Adaptation of User Interface Based on Contextual Feedback

Robertas Damaševičius, Paulius Paškevičius
Department of Software Engineering
Kaunas University of Technology
Kaunas, Lithuania
robertas.damasevicius@ktu.lt

***Abstract*—The spread of social networks and other online collaboration-related practices changes the target of software products from a single user to virtual communities. Such communities view user interfaces of social websites as communication partners (facilitators) rather than mere communication medium. Effective communication requires proper feedback from community-driven systems that create the illusion of active participation and control. Furthermore, feedback combined with community effort and evaluation mechanisms such as crowdvoting can be used as a vehicle for adaptation of interfaces to the requests of community. The implementation of community-driven interfaces requires the extension of existing user interface development architectures and design patterns. In this paper we analyse known user interaction and user interface models and present the contextual feedback based adaptation (CFBA) meta-model, the four-tiered user interface (4TUI) architecture, and the Model-Control-View-Adapter (MCVA) pattern. A case study in the community-driven interface adaptation is presented.**

Keywords—user interface design, feedback modelling, adaptable interface, interface evolution, crowdvoting.

INTRODUCTION[1]

The spectacular rise of social networks and other collaboration-based practices such as crowdsourcing [1] underlined the importance of effective communication in virtual communities. The term "virtual community" refers to a large group of individuals that regularly share and exchange information through computer-mediated mechanisms such as e-mail, weblogs, or forums [2]. Many research studies investigated what motivates people to participate in virtual communities (e.g. see [3]). Out of many contributing factors, the most important ones are common interests (e.g., professional networks such as LinkedIn), status seeking and reputation (e.g., question-and-answer websites such as StackOverflow), and affiliation (e.g., friend networks such as Facebook). All these factors depend upon supporting effective communication between a member of the virtual community and the community represented by its other members and the user interface of the social platform. In virtual communities this communication is mostly computer-mediated, i.e., the user interface of the platform that supports virtual community acts as the „face" of the community or as a partner of conversation. The users are no longer the consumers of media content, but also want to act as producers of content or even co-designers of content delivery platforms [4] to have impact on the face of the virtual community.

The strength of relationships that bind a member to a community can be influenced by the impact a member can make as well as a feedback that a member can receive from a community. The success of a virtual community relies on the voluntary contribution of valuable intellectual property of individuals to a community without explicit compensation [5]. Even if an individual does not receive any explicit reward for his/her contribution, he/she often wants his/her contribution to make impact or at least be seen. Capturing and understanding feedback received from users also is critical in business information systems and customer relationship management [6] as well as in intelligent systems and intelligent user interfaces [7]. This practice has been recognized for long now and content sharing sites such as Flickr or Youtube have been successful very much due to this practice. However, the need to say or show something to a community is paired with a need to obtain answers or feedback from it.

In this paper we analyse feedback models and methods that are aimed to increase the role of feedback in community building and support efforts. To introduce the contextual conditions into user interface evolution process, the contextual model is required that maps contextual requirements from a community of users received through feedback mechanisms to the adaptations of user interfaces. Building user interfaces that dynamically adapt to the context is not new [8-12]. Similar approaches include mediation strategies for integrating the input of multiple crowd workers in real-time [13], the extension of the model-view user interface architecture with an intelligent layer that handles interface events as commands and allows a user to evolve an interface in a way that is entirely independent of applications [14], and the website plug-in that makes use of crowdsourcing to collect context-aware activity data based on which suitable user interface adaptations for different target devices are inferred [15].

Our novelty is the interpretation of the feedback from the community of users of a system as the context of the system's user interface. This interpretation does not contradict the definition of context provided in [16]: "context is any information that can be used to characterize the situation of an entity". Feedback conveys context information (e.g., the

interests, preferences, opinion of user) that has influence over the presentation and functionality of user interface.

We propose the crowdvoting-based contextual feedback meta-model, the four-tier community driven architecture and the extension of the MVC pattern for implementing the community-driven adaptation of user interfaces at the use stage rather than at the design stage (as, e.g., in [17]). We describe its application in the community-driven website user interface project aimed to engage community members in the controlled evolution of website interface.

I. ROLE OF FEEDBACK IN USER COMPUTER INTERACTION

The term 'feedback' originates from the area of cybernetics and refers to the information that a system receives from its environment about the effects or consequences of its actions [18]. In communication, feedback is used for a broad range of responses at various levels of communication. Commonly, feedback is understood as any information about reaction to a product or a person that can be used as a basis for improvement [19]. Allwood et al. [20] claim that feedback is a central functional subsystem of human communication. It consists of methods that allow providing, without interrupting the dialog, information about quality of communication such as ability and willingness to have contact, ability to understand communicated information as well as the emotions and attitudes triggered by the information in the recipient. According to Kotzé [21], feedback has three main elements: 1) response, which serves to confirm that the recipient has received information, 2) modification of behaviour, which reassures the user that his input is relevant and has power to change, and 3) intelligence (see, e.g., social creativity [22], collective intelligence [23] and "wisdom of crowds" [24]) that the opinion or understanding of the community can lead to improved quality of communication and usability of a product.

