Observing Access Control Policies Using Scrabble Games

Suzana Ahmad
Faculty of Computer and Mathematical Sciences,
UiTM Shah Alam
Selangor Malaysia.
suzana@tmsk.uitm.edu.my

Siti Zaleha Zainal Abidin
Faculty of Computer and Mathematical Sciences,
UiTM Shah Alam
Selangor Malaysia.
zaleha@tmsk.uitm.edu.my

Nasiroh Omar
Faculty of Computer and Mathematical Sciences
UiTM Shah Alam
Selangor Malaysia
nasiroh@tmsk.uitm.edu.my

Stephan Reiff-Marganiec
Department of Computer Science,
University of Leicester
United Kingdom
srm13@leicester.ac.uk

Abstract—**Access control is concerned with the policies that manage data sharing activities. It is an important aspect of e-service in many application domains such as education, health and business. However, there is limited support in most programming languages and programming environments for implementing access control policies. High-level features, such as access control management policies are usually hard coded by skilled programmers, who are often scarce in many applications such as e-services. In this paper, we present an abstraction of access control management policies in the form of extended scrabble in its rules. The needs of access control policies program construct for supporting this game are examined. A new relevant program constructs are then incorporated into JACIE (Java-based Authoring language for Collaborative Interactive Environments). The usefulness of these program construct are being demonstrated through the extended scrabble.**

Keywords-component; Access Control Policy, Colloborative Environment, Data Sharing

I. INTRODUCTION

Collaborative data sharing application in e-service requires access control policies managed by designated administrators. Access control policies govern among remote users are rare and hardly implemented in existing applications. Implementation of data sharing application usually involves system-level programming interface. It also requires high level features such as carefully formulated access control management policies for shared data. While many software solutions that have been proposed over the years in the context of various applications, these high-level features are rarely supported by software development tools in a coherent manner. It is hardly to find programming language that features language constructs for providing direct support for access control policies manage by the end users.

This research is concerned with access control policies that govern data sharing activities among multiple users or group of users in collaborative environments. The implementation of the policies is often not a trivial task in the development of an application involving data sharing. The provision of access control policies mechanisms is the weaknesses of most existing programming language and development tools.

In this paper, we attempt to identify a collection of useful access control policies that are common in many data sharing applications. We consider an abstraction of various collaborative data sharing application in the form of variation of rules of scrabble game.

We examine the needs in these games for programming access control policies, and propose a comprehensive collection of program construct for supporting these new constructs into JACIE (Java-based Authoring Language for Collaborative designed to support rapid prototyping and implementation of networked collaborative applications)[1].

Although the concept for access control or also known as interest management was first introduced in JACIE, the language construct provided in its version are limited. This became quite apparent when we attempted to implement variation of rules for the enhanced scrabble game.

These variations provided us with an effective means for identifying different type of policies that can exist and the useful parameters for their customization.

With the corresponding application in mind, it is the study of the game that resulted in a major extension to the access control management features in JACIE. The implementation of this game also helped us verify the correctness of the new access control policies introduced in the extension.

The variation of rules for the extended scrabble game are simple enough for us to concentrate on the access control for data sharing requirement of the language and applications. Such access control policies appear in many real applications.

This paper is organized as; Section 2 gives an overview of the related work. Section 3 discuss the rules in extended scrabble. Section 4 reports the implementation and observation of a case study for enhanced policy model. Section 4 deliberates the results of the implementation and observation. Finally, section 5 gives the concluding remarks.

II. RELATED WORK

Unauthorized access is becoming a major concern when dealing with collaborative data [2] within the rapid explosion of information technology and security. Common models for access control are discretionary, mandatory and non-discretionary or role based [3]. These three access models act as elementary guidance for data access control. Combining or extending such models provides adaptable and secure data collaboration which allows data interchange, sharing and dissemination. Discretionary access control (DAC) model is based on object owner's requirement. A system that uses DAC allows object owner to specify whom or which subjects can access any specific object. The most common implementation of DAC is through access control lists (ACL) which are dictated and set by the owners and enforced by the operating system (OS) [4]. UNIX, Linux and Windows are example of OS that uses DAC as an access control [5]. DAC systems will grant or deny the access based on subject's identity.

