CSCW: Designing Abstract Scripting Language for Role Transition Management

Zainura Idrus

Interactive Computing and Communication Technology (ICCT)
Faculty of Computer and Mathematical Sciences,
Universiti Teknologi MARA, Malaysia.
zainura@tmsk.uitm.edu.my

Zanariah Idrus

Faculty of Computer and Mathematical Sciences,

Universiti Teknologi MARA, Malaysia.

zanaidrus@kedah.uitm.edu.my

Siti Nurbaya Ismail

Faculty of Computer and Mathematical Sciences

Universiti Teknologi MARA, Malaysia.

sitinurbaya@kedah.uitm.edu.my

Siti Z.Z.Abidin

Advanced Analytics Engineering Center (AAEC)

Faculty of Computer and Mathematical Sciences

Universiti Teknologi MARA, Malaysia

sitizaleha533@salam.uitm.edu.my

N.Omar

Interactive Computing and Communication Technology (ICCT)

Faculty of Computer and Mathematical Sciences

Universiti Teknologi MARA, Malaysia

nasiroh@tmsk.uitm.edu.my

ABSTRACT

The nature of tasks and users in computer support collaborative work (CSCW) are dynamic. They usually last for a short period of time and their existence is unpredictable partly due to dynamic data. The uncertainty is suitable to be managed via role-based management where a virtual entity called role is used as a medium between users and tasks. Tasks are assigned directly to roles and not users. Any changes of user will have a minimum impact to role structure. Therefore, users are independent from tasks. The changes of role between users are called role transition and such transition must be flexible to cater for dynamic environment. Thus, the system must support fast development and maintenance. Even though flexibility can be in various forms, this paper supports flexibility through scripting language where the focus is on language abstraction. To achieve the objective, flexibility is explored and viewed from two perspectives which are system and programming language to identify their behaviours via content analysis methods. Since domain specific language is uniquely identified by its abstraction, it is explored in detail. However, most abstraction languages are not abstract enough. Thus, this research has introduced a new model of abstraction. The model introduces a new concept of plug-in and multiple instructions per-statement to increase language abstraction. The knowledge gained is used to design a new set of language constructs through language abstraction which is flexible via plug-in and multiple instructions per-statement mechanisms. Indeed, the mechanisms are very much needed in supporting role transition management in rapid prototyping of CSCW systems.

Key Words: CSCW, role transition, fast programming, abstract programming language

INTRODUCTION

In computer support collaborative work (CSCW), users work together from remote locations virtually. The natures of their works are dynamic and usually last for a short period of time. Thus, collaboration systems must respond to the new requirements. Not only works are dynamic, their existences are also unpredictable and ad-hoc [1][2].

In addition, users are also dynamic and need to be managed. One of the management methods is role-based where works are distributed to roles and not to users directly. Users are responsible to the works only when they play the roles. Any changes to users will not affect the works arrangements. When user change role, role transition occurs and it requires management. Operating in dynamic environment, such applications system must be flexible. They must support fast development and maintenance to cater for frequently changing requirement.

There are many methods to achieve the flexibility such as through tools, system design and development. Unfortunately, tools are limited in term of their capabilities and adaptability to dynamic environment due to lacking of preconceived functionality [3]. Similarly, even though system design and development methods have proven successful rate, they require a lot of logic coding and mainly to be used by expert programmers. Thus, it is costly. Some methods are simple like mashup but could not support comprehensive system development [4] such as role transition management. Therefore, there is a need for a programming language to be flexible as such it can support fast development and maintenance of such system. Hence, at the end of the paper, a new method for flexible language constructs to manage role transition will be designed. The language is flexible in a sense that it can enhance fast system development and maintenance.

The paper is structured with Section 2 highlights the background of the study where the focus is on flexible system as well as flexible programming language. The discussion continues with highly abstraction notation. In Section 3, a new model of abstraction is introduced which supports flexibility through plug-in and multiple instructions per-statement concept. Lastly, in Section 4 we apply the new model by developing a prototype of a call center before we conclude this research in Section 5.

BACKGROUND STUDIES

This section views flexibility through two main perspectives which are system and programming language. The section also discusses various means on how abstraction supports flexibility.

Flexible Systems

A system can be defined as flexible if it has the ability to support incomplete process and continue to support the remaining of the processes during run time [1]. A system can also be defined as flexible if it has the ability to perform on various platforms with minor changes. For instance, Skype, with more 299 million users, run on personal computer, Windows and as well as other platforms such as iOS, Linux, Mac, Window phone, Android, Xbox, Kindle and TV [5].

