Supporting the Design of Flexible Game Systems

David Thue
RISE Research Group, School of Information Technology, Carleton University
Ottawa, Canada
Department of Computer Science, Reykjavik University
Reykjavik, Iceland
david.thue@carleton.ca

ABSTRACT

Designing games is hard. To support player interaction, someone must decide not only what a player might do, but also what they should observe and how the game should respond to each of their actions. While these decisions are often made solely by a game's designers, they are sometimes shared with players, providing one or more ways to change how the game works. How might interacting with such a flexible game system affect a player's experience of the game? Unfortunately, it remains unclear how or whether game design frameworks can be used to help answer this question, as they tend to focus on player actions that change the game's state. As more games blur the lines between settings screens and gameplay or designers and players, new or refined conceptual tools are needed to support the design of flexible game systems. We present our ongoing work to build a conceptual "adapter" for existing tools, allowing game designers to apply their current toolbox to this task.

CCS CONCEPTS

• **Human-centered computing** → **Interaction design theory, concepts and paradigms**; • **Software and its engineering** → **Interactive games**; • **Theory of computation** → *Representations of games and their complexity*; • **Applied computing** → *Computer games*.

KEYWORDS

Game Design; Interaction Design; Conceptual Framework

ACM Reference Format:
David Thue. 2021. Supporting the Design of Flexible Game Systems. In *Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '21), October 18–21, 2021, Virtual Event, Austria.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3450337.3484223

1 INTRODUCTION

Designing a game involves defining what decisions will be available to each player at different times during their experience, along with what information they will be offered (and when) to help inform their choices [20]. These definitions govern which aspects of the game the player can observe (and when) as well as how the game should change over time, particularly in response to player actions. We refer to these two kinds of definitions as *observation rules* and *transition rules*, respectively.

Player decisions often focus on manipulating the game's state to make it reach some desired configuration [2, 20]. However, with the rise of player-driven customization and accessibility-focused design, many games now invite their players to manipulate not only the system's state, but also parts of its observation rules or transition rules. Manipulating observation rules includes actions like changing the field of view in First-Person Shooter or activating modes to support reduced colour vision (e.g., in *World of Warcraft* [4]); such actions alter how the game's state is translated into on-screen observations. Manipulating transition rules includes actions like setting the difficulty level of a game; these actions can change how the game responds when the player performs a variety of other actions, like offering new chances when the player fails. Importantly, both kinds of rule manipulation represent a transfer of decision-making power from the designer to the player – instead of choosing a particular rule to apply (e.g., a certain scheme for colours) and holding it constant for every player's experience, the designer allows the player to choose a rule from a set of potential options.

1.1 Motivation

The benefits of providing players with rule-changing actions are clear from prior work: they empower users [7, 9, 12, 14] and they help mitigate the challenge of creating a successful inflexible design [6]. But how can a designer be sure that the actions they design will have the effects that they intend? While existing design frameworks offer guidance when player actions will modify the game's state [2, 3, 11, 20, 24], they generally say little about actions that can change a game's transition or observation rules.

Is changing a game's state any different from changing one or more of its rules? If not, then existing design frameworks could perhaps be readily applied, by simply treating the game's rules as part of its state. We argue that they *are* different in at least two important ways. First, rules represent a kind of information that is more specific than what states can represent. Specifically, rules describe how something will change when one or more conditions are met (e.g., "The first player to score 10 points wins the game"), while a state describes how something is or could be (e.g., "Kofi has 2 points, Sue has 4 points, …"). As a result, actions that change a game's rules must comply with different design constraints than actions that change a game's state – the result of activating any rule-changing action must be a set of valid rules. Second, thanks to the design principle of Consistency, players will likely assume that a game's rules will change less frequently than its state; the state must change to enable interaction, but rules tend to remain

CHI PLAY '21, October 18–21, 2021, Virtual Event, Austria

ACM ISBN 978-1-4503-8356-1/21/10.
https://doi.org/10.1145/3450337.3484223

constant to help players learn them. Players are thus likely to expect that the effects of changing a rule will last longer than the effects of changing a state, because rules tend to change less often than states. These differences highlight two design considerations that set our problem apart from the challenge of designing for players actions that change a game's state. Our question thus remains unanswered: How can we support the design of actions that can change a game's rules?