Implementation of these access control models are done by numerous programming languages. These include general purpose conventional languages such as Java and C#, and scripting languages such as Perl, Python, VBScript and JavaScript. In addition, there are also many domain-specific languages such as Distributed Oz [6] for network transparency, Yoix [7] for handling broadcast messaging, threaded communications, logging, and screen management, and JCell [8] for distributed object and mobile code. However, the interaction management is mostly achieved with the support of the operating system or by designing a specific algorithm.

III RULES IN EXTENDED SCRABBLE

In this section, we first define a set of abstract notations for modeling the rules in extended scrabble, its variations of rules and the corresponding access control policies in later sections. A summary of the various rules is given in section 3 where we highlight their main policies features, linking them with real data sharing applications

The standard scrabble can be generalized in many different ways increasing the number of players, altering the rules governing the game, changing the definition of data sharing and so on. In our generalizations, we choose to give a high degree of freedom to the specification of game rules (management policies), in order to explore a variety of access control policies and cover a very broad range of applications. At the same time, we restrict ourselves to using only additional control policy from the end user which enables us to maintain a reasonable level of abstraction in order to focus on the access control policies in the games rather than on the games themselves.

a. Extended Scrabble Rules

All players are randomly allocated 7 tiles each at the start of a new games. Each player need to form a word by combining two or more of his or her letters and places it on the board for each turn. Upon success of placing of a new word on the board, score will be counted and announced. Player will then have to draw as many new letters that had been played and at the same time they should always keeping seven letters on player's rack – as long as there are enough tiles left in the letter bag.

All letters played on a turn must be placed in one row across or down the board, to form at least one complete word. New words may be formed by adding one or more tiles to the beginning or end of a word already on the board, or to both the beginning and end of that word. No tile may be shifted or replaced after it has been played and scored. Words cannot be spelt nor read backwards. Player may use a turn to exchange all, some or none of the letters with the tiles in the bag.

No restriction on the number of times a player may exchange tiles during a game. However, there must be at least seven tiles remaining in the bag, regardless of the number of tiles being exchanged. Player may also use a turn to exchange tile with other player. Rules for exchanging tile with other player are: Each player will lose a turn for each exchanged tile and only those players who are interested of changing would be able to view the offered tile. If there are more than one player interested in exchanging the tile, the first one who response will be able to offer his/her tile to be exchange (this should be invisible to other players). The initial player who request to exchange has to determine whether to accept the offered tile or not. Exchange will only take place, if both parties agree on the exchange tiles. Each player would not lose their turn if the exchange operation failed. If the initial player who request to exchange declines to accept the tile from the other player, he/she could not make another offer in the same round. The player should either play or just pass his/her turn and wait for another round to offer his/her tile to be exchanged with other player.

Any player may be challenged before the next player starts a turn. If the play challenged is unacceptable, the challenged player takes back his or her tiles and loses that turn. If the play challenged is acceptable, the challenger loses their turn, and points scored. A player can only challenge the previous players play. If a word is challenged, and with the consent of the player who played the invalid word, then that word can be removed, and its points deducted. This effectively makes the player who played the 'invalid' word lose their turn. If the play is not challenged – gone unnoticed and yet seen later and it cannot be challenged two or more moves after the foul. The game ends when all letters have been drawn and one player uses his or her last letter; or when all possible plays have been made. Passing, exchanging or skip turn, is permitted at any time during the game. If each player passes thrice in succession, the game is declared has ended.

b. Variation of Access Control Policies for Extended Scrabble

Based on extended scrabble game rules, four common data sharing activities are being identified. These activities requires access control policies in order for the next action to proceed. Activities identified are: view, update, exchange and challenge. These activities can be conducted by either individual user or player or group which most likely by the admin of the group. Therefore, eight variation of access control policies are derived (refer to table 1) where all the actions will be either deny or allow.

Access Control Policy	Activities / Event	Condition
P1	View ()	Visibility and ownership true
P2	Update ()	Visibility and ownership true
P3	Exchange ()	Visibility and ownership true; agreement between users valid
P4	Challenge ()	Visibility and ownership true; within time constraint
P5	View ()	Visibility and (group) ownership true
P6	Update ()	Visibility and (group) ownership true
P7	Exchange ()	Visibility and (group) ownership true; agreement between users valid
P8	Challenge ()	Visibility and (group) ownership true; within time constraint

Table 1: Variations of Access Control Policies

However, current JACIE could only accommodate access control policy P1 and P2. In order for the implementation of the case study four language constructs for four activities had been derived and embedded into JACIE.