Others define flexible systems as robust and not easily crash by a single functionality failure (Li, Cheng, Makar, & Mitra, 2013). The flexible systems recover quickly from the failure through two steps which are first error detecting and then self-repairing [6], [7]. In addition, a system is labeled as flexible if it can be fast and easily developed, modified and maintained to meet new requirements [8].

Fast development and maintenance can be achieved through reusable components. The components must be independent from each other, loosely and dynamically bounded among them. As a result, the reusable components offers easy mixing and matching. The components should also support extending and augmenting to generate new components with less afford [9]–[12]. Similarly, the components library should be easily extended to perform more functionality without changing the structure of the base language [13].

Some of the methods used to achieve flexible and maintainable systems are through good system development methods such as readable code, clarity of instruction and easier comprehension of system structure [14], [15]. Another method to support system flexibility is through programming languages.

Flexible Programming Languages

General programming language is defined as flexible if it can support vast problems in wide domain areas. The programming languages fall under this category are C, C#, Ruby, Python and Java. As an example, Java could support standalone systems, networked systems, web applications and applets for internet browser [16]. Moreover, the system developed by Java can be deployed to personal computer, mobile phones, IPod, GPS, and auto components such as air conditional and washing machine [17], [18]. Since general programming languages provide powerful controls, they are usually complex and demand extensive learning, thus they are often left to skilled and professional developers [19].

In addition, flexible means portable where the language supports multiple platforms and not bounded by one platform. This is true for Java and Scripting for Collaborative Online Learning (S-COL) [20]. Moreover, a programming language is categorized as flexible if the language notations are highly abstract. The highly abstract languages have proven to speed up system development and maintenance thus will be explored in depth.

Highly Abstract Notation In Programming Languages

Highly abstract notations hide the detail of coding form users. Example of such language is Unified Modeling Language (UML). As the name implies, programming is about building models. Then, coding is automatically generated based on the models. Since most of modeling languages are based on OOP theory, they involve complicated concepts like relationship between components and inheritance.

Abstraction is also synonym in domain specific language (DSL) where it offers high level of abstraction beyond programming. DSL is commonly known as little language or micro-language [21]. As the name imply, the language is created for a specific domain thus it is usually small and highly focused to solve specific problems. Learning and programming with DSL is faster [22] which is the good source for fast system development and maintenance. Some of the widely known DSL are Backus-Naur Form (BNF), Excel Spreadsheets, HTML, LATEX, MATLAB, and SQL Database queries [23]. The strength of DSL partly lies on its abstraction. As a source for language flexibility, abstraction will be explored in detail in the next subsection.

Flexibility Through Abstraction

Abstraction offers concise as well as powerful notations thus improve testability [24]. It always leads to smaller models with fewer lines of codes. Thus, abstraction contributes to less error rate, easier to identify problems, less debugging and fast modification. As a result, it reduces programming time both during development and maintenance. Programming with abstract notations allows developers to focus on designing solution to a problem and not coding [25], [26]. Emphasis is on what a system does rather that how to develop it. This process reduces switching between design and implementation level which in return reduce error rate and increase system quality, accuracy and reliability [23], [27].

The use of abstract notations promotes simplicity. Simplicity offers rapid prototyping of new ideas and does not involve full development efforts. Prototyping is common among researchers and students for the purpose of experimentation, data collection and simulation [28]. Simplicity also contributes to faster, easier and cheaper of system designing, development and maintenance, consequently, increase productivity [29].

Unfortunately, some DSL do not support abstraction up to its full potentials. Moreover, some DSL mimic structure programming syntax and requires declaration for every variable. On the other hand some DSL adopt OOP notion with class declaration and instantiation. Thus, the abstraction level in some cases turns out to be no higher than GPL. It is also common for

DSL to be similar to scripting language for example XML, thus full of tags. The fully abstraction also cannot be achieved especially if DSL is an internal extension from a base language which are commonly GPL. The notations are tied by the base language and not much different from the base syntax. Similarly, some DSL require improvement in term of their rigid syntax and lack of flexibility in term of spacing.