The main aim of feedback is to induce the change of a software product while the direction of the change itself depends upon the polarity of feedback: positive feedback (agreement) reinforces the change in the same direction; negative feedback (disagreement) causes a change in the opposite direction, while homeostatic feedback maintains equilibrium [25]. In the long term, such change leads to the evolution of a product or its interface and user feedback acts a main driver of such evolution. Software evolution has been recognized as a key issue in software development and use a long time ago: as software application is released for use, the world in which it is situated changes, and therefore new demands constantly arise [26]. Traditionally, software evolution has been dealt with offline using the version-based approach as follows: a version is released, user response is collected, a new (updated, corrected) version is released, and the cycle is repeated again. However, in a modern world of software development, software evolution has an unprecedented speed [27], and feedback can be seen as a means to accommodate and drive the change at the use time.

The understanding of increased importance of feedback mechanisms signifies a shift from consumer cultures (specialized in producing finished artefacts to be consumed passively) to the participation-based cultures in which all people can participate and contribute their solutions [32]. This shift represents a transition from a world in which a small number of experts define rules, create static products, and make decisions for many consumers toward a world in which everyone has interests and opportunities to actively participate in the development of dynamically evolving products [33].

The essential role of feedback in natural communication makes it a crucial issue in the development of human-computer user interfaces [34] where users communicate proactively rather than passively or reactively. An example of the proactive role of the user is so called Split Interfaces, where frequently used functionality is automatically copied to a specially designated adaptive part of the interface [35, 36]. Altered Prominence is another approach to interface adaptation that highlights recently used elements of an interface [37]. Without feedback, a human-computer dialog quickly breaks down while proper feedback can create the illusion of a dialog partner listening [38].

According to [19, 28, 29], in order to be effective, feedback must be 1) persuasive (i.e. influencing future state of community and behaviour of community members), 2) contextual (i.e. include context information by default), and 3) informative (i.e. convey useful information), 4) contributive (i.e. contribute towards benefit of a community as a whole), 5) continual (i.e. to support conversation as narrative of community), 6) expressive (i.e. demonstrate polarity using affective means such as emotions), and 7) effortless (easy to use). In any case, feedback comes as a response to a previous communicative act [30], i.e., in reaction to the status or opinions of a community members or an entire community in order to achieve consensus or alignment [31].

Techniques for collecting user feedback in software systems cover a wide spectrum, ranging from error reporting facilities to the content-related feedback mechanisms of social networks [28]. Examples of such feedback mechanisms are the Facebook "Like" button or the YouTube's thumbs-up/thumbs-down, which allow evaluating content, linking members while require only a minimum amount of effort on the users' side. However, the amount of "likes" and "don't likes" do not have a direct influence how information is presented, i.e., the platform of a virtual community has full control over the presentation while the function of the user is reduced to evaluating other users' content rather than making influence over its presentation. However, if properly implemented feedback could increase affiliation, loyalty and immersion of the community members beyond simple collection of "likes". Examples of such "socially advanced" user interfaces are a crowdsourcing interface that collects user-generated mappings between pairs of web pages [39], an adaptive user interface that is constructed using consensus methods [40], and socially-adaptable interfaces, interfaces that crowdsource the creation of task-specific interface customizations and instantly share them with all users of the application [41]. The development of such socially advanced interfaces requires adequate models of interaction, which we discuss in Section 3.

II. Modeling Interaction and Interface Adaptation

Computers and internet are the media of social interaction in virtual communities. Therefore, the social interaction in virtual communities is mainly guided by the principles of human-computer interaction. When humans interact with computer, they first formulate their goals and then develop a series of steps required to achieve that goal. Such mental model of action is known as Norman's Interaction Cycle (see Fig. 1) [42], which has been used to evaluate the efficiency of a user interface. The model includes both cognitive and physical activities, and includes feedback, which is called "Evaluation" in the model. The Norman's model does not distinguish between the content of the message delivered, its presentation form and its affect (i.e., emotions associated with the message). Therefore, it should be considered only as the simplest approximation of human-computer interaction. Furthermore, the interface in the Norman's model can be interpreted as a metaphor of dialogue between interface designer and its users.

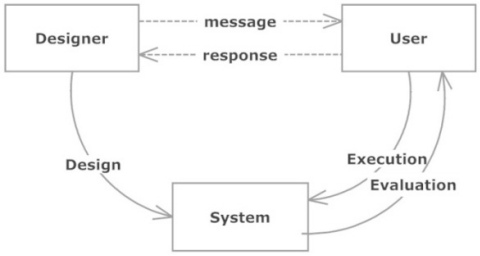

Fig. 1. Interpretation of the Norman's Interaction Cycle [42] as a dialogue between designer and user

The extension of the Norman's mental model is the Isatine model [43] that is also based the Dieterich's taxonomy of user interface adaptation [44]. The model states that three entities are involved in the adaptation of user interface: the user, the interactive system, or any third party. The adaptation is performed as follows: 1. Goals for user interface adaptation are formulated; 2. The user or third party initiates request for adaptation; 3. The adaptation is specified as a sequence of commands issued to the interactive system for execution; 4. The adaptation is applied using the adaptation support mechanisms (e.g., through user interface options, personalization); 5. Transition between the interface before and after adaptation is performed; 6. Information about adaptation is issued to the interested parties; and 7. Adaptation is evaluated. The advantage of the Isatine model is a detailed guideline for performing adaptation of user interfaces.

