easily represented.

- Feature model is used to reduce the gap between the problem space [system specifications established during domain analysis and requirement engineering phases] and the solution space [architecture, design and implementation phases] by managing the variability. This simplifies the management of evolution both in the software systems and in the ACPs [30]. This ensures the consistency of the system after enforcing the evolved ACPs.

After choosing Feature to be ours abstraction, we have to research for further improvements on the Feature Model in order to facilitate the modelling of ACP evolution because the existing works on Features [27, 28, 29] does not address the same. We brief the same in the following sub-sections.

1. ACP Evolution Analysis

We recommend using mathematical models to study the changes due Role evolution, Permissions re-assignment & Resource-set Modification. This would facilitate in detailed analysis of dependencies between Roles, Permissions & Objects during the ACP evolution. Also this kind of formal studies would ensure the consistency in the system design.

2. Separate Analysis of Variability decomposition & Functional decomposition:

A Feature can undergo decomposition in two ways – variability and functional decomposition. Functional decomposition represents that Feature "A" [Whole] is decomposed into Feature "B" & "C". Variability decomposition represents Feature "A" can have variants Feature "A.1", Feature "A.2". This is explained with the following example.

The following diagram shows an example Feature model taken from [31]. This model mainly focuses on modelling the functional requirements represented as Features decomposed into sub-Features; e.g. a Cellular Phone has the functionality of Display that can be either Normal or Touch Screen. This Functional Decomposition defines the whole-part relationship between Features & Sub Features.

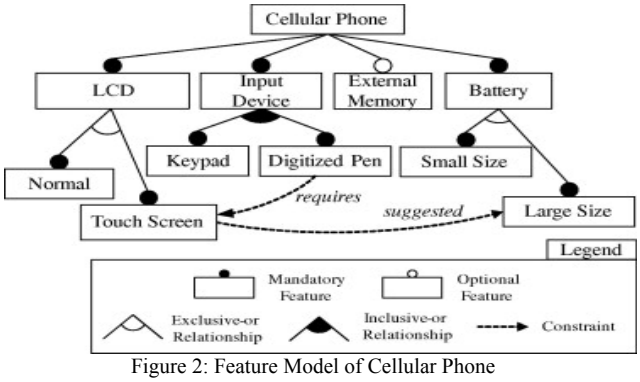

Figure 2: Feature Model of Cellular Phone

Variability is defined as the ability of an entity to evolve into various variants. This decomposition has to be clearly represented as it presents the variants that are needed for evolution. E.g. Functional Feature Touch Screen may further evolve into different variants like Voice enabled Screen or nail touch Screens.

To handle both types of decompositions, research is required to determine how to separately handle the information regarding the functionality & variability, especially during decomposition. This will help in avoiding erroneous modelling of the ACPs and also its evolution [32].

3. Redefinition of Feature without its type

A Feature is always defined with its functional property & also with information that states whether it is mandatory or optional. When a Feature is introduced in the design as a Mandatory Feature, it always remains the same. Its behavior doesn't change in the course of time. After certain evolution, usage of the mandatory feature may not be needed. This has to be recorded in the design also. The existing design approaches doesn't facilitate this change of type information.

We may define ACPs mandatory for accessing certain objects for specific roles; Later those ACPs may become optional for those roles. So we have a hindrance in modelling the ACPs as Features following the standard approaches. Referring to the works of [33], we can research further to define an ACP-Feature without its type.

4. Inclusion of new relationships and constraints

The real time access control policies are very complex and have intricate dependencies. Also the relationships of ACPs with Roles & objects are also not a simple one. Many times, conflicting issues confront the designers.

Since we recommended Feature & Feature model to be our representation, we analyzed the relationships & constraints that are already defined. The existing approach only supports Generalization/Specialization dependency relations and requires/excludes constraints which are not being sufficient to capture all the different types of relationships between ACPs, especially in a situation when ACPs evolve. Hence, research needs to be carried out to identify new relationships and constraints to model the evolving ACPs.

