Adapting Cognitive Task Analysis to Elicit the Skill Chain of a Game

Britton Horn
Northeastern University
Boston, Massachusetts
bhorn@ccs.neu.edu

Seth Cooper
Northeastern University
Boston, Massachusetts
scooper@ccs.neu.edu

Sebastian Deterding
University of York
York, United Kingdom
sebastian.deterding@york.ac.uk

ABSTRACT
Playing a game is a complex skill comprising a set of more basic skills. These skills commonly map onto the main mechanics of the game, and build and depend on each other in a nested learning hierarchy, which game designers have modeled as *skill chains* made of *skill atoms*. For players to optimally learn and enjoy a game, it should introduce skill atoms in the ideal sequence of this learning hierarchy. However, game designers typically construct and use hypothetical skill chains based solely on design intent, theory, or personal observation, rather than empirical observation of players. This risks creating incomplete or suboptimal progression designs. In response, this paper presents an adapted cognitive task analysis method for eliciting the empirical skill chain of a game. A case study illustrates and critically reflects the method. While effective in foregrounding overlooked low-level skills required by a game, its efficiency and generalizability remain to be proven.

ACM Classification Keywords
H.5.m Information interfaces and presentation: Miscellaneous; K.8.0 Personal Computing: General: Games

Author Keywords
cognitive task analysis; game atoms; learning hierarchy; skill atoms; skill chains

INTRODUCTION
Like cooking, driving, and many other everyday activities, playing a video game is a complex skill [60]. *Complex skills* integrate a network of more basic skills: driving, for instance, requires independently mastering braking, steering, and switching gears, but also integrating and fluently switching between them [49]. These constituent basic skills hang together in a *learning hierarchy*: the logical order in which they build and depend on each other and therefore, in which they are ideally learned. For instance, we have to learn counting before we can learn addition and subtraction, and it is easier to learn these before multiplication [26, 4, 64]. Identifying the learning hierarchy of a to-be-taught complex skill is therefore a key task of instructional design, as it directly informs what learning goals to pursue, what outcomes to assess, and what tasks and material to present in what order to optimally support learning [35].

We find the very same need in entertainment and serious game design. No matter if designers want to create good tutorials and level progressions for a game [48, 8]; balance level difficulty or procedurally generate levels that fit player skills [45]; create educational games whose mechanics train targeted capacities [23]; or gamefully restructure everyday activities [21] – they are faced with the question what component skills a given game entails or ought to entail, and in what order the game should introduce these to players. More colloquially, if games are learning machines we enjoy to master [27, 42], it stands to reason they benefit from a well-designed sequence of learning.

Unsurprisingly, game design has developed a range of formal models that describe games as nested networks of game atoms or loops which revolve around specific actions or skills [21]. One particularly popular model, developed by Dan Cook, describes games as *skill chains*, directed graphs of *skill atoms* or core loops that logically build on each other – mirroring learning hierarchies in everything but name [14]. Likewise, there are many design methods such as *Rational Level Design* for prospectively deriving optimal level progression sequences from a given atom model [47, 48].

However, these models and methods provide little if any guidance how to reliably deduce the skill chain or learning hierarchy of a given game. Models are either sketched as blueprints for a new game or based on a designer's or researcher's individual reading of a game. Scarcely any game research methods exist to empirically deduce the skill chain of a game from actual player experience, or assess to what extent the skills and ideal sequencing order predicted by a model matches the actual skills it requires from players, or their actual learning hierarchy. This risks overlooking essential skills, not introducing them to players or introducing them in a suboptimal sequence.

In instructional design, cognitive task analysis (CTA) is a well-established family of methods to identify the skills and knowledge involved in a given task based on empirical observation and interviewing of experts [16, 13]. This includes methods for eliciting and modeling learning hierarchies [63,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
CHI PLAY '17, October 15–18, 2017, Amsterdam, Netherlands
©2017 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-4898-0/17/10…$15.00
DOI: https://doi.org/10.1145/3116595.3116640

35], which makes CTA an ideal candidate for identifying the learning hierarchies or skill chains of games. Although CTA techniques have existed for decades, to our knowledge, they haven't been adopted in games research and design for this purpose. Hence, this paper presents an adapted cognitive task analysis method for extracting the skill chain of a game from empirical gameplay. Akin to prior work on method development [36], we conducted a case study in a mode of critical *reflective practice* [54], continually documenting and reflecting on our method design process to assess its utility, identify future improvements, as well as surface more general issues in eliciting the knowledge and skills involved in playing a particular game.

The next section reviews existing work in games research related to modeling and identifying the component skills of games and introduces CTA. We then lay out our adapted CTA method, including its underlying rationale and a concrete step-by-step procedure for interested users. Our case study – using the method to identify the skill chain of the human computation game *Paradox* – illustrates the method in use and provides material for emerging observations and challenges. We discuss the contribution and limitations of the presented work and derive ramifications for future research.

BACKGROUND

Formal Modeling of Games

Doug Church [11] initiated contemporary work on "formal abstract design tools": developing grammars and tools to describe, analyze, and design the structural components of a game (for recent reviews, see [2, 22]). Following Almeida and da Silva [2], one can roughly distinguish (a) broad models like the MDA framework [34], (b) collections of descriptive terms and patterns (e.g. [6]), (c) design guidelines such as playability heuristics [41], and (d) modeling languages and tools of the core *mechanics* of a game, such as Machinations [22, 1]. Game mechanics describe the "core verbs" or "methods" by which players change the game state, such as moving, shooting, or trading [56]. They form part of *game atoms* [43] or *game loops* [57] – feedback loops between player input (invoking a particular mechanic, e.g. shooting), rules processing (e.g. adjudicating whether the shot hit), and computer output. A game atom is the smallest indivisible functional unit of a game. However, games are usually composed of nested networks of interlinked atoms: In a cover shooter, the "shooting" atom is part of a larger "defeating enemies" atom, which also entails a "cover" atom and may connect to an "upgrading atom," etc.

Skill Chains

As noted, game atom modeling is highly similar to modeling the learning hierarchies of complex skills. Both capture nested relations of basic to complex capacities, mechanics here, skills there – with one crucial difference: most game atom models concern themselves with a *synchronic overview* of the *game* and how the outputs of one atom (e.g. in-game resources like health or experience points) feed into others [2, 22]. They do not capture the *diachronic sequence* in which *players* (ought to) acquire proficiency in each atom. The exception is the *skill atom model* first articulated by Cook [14] and since extended by Deterding [21]. It expressly models game mechanics and their relation from the perspective of player learning. A skill atom describes a game loop between player and game comprising five elements:

1. *action*: the player invoking a mechanic (e.g. shooting);
2. *simulation*: the game processing the action according to rules and changing its internal game state (adjudicating whether the shot hit, changing the location and health score of the hit enemy);
3. *feedback*: the game informing the player (displaying an animation of the hit enemy);
4. *challenge*: the parameters that make executing this particular action differently easy or difficult; and
5. *synthesis*: the player incorporating the feedback, adjusting their mental model of the game state and improving the skill(s) required to master this particular atom (e.g. fast hand-eye coordination to aim and shoot).

Skill atoms exist in nested *skill chains*: directed graphs of the order in which skills build on each other and in which players necessarily or ideally acquire them [14]. For instance, a player has to know how to *equip* a gun before learning how to *aim* and *shoot* with it. Skill chains bottom out in *pre-existing skills*: capacities game designers can assume players already bring to the game. Most PC games assume that players know how to move and click a mouse, for instance. Figure 1 presents a simple skill chain of one pre-existing skill, two basic skills, and one advanced skill that builds on them. Figure 2 shows a skill chain for the game *Tetris*.

