Object-Oriented User Interface Customization: Reduce Complexity and Improve Usability and Adaptation

Le Zhang[1(✉)], Qing-Xing Qu[1,3], Wen-Yu Chao[1], and Vincent G. Duffy[1,2]

[1] School of Industrial Engineering, Purdue University, West Lafayette, USA
zhanl255@purdue.edu

[2] School of Agriculture and Biological Engineering, Purdue University, West Lafayette, USA

[3] Department of Industrial Engineering, Northeastern University, Shenyang, People's Republic of China

Abstract. The purpose of this research is to improve the usability and adaptation of complex information system (CIS) by reducing the complexity. The paper introduces CIS, the complexity of CIS, usability and adaptation issues of CIS, and potential solutions for these issues. Research suggests User Interface (UI) customization can address usability and adaptation issues in CIS. This research proposes the Object-Oriented User Interface Customization (OOUIC) framework to reduce the complexity of CIS, in order to improve the usability and adaptation. The OOUIC approach suggests that classifying users by user roles, e.g., job roles, can reduce the complexity. Use Case Analysis (UCA) can identify actors (job roles) and use cases (goals, tasks, and functions) to develop use case diagrams, tasks diagrams, and function models. Based on model-driven modeling, the mapping between use case diagrams, task diagrams, and function models enables automatic selection of abstract UI and development of concrete UI for each job role. Building connections between vendor UI (V-UI) and concrete UI to generate the adaptable vendor-free UI (MyUI) can ensure the reuse of UI customization on whichever V-UI. The efficiency, robustness, and maintainability of the method had been justified in previous studies. This research proposes a two-phase study by using the product lifecycle management (PLM) system as an example to illustrate that the framework can reduce complexity and improve usability and adaptation.

Keywords: Adaptation · Complex information system · Product lifecycle management · Usability · Use case analysis · User interface customization

1 Introduction

With the growth of computer technology, the information system (IS) becomes a computer-based system which can collect, manage, store, and share information among various users [1]. With current IS technology, multiple users can handle many sets of information in a short time, simultaneously. As the human's demand of managing complex information growing, the IS becomes multifunctional and consists of heterogeneous components. A complex system is defined as any system composed of a great number of heterogeneous entities which may interact with each other to create multiple levels of structure and organization [2]. The heterogeneous entities can be systems, humans, information, and environments elements. Thus, when a IS requires interaction between many heterogeneous systems and users from multiple expertise to manage different sets of information, it can become a Complex Information system (CIS) [2].

As the concept of CIS emerges from recent years, there is not a standard definition of this term. Albers and Still [3] specified four characteristics of CIS: complex work environments, complex information contexts, complex technologies, and complex topics. A complex work environment includes collaborations between multiple users to achieve different goals and often is accompanied by distractions, interruptions, and pauses [4]. This characteristic consists of three aspects: complex users, goals, and environments. The complexity of information presents not only by the variety of information but also by its dynamic feature. The information diversity requires users from different expertise (complex topics) and different technologies or applications to conduct the analysis. Also, the analysis methods, information presentations, and expertise can be varied (complex technologies) due to the dynamic nature of complex information [5]. The complex information overlaps with complex technologies and complex topics. The complex topic is highly associated with users' background and the information [6]. Adapting Albers and Still's definition, this research proposes five complex aspects: user, goal, information, technology, and environment (Fig. 1). This

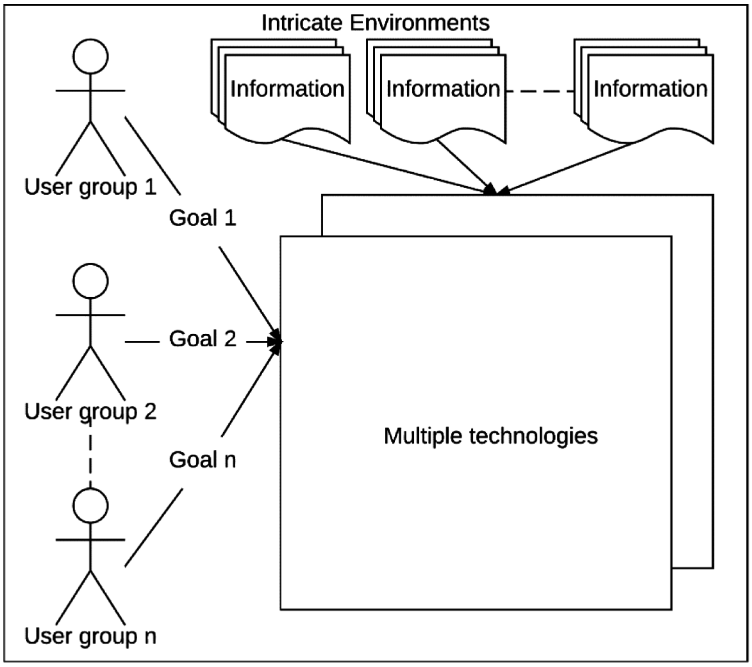

Fig. 1. Five complex aspects of CIS: user, goal, information, technology, and environment (based on Albers and Still's four characteristics of CIS [3])

research defines CIS as any information system which requires collaborations among different users in an intricate environment to use multiple technologies dealing with numerous sets of dynamic information to achieve various goals.