The Taylor's Layered Protocols (LP) model and its elaboration in [45, 46] are based upon the cognitive principle that humans use superimposed layers of abstraction in perception. From this principle the LP model arrives at the architecture for structuring user-system interaction. The model distinguishes between the system's interpretation of their messages (I-feedback), and information the system expects to receive from the user (E-feedback). The advantage of the model is the classification of feedback types (user's feedback and system's feedback).

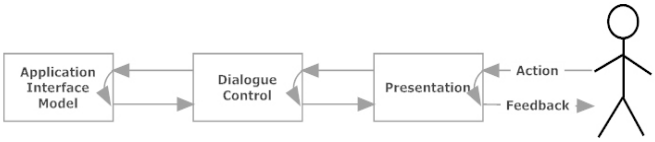

Fig. 2. Seeheim model emphasizing the three different levels of visual feedback

The Seeheim model (see Fig. 2) [47] reveals the linguistic nature of the visual feedback identifying three main software modules (or layers): 1) Dialogue Control module handles the syntactic aspects of the interaction and is responsible for the dynamic behaviour of the system; 2) Application Interface module provides a semantic interpretation of the information received for the dialogue component; 3) Presentation Module handles the lexical aspects of the interaction such in input as well in output and is only aware of the presentation technology. Visual feedback can be formulated at three different levels: lexical (Presentation), syntactical (Dialogue Control) and semantic (Application Interface Model).

The Bezold's model [48] (see Fig. 3) deals with interface adaptation, i.e., the ability of the interface to improve itself for an individual user based on an observation of the user's behaviour. Adaptation to user behaviour comprises two steps: 1) reasoning on the user-system interaction, and 2) adapting the user interface accordingly. The user-system interaction is considered as a linear sequence of basic events, which are emitted by the interactive system. User modelling algorithms extract new knowledge from the user-system interaction and trigger interface adaptations. A semantic layer is introduced as an abstraction of the interactive system that allows implementing reasoning on the user-system interaction. The system-independent logic is defined on the top of the semantic layer. The adaptation layer decides which adaptations can be applied to an interactive system. The advantage of the model is a multi-layered architecture that allows separation of semantic, interaction and adaptation aspects of user interface.

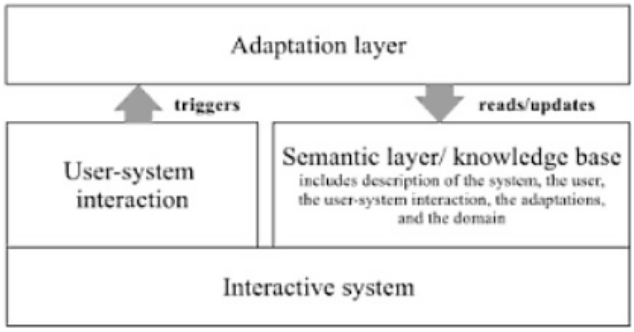

Fig. 3. Bezold's model of interative system adaptation [48]

The Baxley's model of user interface [49] applies the separation of concerns and decomposes user interface into three tiers as follows: Structure (conceptual model, task flow, and organizational model), Behaviour (viewing and navigation, editing and manipulation, user assistance) and Presentation (layout, graphic design style, text). Here, the conceptual model supplies the 'metaphor' that helps users to interact with an application, and the organizational model provides

classification schemes to group and associate application information and interface objects. The model's advantage is a clear separation of the different aspects of user interface.

The RUX (Rich User eXperience) model [50] is used for the systematic adaptation of user interfaces over the existing web applications. The user interface specification is divided into four levels: 1) Concepts and Tasks, 2) Abstract Interface, 3) Concrete Interface and 4) Final Interface. Concepts and Tasks are taken from the underlying web model. Abstract Interface provides a common representation to all devices and interface development platforms without any kind of spatial arrangement or behaviour. Concrete Interface optimizes the presentation of user interface for a specific device or group of devices, and has three Presentation levels: Spatial Presentation allows the spatial arrangement and interface style of to be specified; Temporal Presentation allows the specification of behaviours which require a temporal synchronization; and Interaction Presentation allows modelling the user's behaviour. Final Interface provides code generation of the modelled application. The advantage of the RUX model is a hierarchy of interface entities from most abstract to specific ones, which could be easily mapped to the hierarchy of models according to the model-driven architecture (computation-independent, platform-independent, and platform specific models).