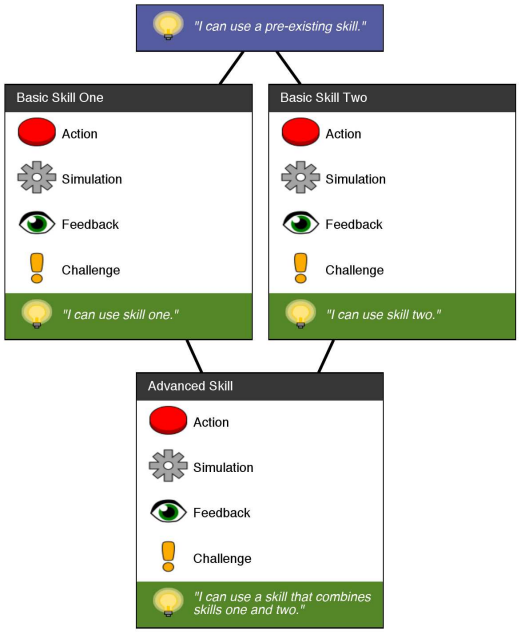

Figure 1. An example skill chain.

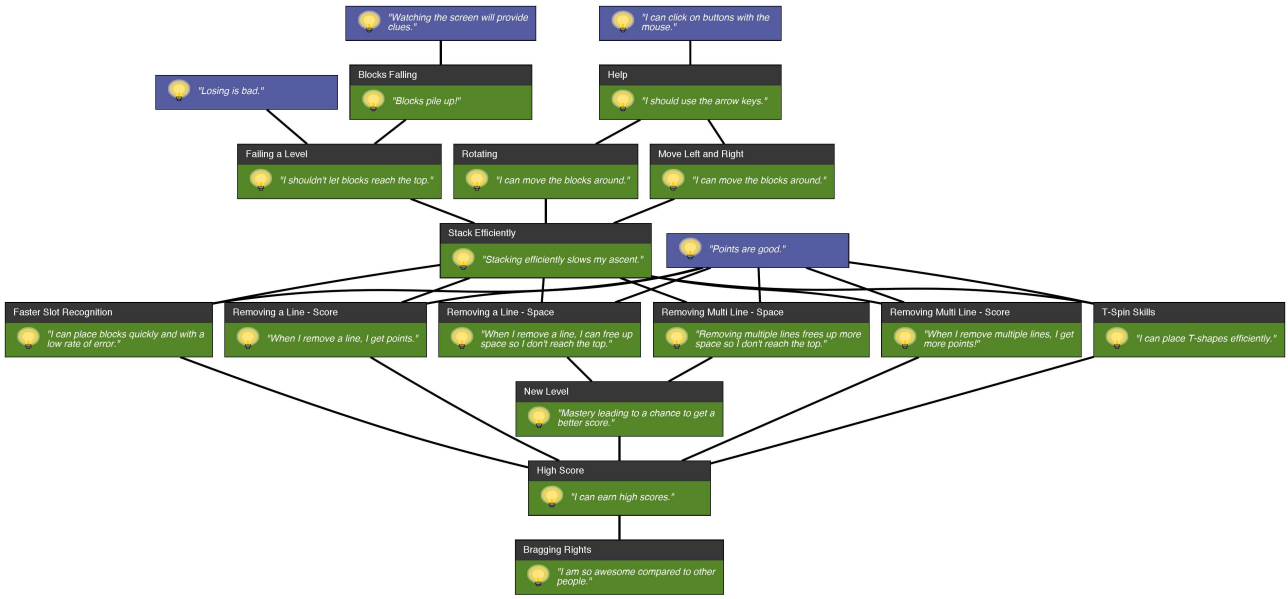

Figure 2. Skill chain for *Tetris*, taken and simplified from Cook [14].

Cook's model has found rich practical application, particularly in applied game design. For instance, Echeverría and colleagues [23] used it to improve an educational physics game. They analyzed which physical concepts the game ought to teach and which concepts it actually incorporated as skill atoms. Redesigning the game to incorporate previously missing skill atoms led to statistically significant learning improvements. Deterding's [21] method for gameful design similarly uses skill atoms to tease out the latent 'mini-games' of existing real-life activities and then redesign these to make them more explicitly and enjoyably game-like.

While not using Cook's [14] explicit articulation, Rational Level Design (RLD) [47, 48] has brought game atom analysis to broad use in entertainment game design, chiefly for difficulty balancing. Following flow theory [17], RLD assumes that players have an optimal or "flow" experience when the difficulty of challenges presented matches players' skill. As player skill grows over time, games have to increase difficulty in lockstep to avoid frustrating or boring players. This raises the question how to systematically design the *difficulty curve* of a game – the rate at which it increases difficulty. To this end, RLD suggests to identify (a) the atoms of a given game and (b) the parameters which affect the challenge of each game atom. For instance, the difficulty of "shooting" may be affected by parameters like enemy distance and speed. Designers should then craft a level sequence that systematically (a) introduces and involves new mechanics and (b) varies and increases the difficulty of the parameters of each atom. RLD essentially translates a synchronic map of game atoms into a recipe for diachronic level progression. However, RLD is chiefly interested in *difficulty* as an *aggregate effect* of the number of atoms involved and the configuration of their parameters. Unlike skill chains, it doesn't concern itself with logical or pragmatic dependencies – how skills build on each other.

Methods for Atom Identification

Either way, both skill chain mapping and RLD require means to elicit the actual skill atoms a game consists of. Cook [14], Echeverría [23] and Deterding [21] are notably silent about how they arrived at the skill atoms and chains they discuss. Where they mention the underlying process, it essentially bottoms out in "expert evaluation". This is a common issue of formal game analysis methods: most are some form of expert interpretation whose content and quality hinges on the unvalidated and tacit expertise of the reviewer. Guidance only concerns the format of the presented result, not the review process, leading to low replicability [44]. RLD [47, 48] similarly provides no method how to initially identify the atoms of a game and its parameters. Only once designers prototype actual levels based on hypothesized atoms and parameters does RLD loop in playtesting to assess the *actual* difficulty of each level as a player fail rate. From there, RLD focuses on iteratively understanding and tweaking the impact of atom parameters (enemy speed and distance) on difficulty. It offers no similar process to identify the game atoms themselves.

Existing playtesting and game user research methods are likewise of little help. No matter if based on player self-report, observation, psychophysiological measures, game telemetry, or a mixture thereof, they revolve around capturing constructs of player *experience* (like flow or immersion) and how game features affect these [5, 19, 24, 10, 39]. Closest to our concerns are heuristic analyses of game approachability – how easy a game is to learn [20] – and methods to balance game difficulty [33]. Yet again, both revolve around approachability or difficulty as aggregate results, not the underlying required

skills. For example, Linehan and colleagues [45] charted difficulty curves for four popular puzzle games by coding Let's Play videos for the number of actions required to solve a given level and when the game required a new skill. This provides an aggregate measure of difficulty and a description of the sequence in which skills are introduced by the game, but not the *learning* sequence in which these build on each other and *should* be introduced to players. The same holds for applied game design [50, 65]. A number of recent methods support the capture of a given game's game and learning mechanics, but chart them in the actual game design sequence, not the ideal learning sequence [3, 9].

A final source of potential methods is recent work merging intelligent tutoring systems with educational games. Butler and colleagues [8] for instance present a system for automatic game progression design for a game teaching fractions. The system models the algorithm required to solve all possible basic fraction problems, generates a large number of game levels, assesses each level's complexity on the number and kind of involved solution features (substrings of the total algorithm required to solve it), and serves players levels matching their measured mastery of solution features. While promising, this approach by definition only works for skills that are easily formalized into an algorithm, and offers no means of empirically identifying what skills a game requires and therefore needs to formalize. Harpstead and Aleven [29] use empirical learning curve analysis, a performance data analysis method from intelligent tutoring systems, to evaluate how well hypothesized models of player skills predict player success in an educational game. While this method does help assess whether there are hidden, non-modeled skills, again, it provides no means to empirically develop initial models.