Previous CIS research hasbeen conducted on product lifecycle management systems, software configuration management systems, resource management systems, healthcare systems, and financial systems [5, 7–11]. These systems are enterprise systems implemented in organizations to manage complex information, such as product data, software configuration, customers' information, and financial data. An organization can involve multiple job roles, various business goals, and intricate environments. Thus it can be seen that enterprise systems follow the CIS's definition and five complex aspects. This research focuses on enterprise systems and uses PLM system as an example to illustrate the usability and adaptability of CIS.

CIS brings not only benefits to progressing large sets of information but also a series of usability and adaptation issues to users using its User Interface (UI). As UI is the software unit that can directly interact with users [12], this research focuses on CIS's UI design. Usability is to ensure effectiveness, efficiency, and satisfaction for a specified set of users to achieve a particular set of tasks in a specific environment [13]. The usability issues caused by complexity include redundancy, unorganized layout, and poor learnability. The purpose of developing multifunctional UI is to enable multiple user roles to achieve various goals and manage different types of information within one single UI. However, the low usage of other functions and information by a particular user role results in redundancy [14]. Redundancy can cause misused functions, inefficiency of locating functions and information, and hard to memorize UI features [15]. Even though designers put efforts in organizing many functions and information in a single UI, users still have trouble to understand the multifunctional UI and blame it for causing unsatisfied experience [9]. Moreover, the difficult nature of CIS applications and multifunctionality result in poor learnability. For instance, converting an engineer from using one PLM system to another PLM system (new vendor or upgraded system) can require six months and $20,000 investment [16]. Adaptation refers to the properties of a system that can automatically adapt its behavior and interaction to suit the user's needs, expertise, and requirements [17]. CIS tends to sacrifice the adaptation and use one single UI to integrate multiple functions and information to satisfy various users' needs [18]. Even though there is research proposes using adaptive UI can improve the adaptation of CIS, the complexity results in difficulties to develop self-adjust UI for various users [19].

These usability and adaptation issues all have connections of the complexity of CIS. The purpose of this research is to investigate the complexity of CIS and improve usability and adaptation. The structure of this paper is as follows. This section introduces CIS and its usability and adaptation issues. The next section reviews relevant works in improving usability and adaptation of CIS. After analyzing potential solutions, the third section proposes the Object-Oriented UI Customization (OOUIC) framework to reduce the complexity of CIS, in order to improve the usability and adaptation. The fourth section provides a study to evaluate the OOUIC approach. The last section suggests future works.

2 Relevant Research

The ultimate goal of this research is to improve usability and adaptation of CIS. However, adaptive UI and common User-Centered Design (UCD) methods might not be sufficient to address issues in the CIS's unique atmosphere. In order to identify appropriate methods for CIS, this section reviews UI adaptation, UCD methods, and UI customization approach.

2.1 User Interface Adaptation, Customization, and Personalization

Adaptation is associated with personalization and customization [20]. Similar to adaptive UI, personalization relies on the artificial intelligence and machine learning to predict user preference and tailor the UI to increase its personal relevance to an individual or a category of individuals [21]. Not to mention the faultiness of machine learning technology, identifying complex user requirements in CIS is a great challenge. Relying on UI to do self-adaptive to fit complex customers' needs is impractical.

In contrast, UI customization, which refers to the capability of enabling users to adapt the UI to meet their requirements on specific tasks [22], can be a potential solution to improve both usability and adaptation of CIS. Against static and adaptive UI, UI customization builds an adaptable UI which enables users to control UI design to provide higher perceived efficiency levels, thus greater satisfaction [23]. The research found that UI customization can improve users' perceived ease of use, satisfaction, user experience, willingness to use, and performance [24]. Thus, UI customization is a potential solution to improve both adaptation and usability. The challenge to implement UI customization is to ensure robustness, efficiency, and maintainability in a complex setting [25].

2.2 User-Centered Design

User-Centered Design suggests placing users at the center of the design to allow them to influence the design shape and giving extensive attention to their needs in each stage of design processes [26]. UCD emphasizes user involvement and provides many methods to evaluate and develop usability design, such as questionnaire, interview, focus group, use case, scenario analysis, cognitive walkthrough, and heuristic evaluation [27]. However, research indicates that UCD approach might not be sufficient to identify user requirements in CIS [3].