IV IMPLEMENTATION AND OBSERVATION

JACIE had been embedded with new language construct to cater for the variations of access control policies identified. Normally one would aspect a complex set of pre-defined library functions or objects for managing collaborative activities and for interfacing with communication sub-functions. In JACIE, however, for an ordinary programmer, the programming interface to these predefined sub-functions is largely declarative, that is, in the form of protocol specifications. Due to space limitations, the full specification of all language constructs in JACIE are omitted. In the following subsections, we give only the syntactic specification of the language construct for each access control policy. As JACIE is a scripting language, most arguments (or extensions) of a protocol are optional, which facilitate 'fast scripting' for simple and commonly used policies, and the extensibility when introducing new variations and extensions.

P1 and P5 statement

```
view <shared variable> by [<user list>|<group list>]
[to visible |not to visible]
```

P1 and P5 are the most common policies that are being used, these policies will invoke view language construct. In this construct, user can choose to let the data viewed by individuals, groups or both of it.

P2 and P6 statement

```
update <shared variable> by [<user list>|<group list>]
[to own |not to own]
[to read [with password <string>] |[not to read]]
[to write [with write <string>] |[not to write]]
```

Update language construct will triggered P2 or P6 policies or both the policies. This construct will give the user to choose either to let the other parties own and update the data or not to give permission to do any amendments.

P3 and P7 statement

```
exchange <shared variable> by [<user list>|<group list>]
check <conditional sv-expression> <statement>
{else check <condition sv-expression> <statement>
[else <statement>] [default <statement>]
[to own |not to own]
```

Exchange language construct is to handle P3 and P7 policies. Unlike previous construct, exchange construct requires an operation to validate the access permissions for all shared variables involved. Shared variables denotes as <sv-expression>. Exchange operations will be performed only if the validations is successful.

P4 and P8 statement

```
challenge <shared variable> by [<user list>|<group list>]
check <conditional sv-expression> <timer><statement>
{else check <condition sv-expression><timer> <statement>
[else <statement>] [default <statement>]
[to own |not to own]
```

Similar to exchange construct, challenge construct also need to perform validation in order to complete operation. Though this construct do need another parameter to be true which is time. Challenge construct with successful validation will revoke back specified sharing operation to previous state.

With JACIE which had been extended language construct a scrabble application game had been developed as a case study. Focus of this case study is to analyze all access control policies that could be done by the player or group of player

These four actions of the main activities will be the main event to trigger the invocation of the access control policies. Access control policies together with the conditions will determine whether an action could be carried out or not.

This game could be played by two to four players on a square board with a 15-by-15 grid of cells. Each of the cells accommodates a single letter tile.

bag : 100 tiles : B ($T_{[1-100]}$)
Brd : 15 x 15
Four players : P = { P_w , P_x , P_y , P_z} , ƥ ∈ P
Each player – 7 tiles : ƥ $_n$ [T_1 – T_7]
1 round – each player got a turn : Rd_n = { ƥ $_w$[Rd_n] , ƥ $_x$[Rd_n] , ƥ $_y$[Rd_n] , ƥ $_z$[Rd_n] }
Each turn – player could either view(), update(), exchange() and challenge()

Listed in table below are the examples of activities that run under four main events.
Example of activities are successfully handled by enhanced JACIE.