In order to achieve a better language abstraction, this paper proposed abstraction through plug-in method and multiple instructions per-statement concepts. Instruction can be changed by just plug out old statements and plug in new statements. Even though there are multiple instructions in a single line yet they are simple. Prior to that, role transition management has been studied in our previous research where a monopoly type of game has been chosen as the case study.

THE LANGUAGE MODELING

The knowledge gathered on role transition management must be represented by some form of abstract notations. The notation must be as close as possible to domain concepts as well as descriptive and easily identifiable.

Syntax Of The Abstract Language In Backus-Naur Form (BNF)

The language designed is on *ctrTransition* where it is to control user changing role. Below is the design of *ctrTransition* using the new introduced models of plug-in and multiple instructions per-statement. The design is highly abstract.

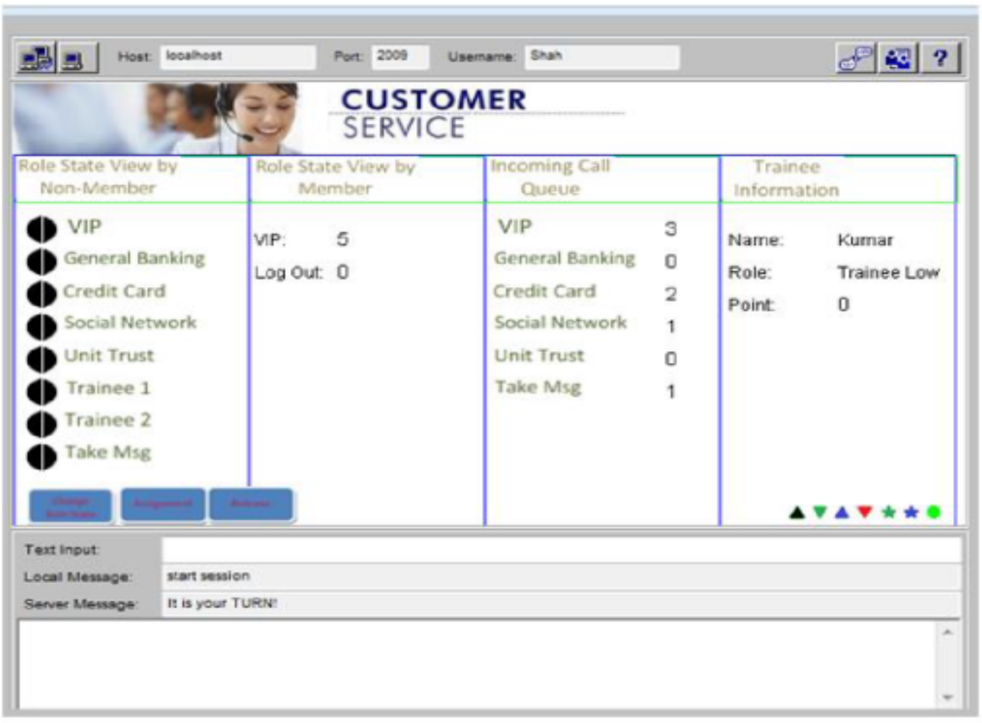
<role transition management>	:=	***ctrTtransition*** { < role transition definition option >}
< role transition definition option >	:=	<role transition definition> |
<role transition definition>	:=	<set role_t > | < role transition definition> , < set role transition > <lock>
< set role transition >	:=	<set state > <set mode> <set role> <set duration condition>
<set state>	:=	**lock** | **unlock** |
<set mode>	:=	**in** | **out** |
<set role>	:=	**role**< select role definition option> |
<set duration condition>	:=	<duration condition>
<duration condition>	:=	**duration**<pair time expression>
<pair time expression>	:=	<integer>

ctrTransition is supported by four parameters which are state, mode, role and duration condition as depicted in Table 1. State and mode are responsible to determine if a role is allowed to receiver a new user or prevent the current user from leaving a role. Both notations come with default function. If state and mode are not stated during development, state will automatically be set to lock while mode is set to both in and out as such the role is lock for both inbound and outbound. Base from the design, a new set of language construct has been developed. The language can support multiple mode of controlling via plug in and plug out concepts.

Table 1 *Ctrtransition* and Its Parameters

State	Mode	Role	Duration Condition
unlock	in	one or more roles	one or more duration
unlock	out	one or more roles	one or more duration
lock	in	one or more roles	one or more duration
lock	out	one or more roles	one or more duration

State and mode can be applied to one or many roles as well as one or many *duration condition*. *Duration condition* determines the duration in which a role must be in its new state. The notation not only controls the role state; it also updates the state in a predefined database. For that purpose, no extra coding is required from programmer. The language constructs are developed by extending JACIE Scripting Language. The language constructs are then used to develop a prototype of a call centre management system as a mean to assess its functionality. Figure 1 is a sample of the prototype.