As potential users of CIS can come from different backgrounds, simply applying questionnaire, interview, and focus group can collect very diverse user requirements. Satisfying different expert requirements in a single UI is hard. Forcing a single UI to include all the needs can cause functional conflicts and redundancy [7]. Another limitation of UCD methods is the development of CIS requires domain expertise. It limits the application of user-free formative methods, e.g., cognitive walkthrough and heuristic evaluation, since designers are not familiar with terminologies and functionalities of the CIS [28]. Applying UCD method in CIS requires reducing the complexity, e.g., separating users into groups to collect requirements and designing for particular user roles. The Use Case Analysis (UCA), which can identify user goals and group requirements by goals, is one UCD method that can handle complexity [29]. Thus, this research suggests using UCA to reduce the complexity of CIS.

2.3 Object-Oriented Design Method

Based on UCA, Lin and Lee [30] proposed the Object-Oriented Analysis (OOA) method to investigate the complex relationship between users' objects and UI elements for the development of CIS. The approach suggests end-users' objects are associated with their requirements and desired system behaviors.

As shown in Fig. 2, the approach classifies user roles (in a PLM system, user role can be an engineer, manager, and customer) and execution platforms (a PLM system can be smart device application, desktop application, web application) and groups them into client units, e.g., customers using an iPhone application. When customizing UI for a specific user in a user role, the user and a platform can be a unit, e.g., a specific engineer called Jack using a desktop application. For each client unit, identify which trigger event enacts users to accomplish a specific use case. A trigger event can be a message or another use case. The use case is a set of scenarios that narratively describe interactions between the user and the system to achieve a specific goal. One client unit can have multiple trigger events and use cases. Based on the use case and the user role's characteristics, designers can build user role profile and functions for this user role, e.g., engineers' profile and engineers' functions. For a specific user in a user role who wants to customize functions, the user's characteristics will be used to generate customization profile, e.g., an engineer called Jack has Jack's profile and Jack's customized functions. All information will be stored in a use case bank for the system to determine which UI should be provided to the user.

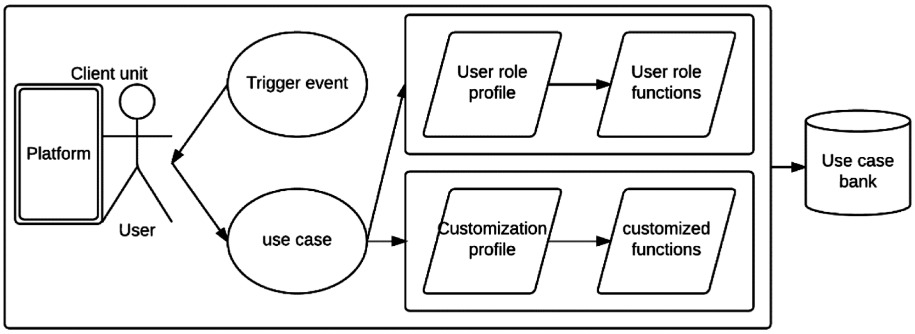

Fig. 2. The framework of object-oriented analysis in UI customization (modified based on Lin and Lee's diagrams [30])

With the use case bank, a system can judge which platform the user is using. Since the widget and size in different platforms are different, the UI should be designed for each individualplatform. Based on user role profile and customization profile, the system judges: what is the user' role, which user role functions should be provided, whether this user has customized functions, and which customized functions should be provided. When a trigger event occurs, the system judges which use case will happen. Based on user role profile and customization profile, the system provides UI for the user to accomplish the use case.

The OOA method can customize UI for each user role, platform, use case, and aspecificuser. The advantage of this approach is that it classifies user requirements into detail groups. For this reason, it can ensure the robustness of identifying complex requirements in CIS. However, customize UI for every individual user is time-consuming and not economical. A more efficient approach is required.

2.4 Model-Driven Design

Model-Driven Development (MDD) proposed automatic UI customization to increase efficiency and consistency when developing variants of an UI [31]. However, purely automatic UI customization relies on unstable machine learning technology and can cause usability issues as adaptive UI does. To avoid this dilemma, Pleuss [32] proposed Semi-Automatic UI Customization (SAUIC) on the basis of Model-Based User Interface Development (MBUID) and Software Product Line (SPL) [33, 34]. The MBUID suggests derivation of an UI from a set of UI models by considering users, platforms, and environments. The SPL's idea is to build a series of templates of an UI to achieve mass production by using templates. Merging two methods, SAUIC inherits efficiency and usability.