5. Feature Warehouse

Generally, in Policy based access control mechanisms, ACPs are defined and stored in policy repositories. Later they are extracted & applied to check the authorization. Research can be directed to define a similar kind of Feature warehouse that would store the defined ACP Features. A Logical model for Feature storage along with its storing & retrieving methodologies can be defined

6. Scalable & Large Feature models

The ACPs of an organization will scale along with the organization's growth. When these ACPs are represented as Feature models, it would be a large and cramped model. It will lack the required clarity and presentation. The maintenance of Large ACP-Feature models would be

a laborious task. From [34] we found that researchers tried to automate the analysis of large Feature models either using satisfiabililty [SAT] solvers or Binary Decision Diagrams [BDD]. But our objective is to model the scaling ACPs efficiently using Feature models. So that it is possible that even a large Feature model can be split or modularized into small & multiple Feature models to enable their easy maintenance and analysis. But we need to take care of the constraints and interdependencies during splitting. Hence there should be a justifiable factor on which the Feature models can be split up into small Feature models.

According to our objective, we can split feature models based on Roles, or based on objects. The works of [33] suggested considering "Perspectives" to handle large & scalable Feature models. The work in [35] explained that business or legal or technical concerns may drive to reduce the overall Feature model to a representative subset for efficiently testing the complete Feature model. And [36] suggested Fragmenting the changing Features alone and thereby handling the large feature models. These works give motivation for further research works in this direction.

CONCLUSION

In this position paper, we have presented the issues in modelling the evolving ACPs using the existing modelling techniques. We have also listed the pointers for future work that are required to be done in order to handle the issues pertaining to evolving ACPs efficiently.

REFERENCES

[1] Premkumar T. Devanbu and Stuart Stubblebine, "Software engineering for security: a roadmap," in *Proceedings of the Conference on The Future of Software Engineering (ICSE '00)*, 2000, pp. 227-239.

[2] G Georg, I Ray, and R France, "Using aspects to design a secure system," *In Proceedings of the International Conference on Engineering Complex Computing Systems (ICECCS 2002), Greenbelt, MD, ACM Press.*, 2002.

[3] K ShanthaKumari and T Chithralekha, "A Comparative Analysis of Access Control Policy Modeling Approaches," *International Journa lof Secure Software Engineering*, pp. 65-83, 2012.

[4] Qingfeng He and Annie I. Antón, "Deriving Access Control Policies from Requirements Specifications and Database Designs," 2004.

[5] Romuald Thion and Stéphane Coulondre, "Integration of access control in information systems : From role engineering to implementation," *Informatica*, vol. 30, pp. 87-95, 2006.

[6] Jerome H. Saltzer and Michael D. Schroeder, "The Protection of Information in Computer Systems," in *Proceedings of the IEEE*, vol. 9, 1975, pp. 1278-1308.

[7] Stefan Savage Brian N. Bershad, Przemyslaw Pardyak, David Becker, Marc Fiuczynski, and Emin Gün Sirer, "Protection Is A Software Issue," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, Orcas Island, Washington, 1995.

[8] Robert Grimm and Brian Bershad, "Separating Access Control Policy, Enforcement and Functionality in Extensible Systems," *ACM Transactions on Computer Systems*, pp. 36-70, 2001.

[9] Eunjee Song et al., "Verifiable composition of access control features and applications," in *Proceedings of 10th ACM Symposium on Access Control Models and Technologies (SACMAT 2005)*, 2005.

[10] S Barker, "Security Policy Specification in Logic," in *International Conference on Artificial Intelligence (ICAI2000)*, Las Vegas, Nevada, USA., 2000.

[11] Steve Barker and Arnon Rosenthal, "Flexible security policies in SQL," in *Proceedings of the fifteenth annual working conference on Database and application security*, Niagara, Ontario, Canada, 2001, pp. 167-180.

[12] Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari, "TRBAC: a temporal role-based access control model," in *RBAC '00 Proceedings of the fifth ACM workshop on Role-based access control*, Berlin, Germany, 2000, pp. 21–30.

[13] Fang Chen and Ravi S. Sandhu, "Constraints for role-based access control," in *RBAC '95 Proceedings of the first ACM Workshop on Role-based access control* , 1995, p. Article No. 14.

[14] R Hayton, J Bacon, and K Moody, "Access Control in an Open Distributed Environment," in *IEEE Symposium on Security and Privacy*, 1998, pp. 3-14.

[15] R Ortalo, "A Flexible Method for Information Systems Security Policy Specification," in *Proceedings of the 5th European Symposium on Research in Computer Security*, Louvain-la-Neuve, Belgium, 1998.