In summary, skill atom chains formally model the component mechanics and skills of a game and their logical dependencies. Thus, they lend themselves readily to map a game's learning hierarchy. Current game user research, applied gaming, and intelligent tutoring research provide no reliable empirical method to identify the learning hierarchy or skill chain of a given game – the *actual* skills a player needs to acquire to master a game, and the *actual* order in which they build on each other and therefore *should* be introduced. Existing methods are limited to either (a) charting what mechanics a given game practically includes and introduces in what order (rather than *should* include or sequence to match the empirical learning hierarchy), (b) generating, testing, and optimizing level progression in terms of difficulty given an initial model, or (c) testing the statistical fit of a given model.

Cognitive Task Analysis

Faced with the same question – how to identify the skills involved in a domain – instructional design has developed a cluster of methods called Cognitive Task Analysis (CTA). CTA involves a variety of interview, observation, and modeling techniques to elicit and describe the knowledge and skills experts use to solve complex tasks [15]. CTA is the currently prevalent method for determining how people solve complex problems and for eliciting their learning hierarchies, forming the bedrock of any instructional design [35]. Recent systematic reviews suggest that basing instruction on CTA has strong positive effects on learning outcomes [62].

That said, there is no one single CTA. With over 100 CTA techniques available [15], choosing an appropriate method is challenging. A review by Wei and Slavendy [63] distinguishes four families of CTA methods and derive guidelines when to apply which: (1) more informal *observations and interviews* are advisable when the domain in question is very broad, ill-defined, or ill-understood; (2) more rigorous *process tracing* captures the actual steps and involved knowledge and skills of performing a given task through think-aloud or stimulated recall techniques, and is advised when exemplary tasks are easily identified; (3) *conceptual techniques* generate structured representations of domain concepts and their relations and are used to either analyze and represent data collected through other techniques, or when the domain in question mainly involves conceptual knowledge; (4) computer simulations testing *formal models* are used when task models already exist and quantitative predictions or measures are required. Combining multiple techniques is generally recommended to reduce errors and improve validity.

No matter what technique, CTA generally involves an iterative five-step process [13]:

1. *Collect preliminary knowledge to identify learning goals, tasks and subjects*: The analyst familiarizes themselves with the domain and desired learning outcomes to identify tasks to analyze and experts to recruit through e.g. document analysis, observation, or initial interviews.

2. *Identify knowledge types*: The analyst determines what kind of knowledge and skills the given tasks comprise and therefore, what specific elicitation, analysis and representation techniques are best suited (e.g., cooking a meal is a highly sequential task involving lots of tacit skills around preparation techniques, suggesting close observation and a flow chart as a representation).

3. *Apply focused knowledge elicitation methods*: The analyst uses the chosen techniques to elicit the knowledge and skills involved in the observed tasks. These typically involve some form of verbal report by the expert to surface covert cognitive processes.

4. *Analyze and verify data*: The analyst codes the generated data following the chosen method and produces initial representations of the involved skills and knowledge. Data and representations are cross-checked with the involved experts for potential errors and unclear points, and compared and contrasted between multiple elicitations to arrive at a final, integrated model.

5. *Format results for intended application*: The analyst prepares a formal presentation of the resulting model depending on the intended purpose of the CTA.

ADAPTING CTA FOR SKILL CHAIN ELICITATION

Given the maturity of CTA as a means for eliciting the skills involved in a given task, we decided to develop an adapted CTA method to identify the skill chain of a game. We were

especially encouraged in this as the skill atom model frames gameplay as a learning process of moving through an implicit learning hierarchy [14], and CTA is recommended specifically to identify learning hierarchies [35]. For each of the five steps of CTA methods, we will first explain why we chose specific techniques and adaptations. We will then give idealized step-by-step instructions for our final procedure to allow easy replication.

Method Development and Rationale

1. Collect preliminary knowledge to identify learning goals, tasks and subjects. In the case of game analysis, domain and learning goals are determined by the game in question and what counts as successfully completing it. The task is naturally a stretch of gameplay, which should be long enough for players to demonstrate the skills in question without putting undue hardship on subject or analyst. Depending on the size of the game, analysts may therefore want to focus on a particular aspect or stretch of the game. For instance, to analyze end-game raiding in an online role-playing game, which players often only access after dozens or hundreds of hours of gameplay, it may be advisable to focus on the first half an hour of an exemplary raid.

In terms of subject recruitment, most CTA techniques rely on subject matter experts. However, especially basic, low-level gameplay (like using controls) is a highly automated skill [12] that experts are rarely able to consciously explicate. A proven technique for foregrounding these skills is comparing novice and expert performance [55]. We conclude that unless the focus is on 'high-end' gameplay (such as end game raiding), recruiting a diverse set of expert *and* novice players of the game in question is a preferable strategy. As regards sample size, CTA gives no hard recommendations beyond recruiting more than one expert [13]. Given most CTA techniques are qualitative suggests adopting sampling criteria from qualitative research paradigms, most notably theoretical saturation: data collection should cease when additional data doesn't challenge the developed model anymore. Since a recent meta-analysis of qualitative interview methods suggest theoretical saturation is typically reached at around 12 or more participants [28], we chose 12 participants as our lower bound.

2. Identify knowledge types: Playing any game is a well-defined task that usually involves complex problem-solving with a wide variety of required skills and knowledge types [12], suggesting a process tracing technique [63]. Given our particular interest in skills, we chose Seamster and colleagues' [55] skill-based CTA (SBCTA) as a starting point. SBCTA combines a number of specific techniques to identify five types of cognitive skills that capture the range of skills required by video game play well: *automated* (e.g. hand-eye coordination), *procedural* (e.g. how to open menus), *representational* (mental models like predictions of enemy movement patterns), *decision-making*, and *strategies*. Seamster and colleagues and Wei [63] suggest to elicit automated and procedural skills through process tracing combined with verbal reports such as think-aloud. However, gameplay is highly cognitively involving, making parallel think-aloud problematic [31]. We therefore chose to use *stimulated recall*, likewise a common process tracing technique in CTA [16]. Here, the subject is video-recorded while performing the task in question. Afterwards, the analyst replays the video to the subject, stopping the video at relevant moments to ask the subject to explicate their thoughts and decision-making processes at the recorded time. This method allows the subject to perform tasks without interruption in a more natural setting while also cueing fresh memories and double-checking recall against actual recorded behavior, reducing false memories and post-rationalization [18]. For these and other reasons, variants of stimulated recall have been in active use in game research for some time [52, 7, 37, 38]. Following SBCTA, *representational* and *decision-making* skills are captured through the critical decision method [40] and error analysis, focused interview probing of moments in task performance when subjects made decisions or errors. Finally, *strategy* skills are likewise elicited with structured interview probing on decision points and/or scenarios [55].

3. Apply focused knowledge elicitation methods: Each subject is video-recorded playing the gameplay stretch investigated. Since gameplay occurs both on and in front of the screen, both should be captured and merged into a single picture-in-picture or picture-next-to-picture video file for replay and analysis [52, 59]. We decided to instruct players to think-aloud while they play *to the extent possible*, since think-aloud data provides additional cues and checks on the player's memory during stimulated recall [58, 61]. To elicit representational and decision-making skills via critical decisions and errors, the analyst watches the unfolding play and makes time-stamped notes on these incidents for focused follow-up. Indicators for relevant incidents are moments such as the player taking additional time to figure something out; struggling, pausing, or making errors; expressing an "aha" moment verbally or through body language; making a decision; or deviating from expected gameplay.