The SAUIC process (Fig. 3) has a family of abstract UI templates for one kind of software. Abstract UI is the unit that includes the most basic elements of an UI to achieve a task, e.g., abstract UI of a normal searching function includes an input box, a search button, and a list of search results. One UI can have different abstract UI templates, e.g., search by name, search by ID, advanced search. Then designers define UI configuration of which elements are required for the software, e.g., the UI only needs search by name. After having the specific abstract UI for the software, designers can manually adjust the UI layout for a user role, a platform, and an environment. All

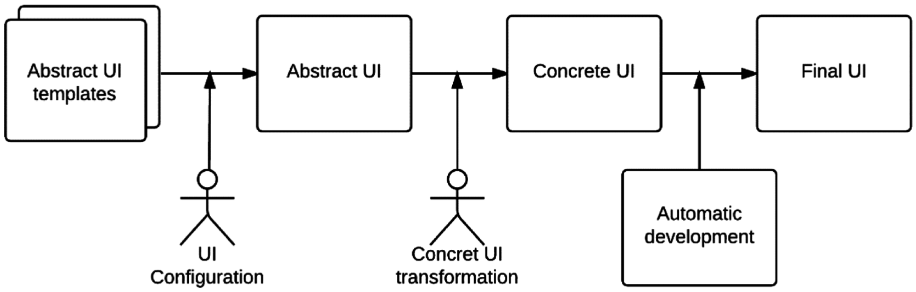

Fig. 3. The framework of semi-automatic UI customization (modified based on Pleuss's Model-Driven Development model [25])

manual adjustments can automatically convert to a concrete UI, which is a preview of the adjusted UI. Last, the concrete UI will automatically transfer to the final UI, which is the UI for users.

Research justified that the SAUIC approach ensures not only the efficiency but also the usability of UI customization [25]. However, the model did not specify a robust method for manual customization. Thus, OOA should be integrated into the SAUIC approach.

2.5 User-Centered User Interface Customization

Implementing OOA in SAUIC can ensure the efficiency and robustness in UI customization of one software. However, switching software and upgrading software are common in an organization. Maintainability is essential for UI customization. The maintainability is to ensure the UI customization can be implemented in a dynamic environment, including changing the software vendor, upgrading software, and switching to another type of software. Wu [35] and his colleagues propose a User-Center UI Customization (UCUIC) to enable the extension of a customized UI to other vendor UIs.

The UCUIC method (Fig. 4) allows users to select desired functions from a vendor UI (V-UI) to customize a MyUI for a specific user. The MyUI also allows the user to adjust the UI configuration. All customized behaviors are stored in a database so that the tool can regenerate the MyUI when the software switches to a new V-UI.

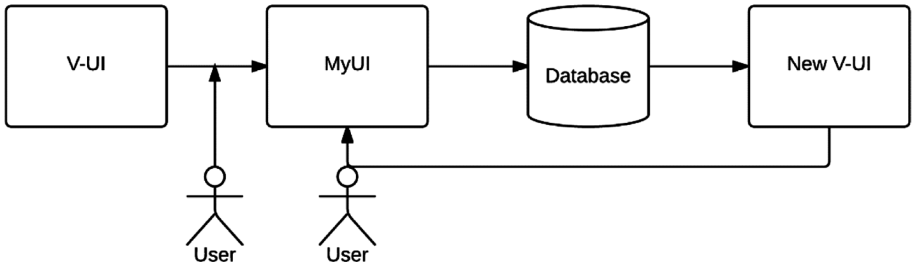

Fig. 4. The framework of user-centered UI customization (modified based on Wu's user interface customization model [35])

3 Objective-Oriented User Interface Customization Framework

This research suggests that UCA can reduce the complexity by grouping users by specific characteristics, e.g., job roles. Also, UI customization can improve the usability and adaptation of CIS. Robustness, efficiency, and maintainability are the keys to the implementation of UI customization in CIS. Thus, this research integrates OOA, SAUIC, and UCUIC into a systematic approach to achieve Objective-Oriented UI Customization (OOUIC) in CIS. This section describes the framework of OOUIC (Fig. 5).