1.2 Contribution

In this paper, we present our progress toward broadening the utility of existing game design frameworks, toward ultimately showing how they can be used to design actions that change a game's rules. Specifically, we build on our prior method for modelling player interaction [23] by introducing two new notions (state-likeness and influence paths) that each point to new ways in which game design frameworks can be used.

2 BACKGROUND & RELATED WORK

Our work draws on research in End User Development and Meta-design, and builds on a recent framework from the field of Interactive Storytelling.

2.1 End User Development

Empowering users to change a system's rules is a key goal of End User Development (EUD) [12], which grew from the assertion that the functions of any given software should be adaptable; i.e., its users should have the power to customize or extend the capabilities of the software's interactive system. Notable successes of EUD include the programmability of spreadsheets and the integration of "modding" culture into the design and development of many digital games [19].

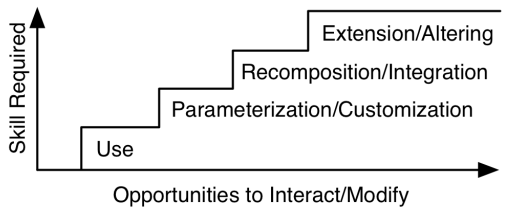

Figure 1: Ludwig et al.'s proposed relationship between required user skill and the adaptability of a system (based on [14]).

More recently, there has been work toward generalizing EUD beyond the development of software, such that it could apply to the development of any sort of tool [14]. In particular, Ludwig et al. [14] offered a revision of MacLean's earlier "tailorability mountain" [15], wherein the degree of adaptability of a system was compared to the level of user skill required to act at each of four levels of sophistication (Figure 1). Following Ludwig et al.'s goal of generalization, we propose that the upper three levels of sophistication (above "use" in the figure) are useful for describing the different kinds of rule-changing actions that might be offered in *any* game – digital or analog. Furthermore, while we have thus far discussed rule-changing actions using examples of parameterization and customization (e.g., a game's difficulty settings), our work is agnostic to the kind of rule-manipulation that players might perform. For example, we view *World of Warcraft*'s support of player scripting via Lua [4] (an example of "Extension/Altering", in Figure 1) as a way to offer a very large set of rule-changing actions for player to consider.

2.2 Meta-design

The ideas of meta-design are broad and have diverse origins. We focus on the version of meta-design that was proposed by Fischer and Scharff [6], synthesized by Giaccardi [8], and then discussed with respect to End User Development by both Fischer and Giaccardi [5]. In particular, two concepts of meta-design are relevant to our work: underdesign and design time versus use time.

As we discussed in Section 1, providing rule-changing actions to a player amounts to a direct transfer of power from the designer to the player: decisions that a designer could have made are given to the player to make instead. The transfer of decision-making power from designer to user is the key idea of *underdesign* [1], which asserts that a meta-designer should create "design spaces for others" ([5], pg. 432).

A second effect of this transfer of power is that the timing of the transferred decisions also changes, shifting from *design time* (before any user experience begins) to *use time* (during a user's experience) [6]. This time shift is important. Unlike Participatory Design, which involves a similar transfer of power but *no* shift in decision time [21], the design of rule-changing actions implies a temporal disconnect between designers and users. We discuss the implications of this disconnect in Section 4.

2.3 Interactive Processes

The core of our approach builds on our own recent work [23] in the domain of Interactive Storytelling, which focuses on the design and production of interactive narrative systems. Specifically, we introduced a method for modelling interactive systems that incrementally creates a collection of connected *interactive processes*.

As shown on the left of Figure 2, an *interactive process* (IP) is a connected collection of functions and data that define how one or more actors (e.g., players) can interact with some particular target object. Such a process offers a general way to represent "acted change" in an interactive system, because the target object can be set freely according to the designer's needs. For example, when the target object of an IP is set to be the state of a narrative world, then the process can be used to represent how readers interact as they read a Choose Your Own Adventure book [23].

Notably for our purpose, two interactive processes can be connected by having the target object of one process be an element of another process. For example, the right side of Figure 2 shows a situation where IP1's transition function is the target object of IP2. Each IP offers nine elements as potential targets, as shown by the abbreviated labels on each process in the right of Figure 2. This kind of representation is useful for our current work because it gives us a way to model how the rules of a game might be changed, and the resulting model can offer clarity over how rule-changing actions should be designed.