The play session is followed by a video-aided recall session that is also recorded. These generally follow a semi-structured interview pattern of initial scripted questions to elicit the subject's thinking at a given point, followed up by further, more open probing [46]. Concretely, we decided to show the player the record of each point in gameplay marked by the interviewer, and ask them (a) what elements of the game they interacted with or paid attention to, (b) what they were thinking at this point, and (c) why they took the action they took. These questions try to elicit procedural and automated knowledge around low-level gameplay as well as representational decision-making and strategy skills. Finally, subjects are asked what aspects of the game made it more or less difficult to complete the particular game goal at that point in order to identify the "challenge" component of the skill atom.

4. Analyze and verify data: Following standard procedures for stimulated recall analysis [52], the recall session record is transcribed as a structured, time-coded script of (a) the recall dialogue and (b) recorded gameplay and think-aloud verbalizations it refers to. To conduct analysis and cross-check transcripts against video data, we suggest using a computer-aided qualitative data analysis (CAQDA) software that can code and display text and video data. As skill atoms already

prescribe a clear unit of analysis, we adopted a directed qualitative content analysis method [32]: each unit of the first transcript is parsed for any *actions* the player takes in the game at that point. These actions are then contextualized in the video record and transcript to assess whether it forms part of a skill atom, meaning it involves some *simulation*, *game feedback*, *challenge*, and player *synthesis*.

The *simulation* portion of a coded atom can be determined by observing audiovisual feedback indicating a game state change, or additional knowledge of the game itself. *Feedback* is determined by observing the audiovisual record of gameplay and analyzing a subject's statements directly after an action is performed to see what feedback (if any) they noticed and (rightly or wrongly) interpreted as a result of their action. *Challenge* is explicitly derived from subject's statements about what makes a given moment of gameplay hard or easy to master, and implicitly from moments of pausing or failures at performing a given action. *Synthesis* can be derived from moments where the player explicitly voices a particular "aha" moment or demonstrates newly competent performance of an action. Each instance showcasing all five elements is coded as a skill atom. A second pass through the transcript codes for further instances of the identified skill atoms or their components, e.g. additional dimensions of challenge.

Dependencies between atoms in the skill chain are discovered through (a) analyzing the transcript for the sequence in which players showed or reported to learn a given atom, and (b) subsequent logical challenging whether the observed sequence expresses a necessary dependency or not. After analyzing the first transcript, transcripts of additional subjects are coded for the already-identified and additional skill atoms, also revising or refining prior skill atoms as needed.

5. Format results for intended application: Wei and Slavendy recommend conceptual CTA techniques such as visual diagramming to articulate and present the structure of knowledge of a domain [63]. Cook [14] already provides a visual diagramming language for skill chains, which we chose to adopt. Interestingly, skill chains parallel the graphical structure of concepts maps, a common conceptual tool for diagramming results of a CTA [51]: both are constructed of nodes representing a specific concept and edges connecting nodes that represent their relationships. We took this as convergent support for our choice. We programmed a script to automatically generate a visualization from a simple JSON file. True to the iterative nature of CTA [13], we found it useful to already sketch and iteratively revise and refine a draft diagram skill chain in parallel to data analysis.

Method Procedure

The following is a streamlined set of instructions for replicating the final methodology of our adapted CTA.

1. Identify analysis goals, tasks and subjects. Determine which game and particular aspect of its gameplay you wish to map as a skill chain. Choose a portion of gameplay that requires players to learn and/or demonstrate mastery of the focused aspect and does not overburden subjects – assume that interview sessions last at least double the time of the recorded gameplay stretch plus 20 minutes of briefing and debriefing. Unless you focus on a particular audience or gameplay aspect (e.g. end-game content), recruit a diverse pool of 12+ subjects that involves both novices and experts at the game.[1]

2. Elicit knowledge. Instruct subjects to play the selected stretch of gameplay, verbalizing what is going through their head as they do so. During gameplay, audiovisually record both on-screen game events and off-screen player activity and take notes including time stamps on critical moments when players (a) seem to make a decision; (b) struggle, pause, or make an error; (c) express an "aha" moment; or (d) deviate from expected gameplay. After the play session, replay the video recording to the subject. Fast forward to and play each critical moment you noted and ask the subject to verbalize (a) what game elements they were paying attention to or interacting with, (b) what was going through their mind at that point, (c) why they took the action they took, and (d) what aspects of the game made it more or less difficult to complete the particular game goal at that point.

3. Analyze data. Transcribe all stimulated recall sessions with time codes, noting (a) the recall dialogue and (b) recorded gameplay and think-aloud verbalizations it refers to. Upload video data and transcript to a CAQDA software that can display and code both. For analysis, parse each unit of the first transcript for *actions* the player takes. Contextualize each action in the video recording and transcript to determine whether it forms part of a skill atom comprising:

- an *action*,
- *simulation* or rule processing and game state change, based on recorded game screen feedback,
- game *feedback*, based on recorded game feedback and subject statements directly after an action indicating whether they (in)correctly perceived a game state change as feedback on their action,
- dimensions of *challenge*, based on subject statements about what makes a given moment of gameplay hard or easy, as well as play pauses or failures at performing a given action,
- moments of *synthesis* where the player voices insight into or demonstrates competent enactment of some required knowledge or skill connected to the action.

Code each instance showcasing all five elements as a skill atom and label it based on the main synthesis knowledge or skill. Cross-validate player-derived *simulation* with the actual game rules, code, and/or game designer to ensure these aren't player misconceptions. In a second pass, code the transcript for further instances of the identified skill atoms or their components. After identifying skill atoms, parse the transcript for *dependencies* between atoms expressed in when and/or what order players showed or reported to learn a given atom. Challenge each derived dependency by questioning whether the documented order is an incidental result of the game's design, or a logically necessary dependency. After analyzing the first transcript, code transcripts of additional subjects for

[1] Our results indicate a smaller n of 5+ may be sufficient, see below.

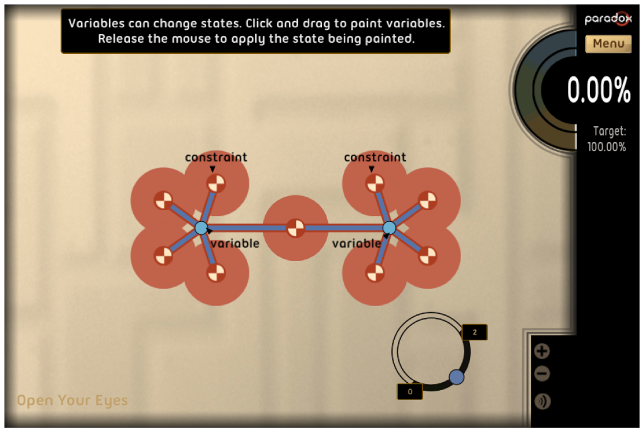

Figure 3. A screenshot of the game *Paradox* analyzed in our case study.

already-identified and additional skill atoms, iteratively revising or refining prior skill atoms and re-coding prior transcripts as needed.

4. Visualize skill chain. Already during data analysis, draft a first skill atom list and skill chain diagram as a reference and cross-check for coding. Once all transcripts are analyzed and have informed the draft skill chain, draw up a final clean skill chain diagram using the provided script[2].

CASE STUDY

Game and background

We developed and tested the above CTA method by eliciting the skill chain of the human computation game (HCG) *Paradox* (Figure 3). This was part of a larger project aimed at developing automatic level progression algorithms for HCGs that crowdsource scientific tasks like classifying images of galaxies. Notably, HCGs suffer from poor player retention at least partially due to poor progression design: instead of sequencing tasks in an order matching the learning curve of players, they predominantly serve tasks at random, risking both player frustration and boredom [53]. To inform machine learning algorithms that would automatically assess the difficulty of each task in *Paradox*, we wanted to get a grounded understanding of what component skills are required to play the game and thus, how difficult each *Paradox* task would be, depending on what skills it required (akin to [8]). *Paradox* was designed to crowdsource software verification, checking how many given conditions a given piece of code could satisfy. Each level or task presents players with a visualization of an underlying code piece as a graph of variables (displayed as nodes that can take different states) and conditions pertaining to variables (visualized as edges). Players can manually click individual variables to change their states or use various "brushes" to select subsets of variables to be modified. Different brushes trigger different approaches to modifying variables, from simple brushes that immediately set all to a certain value, to brushes that run specific optimization algorithms. The player's goal is to configure variables so that the highest possible number of conditions are satisfied. To complete a level, a player must reach a given target score or percentage of satisfied conditions. In general, it is not known in advance whether the target percentage of conditions (let alone all conditions) of a level can be satisfied.