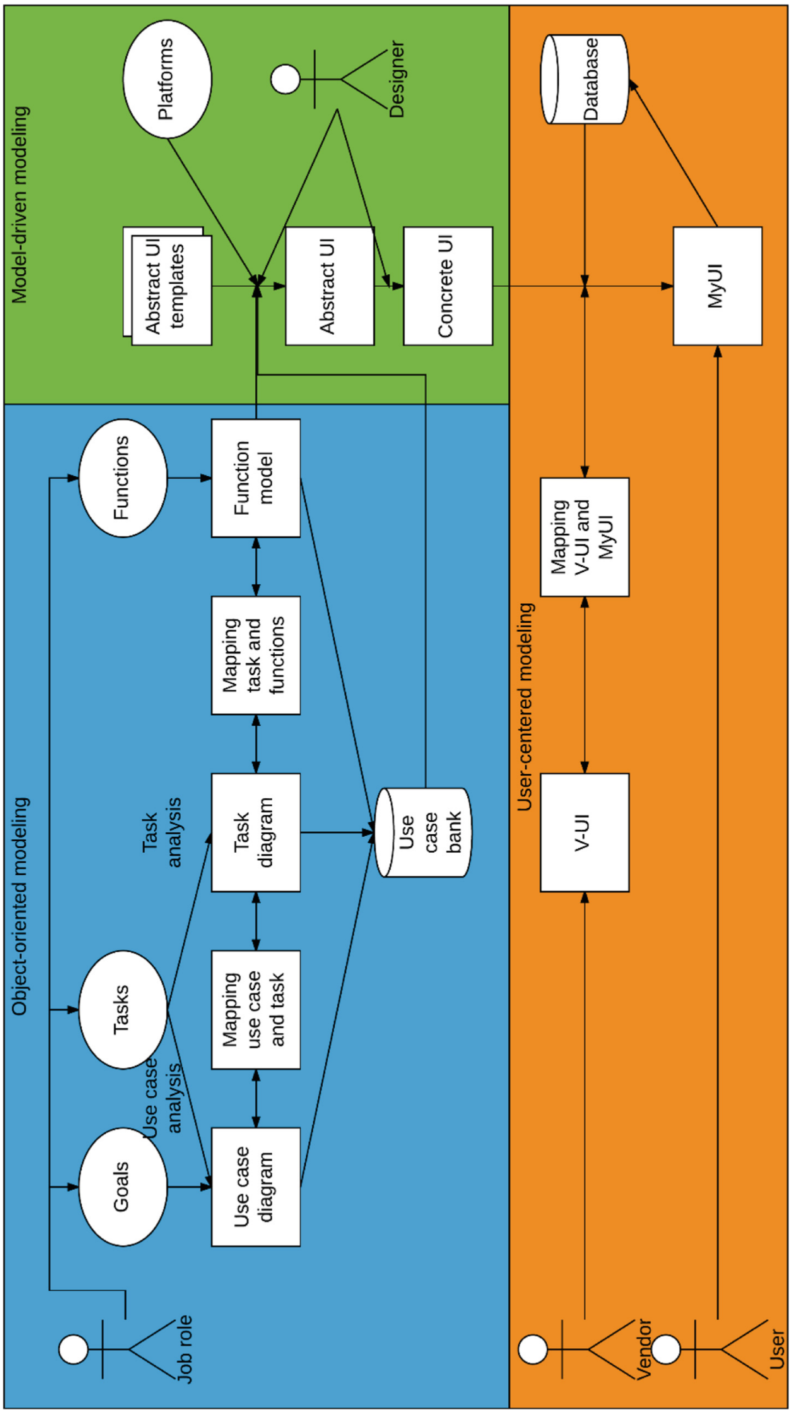

Fig. 5. The framework of objective-oriented UI customization

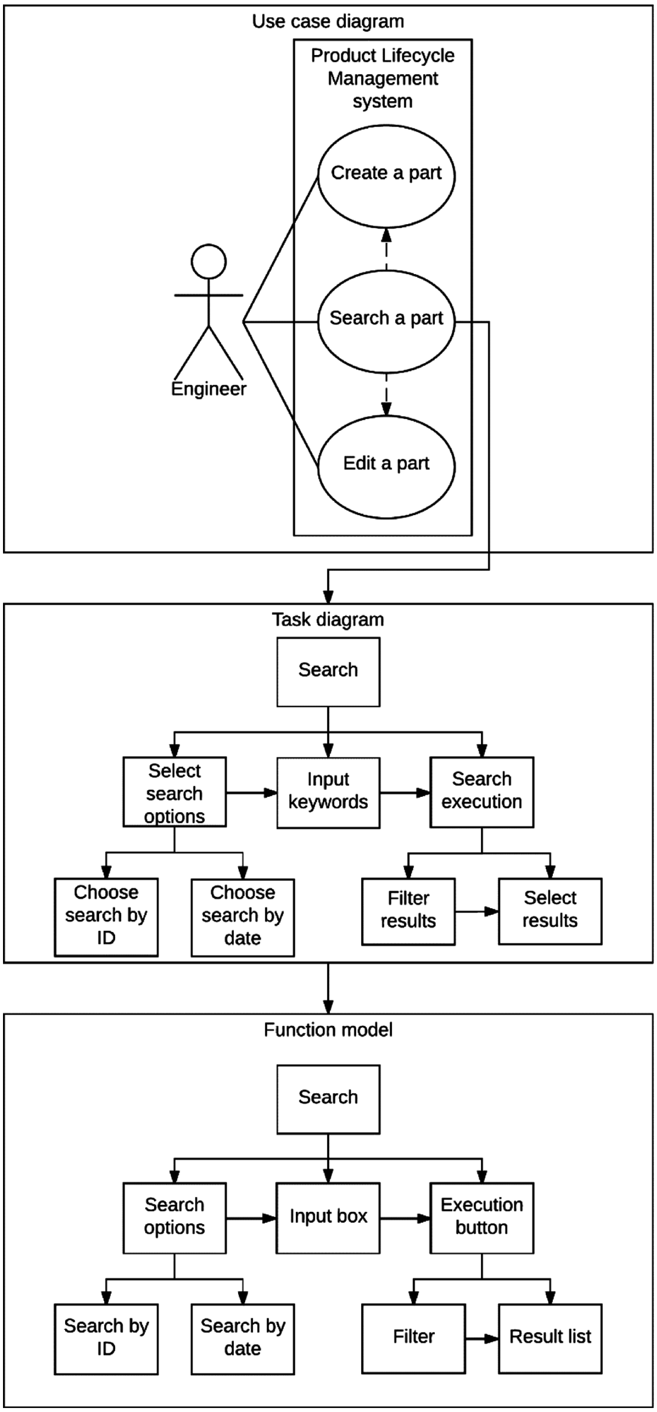

Fig. 6. Use case diagram, task diagram, and function model

3.1 Object-Oriented Modeling

The OOUIC framework includes three phases: Object-Oriented Modeling (OOM), Model-Driven Modeling (MDM), and User-Centered Modeling (UCM). The first phase is OOM. The OOUIC framework can classify users by their job role, e.g., engineers, managers, customers. A job role is a description of what a person does in an organization [36]. Most companies use a job title to name a job role.

UCA will be applied to collect each job role's goals, tasks, and required functions to develop use case diagram, task diagram, and function model (Fig. 6). Developing a use case diagram can identify a job role's goals and the required tasks to achieve these goals. For example, an engineer's goal is to design a part in a PLM system. The engineer needs to create a part, edit the part, and search the part.

A task diagram illustrates all detailed steps to achieve a task. In Fig. 6, the task diagram shows steps to achieve a search task. Vertical arrows connect sub-steps, e.g. to select a search option, a user can choose search by ID or search by name. Horizontal arrows show the sequence of a series of steps, e.g., filter results first, then select the desired result. The reason to have a task diagram is to help designers eliminate unintended steps. For example, a job role does not require search options, the select search options and its sub-steps can be removed from the diagram.

A function model is to map task diagram to a software's functions. Each step in task diagram has a corresponding function in the function model, and the function is described in a way that system can understand. Thus, the system can eliminate the unintended functions from the UI.

There are mappings between use case diagram, task diagram, and function model so that other diagrams can make a corresponding change to the adjustment in one diagram. These diagrams and mappings are stored in the use case bank for designers to create abstract UI.

3.2 Model-Driven Modeling

The next phase is MDM. A group of abstract UI templates is designed for achieving each function in different platforms. Based on the platform and the user's job role, the system will automatically develop abstract UI for the user role. To ensure the usability, designers can manually adjust the UI configuration during the process of creating abstract UI. With the abstract UI, designers can manually adjust UI layout to develop a concrete UI.