[16] Michael Hitchens and Vijay Varadharajan, "Tower: A Language for Role-Based Access Control," in *POLICY '01 Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, Bristol, U.K. , 2001, pp. 88 - 106.

[17] James A Hoagland, Raju Pandey, and Karl N Levitt, "Security Policy Specification Using a Graphical Approach," 1998.

[18] OASIS. (2002) http://www.oasis-open.org/committees/xacml.

[19] C Ribeiro, A Zuquete, and P. Ferreira, "SPL: An Access Control Language for Security Policies with Complex Constraints.," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA., 2001.

[20] D Kim, I Ray, R France, and N Li, "Modeling Role-Based Access Control Using Parameterized UML Models," in *FASE 2004. LNCS, vol. 2984*, 2004, pp. 180–193.

[21] Torsten Priebe, Eduardo B. Fernandez, Jens I. Mehlau, and Günther Pernul, "A PATTERN SYSTEM FOR ACCESS CONTROL," in *Proceedings of Eighteenth Annual Conference on Data and Applications Security*, Catalonia, Spain, 2004, pp. 235-249.

[22] I Ray, N Li, R. B France, and D. K Kim, "Using UML to visualize role-based access control constraints," in *Proceedings of the Symposium on Access Control Models and Technologies (SACMAT)*, 2004, pp. 31-40.

[23] T. Lodderstedt, D. Basin, and J. Dose, "Secureuml: A uml-based modeling language for model-driven security," in *Proceedings of the International Conference on the Unified Modeling Language, UML'2002*, 2002, pp. 426-441.

[24] Jan Jürjens, "UMLsec: Extending UML for Secure Systems Development.," in *UML '02 Proceedings of the 5th International Conference on The Unified Modeling Language*, Germany, 2002, pp. 412-425.

[25] Robert E.Filman, Tzilla Elrad, Siobhan Clarke, and Mehmet Aksit, *Aspect-Oriented Software Development*.: Addison-Wesley Professiona, 2004.

[26] Geri Georg, Robert France, and Indrakshi Ray, "An Aspect-Based Approach to Modeling Security Concerns," in *Proceedings of the Workshop on Critical Systems Development with UML*, Dresden, Germany, 2002.

[27] S Kim, D K Kim, L Lu, S Kim, and S Park, "A Feature-Based Approach for Modeling Role-Based Access Control Systems," *Journal of Systems and SoftwareVol. 84, No. 12*, pp. 2035-2052., 2011.

[28] D K Kim, L Lu, and S Kim,".A verifiable modeling approach to configurable role-based access control," in *In the proceedings of FASE 2010. LNCS, vol. 6013,* 2010, pp. 188–202.

[29] L Sun and G. Huang, "Modeling Access Control Requirements in Feature Model," in *Software Engineering Conference, APSEC '09,Asia-Pacific*, 2009, pp. 241 - 248.

[30] Kathrin Berg and Judith Bishop, "Tracing Software Product Line Variability – From Problem to Solution Space," in *Proceedings of SAICSIT 2005*, 2005, pp. 111-120.

[31] Suntae Kim, Dae-Kyoo Kim, Lunjin Lu, and Sooyong Park, "Quality-driven architecture development using architectural tactics," *Journal of Systems and Software*, vol. 82, no. 8, pp. 1211-1231, 2009.

[32] Thomas von der Maßen and Horst Lichter, "Deficiencies in Feature Models," in *Workshop on Software Variability Management for Product Derivation - Towards Tool Support*, 2004.

[33] L Abo Zaid, F Kleinermann, and O De Troyer, "Feature Assembly Framework: Towards Scalable and Reusable Feature Models," in *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems*, Namur, Belgium, 2011, pp. 1-9.

[34] M. Mendonça, "Efficient Reasoning Techniques for Large Scale Feature Models," School of Computer Science, University of Waterloo, PhD 2009.

[35] Julia Schroeter, Malte Lochau, and Tim Winkelmann, "Multi-perspectives on feature models," in *Proceedings of the15th international conference on Model Driven Engineering Languages and Systems (MODELS'12)*, 2012, pp. 252-268.

[36] Andreas Pleuss, Goetz Botterweck, Deepak Dhungana, Andreas Polzer, and Stefan Kowalewski, "Model-driven support for product line evolution on feature level," *Journal of Systems & Software*, vol. 85, no. 10, pp. 2261-2274, 2012.