Currently commonly used software interface patterns (such as MVC or PAC) are derived from the Seeheim model. The Presentation–Abstraction–Control (PAC) pattern [51] separates an interactive system into three types of components responsible for specific aspects of the application's functionality. The abstraction component retrieves and processes the data, the presentation component formats the visual and audio presentation of data, and the control component handles things such as the flow of control and communication between the other two components. In the Model–View–Controller (MVC) pattern (Fig. 4), the Model consists of application data and business rules, the Controller acts as mediator like the Dialogue component in the Seeheim architecture, and a View can be any output representation of data [52]. Model–View–Presenter (MVP) is a derivative of MVC, where the presenter assumes the role of the Controller, retrieves and formats data for the View, the View is responsible for handling the user interface events, which is the controller's role in MVC, and the Model is strictly a domain model. In Model-View-ViewModel (MVVM), the ViewModel is responsible for providing access to data objects and backend logic from the Model. View is all elements displayed by the user interface. Model is either a domain model which represents the real state content, or the data access layer that represents that content.

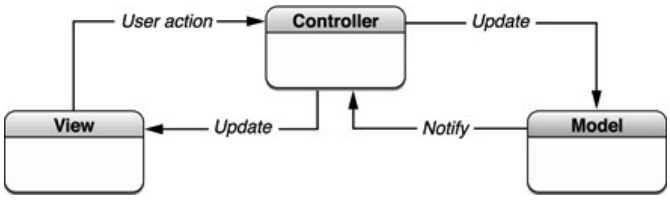

Fig. 4. The MVC pattern

All analysed models of interface interaction and adaptation emphasize the role of feedback in the interface adaptation process. For interface customization, a separate dedicated interface (or interface layer) is required. We call this interface, the meta-interface, since it is overlaid on top of the software product's interface and allows making configuration choices on the product's interface. One example of such meta-interface is the Facebook's Like button (see Fig. 5, a), which allows the Facebook users to express their opinion on the content of a website. The button provides a one-click shortcut to express and externalize the affective reactions of a user. Another example of meta-interface is provided by Usabilla (www.usabilla.com), which is a service for real-time visual user feedback tracking. The users of the website click the feedback button and can select any part of the page to evaluate it (see Fig. 5, b). Google provides a similar mechanism, where users can highlight any areas of web interface, black out personal information, comment on relevant issues and send it to Google (see Fig. 5, c).

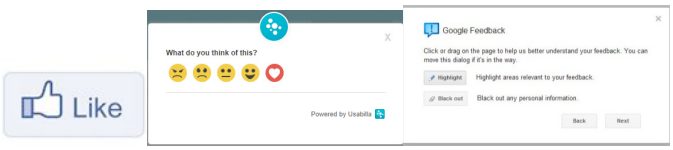

Fig. 5. Feedback in (left to right): Facebook, Usabilla and Google.

Summarizing, two possible implementations of feedback are usually considered [53]: 1) Emoticons-based feedback: aiming at expressing the emotionally affected satisfaction degrees among the end-users via picking an emoticon (virtual facial expression) for judging his user experience [6]; 2) Recommendation frames: a simple interaction illustrated differently (e.g. pop-up window, sliding area), which is mainly used in e-commerce to provide client recommendations. The implementation of such meta-interface together with the need for handling community requests and implementation of conflict resolution and opinion aggregation mechanisms for crowdvoting, requires the extension of existing user interface development architectures and design patterns.

III. Framework of Community-Driven User Interface Development

The proposed framework of community-driven user interface development consists of 1) the contextual feedback based adaptation (CFBA) metamodel, 2) the four-tiered user interface (4TUI) architecture, and 3) the Model-Control-View-Adapter (MCVA) pattern.

The CFBA metamodel (see Fig. 6) describes the relationship between different entities and models in the modelling and implementation of adaptable and evolvable user interfaces. The metamodel is based on the Norman's Model [42], Taylors Layered Protocols [45], Seeheim model [47] and its implementations as user interface design patterns (PAC, MVC and their variants), Bezold's model [48], Baxley's model [49] and RUX model [50]. The CFBA metamodel actually features two interaction cycles: 1) a traditional cycle, where a user and a system exchange with messages and feedbacks during the system's use, and 2) a community-driven cycle,

where a crowdvoting entity collects feedback from a community of users and changes the presentation of the interface according to the needs of the majority of users.

The community (crowd) is treated as a part of context that depends on the Context Model. To collect the opinion and judgments of the community of user interface, the crowdvoting [23] mechanism is used. The implementation of the user interface is described by the adaptation of the Seeheim model.