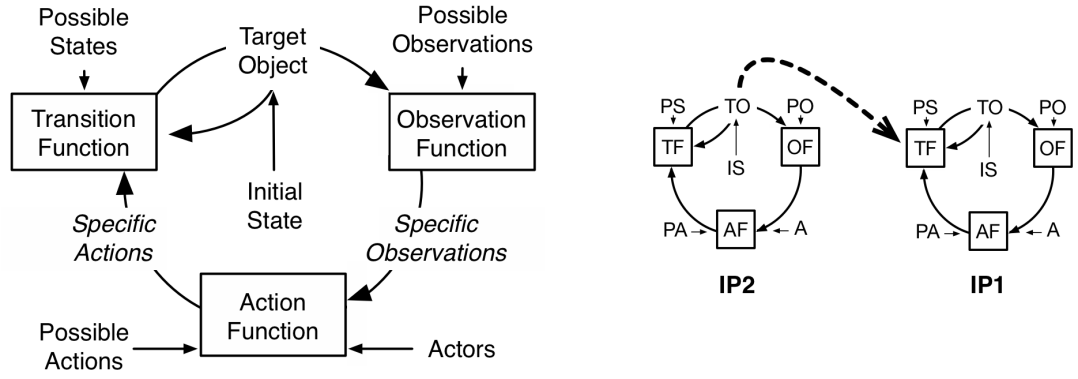

Figure 2: At left: An interactive process, as given in our prior work ([23]). Interaction with a target object occurs as actors receive specific observations and perform specific actions via the action function. The transition and observation functions uphold the system's transition and observation rules, respectively, producing a new state of the target object and a new observation of that state. Interaction begins with the target object in a given initial state. At right: One interactive process (IP2) has its target object set as the transition function of a different interactive process (IP1).

Finally, our prior work contained a method for modelling an interactive system as a graph of connected interactive processes [23]. To keep our writing self-contained, we summarize the method here:

- The modeller begins with a model that consists of single interactive process, where the target object of that process is the state of the interactive system to be modelled (e.g., the state of a game world).[1]
- For each element in the model that has not been considered, the modeller asks themselves the yes/no question: "Can or should any agent change this element?"
  - Whenever the answer is "yes", the modeller creates a *new* interactive process and adds it to the model, setting its target object equal to the element for which the modeller just answered "yes". This new IP will host all interactions that can directly change the targeted element.
  - When the answer is "no", the modeller considers and questions a different element in the model.
- The method and model are complete once no unconsidered elements remain in the model.

This method is useful to our approach because it offers a systematic way to build or modify a model of a game, where the resulting model explicitly addresses how players might change something other than the game's state. As an example, Figure 3 shows several stages of this process for modelling the card game *Fluxx* [16]. Interestingly for our example, the rules of *Fluxx* are modified by its players as a part of regular play. These rules begin as "Draw 1, Play 1", but can change as players play "New Rule" cards such as "Draw 2", "Play 3", or "Draw +1 if you Talk like a Pirate during your turn".

3 TWO KEY CONCEPTS: STATE-LIKENESS AND INFLUENCE PATHS

We now consider our core question: "How can we support the design of actions that can change a game's rules?" In this section, we introduce and develop two concepts related to interactive processes. These concepts help show how existing design frameworks can be adapted to offer the support that we seek.

3.1 Process Elements can be State-like

Our approach relies on the following insight: when connecting interactive processes (following Section 2.3), the target object of any connection gets treated in a *state-like* way; regardless of whether it is a function or data, it exhibits properties that can be changed by actions. As a concrete example, Figure 4 shows an annotated version of our model of the card game *Fluxx*, as we developed it in Figure 3.

Referring to Figure 4, the base game's transition function (TF, right side) becomes state-like when it is set as the target object (TO, left side) of the game's rule-changing process (see the dashed arrow). More specifically, it becomes state-like because it has particular, changeable properties that govern how it works (namely, which "New Rule" cards are in play), and those properties can be observed and altered though actions in an interactive process. These properties (shown as 'Current "New Rule" Cards' on the left side of the figure) determine the current "state" of the base game's transition function. In general, the state-like aspect of a transition function represents the transition rules that it upholds, and the state-like aspect of an observation function represents the observation rules that it upholds. To simplify our presentation, we avoid discussing the state-like aspects of the other elements of an interactive process.

