

Languages of Games and Play: A Systematic Mapping Study

RIEMER VAN ROZEN, Centrum Wiskunde & Informatica, The Netherlands and Amsterdam University of Applied Sciences, The Netherlands

Digital games are a powerful means for creating enticing, beautiful, educational, and often highly addictive interactive experiences that impact the lives of billions of players worldwide. We explore what informs the design and construction of good games to learn how to speed-up game development. In particular, we study to what extent *languages*, *notations*, *patterns*, and *tools*, can offer experts theoretical foundations, systematic techniques, and practical solutions they need to raise their productivity and improve the quality of games and play. Despite the growing number of publications on this topic there is currently no overview describing the state-of-the-art that relates research areas, goals, and applications. As a result, efforts and successes are often one-off, lessons learned go overlooked, language reuse remains minimal, and opportunities for collaboration and synergy are lost. We present a systematic map that identifies relevant publications and gives an overview of research areas and publication venues. In addition, we categorize research perspectives along common objectives, techniques, and approaches, illustrated by summaries of selected languages. Finally, we distill challenges and opportunities for future research and development.

CCS Concepts: • General and reference → Surveys and overviews; • Applied computing → Computer games; • Software and its engineering → Domain specific languages; Visual languages; Design languages;

Additional Key Words and Phrases: Game development, game design, languages, notations, patterns, tools, systematic map

ACM Reference format:

Riemer van Rozen. 2020. Languages of Games and Play: A Systematic Mapping Study. *ACM Comput. Surv.* 53, 6, Article 123 (December 2020), 37 pages.

<https://doi.org/10.1145/3412843>

1 INTRODUCTION

In the past decades, digital games have become a main podium for creative expression enabling new forms of play and interactive experiences that captivate and enchant like the works of great historical writers, painters, artists, and composers. The game development industry is a vast and lucrative branch of business that eclipses traditional arts and entertainment sectors, outgrowing even the movie industry [53]. Games reach audiences around the world, unite players in common activities, and give rise to subcultures and trends that impact pastime, awareness, and policies of modern societies.

nwo/SIA grants: Early Quality Assurance in Software Production, Automated Game Design and Live Game Design.
Authors' address: R. van Rozen, Centrum Wiskunde & Informatica, P.O. Box 94079, 1090 GB, Amsterdam, The Netherlands, Amsterdam University of Applied Sciences, Rhijnspoorplein 1, 1091 GC, Amsterdam, The Netherlands; email: rozen@cwi.nl.



This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2020 Copyright held by the owner/author(s).

0360-0300/2020/12-ART123

<https://doi.org/10.1145/3412843>

However, for every outstanding success exist many games with unrealized potential and failures that preceded bankruptcy. Developing high-quality games is dreadfully complicated, because game design is intrinsically complex. We wish to learn what informs the design of good games to help speed-up the game development process for creating better games more quickly. In particular, we study to what extent *languages*, *structured notations*, *patterns*, and *tools*, can offer designers and developers theoretical foundations, systematic techniques, and practical solutions they need to raise their productivity and improve the quality of games and play.

We propose the term “*languages of games and play*” for language-centric approaches for tackling challenges and solving problems related to game design and development. Despite the growing number of researchers and practitioners that propose and apply these languages, there is currently no overview of publications that relates languages, goals, and applications. As a result, publications on the topic lack citations of relevant related work. In addition, lessons learned are overlooked and available methods, and techniques for language development often remain unused. As a consequence, it remains difficult to compare and study games, designs, and research contributions to build bodies of knowledge that describe best practices and industry standards.

We aim to map the state-of-the-art of languages of games and play in an understandable way, such that it is accessible to a wide audience. Our goal is to provide a means for (1) informing practitioners and researchers about the breadth of related work; (2) sharing knowledge between research areas and industry for improved results and collaboration; (3) enabling the application of available techniques; and (4) identifying opportunities for future research and development.

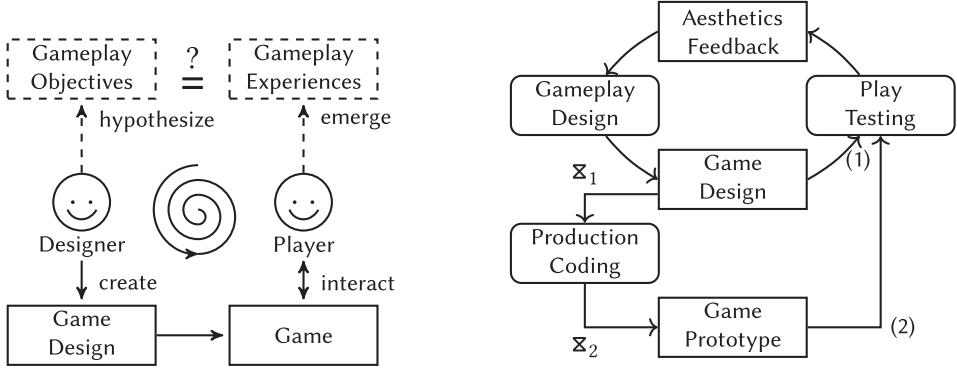
Our research questions are summarized as follows:

- Which publication venues include papers on languages of games and play?
- How do the various approaches compare?
- What are open research challenges and opportunities for future work?