Host: localhost Port: 2009 Username: Shah

Role State View by Member	
VIP:	5
Log Out:	0

Incoming Call Queue	
VIP	3
General Banking	0
Credit Card	2
Social Network	1
Unit Trust	0
Take Msg	1

Trainee Information	
Name:	Kumar
Role:	Trainee Low
Point:	0

Text Input:
Local Message: start session
Server Message: It is your TURN!

Figure 1 A Prototype of a Call Centre

The sample of the program flow of the role transition management in the call centre is as follows:

```
configuration
{ …   }
roleTransit
{ …
unfreeze in role rGB condition "(q > 5 && dayState == true)",
freeze in role rGB condition "(q <= 5 && dayState == true)",
unfreeze inout role rGB condition "(dayState == false)",
…   }
on NEWMESSAGE //receive message from server
{ …  }
```

The prototype which was developed with the newly developed language has successful control role transition using various modes of controlling and supported multiple instruction per-statements.

In short, with the new set of language construct, flexible system to manage role transition has been achieved. The system is flexible in two directions:

- The controlling of role transition can be realized faster through plug in plug out concept which is part of the language features. Thus, development and maintenance can be achieved faster.
- Automation creating and updating database in a single instruction definitely speed up programming time.

CONCLUSION

In order for a system to survive in computer support collaborative work (CSCW), it must be flexible and easily adapt to new requirements. Role-based management is one of the methods to manage users in such environment. The system must also be flexible to support the consistently changing requirement. This paper supports flexibility of managing role transition through programming language. A new model of abstraction is introduced which is plug-in and multiple instruction per-statement. The model is then implemented in a set of language construct. The language is to control user from entering or leaving a role. In conclusion, the new set of language construct can support system flexibility in two ways which are through the plug-in and multiple instruction per-statement mechanism.

ACKNOWLEDGEMENT

The authors would like to thank Universiti Teknologi MARA, Malaysia for the facilities and financial support under the LESTARI grant 600-IRMI/DANA KCM 5/3/LESTARI (158/2017)

REFERENCES

[1] S. W. Sadiq, M. E. Orlowska, and W. Sadiq, “Specification and validation of process constraints for flexible workflows,” Inf. Syst., vol. 30, pp. 349–378, 2005.

[2] Z. Idrus, S. Z. Z. Abidin, N. Omar, and Z. Idrus, “A Collaborative Space and Its Entities: The Importance of Roles and Relationship of Entities,” in 2016 International Conference on Platform Technology and Service (PlatCon), 2016, no. February, pp. 1–6.

[3] S. Maad, “An Empirical Modelling Approach to Software System Development in Finance: Applications and Prospects,” University of Warwick, 2002.

[4] S. Aghaee and C. Pautasso, “Live mashup tools: Challenges and opportunities,” 2013 1st Int. Work. Live Program. LIVE 2013 - Proc., pp. 1–4, 2013.

[5] M. Swider, “Microsoft highlights 299m skype users, 1.5b halo games playe.” Operating systems, 2013. [Online]. Available: http://www.techradar.com/news/software/operating-systems/xbox-live-upgrade-includes-300-000-servers-600-times-more-than-its-debut-1161749. [Accessed: 15-Aug-2016].

[6] Y. Li, E. Cheng, S. Makar, and S. Mitra, “Self-repair of uncore components in robust system-on-chips: An OpenSPARC T2 case study,” Proc. - Int. Test Conf., pp. 1–10, 2013.

[7] S. Mitra, V. Narayanan, L. Spainhower, and Y. Xie, “Robust system design from unreliable components. In Intl. Symp. on Computer Architecture.,” 2005.

[8] M. B. Rosson, “A flexible programming language for generating stimulus lists for cognitive psychology experiments,” Behav. Res. Methods, Instruments Comput., vol. 20, no. 2, pp. 119–128, 1988.

[9] Q. Boucher, A. Classen, P. Faber, and P. Heymans, “Introducing TVL, a text-based feature modelling language,” in The Fourth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS’10), 2010, pp. 27–29.