The notion of state-likeness is useful because it suggests how a design framework that deals with state-changing actions (e.g., MDA) might be adapted to deal with rule-changing actions instead – by retargeting the framework's notion of "state" to the state-like aspect of those rules. Still, this solution is incomplete. While we could apply an existing design framework independently to each interactive process in a model (e.g., once for *Fluxx*'s base game and once for its rule-changing process), doing so would fail to

[1]We assume that every game has a "state" that describes the current configuration of everything in the game world.

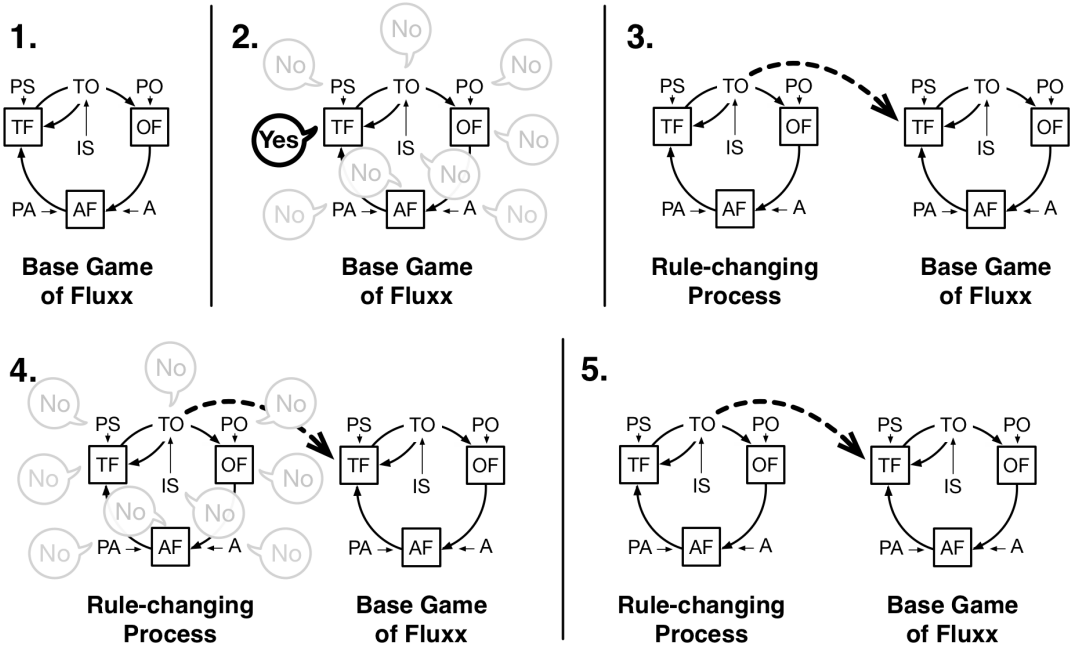

Figure 3: A novel demonstration of our modelling method [23], as applied to the game of *Fluxx* [16]. 1. The method begins with a single interactive process to represent the base game. 2. The modeller asks themselves "Can or should any agent change this element?" for each element in the current model. 3. For each element where the answer was "Yes", a new interactive process is created with that element as its target object. 4. The modeller asks and answers the same question from step 2 for every new element in the model. 5. All elements have been considered, and thus the model is complete.