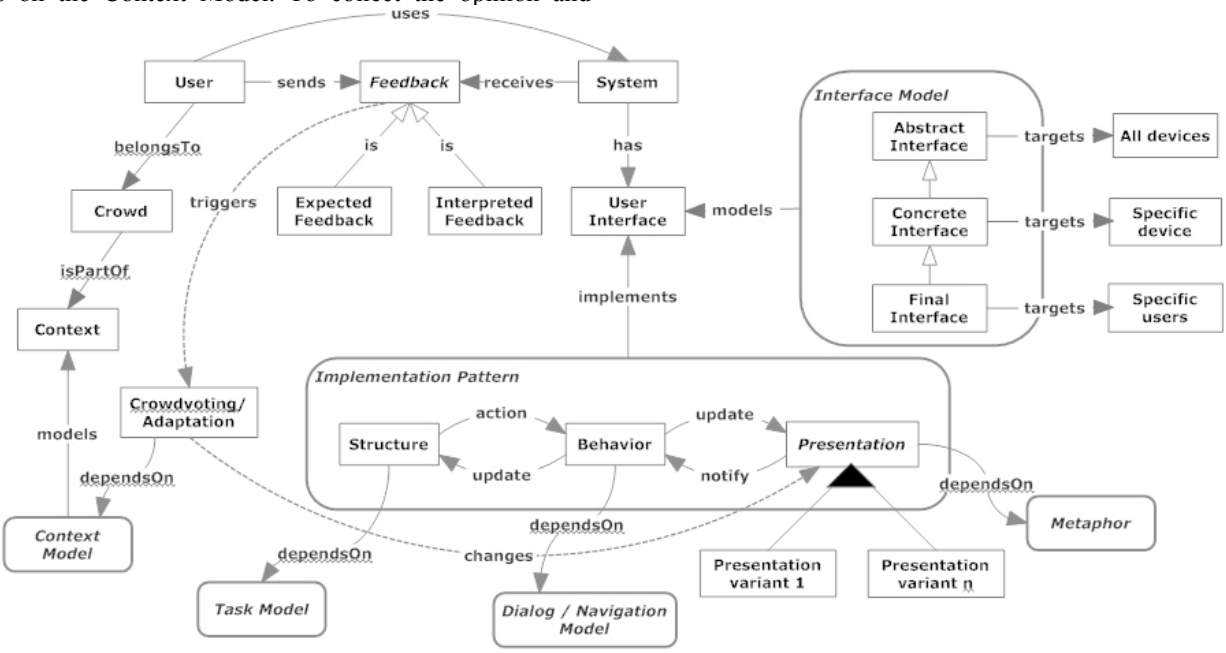

Fig. 6. Contextual feedback based user interface adaptation metamodel

However, where is a difference, the Presentation entity is variable and different variants of interface presentation can be selected according to the requests of the community aiming to guarantee usability while achieving adaptation to the changing needs of the community of users. The relationship of interface to other models is given by the adaptation of the Baxley's model: Structure depends upon Task Model, Behaviour depends upon Dialog and Navigation Model, and Presentation depends upon interface Metaphor.

Modelling of user interface at different levels of abstraction is represented by the adaptation of the RUX model, where a hierarchy of interfaces is used to represent interface independence and specificity with respect to different platforms and / or user groups, while allowing to implement automatic generation of interfaces [54]. We adopt the elements of crowdsourcing, i.e. crowdvoting, as a model [55] for solving a problem of interface adaptability and evolvability at use time.

The 4TUI architecture (see Fig. 7) describes the structural organization of the community-driven user interface system. The proposed architecture is an extension of the classic three-tiered architectures and models (such as MVC pattern), which include Persistence layer for data storage and handling, Business layer for business logic; and Presentation layer for content delivery. The additional fourth layer (tier) is proposed for managing community requests for interface representation and community-driven reasoning based on crowdvoting. This layer performs the functions of the semantic layer in the Bezold's model [48]. The functions of layers in the four layered system are summarized in Table 1.

The MVCA pattern (see Fig. 8) is an extension of the MVC family of patterns with an additional class for managing the community-driven requests for user interface modification. Since the community may include users with conflicting interests, a mechanism for solving these conflicts is required.

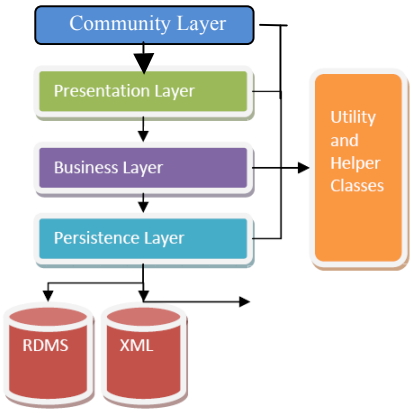

Fig. 7. Proposed four layered architecture of community-driven business applications

TABLE I. FUNCTIONS OF LAYERS IN FOUR LAYERED SYSTEM

Layer	Function
Persistence	Stores data and handles requests for data
Business	Specifies business objects and business logic rules, and handles interfacing between presented

	information and stored data
Presentation	Delivers content to browser
Community	Manages community requests for interface change and provides representation conflict resolution