Procedure, Observations and Reflections

In the following, we report on how we concretely implemented our method step by step and what generally relevant observations we made for that step.

Identify analysis goals, tasks and subjects. To make sure subjects were exposed to the same gameplay, we used a stable local version of *Paradox*[3] that featured seven tutorial levels introducing gameplay, followed by a fixed series of 20 challenge levels, which were generally larger, more open-ended and more difficult than the tutorials. Levels were chosen to cover a range of level sizes and likely solution strategies. Tutorial levels were gated: players had to complete a level by reaching its target score before being able to proceed to the next tutorial level. In contrast, players were able to skip challenge levels without completing them if they desired. We asked participants to play the game for 30 minutes, immediately followed by a 30 minute stimulated recall session. Gameplay length was determined by estimating how long players typically take to get through the tutorial and five challenge levels, which we assumed sufficient for novice players to acquire and demonstrate basic gameplay skills and for expert players to be challenged in the breadth of their expertise.

To record at least 12 subjects (ensuring theoretical saturation [28]), we recruited 15 subjects, preparing for a number of no-shows. We recruited 5 "expert" subjects who had played *Paradox* extensively in the past and 10 "novice" players who had never seen *Paradox* before, otherwise aiming for maximum diversity in gender, age, and socio-economic background.

Observations and Reflections. *Novices are more valuable than experts.* Interestingly, novices proved much more valuable for discovering low-level interface and gameplay skills than expert players. Indeed, analyzing expert gameplay only added minor refinements to the emerging skill chain. This somewhat contradicts standard CTA philosophy to rely on experts, but may be at least partially due to the relatively simple gameplay of *Paradox* or the fact that expert traces were analyzed last.

Quick saturation. We identified the vast majority of skill atoms during analysis of the first five recall transcripts, with subsequent transcripts adding only about one additional skill atom (3 percent of all codes) each. This suggests that future analyses may work sufficiently with a smaller number of subjects than we used – although this has to be tested with larger, more complex games.

Recognize and bracket shortcuts. At the conclusion of the first three recall sessions, we noticed that players heavily relied on the so-called optimizer brush. This brush automatically maximized satisfied conditions in a given graph area. While

[2]Available at http://github.com/crowdgames/skillchain.

[3]http://paradox.centerforgamescience.org/

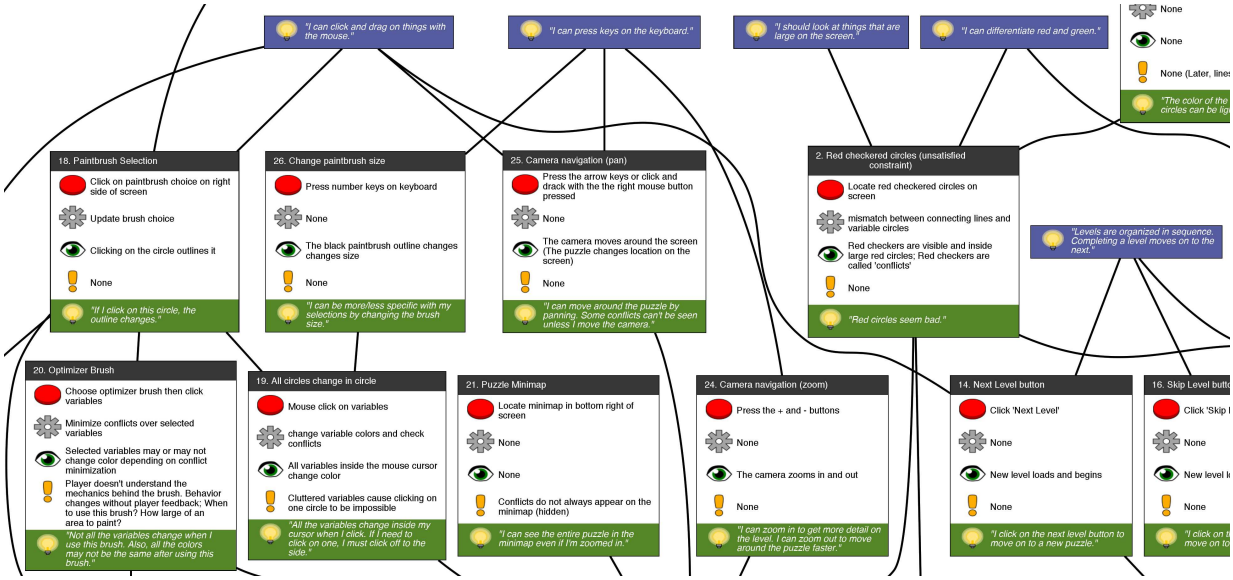

Figure 4. Detail of *Paradox* skill chain produced by our method.

generating a good first score, the global maximum of possible satisfied conditions cannot usually achieved with the optimizer; it requires deeper analysis and probing of the total graph. However, since the optimizer brush was introduced early on in the tutorial and was enough to complete early tutorial levels, novices tended to learn the heuristic to simply use the brush to clear each level, rather than learning how the constraint satisfaction mechanic worked and how to manually analyze and manipulate the graph. Hence, they would often become frustrated in later challenge levels when the brush alone didn't suffice, and were not able to switch to manual optimization. (One participant even said that the optimizer felt "like cheating" because it would do all the work without players understanding how.) In terms of CTA, this highlights that the availability of "power tools," "exploits," or "shortcuts" as part of the analyzed task can prevent certain procedural skills from being actively performed and thus made observable. Observed tasks should therefore ideally be trialled in advance of actual analysis to check for and eliminate undesired shortcuts. In our case, we later on manipulated the game and restricted two novices and one expert from using the optimizer brush at all, requiring them to manipulate each variable of a level individually. This helped discover particular skill atoms for novices as well as challenge features of graph layouts we hadn't observed before.

Elicit knowledge and analyze data. We began stimulated recall sessions with novice players as we assumed that their play would feature many critical moments foregrounding basic *Paradox* skills which expert players had already perfected and would therefore be hard to notice. Gameplay sessions were captured using *Morae*[4], which allows to make categorized notes during screen and camera recording that are logged on the recording timeline. We used this to log critical moments we would then replay to subjects to stimulate recall after the play session. Stimulated recall sessions were recorded with *Camtasia Studio*[5], as this program allowed to display and screen capture play session video and audio in addition to the camera video and audio of analyst and subject conversing. Stimulated recall sessions were transcribed and coded using *MaxQDA*[6].

Observations and Reflections. *Skill dependencies are unclear and confounded by level design.* We found it hard to identify clear dependencies between skill atoms and to disentangle (a) the order in which the game's progression design required certain skills, (b) the order in which players developed insights, and (c) the ideal learning hierarchy in which both should occur. Instead, the order in which the game tutorial introduced skill atoms strongly shaped player perceptions and analyst coding: both tended to state that the order in which the game presents skill atoms is the order in which players learned them. This indicates a strong limitation in using qualitative analysis of fixed games with fixed progressions for eliciting skill chains. Ideally, players would be exposed to a randomized multivariate ordering of skill atoms to observe which order empirically produces the fastest learning gains.