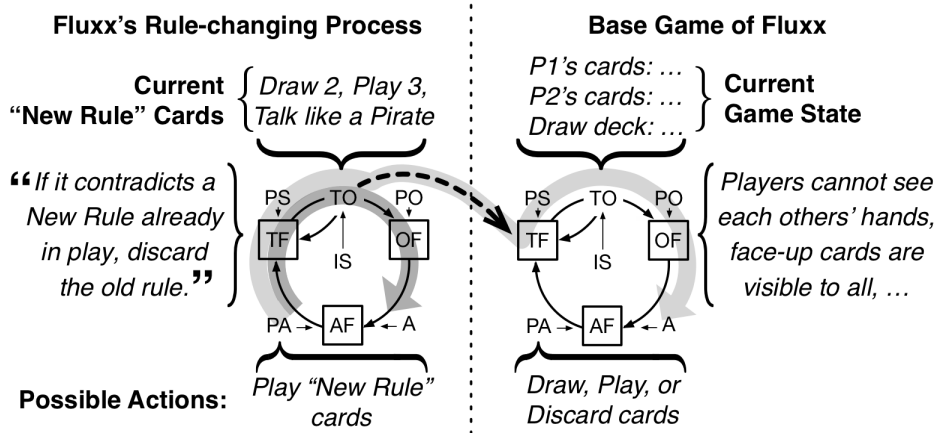

Figure 4: An annotated example of a card game (*Fluxx* [16]) wherein players can change the transition rules that govern the game's state. For the sake of simplicity, only a subset of the possible annotations are shown. To understand the abbreviations, see Figure 2. Some game details are omitted for the sake of clarity.

consider how the processes are connected. We explore this concern in Section 3.2.

3.2 Influence Paths within and across Processes

Game design frameworks are fundamentally concerned with player experiences. In terms of the models that we can build using interactive processes, a player's experience arises from the sequence of observations they receive and the actions they perform as they interact with a game. For a game whose rules *can't* be changed, this interaction can be modelled using a single interactive process; our method ends almost immediately because the modeller's answer for every element in the base process is "No" (see Figure 5: Left).

When a model contains only one IP, existing design frameworks can be readily applied because there is only one way in which a

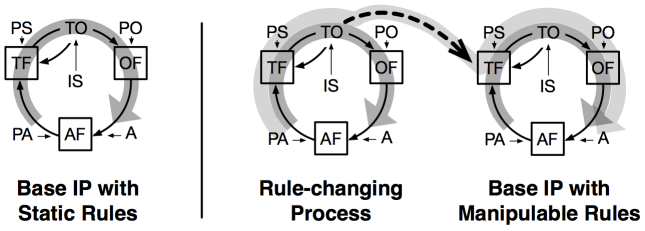

Figure 5: Two diagrams showing the paths of influence that a player has access to, beginning with a player action and ending with a player observation. At left: When an interactive system's rules cannot be changed, it has a single interactive process and there is only one influence path. At right: When an interactive system's transition rules can be changed, there are three influence paths, one of which crosses from one process to the other via the target object connection.

player can influence their future observations: their actions lead directly (and exclusively) through the transition function to produce a new state, which the observation function then uses to produce a new observation (see the dark grey arrow in Figure 5: Left). We say that this simple model has a single *influence path* – a single way for a player's actions to affect their observations.

For games whose rules *can* be changed, however, there are multiple influence paths to consider (see the three shaded arrows in Figure 5: Right). For example, when only the base IP's transition rules can be changed (like in the figure), a player can affect their own experience via three possible paths – one in each of the model's two processes and one that crosses from one process to the other. This crossing happens because a player's action can change a transition rule, and that rule could have a direct effect on how their future actions in the base IP will affect the base IP's state – the player can change how the base IP "works" by changing its rules. As an example of this path from *Fluxx*, consider the action of playing a "New Rule" card that causes the "Draw 3" rule to replace an existing "Draw 2" rule. This action is performed in the Rule-changing Process and (via the target object link) changes the rules of the game's Base IP. As a result, the next time a player performs a "Draw" action in the base IP, they will receive three cards instead of two. Tracing and thinking about influence paths allows a designer to reason about how a player's observation in one process might lead them to act in another. But how can we trace them?

When modelled using our method, a game has a set of influence paths, each of which begins with a player's ability to act (just after an action function, AF) and ends with an observation being sent to the player (just before an AF). To find a given game's set of influence paths for a given set of players, one must first identify every action function at which those players may receive observations and perform actions. Every influence path will: (i) begin just after one those action functions, (ii) follow the directional arrows along each interactive process (and across them, via target object links), and (iii) end as soon as one of the identified action functions is reached. The ending AF might be the same as the starting AF (as in Figure 5: Left), or it might be a different AF (as a result of crossing from one process to another). Once the set of AFs has been identified, a simple graph search across the model can identify the set of influence paths.

Knowing about a game's influence paths is useful because it allows designers to distinguish between multiple interaction loops in a single game. In practice, this means that the considerations of a design framework could extend along multiple paths of influence - one for each path that affects a target object.