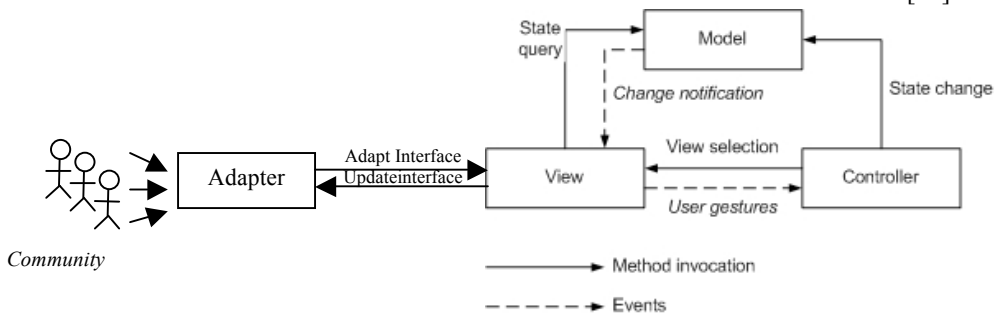

Fig. 8. Handling interface adaptation requests using the MVCA pattern

The proposed approach for the community-driven user interface adaptation combines elements of the following methodologies: Design-for-change [56] for developing interface that is evolvable and adaptable to anticipated and unanticipated changes; Meta-design [57] and related methodologies (End-User Development [58], Participatory Design [59], Collaborative Design [60], etc.) as a theoretical foundation for involvement of end-users as co-designers of a product; and Crowdsourcing [1] for enabling the participation of a crowd (i.e., a community of dedicated users) in the user interface improvement and evolution process.

IV. Conclusions

The social collaboration based approach can be applied to the process of user interface development. The community of users can drive the evolution of user interfaces to increase their flexibility (the interface adapts flexibly to change requirements of the users), plasticity (the interface's capacity of adaptation to cope with changing context), quality (the interface of a product represents the combined efforts aka collective intelligence of the community of users), usability (the community itself selects and adopts via natural selection the best practices of interface design), cultural acceptance (the interface reflects the culture of its users), user satisfaction (opportunity to have a say and an impact increases user satisfaction) and product popularity (the users have control over what they see and what they get).

The advantages of the proposed framework for community-driven user interface adaptation are as follows: iterative refinement of user interface enables interface evolution; the community performs interface evaluation; interface dynamically changes in response to the changing requirements of the community; the user interface evolution process is outsourced to the community of users and is fully automated; community feedback ensures that interface of the product is prevented from ageing and decay so long as the community is interested in the services provided by the product itself; collective intelligence allows to evolve a user interface that is inherently usable, culturally-acceptable and visually-pleasant; direct real-time user participation in the user interface evolution increases user engagement and satisfaction; interface quality is ensured by the intelligent voting mechanisms that harnesses "wisdom of crowds" and retargets interface for a specific groups of users.

The proposed solution is finding sub-communities or groups of users with similar interests based on their profiles and interface preferences, and providing customized variants of interfaces to these particular groups. The mechanism for conflict solving is based on the consensus-based user profile determination method [40].

References

[1] J. Howe, "The rise of crowdsourcing", Wired 14(6), 2006.

[2] P. Waterson, "Motivation in Online Communities", in S. Dasgupta (ed.) Encyclopedia of Virtual Communties, 2006.

[3] P. Kollock, "The economies of online cooperation: gifts and public goods in cyberspace", in M.A. Smith, P. Kollock, (Eds.), Communities in Cyberspace, pp. 220-238. London: Routledge.1999.

[4] D. Schuler and A. Namioka, Participatory design: Principles and practices. Hillsdale, NJ: Erlbaum, 1993.

[5] J. Roberts, I.-H. Hann and S. Slaughter, "Understanding the Motivations, Participation and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects", Management Science 52(7), July 2006, pp. 984 - 999.

[6] S. Abeyratna, G.V. Paramei, H. Tawfik and R. Huang, "An Affective Interface for Conveying User Feedback", Proc. of 12th International Conference on Computer Modelling and Simulation (UKSim), pp. 369-374, 2010.

[7] S. Stumpf, E. Sullivan, E. Fitzhenry, I. Oberst, W.-K. Wong and M.M. Burnett, "Integrating rich user feedback into intelligent user interfaces", Proc. of International Conference on Intelligent User Interfaces (IUI 2008), pp. 50-59, 2008.

[8] J. Eisenstein, J. Vanderdonckt and A. Puerta, "Adapting to mobile contexts with user-interface modeling", Proc. of the Third IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'00), pp. 83-92, 2000.

[9] G. Menkhaus and W. Pree, "User interface tailoring for multi-platform service access", Proc. of the 7th international conference on intelligent user interfaces (IUI '02), pp. 208-209, 2002.

[10] P. Repo, "Facilitating user interface adaptation to mobile devices", Proc. of the third Nordic conference on Human-computer interaction (NordiCHI '04), pp. 433-436, 2004.

[11] M. Bisignano, G. Di Modica and O. Tomarchio, "Dynamic User Interface Adaptation for Mobile Computing Devices", Proc. of the 2005 Symposium on Applications and the Internet Workshops (SAINT-W '05), pp. 158-161, 2005.

[12] G. Calvary, J. Coutaz and D. Thevenin, "Supporting Context Changes for Plastic User Interfaces: a Process and a Mechanism", Proc. of the Joint AFIHM-BCS Conf. on Human-Computer Interaction IHM-HCI 2001, vol. I, pp. 349-363. Springer, London, 2001.

[13] W.S. Lasecki, K.I. Murray, S. White, R.C. Miller and J.P. Bigham, "Real-time Crowd Control of Existing Interfaces", Proc. of the ACM

Symposium on User Interface Software and Technology (UIST 2011), pp. 23-32, 2001.