Based on SPL, the system can leverage abstract UI templates and use case bank to build concrete UI automatically. All manual designs are not necessary. However, when a new job role or specific requirements come in, manual design can satisfy special needs and improve the usability.

3.3 User-Center Modeling

The UCM phase is to ensure the maintainability of the model. A CIS can have multiple applications, and these applications can change dynamically. For example, in a PLM system, product development engineers need CAD software to design 3D model; managers need ERP software to manage resources; sales need CRM software to communicate with customers. In addition, companies can change software vendors or upgrade software to another version. It is inefficient to redo customization when there is a change in the software.

User-center modeling suggests developing MyUI for each user. A MyUI is converted from the concrete UI. Users can manually customize MyUI. The MyUI's information (concrete UI and user's customization behaviors) is stored in a database. Based on the database, the system can automatically build a mapping between V-UI and MyUI. Thus, the user only needs to use one MyUI; the system can take care of the dynamic changes in V-UIs.

4 Application in PLM System UI Customization

The purpose of building the OOUIC framework is to emerge with an efficient, robust, and maintainable approach to improve the usability and adaptation of CIS. Previous studies had validated the efficiency, robustness, and maintainability of the methods implemented in OOUIC [25, 30, 35]. This research proposes using an observation of PLM system UI customization to investigate whether the approach can improve usability and adaptation. As complexity is a major factor that can influence the usability and adaptation, this research also will assist in determining the relationship between complexity, usability, and adaptation.

4.1 Hypotheses

Hypothesis 1: Comparing to V-UI, the MyUI can significantly reduce the UI complexity.

Hypothesis 2: Comparing to V-UI, the MyUI can significantly improve user performance and perceived usability. They can highlight the relationship with usability and adaptation.

4.2 Methods

The observation includes two phases. The first phase will apply OOUIC method to develop MyUI for the PLM system. As previous research indicates that "fifteen to twenty participants are required to elicit user requirements in Use Case Analysis [37] ", this study will conduct the UCA with 20 participants to develop case diagrams, task diagrams, and function models. Abstract UI templates will be developed based on Siemens Teamcenter and PTC Windchill. Prototypes of MyUI for multiple job roles will be developed by using UCA results and Abstract UI templates.