Strategy skills are unvalidated. In alignment with Seamster and colleagues' [55] hierarchy of cognitive skills, we found *strategies* to sit at the 'highest' level of our skill chain. The distinction between procedural and strategy skills is fuzzy. We found a good indicator for strategy skills to be that players consciously identified different approaches or composite applications of procedural skills and chose between them based on context. For example, players chose from what location to begin solving *Paradox* levels and in which order to move through the graph depending on level geometry. Sometimes a player would work from the periphery to the center, other times a player would choose to start in an area that had the

[4]https://www.techsmith.com/morae-features.html

[5]https://www.techsmith.com/camtasia.html

[6]http://www.maxqda.com/

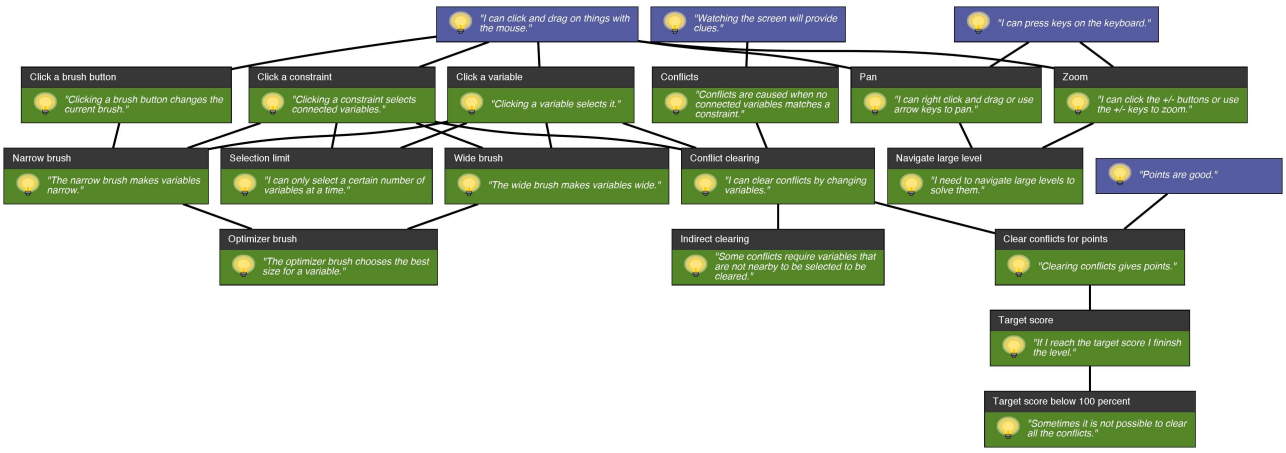

Figure 5. Simplified *Paradox* skill chain hand-authored by one of the game's designers.

most conflicts, and players would generally verbalize that and why they chose this particular strategy. That said, it is hard to tell from our data whether and how optimal any of these strategies in a particular context actually are. (Notably, the same holds for CTA, which simply assumes that expert practice is self-validated as best practice.) It would be good to triangulate our qualitative data with quantitative data on the relative performance of different strategies, in the same way players and teams in eSports analyze the performance of different characters or items[7].

Also, conceptual CTA techniques focus on mapping the cognitive skills and knowledge of a task rather than individual strategies, meta-strategies, and conditions when to employ them [63]. To a certain extent, one could argue that the three strategy skill atoms we mapped are really on the skill atom "choosing optimal strategies". Hence, CTA may be less apt at analyzing and visualizing strategies and strategy-heavy games.

Visualize skill chain. A detail of the final skill chain we created for *Paradox* can be seen in Figure 4. The full skill chain is given as supplementary information. To assess the produced skill chain, we asked one of the designers of *Paradox* to draw a skill chain based off of their understanding of the skills necessary for the game, which can be seen in Figure 5.

Observations and Reflections. *Skill chain analysis surfaces low-level and pre-existing skills.* While both designer and CTA-derived skill chain covered the same basic mechanics, the CTA-generated chain is far more detailed and comprehensive. First, it entails many required pre-existing skills. For instance, we discovered that the game was not accessible for people with blue-yellow and green-red color blindness, who had difficulty recognizing the color-coded states of the variables. Second, comparing novice and expert gameplay brought to light that experts used certain low-level skills that were not explicitly conveyed to players in the tutorial and therefore not used by novice players. One example is changing brush sizes. Because expert players had explored *Paradox* and its interface more deeply, they had discovered how to change the brush size, which allowed them more control over the variables selected. This observation arguably demonstrates the most direct value of our method: the tutorial, based on the designers' hypothetical skill chain, overlooked parts of the actual empirical skill chain of players – quite possibly since designers are expert players who therefore have difficulty recognizing the low-level, highly automated skills they possess but novices don't. That said, it is an open question whether the same insights could not be generated more efficiently through standard usability and playability testing.

Skill chains run together in a core mechanic. Interestingly, unlike the designer-generated skill chain, the CTA-generated chain eventually runs together in one central node, "efficiently reduce number of conflicts", which then branches out into strategies for achieving such efficient reduction. Discussions with the game's designers and our own gameplay experience suggest that this central node is indeed the "core mechanic" of *Paradox* [56]. We take this as further validation of our method and find it suggestive for formal game analysis more broadly: core mechanics or loops are the graph-theoretically most central nodes in which all dependencies and subskills run together.

Skill chains remain flat. Overall, the skill chain we elicited had a flat, "pancake" quality: it consists of many fundamental skills around using the interface without many dependencies between or beyond them. The same flat structure can be seen in Cook's skill chain of *Tetris* [14], while the skill chain of *Pac-Man* shows greater depth [25]. This may be due to many things: the relative simplicity of *Paradox* and *Tetris* compared to a greater gameplay depth of *Pac-Man*; the subjectivity of involved analysts; or the general complexity of the underlying genre. It is worth noting that all published uses of skill atoms cover relatively simple puzzle and action games. Hence, it is an open question whether (a) different game genres and more complex games would produce different skill chains, and whether (b) skill chain mapping is feasible or productive for

[7] See e.g. https://www.dotabuff.com/heroes/winning

more complex games or whether graphs become too unwieldy to be of much use.

DISCUSSION

Reflecting on our case study, we think it warrants the conclusion that our method worked: we were able to elicit a skill chain from gameplay that roughly mapped the understanding of one of the game's designers, identified the correct "core loop", and produced useful design insight. Particularly, it surfaced a range of overlooked prerequisite and low-level skills that had eluded the designer's attention and made the game more challenging to learn and play for novices. A second main observation vis-à-vis CTA is that observing novices learning how to play a game proved potentially more valuable than observing smooth expert performance.

That said, our case study also surfaced a series of major challenges and limitations. First, it is unclear whether standard usability and playability testing methods wouldn't be more efficient in producing the same insights into overlooked low-level gameplay skills. Although our case study suggests that skill chains can be elicited with a smaller (5+) *n* than we used, the method remains quite involved, using about 10 hours per subject (1 interview, 6 transcription, 3 analysis). Likewise, it is unclear how approachable our method is for designers and game researchers. In future work, we would therefore like to run method variants with other analysts to assess ease of use, approachability, perceived efficacy, and outcomes, and to trial the method with smaller *n*'s and with direct video analysis that doesn't rely on transcription.

Now standard playability and usability testing don't provide learning hierarchies to inform e.g. level design or procedural content adaptation or generation. However, this arguable main differentiator of our method – identifying the dependencies between skills – also proved the most elusive. Actual ideal dependencies were hard to ascertain, and the resultant skill chain featured little depth. We assume this is partially due to the simplicity of *Paradox* and the fact that the game's actual fixed progression sequence strongly biased observation. In future work, we would therefore want to use process tracing on experimental variations of skill sequencing, and replicate our method with more complex games, which brings us to a further limitation: We tested the method with a very simple puzzle game. It is unclear whether it would work with different genres or more complex games. Action games rich in automated 'twitch' skills may prove harder to analyze, and 'big' games like the MMORPG *EVE Online* may involve so many skill atoms that eliciting them in short gameplay sessions or mapping them in a single chain could prove unwieldy.