[14] J. Dicker and B. Cowan, “Platforms for Interface Evolution”, Proc. of ACM CHI 2008 Conference on Human Factors in Computing Systems (CHI’08).

[15] M. Speicher, Crowdsourced Evaluation and Adaptation of Web Interfaces for Touch. Master thesis. Global Information Systems Group, ETH Zurich, 2012.

[16] A.K. Dey, “Understanding and Using Context”, Personal and ubiquitous computing, Vol. 5, February 2001, pp. 4-7.

[17] S. Hennig, J. Van den Bergh, K. Luyten and A. Braune, “User driven evolution of user interface models - The FLEPR approach”, Proc. of the 13th IFIP TC 13 international conference on Human-computer interaction - (INTERACT'11), Part III. Springer-Verlag, pp. 610-627, 2011.

[18] N. Wiener, Cybernetics: or Control and Communication in the Animal and the Machine. Cambridge: MIT Press, 1948.

[19] M. Norgaard and K. Hornbaek, Exploring the value of usability feedback formats. International Journal of Human-Computer Interaction 25(1), pp. 49-74, 2009.

[20] J. Allwood, J. Nivre and E. Ahlsén, “On the semantics and pragmatics of linguistic feedback”, Journal of Semantics 9(1), 1992, pp. 1-26.

[21] P. Kotzé, “Feedback And Task Analysis For E-Commerce Sites”, Proc. of the ISSA 2002 Information for Security for South-Africa 2nd Annual Conference, 10-12 July 2002, Muldersdrift, South Africa, pp. 1-17.

[22] A. Warr and E. O'Neill, “Getting Creative with Participatory Design”, Proc. of Participatory Design Conference, 2004, pp. 57-61.

[23] J.M. Leimeister, “Collective Intelligence”, Business & Information Systems Engineering, 2(4), 2010, pp. 245-248.

[24] J. Surowiecki, The Wisdom of Crowds, Anchor, 2005.

[25] A. Spink and T. Saracevic, “Human-computer interaction in information retrieval: Nature and manifestations of feedback”, Interacting with Computers, 10(3), pp. 249-267, 1998.

[26] M.M. Lehman and L.A. Belady (eds.), Program evolution: processes of software change. Academic Press Professional, 1985.

[27] C. Ghezzi, P. Inverardi and C. Montangero, “Dynamically Evolvable Dependable Software: From Oxymoron to Reality”, in P. Degano, R. Nicola and J. Meseguer (Eds.), Concurrency, Graphs and Models, LNCS vol. 5065. Springer-Verlag, pp. 330-353, 2008.

[28] F. Heller, L. Lichtschlag, M. Wittenhagen, T. Karrer and J. Borchers, “Me Hates This: Exploring Different Levels of Use”, Proc. of ACM CHI 2011 Conference on Human Factors in Computing Systems (CHI 2011), 2011, pp. 1357-1362.

[29] R. Mendoza Gonzalez, J. Munoz Arteaga, F.J Alvarez and M. Vargas Martin, “Integration of auditive and visual feedback in the design of interfaces for security applications”, in Workshop on Perspectives, Challenges and Opportunities for Human-Computer Interaction in Latin America (CLIHC), Rio de Janeiro, Brazil, 2007.

[30] L. Cerrato, “A comparison between feedback strategies in human-to-human and human-machine communication”, Proc. of 7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002.

[31] R.-J. Beun, R.M. van Eijk and H. Prust, “Ontological Feedback in Multiagent Systems”, Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '04), Vol. 1, pp. 110-117, 2004.

[32] G. Fischer, “Understanding, fostering, and supporting cultures of participation”, Interactions 18(3), pp. 42-53, 2011.

[33] G. Fischer, “End User Development and Meta-Design: Foundations for Cultures of Participation”, Journal of Organizational and End User Computing, 22(1), pp. 52-82, 2010.

[34] S. Kopp, J. Allwood, K. Grammer, E. Ahlsén and T. Stocksmeier, “Modeling Embodied Feedback with Virtual Humans”, ZiF Workshop, 2006, pp. 18-37.

[35] A. Sears and B. Shneiderman, “Split menus: effectively using selection frequency to organize menus”, ACM Transactions on Computer-Human Interaction (TOCHI), v.1 n.1, p.27-51, March 1994.

[36] K.Z. Gajos, D.S. Weld, and J.O. Wobbrock, “Decision-theoretic user interface generation”, Proc. of the 23rd national conference on Artificial intelligence (AAAI'08), Vol. 3, pp. 1532-1536, 2008.

[37] K. Gajos, D. Christianson, R. Hoffmann, T. Shaked, K. Henning, J.J. Long and D.S. Weld, “Fast and robust interface generation for ubiquitous applications”, Proc. of the 7th international Conference on Ubiquitous Computing (UbiComp'05), pp. 37-55, 2005.

[38] M.A. Pérez, “Conversational dialogue in graphical user interfaces: interaction technique feedback and dialogue structure”, in CHI 95 Conference Companion, 1995, pp. 71-72.