The second phase is to conduct a scenario analysis to measure and compare the complexity, usability, and adaptation between MyUI and two V-UIs (Siemens

Teamcenter and PTC Windchill). According to UCA, one task will be chosen as the scenario. Previous research suggests having 30 participants can receive statistically significant results [38]. Thus, this study will recruit 30 to 40 PLM users from the manufacturing company to conduct the task on MyUI and two V-UIs. Alemerien's [39] UI complexity metrics (alignment, grouping, size, density, and balance) will be used to measure complexity. The metrics include both objective measures of UI structure and participants' subjective rating. Previous studies indicate that adaptation is associated with user performance and user experience measures, such as task time, errors, and perceived usability [23, 40]. According to Albert's book [38], this study defines task success, task time, and errors as performance measures, and usefulness, ease of use, satisfaction, and ease of learning as self-reported measures (perceived usability). The measures can highlight the relationship with usability and adaptation. ANOVA will compare the complexity, user performance, and perceived usability between MyUI and two V-UIs. The expected result is there is a significant difference between MyUI and V-UIs.

5 Conclusion

In conclusion, UI customization can be an effective way to improve usability and adaptation of CIS. This research proposes the OOUIC framework by combining OOA, SAUIC, and UCUIC as an efficient, robust, and maintainable approach to customize UI for CIS. However, further study is required to investigate the improvement of complexity, usability, and adaptation of MyUI, in the comparison of V-UI.

The complexity of user, goal, information, and technology can be reduced by applying the OOUIC framework for UI customization. However, a complex environment can still cause serious usability issues, such as interruptions, distractions, and unintended pauses. Among these complex environment issues, the interruption can easily cause harmful effects on users' cognitive and physical work process and result in barriers to system adoption [41]. In the future work, the effect of interruptions on MyUI and V-UI should be studied.

Acknowledgement. Authors are grateful to the support of Professor Nathan Hartman (Purdue Polytechnic Institute) and Product Lifecycle Management Center of Excellence at the Purdue Polytechnic Institute. Authors also want to thank master students Shefali Rana and Apoorva Sulakhe (Purdue Industrial Engineering) for providing valuable advice.

References

1. Davis, G.B., Olson, M.H.: Management Information Systems: Conceptual Foundations, Structure, and Development. McGraw-Hill Inc., New York City (1984)
2. Bihanic, D., Polacsek, T.: Models for visualisation of complex information systems. In: Proceedings of the International Conference on Information Visualisation, pp. 130–135. IEEE (2012)

3. Albers, M., Still, B.: Usability of Complex Information Systems: Evaluation of User Interaction. CRC Press, Boca Raton (2010)
4. Spaulding, C.R., Gibson, W.S., Schreurs, S.F., et al.: Systems engineering for complex information systems in a federated, rapid development environment. Johns Hopkins APL Tech. Dig. **29**, 310–326 (2011)
5. Wang, W., Hsieh, P.: Beyond routine: symbolic adoption, extended use, and emergent use of complex information systems in the mandatory organizational context. In: ICIS 2006 Proceedings, vol. 48, pp. 733-750 (2006). http://aisel.aisnet.org/icis2006/48
6. Tiwari, S., Gupta, A.: A systematic literature review of use case specifications research. Inf. Softw. Technol. **67**, 128–158 (2015). doi:10.1016/j.infsof.2015.06.004
7. Mirel, B.: Interaction Design for Complex Problem Solving: Developing Useful and Usable Software. Morgan Kaufmann, Burlington (2004)
8. Levary, R.R.: Computer integrated manufacturing: a complex information system. Prod. Plan. Control **7**, 184–189 (1996). doi:10.1080/09537289608930340
9. Lu, X., Wan, J.: Model Driven Development of Complex User Interface, MDDAUI (2007)
10. Song, X., Li, B.H., Chai, X.: Research on key technologies of complex product virtual prototype lifecycle management (CPVPLM). Simul. Model. Pract. Theory **16**, 387–398 (2008). doi:10.1016/j.simpat.2007.11.008
11. Warman, A.R.: Developing complex information systems: the use of a geometric data structure to aid the specification of a multi-media information environment. London School of Economics and Political Science (United Kingdom). Doctorate dissertation (1990)
12. Laurel, B., Mountford, S.J.: The Art of Human-Computer Interface Design. Addison-Wesley Longman Publishing Co., Inc., Boston (1990)
13. ISO FDIs: ISO 9241-210: 2009. Ergonomics of human system interaction-Part 210: human-centered design for interactive systems. Int. Organ. Stand. (ISO) Switz. (2009)
14. Veneziano, V., Mahmud, I., Khatun, A., Peng, W.W.: Usability analysis of ERP software: education and experience of users' as moderators. In: SKI 2014 - 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA) (2014). doi:10.1109/SKIMA.2014.7083560
15. Allanic, M., Durupt, A., Joliot, M., et al.: Towards a data model for PLM application in bio-medical imaging. In: 10th International Symposium on Tools Methods Competitive Engineering TMCE, pp. 365–376 (2014)
16. Bartholomew, D.: PLM: Boeing's dream, airbus' nightmare. Baseline (2007)
17. Schneider-Hufschmidt, M., Malinowski, U., Kuhme, T.: Adaptive User Interfaces: Principles and Practice. Elsevier Science Inc., Amsterdam (1993)
18. DeLone, W.H., McLean, E.R.: Information systems success: the quest for the dependent variable. Inf. Syst. Res. **3**, 60–95 (1992)
19. Akiki, P.A., Bandara, A.K., Yu, Y.: Adaptive model-driven user interface development systems. ACM Comput. Surv. **47**, 1–33 (2014). doi:10.1145/2597999
20. Blom, J.: Personalization - a taxonomy. Conf. Hum. Factors Comput. Syst. ACM 313–314 (2000). doi:10.1145/633292.633483
21. Fan, H., Poole, M.S.: What is personalization? Perspectives on the design and implementation of personalization in information systems. J. Organ. Comput. Electron. Commer. **16**, 179–202 (2006). doi:10.1207/s15327744joce1603&4_2
22. Rivera, D.: The effect of content customization on learnability and perceived workload. In: CHI 2005 Extended Abstracts on Human Factors in Computing Systems, pp. 1749–1752 (2005). doi:10.1145/1056808.1057013
23. Findlater, L., McGrenere, J.: A comparison of static, adaptive, and adaptable menus. In: Proceedings ACM CHI 2004, vol. 6, pp. 89–96 (2004). doi:10.1145/985692.985704