A final challenge and limitation concerns strategy skills. While we could elicit a number of emergent strategies actively used by players, our method cannot speak to how empirically optimal these strategies actually are. Here, combinations of our qualitative analysis and quantitative game analytics would be useful. Indeed, we view novel mixed method combinations of qualitative 'thick data' methods like CTA with 'big data' analytics to be a highly promising future direction, as many data-driven methods of tutorial or progression design and analysis [60, 8, 30] in turn lack exactly the specificity and insight into what particular skills to analyze for or generate from that qualitative data provides.

CONCLUSION

Like designing good instruction, designing an enjoyable, easy to learn game requires understanding what skills the game's mechanics require, how these build and depend on each other, and thus, in what order to introduce them to players. Learning hierarchies in instructional design and skill chains in game design are common formal models to map these relations. Yet where instructional design can rely on cognitive task analysis to empirically identify the learning hierarchy of a task, game design so far relied on expert interpretation to identify the skill chains of games. Given that experts are typically blind to the full range of tacit skills they have mastered, this risks overlooking crucial skills that novices need to be taught. In this paper, we therefore developed and presented an adapted cognitive task analysis method to elicit a game's skill chain from empirical player observation and interviewing. The method combines stimulated recall interviews on targeted stretches of gameplay with directed qualitative content analysis of the generated data. We demonstrated and critically reflected on the method through a case study use on the game *Paradox*. The skill chain elicited for *Paradox* with our method indeed proved aligned with but more comprehensive than a designer-crafted skill chain produced without empirical player observation. Specifically, it included critical missing pre-requisite and low-level skills.

While principally effective in our case study, the method also showed major limitations and open questions regarding its efficiency, generalizability across genres and more complex games, and ability to reliably elicit skill dependencies, and validity of captured emergent player strategies, which we hope to address in future applications with replications and mixed method approaches.

ACKNOWLEDGEMENTS

This work was supported by a Northeastern University TIER 1 grant and partly conducted in the Digital Creativity Labs (digitalcreativity.ac.uk), jointly funded by EPSRC/AHRC/InnovateUK under grant no. EP/M023265/1. This material is based upon work supported by the National Science Foundation under grant no. 1652537. We would like to thank the University of Washington's Center for Game Science for initial *Paradox* development.

REFERENCES

1. Ernest Adams and Joris Dormans. 2012. *Game Mechanics: Advanced Game Design*. New Riders, Berkeley, CA.
2. Marcos Silvano Orita Almeida and Flavio Soares Correa da Silva. 2013. A Systematic Review of Game Design Methods and Tools. In *ICEC 2013*, J.C. Anacleto (Ed.). Springer, 17–29.
3. Sylvester Arnab, Theodore Lim, Maira B. Carvalho, Francesco Bellotti, Sara De Freitas, Sandy Louchart, Neil Suttie, Riccardo Berta, and Alessandro De Gloria. 2015. Mapping learning and game mechanics for serious games

analysis. *British Journal of Educational Technology* 46, 2 (2015), 391–411.

4. Melissa Baralt, Roger Gilabert, and Peter Robinson. 2014. An introduction to theory and research in task sequencing and instructed second language learning. *Task sequencing and instructed second language learning* (2014), 1–50.
5. Regina Bernhaupt. 2010. User Experience Evaluation in Entertainment. In *Evaluating User Experience in Games: Concepts and Methods*, Regina Bernhaupt (Ed.). Springer London, London, 3–7.
6. Staffan Björk and Jussi Holopainen. 2005. *Patterns in Game Design*. Charles River Media, Boston, MA.
7. Joceran Borderie and Nicolas Michinov. 2016. Identifying Flow in Video Games: Towards a New Observation-Based Method. *International Journal of Gaming and Computer-Mediated Simulations (IJGCMS)* (2016).
8. Eric Butler, Erik Andersen, Adam M Smith, Sumit Gulwani, and Zoran Popovic. 2015. Automatic Game Progression Design through Analysis of Solution Features. *Chi'15* (2015), 2407–2416.
9. Maira B. Carvalho, Francesco Bellotti, Riccardo Berta, Alessandro De Gloria, Carolina Islas Sedano, Jannicke Baalsrud Hauge, Jun Hu, and Matthias Rauterberg. 2015. An activity theory-based model for serious games analysis and conceptual design. *Computers and Education* 87 (2015), 166–181.
10. Judeth Oden Choi, Jodi Forlizzi, Michael Christel, Rachel Moeller, MacKenzie Bates, and Jessica Hammer. 2016. Playtesting with a Purpose. *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play - CHI PLAY '16* (2016), 254–265.
11. Dough Church. 1999. Formal Abstract Design Tools. (1999).
12. D. B. Clark, E. E. Tanner-Smith, and S. S. Killingsworth. 2016. Digital Games, Design, and Learning: A Systematic Review and Meta-Analysis. *Review of Educational Research* 86, 1 (2016), 79–122.
13. Richard E. Clark, D. Feldon, Jeroen JG van Merriënboer, Kenneth Yates, and Sean Early. 2008. Cognitive task analysis. *Handbook of research on educational communications and technology* 3 (2008), 577–593.
14. Daniel Cook. 2007. The chemistry of game design. `http://www.gamasutra.com/view/feature/1524/the_chemistry_of_game_design.php`. (2007).
15. Nancy J Cooke. 1994. Varieties of knowledge elicitation techniques. *International Journal of Human-Computer Studies* 41, 6 (1994), 801–849.
16. Beth Crandall, Gary Klein, and Robert Hoffman. 2006. *Working Minds: A Practitioner's Guide to Cognitive Task Analysis*.
17. Mihaly Csikszentmihalyi. 1990. *Flow: the psychology of optimal experience*. Harper and Row, New York.
18. Nicholas P. Dempsey. 2010. Stimulated Recall Interviews in Ethnography. *Qualitative Sociology* 33, 3 (2010), 349–367.
19. Heather Desurvire and Magy Seif El-Nasr. 2013. Methods for Game User Research: Studying Player Behavior to Enhance Game Design. *IEEE Computer Graphics and Applications* 33, 4 (2013), 82–87.
20. Heather Desurvire and Charlotte Wiberg. 2015. User Experience Design for Inexperienced Gamers: GAP – Game Approachability Principles. In *Game User Experience Evaluation*, Regina Bernhaupt (Ed.). Springer, London, 131–147.
21. Sebastian Deterding. 2015. The lens of intrinsic skill atoms: a method for gameful design. *Human-Computer Interaction* 30, 3-4 (2015), 294–335.
22. Joris Dormans. 2012. *Engineering Emergence: Applied Theory for Game Design*. PhD Thesis. Universiteit van Amsterdam.
23. Alejandro Echeverría, Enrique Barrios, Miguel Nussbaum, Matías Améstica, and Sandra Leclerc. 2012. The Atomic Intrinsic Integration Approach: A structured methodology for the design of games for the conceptual understanding of physics. *Computers & Education* 59, 2 (2012), 806–816.
24. Magy Seif El-Nasr, Anders Drachen, and Alessandro Canossa (Eds.). 2013. *Game Analytics: Maximizing the Value of Player Data*. Springer, London. 815 pages.
25. firecorth. 2013. Game Design Alchemy the Second. (2013).
26. Robert M Gagne. 1968. Presidential address of division 15 learning hierarchies. *Educational psychologist* 6, 1 (1968), 1–9.
27. James-Paul Gee. 2003. *What Video Games Have to Teach Us About Learning and Literacy*. Palgrave Macmillan, New York.
28. Greg Guest, Arwen Bunce, and Laura Johnson. 2006. How Many Interviews Are Enough?: An Experiment with Data Saturation and Variability. *Field Methods* 18, 1 (2006), 59–82.
29. Erik Harpstead and Vincent Aleven. 2015. Using Empirical Learning Curve Analysis to Inform Design in an Educational Game. In *CHI Play 2015*. 197–207.
30. Erik Harpstead, Thomas Zimmermann, Jose J Guajardo, Ryan Cooper, Tyson Solberg, and Dan Greenawalt. 2015. What Drives People : Creating Engagement Profiles of Players from Game Log Data. (2015), 369–379.
31. Jettie Hoonhout. 2008. Let the Game Tester Do the Talking: Think Aloud and Interviewing to Learn About the Game Experience. In *Game Usability: Advice from the Experts for Advancing the Player Experience*, Katherine Isbister and Noah Schaffer (Eds.). Morgan Kaufmann, Amsterdam et al., 65–78.