4 DISCUSSION AND FUTURE WORK

In this work, we explored a particular question: How can we support the design of actions that can change a game's rules? To help answer this question, we leveraged an existing method for modelling interactive systems and introduced two new related concepts: state-likeness and influence paths.

The idea that rules can be treated and modified like state is common in the context of Genetic Algorithms [13], but only a small proportion of games have invited rule modification as a core gameplay activity. The video game *Baba is You* [10] offers some evidence of this claim, as it is both relatively recent and critically recognized for its originality in having the player solve puzzles by altering the game's rules [18]. It would be interesting to interview its creator to understand what design frameworks (if any) they used, and see how our model of the game's interaction would compare to how they conceive it.

For our work, the idea of state-likeness offers a way to adapt a given game design framework to address rule-changing actions: by redefining the framework's notion of game state to target the rules themselves instead. This allows a designer to use the frameworks that they know (e.g., MDA [11]) to reason about rule-changing actions. For example, one can consider what Mechanics allow a player to establish new rules in *Fluxx*, what Dynamics those rules lead to, and what Aesthetics arise as a result of those Dynamics.

The interactive loops created by our influence paths (even in a single interactive process) are somewhat similar to the game loops proposed by Sicart [22]. The key difference between the ideas is that Sicart's loops distinguish between particular ways of changing a game's state (e.g., in football, the offensive-play loop is different from the defensive-play loop), while our interactive processes distinguish between different target objects. One target object is always the game's state (when a game is modelled), but there might be many other target objects that target other process elements (such as the game's transition rules). Furthermore, influence paths can be used to identify interactive loops that span multiple interactive processes, supporting the designer in reasoning about how a player's various observations (from every process they take part in) might lead them to act in any of the ways that they have available.

Put another way, influence paths help the designer reason about the impacts of the system's flexibility, along with their capacity and responsibility to shape how that flexibility is used. Here is where the meta-design concepts of design time, use time, and underdesign apply, as we discussed them in Section 2.2: A flexible game system is necessarily underdesigned because it converts what would ordinarily be design-time, designer-made decisions into use-time, player-made decisions. More specifically, such a system is underdesigned in the sense that its design work focuses on system elements

that are *outside of* the base interactive process. The designer creates "meta-rules" (see the left TF and OF in Figure 4) that govern what a player's rule-changing actions can do and how they will perceive the rules that they can change. Meanwhile, the player does some work that might have otherwise been done by the designer, acting in the interactive processes that target the base process's rules. We are keen to explore how our modelling method might offer new perspectives on both meta-design and end user development, and how these might be applied back to the context of games.

We see several ways to carry this work forward. First, our modelling method is relatively new, and seems likely to be underspecified as a result. Nevertheless, we believe that the concepts we have developed in this work should remain useful even as the method evolves. Second, while our notion of influence paths seems useful on its own, it remains to be seen whether such paths are at all related to how players reason about how their actions when faced with a flexible game system. A study of this relationship is planned as future work. Third, given that our method is agnostic with respect to the kind of rule-manipulation that users might perform (recall Section 2.1), it could be valuable to explore how this dimension of analysis might affect how different design frameworks should be adapted. Finally, we would welcome the chance to refine and validate our modelling method and its related concepts with both academics and practitioners alike.

REFERENCES

[1] Stewart Brand. 1995. *How Buildings Learn: What Happens After They're Built.* Penguin Books.

[2] Rogelio E Cardona-Rivera, José P Zagal, and Michael S Debus. 2020. GFI: A formal approach to narrative design and game research. In *International Conference on Interactive Digital Storytelling*. Springer, 133–148.

[3] Roberto Dillon. 2011. The 6-11 Framework: a new approach to video game analysis and design. *GAMEON-ASIA 2011 Proceedings* 2011 (2011), 25–29.

[4] Blizzard Entertainment. 2021. World of Warcraft. https://worldofwarcraft.com/.