Event	Example by example
View()	Viewing tiles on the game board ƥ → Brd : { $T_{n,...}$ T_n}
	Viewing player's own tiles ƥ → ƥ {T_1, T_2, T_3, T_4, T_5, T_6, T_7}
Update()	Construct New Word ƥ = {T_1, T_2, T_3, T_4, T_5, T_6, T_7} → { T_2, T_4, T_5} Brd : { T_2, T_4, T_5}. ƥ = {T_1, T_3, T_6, T_7} + {T_n,T_n,T_n} .bag {T_{n-3}}
	Update existing Word ƥ = {T_1, T_2, T_3, T_4, T_5, T_6, T_7} → { T_4} Brd : { $T_{n...+}$ T_4}. ƥ = {T_1, T_2, T_3 T_5 T_6, T_7} + {T_n}.bag {T_{n-1}}
Exchange()	Exchange tile/s with tile/s in bag ƥ = {T_1, T_2, T_3, T_4, T_5, T_6, T_7} → { T_2, T_4, T_5} bag {T_{n-3}+ T_2, T_4, T_5}. ƥ = {T_1, T_3, T_6, T_7} + {T_n,T_n,T_n}
	Exchange tile with other player P_x = {T_{x1}, T_{x2}, T_{x3}, T_{x4}, T_{x5}, T_{x6}, T_{x7}}. P_y = {T_{y1}, T_{y2}, T_{y3}, T_{y4}, T_{y5}, T_{y6}, T_{y7}} P_x = { T_{x5} } exchange with P_y = { T_{y3}} → P_x = {T_{x1}, T_{x2}, T_{x3}, T_{x4}, T_{y3}, T_{x6}, T_{x7}}. P_y = {T_{y1}, T_{y2}, T_{x5}, T_{y4}, T_{y5}, T_{y6}, T_{y7}}
Challenge()	Challenge opponent to withdraw his/her tiles (update word/ new word) on the game board ƥ$_x$ = {T_1, T_2, T_3, T_4, T_5, T_6, T_7} → { T_2, T_4, T_5} Brd : { T_2, T_4, T_5}. ƥ = {T_1, T_3, T_6, T_7} + {T_n,T_n,T_n} .bag {T_{n-3}} ƥ$_x$ challenge that the word is not correct, if challenge is **acceptable:** Brd : { T_2 T_4 T_5}. ƥ = {T_1 T_3, T_6, T_7}+ {T_n,T_n,T_n} .bag {T_{n-3}} → { T_2, T_4, T_5} Brd : { T_{n-3} }. ƥ = {T_1, T_2, T_3, T_4, T_5, T_6, T_7} .bag {T_{n+3}}

Table 2: Example of activities are successfully handled by enhanced JACIE.

V CONCLUSIONS

In this paper, we have built a discussion around variations of access control policies to complement an extended scrabble, which serve as an 'abstract' collection of data sharing activities, and enable us to focus on the policies, rather than the context specific details in the applications.
Based on the formal notations and consideration of extended scrabble developed, we have developed a collection of access control policies and have incorporated them into JACIE, a scripting language purposely designed for prototyping networked collaborative applications. Our main contribution in this aspect is the adventurous attempt in providing language constructs for specifying a variety of access control policies. The implementation of such functionality typically requires the skills of experienced network programmers.

REFERENCES

[1] Abidin S. Z. Z. (2006). Interaction and Interest Management in a Scripting Language. Ph.D. Thesis. University of Wales, Swansea; UK.

[2] Ahmad S., Abidin S.Z.Z. and Omar N., "Data Sharing in Networked Environments: Organization, Platforms and Issues", Proceeding WSEAS 2011, pp 207-213.

[3] Samarati, P. and Vimercati, C. S., "Access Control: Policies, Models, and Mechanisms", Lecture Notes in Computer Science Vol 2171. 2001. Pp 137-196 2001

[4] Sandhu, R.S.; Coyne, E.J.; Feinstein, H.L.; Youman, C.E., "Role-based access control models," *Computer* , vol.29, no.2, pp.38,47, Feb 1996

[5] Bhatti R., Bertino E. and Ghafoor A., "A Trust-Based Context-Aware Access Control Model for Web-Services", Distributed and Parallel Databases July 2005, vol 18, Issue 1, pp 83-105 2005

[6] Roy PV, Haridi S, Brand P, Smolka G, Mehl M, Scheidhauer R, Mobile objects in distributed oz. ACM Trans. Programming Language Systems (TOPLAS) 1997;19(5):804-51

[7] Drechsler RL, Mocenigo JM, The Yoix scripting language as a tool for building web-based systems. In:Gregori E. Cherkasova L, Cugola G, Panzieri F, Picco G, editors, Web engineering and peer-to-peer computing: NETWORKING 2002, Lecture notes in computer science, vol. 2376. Berlin and Heidelberg, Pisa, Italy: Springer; 2002. P. 90-103

[8] Rinat R, Smith S, Modular internet programming with cells. In: Magnusson B, ediotr, ECOOP 2002 –object oriented programming: 16th european conference, Lecture notes in computer science, vol. 2374. Berlin and Hedelberg, Malaga, Spain: Springer: 2002. P. 257 - 80