24. Hui, S.L.T., See, S.L.: Enhancing user experience through customisation of UI design. Procedia Manuf. **3**, 1932–1937 (2015). doi:10.1016/j.promfg.2015.07.237
25. Pleuss, A., Wollny, S., Botterweck, G.: Model-driven development and evolution of customized user interfaces. In: EICS 2013 – Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 13–22 (2013). doi:10.1145/2480296.2480298
26. Abras, C., Maloney-Krichmar, D., Preece, J.: User-centered design. Bainbridge, W. Encycl. Hum.-Comput. Interact. Thousand Oaks Sage Publ. **37**, 445–456 (2004). doi:10.3233/WOR-2010-1109
27. Williams, A.: User-centered design, activity-centered design, and goal-directed design: a review of three methods for designing web applications. In: Proceedings of the 27th ACM International Conference on Design of Communication, pp. 1–8. ACM (2009)
28. Albers, M.J.: Communication of Complex Information: User Goals and Information Needs for Dynamic Web Information. Routledge, Abingdon (2004)
29. Lee, J., Xue, N.: Analyzing user requirements by use cases: a goal-driven approach. IEEE Softw. **16**, 92–101 (1999)
30. Lin, J., Lee, M.-C.: An object-oriented analysis method for customer relationship management information systems. Inf. Softw. Technol. **46**, 433–443 (2004). doi:10.1016/j.infsof.2003.09.001
31. Voelter, M., Groher, I.: Product line implementation using aspect-oriented and model-driven software development. In: Software Product Line Conference 2007, SPLC 2007, 11th International, pp. 233–242 (2007). doi:10.1109/SPLINE.2007.23
32. Pleuss, A., Botterweck, G., Dhungana, D.: Integrating automated product derivation and individual user interface design. In: Proceedings of the 4th International Workshop on Variability Modelling of Software-Intensive Systems, Linz, Austria, 27–29 January 2010, pp. 69–76 (2010)
33. Szekely, P.: Retrospective and challenges for model-based interface development. In: Bodart, F., Vanderdonckt, J. (eds.) Design, Specification and Verification of Interactive Systems 1996. (EUROGRAPH), pp. 1–27. Springer, Vienna (1996). doi:10.1007/978-3-7091-7491-3_1
34. Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns. Addison-Wesley, Boston (2002)
35. Wu, L., Yu, P., Wei, J., et al.: A method for user-centered interface customization and development of a prototype system. In: Proceedings of the 2010 2nd WRI World Congress on Software Engineering (WCSE), vol. 1, pp. 197–200 (2010). doi:10.1109/WCSE.2010.46
36. Sandhu, R.S.: Role-based access control. Adv. Comput. **46**, 237–286 (1998)
37. Adolph, S., Cockburn, A., Bramble, P.: Patterns for Effective Use Cases. Addison-Wesley Longman Publishing Co., Inc., Boston (2002)
38. Albert, W., Tullis, T.: Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics. Newnes, Burlington (2013)
39. Alemerien, K., Magel, K.: GUIEvaluator: a metric-tool for evaluating the complexity of Graphical User Interfaces. In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering SEKE, pp. 13–18 (2014)
40. Van Velsen, L., Van Der Geest, T., Klaassen, R., Steehouder, M.: User-centered evaluation of adaptive and adaptable systems: a literature review. Knowl. Eng. Rev. **23**, 261–281 (2008). doi:10.1017/S0269888908001379
41. Lee, B.C.: Human cognitive performance in healthcare information system environment and its application on nursing tasks. Purdue University. Doctorate dissertation (2013)