[5] Gerhard Fischer and Elisa Giaccardi. 2006. Meta-design: A Framework for the Future of End-User Development. In *End User Development*, Henry Lieberman, Fabio Paternò, and Volker Wulf (Eds.). Springer Netherlands, Dordrecht, 427–457. https://doi.org/10.1007/1-4020-5386-X_19

[6] Gerhard Fischer and Eric Scharff. 2000. Meta-Design: Design for Designers. In *Proceedings of the 3rd Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques* (New York City, New York, USA) *(DIS '00)*. Association for Computing Machinery, New York, NY, USA, 396–405. https://doi.org/10.1145/347642.347798

[7] Giuseppe Ghiani, Marco Manca, Fabio Paternò, and Carmen Santoro. 2017. Personalization of Context-Dependent Applications Through Trigger-Action Rules. *ACM Trans. Comput.-Hum. Interact.* 24, 2, Article 14 (April 2017), 33 pages. https://doi.org/10.1145/3057861

[8] Elisa Giaccardi. 2003. *PRINCIPLES OF METADESIGN: Processes and Levels of Co-Creation in the New Design Space.* Ph.D. Dissertation. Faculty of Science and Technology, University of Plymouth. https://doi.org/10026.1/799

[9] Mona Haraty and Joanna McGrenere. 2016. Designing for Advanced Personalization in Personal Task Management. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) *(DIS '16)*. Association for Computing Machinery, New York, NY, USA, 239–250. https://doi.org/10.1145/2901790.2901805

[10] Hempuli. 2019. Baba is You. https://hempuli.com/baba/.

[11] Robin Hunicke, Marc G. Leblanc, and Robert Zubek. 2004. MDA : A Formal Approach to Game Design and Game Research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*. San Jose, CA, USA.

[12] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. 2006. End-user development: An emerging paradigm. In *End user development*. Springer, 1–8.

[13] Siew Mooi Lim, Abu Bakar Md Sultan, Md Nasir Sulaiman, Aida Mustapha, and Kuan Yew Leong. 2017. Crossover and mutation operators of genetic algorithms. *International journal of machine learning and computing* 7, 1 (2017), 9–12.

[14] Thomas Ludwig, Julian Dax, Volkmar Pipek, and Volker Wulf. 2017. *A Practice-Oriented Paradigm for End-User Development.* Springer International Publishing, Cham, 23–41. https://doi.org/10.1007/978-3-319-60291-2_2

[15] Allan MacLean, Kathleen Carter, Lennart Lövstrand, and Thomas Moran. 1990. User-Tailorable Systems: Pressing the Issues with Buttons. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Seattle, Washington, USA) *(CHI '90)*. Association for Computing Machinery, New York, NY, USA, 175–182. https://doi.org/10.1145/97243.97271

[16] Looney Labs. 2013. Fluxx. https://www.looneylabs.com/games/fluxx.

[17] Valve Corporation and Hidden Path Entertainment. 2012. Counter-Strike: Global Offensive. https://counter-strike.net.

[18] Chris Plante. 2019. Baba Is You review: one of the best puzzle games in years. *Polygon* (2019). https://www.polygon.com/reviews/2019/3/13/18263824/baba-is-you-review-nintendo-switch-pc

[19] Lev Poretski and Ofer Arazy. 2017. Placing Value on Community Co-Creations: A Study of a Video Game 'Modding' Community. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (Portland, Oregon, USA) *(CSCW '17)*. Association for Computing Machinery, New York, NY, USA, 480–491. https://doi.org/10.1145/2998181.2998301

[20] Jesse Schell. 2008. *The Art of Game Design: A book of lenses.* CRC press.

[21] Douglas Schuler and Aki Namioka. 1993. *Participatory design: Principles and practices.* Lawrence Erlbaum Associates, Inc, Hillsdale, NJ, US.

[22] Miguel Sicart. 2015. Loops and Metagames: Understanding Game Design Structures. In *Proceedings of the 10th International Conference on the Foundations of Digital Games*. Pacific Grove, CA, USA.

[23] David Thue. 2020. What might an action do? Toward a Grounded view of Actions in Interactive Storytelling. In *Proceedings of the 13th International Conference on Interactive Digital Storytelling (ICIDS'20) (LNCS Vol. 12497)*. Springer, Cham, 212–220.

[24] Wolfgang Walk, Daniel Görlich, and Mark Barrett. 2017. Design, Dynamics, Experience (DDE): An Advancement of the MDA Framework for Game Design. In *Game Dynamics: Best Practices in Procedural and Dynamic Game Content Generation*, Oliver Korn and Newton Lee (Eds.). Springer International Publishing, Cham, 27–45.