32. Hsiu-Fang Hsieh and Sarah E Shannon. 2005. Three approaches to qualitative content analysis. *Qualitative health research* 15, 9 (Nov 2005), 1277–88.
33. Robin Hunicke. 2005. The case for dynamic difficulty adjustment in games. *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology - ACE '05* (2005), 429–433.
34. Robin Hunicke, Marc LeBlanc, and Robert Zubek. 2004. MDA: A Formal Approach to Game Design and Game Research. In *Papers from the 2004 AAAI Workshop "Challenges in Game Artificial Intelligence"*, Dan Fu, Stottler Henke, and Jeff Orkin (Eds.). AAAI Press, Menlo Park, Ca, 1–5.
35. David H. Jonassen, Martin Tessmer, and Wallace H. Hannum. 1998. *Task Analysis Methods for Instructional Design*. Routledge.
36. Rilla Khaled and Gordon Ingram. 2012. Tales from the front lines of a large-scale serious game project. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 69–78.
37. David Kirschner and J Patrick Williams. 2013. Experts and Novices or Expertise? Positioning Players through Gameplay Reviews. In *DiGRA 2013: DeFragging Game Studies*. DiGRA, Snowbird, UT.
38. D. Kirschner and J. P. Williams. 2015. Measuring Video Game Engagement Through Gameplay Reviews. *Simulation & Gaming* 45, 4-5 (2015), 593–610.
39. J. Matias Kivikangas, Guillaume Chanel, Ben Cowley, Inger Ekman, Mikko Salminen, Simo Järvelä, and Niklas Ravaja. 2011. A review of the use of psychophysiological methods in game research. *Journal of Gaming & Virtual Worlds* 3, 3 (2011), 181–199.
40. G. A. Klein, R. Calderwood, and D. MacGregor. 1989. Critical decision method for eliciting knowledge. *IEEE Transactions on Systems, Man, and Cybernetics* (1989).
41. Christina Koeffel, Wolfgang Hochleitner, Jakob Leitner, Michael Haller, Arjan Geven, and Manfred Tscheligi. 2010. Using Heuristics to Evaluate the Overall User Experience of Video Games and Advanced Interaction Games. In *Evaluating User Experience in Games*, Regina Bernhaupt (Ed.). Springer, London, 233–255.
42. Raph Koster. 2004. *A Theory of Fun for Game Design*. Paraglyph Press, Scottsdale, AZ.
43. Raph Koster. 2013. *Theory of fun for game design*. O'Reilly Media, Inc.
44. Petri Lankoski and Staffan L Björk. 2015. Formal analysis of gameplay. In *Game Research Methods: An Overview*, Petri Lankoski and Staffan L Björk (Eds.). ETC Press, Pittsburgh, PA, 23–36.
45. Conor Linehan, George Bellord, Ben Kirman, Zachary H Morford, and Bryan Roche. 2014. Learning Curves: Analysing Pace and Challenge in Four Successful Puzzle Games. In *CHI Play'14*. 181–190.
46. John Lyle. 2010. Stimulated recall: a report on its use in naturalistic research. *British Educational Research Journal* 29, 6 (2010), 861–878.
47. Chris McEntee. 2012. Rational Design: The Core of Rayman Origins. `http://www.gamasutra.com/view/feature/167214/rational_design_the_core_of_.php`. (2012).
48. Luke McMillan. 2013. The Rational Design Handbook: An Intro to RLD. (2013).
49. Jeroen J. G. van Merriënboer, Richard E. Clark, and Marcel B. M. de Croock. 2002. Blueprints for complex learning: The 4C/ID-model. *Educational Technology Research and Development* 50, 2 (2002), 39–61.
50. Lennart Nacke, Anders Drachen, and Stefan Gobel. 2010. Methods for Evaluating Gameplay Experience in a Serious Gaming Context. *International Journal of Computer Science in Sport* 9, 2 (2010).
51. Joseph D Novak and Alberto J Cañas. 2008. The theory underlying concept maps and how to construct and use them. (2008).
52. Jori Pitkanen. 2015. Studying thoughts: Stimulated recall as a game research method. In *Game Research Methods*, Petri Lankoski and Staffan L Björk (Eds.). ETC Press, 117–132.
53. Anurag Sarkar, Michael Williams, Sebastian Deterding, and Seth Cooper. 2017. Engagement Effects of Player Rating System-Based Matchmaking for Level Ordering in Human Computation Games. In *FDG'17*. ACM Press, New York.
54. Donald A Schon. 1984. *The reflective practitioner: How professionals think in action*. Basic books.
55. Thomas L. Seamster, Richard E. Redding, and George L. Kaempf. 2000. A skill-based cognitive task analysis framework. In *Cognitive task analysis*, Jan Maarten Schraagen, Susan F. Chipman, and Valerie L. Shalin (Eds.). Lawrence Erlbaum, Mahwah, NJ, 135–146.
56. Miguel Sicart. 2008. Defining game mechanics. *Game Studies* 8, 2 (2008), 1–14.
57. Miguel Sicart. 2015. Loops and Metagames: Understanding Game Design Structures. In *Foundations of Digital Games*.
58. M. W. Someren, Y. F. Barnard, and J. a. C. Sandberg. 1994. *The think aloud method: a practical approach to modelling cognitive processes*. Academic Press.
59. Reed Stevens, Tom Satwicz, and Laurie McCarthy. 2008. In-Game, In-Room, In-World: Reconnecting Video Game Play to the Rest of Kids' Lives. In *The Ecology of Games: Connecting Youth, Games, and Learning*, Katie Salen (Ed.). MIT Press, Cambridge, MA, 41–66.

60. Joseph J Thompson, Mark R Blair, Lihan Chen, and Andrew J Henrey. 2013. Video Game Telemetry as a Critical Tool in the Study of Complex Skill Learning. *PLoS ONE* 8, 9 (2013), e75129.

61. Bonnie L Tjeerdsma. 1997. A comparison of teacher and student perspectives of tasks and feedback. *Journal of teaching in physical education* 16, 4 (1997), 388–400.

62. Colby Tofel-Grehl and David F. Feldon. 2013. Cognitive Task Analysis–Based Training. *Journal of Cognitive Engineering and Decision Making* 7, 3 (2013), 293–304.

63. June Wei and Gavriel Salvendy. 2004. The cognitive task analysis methods for job and task design: review and reappraisal. *Behaviour & Information Technology* 23, 4 (2004), 273–299.

64. Richard T White and Robert M Gagné. 1974. Past and future research on learning hierarchies 1. *Educational psychologist* 11, 1 (1974), 19–28.

65. Pieter Wouters, Erik D. van der Spek, and Herre van Oostendorp. 2009. Current practices in serious game research: A review from a learning outcomes perspective. In *Games-Based Learning Advancements for Multi-Sensory Human Computer Interfaces: Techniques and Effective Practices*, Thomas Connolly, Mark Stansfield, and Liz Boyle (Eds.). IGI Global, Hershey, PA, 232–250.