[39] J. Kim, R. Kumar and S.R. Klemmer, “Crowdsourcing Interface for Collecting Correspondences of Web Pages”, UIST '09 Poster, Victoria, BC, Canada 2009.

[40] J. Sobecki and N.T. Nguyen, “Consensus-based adaptive interface construction for multiplatform Web applications”, Proc. of 4th International Conference on Intelligent Data Engineering and Automated Learning, IDEAL 2003. Springer LNCS Vol. 2690, 2003, pp. 457-461.

[41] B. Lafreniere and M. Terry, “Socially-Adaptable Interfaces: Crowdsourcing Customization”, Proc. of ACM CHI 2011 Conference on Human Factors in Computing Systems (CHI 2011), 2011.

[42] D.A. Norman, “Cognitive Engineering”, in D.A. Norman and S.W. Draper (eds.), User Centered System Design. Lawrence Erlbaum Associates, Hillsdale, 1986, pp. 31–61.

[43] V. Lopez-Jaquero, J. Vanderdonckt, F. Montero, and P. Gonzalez, “Towards an Extended Model of User Interface Adaptation: The Isatine Framework”, in J. Gulliksen, M.B. Harning, P. Palanque, G.C. Veer and J. Wesson (Eds.), Engineering Interactive Systems, LNCS, Vol. 4940. Springer-Verlag, Berlin, Heidelberg, pp. 374-392, 2008.

[44] H. Dieterich, U. Malinowski, T. Kühme and M. Schneider-Hufschmidt, “State of the Art in Adaptive User Interfaces”, in: M. Schneider-Hufschmidt, T. Khüme and U. Malinowski (eds.), Adaptive User Interfaces: Principle and Practice. North Holland, Amsterdam, 1993.

[45] J.H. Eggen, R. Haakma and J.H.D.M. Westerink, “Layered Protocols: hands-on experience”, International Journal on Human–Computer Studies, 1996, 44, pp. 45–72.

[46] R. Haakma, Layered Feedback in User-System Interaction, Master thesis, Eindhoven University of Technology.

[47] A.J. Dix, B. Russell and A. Wood, “Architectures to make Simple Visualizations using Simple Systems”, Proc. of Advanced Visual Interfaces, AVI2000, 2000, pp. 51-60.

[48] M. Bezold, “A Semantic Framework for Adapting Interactive Systems in Intelligent Environments”, Proc. of Intelligent Environments 2009, pp. 204-211.

[49] B. Baxley, “Universal model of a user interface”, Proc. of Conference on Designing for user experiences (DUX '03), pp. 1-14, 2003.

[50] J.C. Preciado, M.L.Trigueros and F. Sánchez-Figueroa, “An Approach to Support the Web User Interfaces Evolution”, Proc. of the 2nd Int. Workshop on Adaptation and Evolution in Web Systems Engineering AEWSE'07, 2007. CEUR Workshop Proc. 267.

[51] J. Coutaz, “PAC: an Implementation Model for Dialog Design”, Proc. of the Interact'87 conference, 1987, pp. 431–436.

[52] O. Moravcik, T. Skripcak, D. Petrik and P. Schreiber, “Approaches of the Modern Software Development”, International Journal of Machine Learning and Computing, Vol. 1, No. 5, December 2011, pp. 479-487.

[53] N. Mezhoudi, “User interface adaptation based on user feedback and machine learning”, IUI Companion, 2013, pp. 25-28

[54] V. Štuikys, R. Damaševičius, J. Valančius, G. Ziberkas, V. Limanauskienė and E. Toldinas, “Generation of Database Interfaces for Nomadic Users”, Information Technology & Control, No. 2(27), pp. 41-50, 2003.

[55] D. Brabham, “Crowdsourcing as a model for problem solving: An introduction and cases”, Convergence: The International Journal of Research into New Media Technologies, 14(1), pp. 75-90, 2008.

[56] V. Štuikys, R. Damaševičius, M. Montvilas, V. Limanauskiene and G. Ziberkas, “Educational Portal Development Model for Implementing Design for Change”, Information Technology and Control, 35(3), pp. 222-228, 2006.

[57] G. Fischer, “Meta-Design: A Conceptual Framework for End-User Software Engineering”, in M.M. Burnett, G. Engels, B.A. Myers and G. Rothermel (eds.), End-User Software Engineering, Dagstuhl Seminar Proc. 07081, Schloss Dagstuhl, Germany, 2007.

[58] H. Lieberman, F. Paternó, M. Klann and V. Wulf, “End-user development: An emerging paradigm”, in H. Lieberman, F. Paterno and V.Wulf (eds.), End-user development. Springer, pp. 9-15, 2005.

[59] Y. Dittrich, S. Eriksén and C. Hansson, “PD in the Wild; Evolving Practices of Design in Use”, Proc. of the Participatory Design Conference (PDC 02), 2002, pp. 124-134.

[60] L. Zhu, “Cultivating collaborative design: design for evolution”, Proc. of the Second Conference on Creativity and Innovation in Design (DESIRE '11), pp. 255-266, 2011.