[10] H. Fuks, A. B. Raposo, L. P. Magalhães, and I. L. M. Ricarte, “Coordination of collaborative activities : A framework for the definition of tasks interdependencies,” in Groupware, 2001. Proceedings. Seventh International Workshop on IEEE, 2001, pp. 170–179.

[11] C. H. Ganoe, J. P. Somervell, D. C. Neale, P. L. Isenhour, J. M. Carroll, M. B. Rosson, and D. S. Mccrickard, “Classroom BRIDGE : Using collaborative public and desktop timelines to support activity awareness,” vol. 5, no. 2, pp. 21–30, 2003.

[12] R. Patnayakuni and C. P. Ruppel, “A socio-technical approach to improving the systems development process,” Inf. Syst. Front., vol. 12, pp. 219–234, 2010.

[13] G. Karsai, B. Rumpe, M. Schindler, and S. Völkel, “Design guidelines for domain specific languages,” in The 9th OOPSLA Workshop on Domain-specific Modeling, 2009, pp. 7–13.

[14] S. Bartsch, K. Sohr, and C. Bormann, “Supporting agile development of authorization rules for SME applications,” Collab. Comput. Networking, Appl. Work., pp. 461–471, 2009.

[15] I. Jahnke, “How to foster creativity in technology enhanced learning ?” 2011.

[16] S. Imai, Y., Moritoh, Y., & Imai, M. (2014, “Utilization of Java-based web application for educational visualization” in In 2014 International Conference on Education Technologies and Computers (ICETC), IEEE., 2014, pp. 103–109.

[17] M. Masoodian, E. André, M. Kugler, F. Reinhart, B. Rogers, and K. Schlieper, “USEM: A Ubiquitous Smart Energy Management System for Residential Homes,” Int. J. Adv. Intell. Syst., vol. 4, no. 3, pp. 519–532, 2014.

[18] P. Kaur and S. Sharma, “Google Android a mobile platform: A review,” Eng. Comput. Sci. (RAECS), 2014 Recent Adv., no. August, pp. 1–5, 2014.

[19] B. De Alwis, C. Gutwin, and S. Greenberg, “GT / SD : Performance and simplicity in a groupware toolkit,” in Proceedings of the 1st ACM SIGCHI Symposium on Engineering Iteractive Computing Systems, 2009, pp. 265–274.

[20] C. Wecker, K. Stegmann, F. Bernstein, M. J. Huber, G. Kalus, I. Kollar, S. Rathmayer, and F. Fischer, “S-COL: A copernican turn for the development of flexibly reusable collaboration scripts,” Int. J. Comput. Collab. Learn. vol. 5, no. 3, pp. 321–343, Jul. 2010.

[21] J. Bentley, “Programming pearls: little languages” Commun. ACM, vol. 29, no. 8, pp. 711–721, 1986.

[22] M. Mernik, J. A. N. Heering, and A. M. Sloane, “When and how to develop domain-specific languages,” ACM Comput. Surv., vol. 37, no. 4, pp. 316–344, 2005.

[23] T. Kosar, M. Mernik, and J. C. Carver, “Program comprehension of domain-specific and general-purpose languages: Comparison using a family of experiments,” Empir. Softw. Eng., vol. 17, no. 3, pp. 276–304, Aug. 2011.

[24] T. Lodderstedt and D. Basin, “SecureUML : A UML-based modeling language form model-driven security,” in The Unified Modeling Language., 2002, pp. 426–441.

[25] L. M. Bibbo, D. García, and C. Pons, “A domain specific language for the development of collaborative systems,” 2008 Int. Conf. Chil. Comput. Sci. Soc., pp. 3–12, Nov. 2008.

[26] H. Panetto and L. Whitman, “Knowledge engineering for enterprise integration, interoperability and networking: Theory and applications,” Data Knowl. Eng., p. , 2016.

[27] T. Kosar, S. Bohra, and M. Mernik, “Domain-specific languages: A systematic mapping study,” Inf. Softw. Technol., vol. 71, pp. 77–91, 2015.

[28] S. Z. Z. Abidin, M. Chen, and P. W. Grant, “Designing interaction protocols using noughts and crosses type games,” J. Netw. Comput. Appl., vol. 30, no. 2, pp. 586–613, 2007.

[29] M. Jiménez, M. Piattini, and A. Vizcaíno, Challenges and improvements in distributed software development: A systematic. Data structure and software engineering: Challenges and improvements. 2016.