For answering these questions, we conduct a survey of languages of games and play called a *systematic mapping study*. Mapping studies provide a wide overview of a research area by identifying, categorizing, and summarizing all existing research evidence that supports broad hypotheses and research questions [26]. In contrast, systematic literature reviews usually have a more narrow focus, and instead perform in-depth analyses to answer particular research questions. Both enjoy the benefits of a well-defined methodology for (re)producing high-quality results and reducing bias.

We identify and analyze relevant publications on languages of games and play. First, we motivate the need for this study by describing its scope in Section 2. Next, we describe the methodology with research questions, sources, queries and inclusion criteria, and a review protocol in Sections 3 and 4. We contribute the following:

- (1) A systematic map on languages of games and play that provides an overview of research areas and publication venues, presented in Section 5.
- (2) A set of 14 complementary research perspectives on languages of games and play synthesized from summaries of over 100 distinct languages we identified in over 1,400 publications, presented in Section 6.
- (3) An analysis of general trends and success factors of the identified research, and one unifying specific perspective on “*automated game design*,” which discusses challenges and opportunities for choosing directions in future research and development, presented in Section 7.



(a) Game designers form hypotheses about how a game’s parts realize gameplay goals, but usually repeatedly find intended and actual experiences differ.

(b) Developing a high-quality game entails iteratively designing, play-testing, and improving its design as a paper (1) or a software prototype (2)

Fig. 1. Game development aims for games with high-quality player experiences.

We describe related work in Section 8, discuss threats to validity in Section 9, and conclude in Section 10. Our map provides a good starting point for anyone who wishes to learn more about the topic. We maintain an accompanying website here: <https://vrozen.github.io/LoGaP/>.

2 RESEARCH VISION

A mapping study on games and play can be approached from different research perspectives, each with different goals and needs. We introduce fundamental concepts and illustrative challenges of digital game design in Section 2.1, and we formulate two general hypotheses that drive this study in Section 2.2. Our specific motivation is to automate game design and investigate how domain-specific language technology, introduced in Section 2.3, can offer solutions. We clarify our position and motivate this study in Section 2.4.

2.1 Games and Play

Games and play are inextricably intertwined concepts. Games bring about experiences such as enjoyment, persuasion, and learning. Developing digital games such as puzzles, adventures, lessons, or treatments requires combining diverse technical and non-technical expertise. Game development teams typically consist of game designers, artists, and software engineers. However, the collaborative process may also include educators, level designers, AI programmers, narrators, or healthcare professionals. Perhaps the most crucial for a successful outcome is the role of the *game designer*, who is primarily responsible for the quality of player experiences.

Game design [17, 40], the discipline and process of iteratively designing and improving games, is an instance of a so-called *wicked problem*, a problem that is “*difficult to solve in general due to incomplete, contradicting and evolving requirements*” [11, 13, 31]. We highlight several challenges as examples that illustrate its inherent complexity.

Improving a game’s qualities depends on gradually improving insight, as illustrated by Figure 1(a). Game designers use *paper prototyping* to explore and understand the problem at hand, abstracting away a game’s details until what remains is essential. They form hypotheses about play, experiment with rules and objectives to evolve a game’s design, and learn what the solution can become. For instance, designers create interaction mechanisms (a.k.a. game mechanics, or rules) offering playful affordances [39].

Players interact with games via these mechanisms during a game's execution. Playful acts result in dynamic interaction sequences. Ideally, these also represent aesthetically pleasing experiences called *gameplay*, e.g., fellowship, challenge, fantasy, narrative, discovery, or self-expression [24]. However, opinions on a game's quality differ from person to person, e.g., with age, gender, and beliefs.

For game designers *play-testing* is essential for verifying assumptions and learning if a game meets its objectives. More often than not, designers discover that the realized and intended gameplay differ. Unfortunately, even well-prototyped games may fail to meet expectations as fully developed software. In general, it is hard to predict the outcome of modifying a game's parts, e.g., how changing the rules affects the dynamics and aesthetics of play. As a result, steering towards new goals is difficult.

Improving a game is never truly done. The maximum number of game design iterations determines the achievable quality. Efforts on balancing, fine-tuning, and polishing are limited only by time and money. Resource-wise, AAA studios have a competitive advantage over indie game developers. However, developing novel high-quality games in a time-to-market manner is universally hard, because game design iterations take simply too much time. Figure 1(b) shows an abstract game development process that illustrates two important root causes of delay (shown as χ).

Game designers and software engineers usually live on opposite sides of the fence [27]. Both lose time when adjustments best understood by designers have to be implemented by software engineers (χ_1). To evolve a game, designers have to explore alternative gameplay scenarios, constantly requiring changes.

As time progresses, more and more choices become fixed, and frequent changes to the source code become more difficult, time-consuming, and error-prone (χ_2). The evolution of digital games, like other software, suffers from a well-known phenomenon called *software decay* [33]. The software quality deteriorates with frequent changes to the source code made to accommodate evolving requirements. As a consequence, game designers have precious few chances to experiment with design alternatives. This seriously compromises their ability to design, prototype, and play-test. Unfortunately, the complexity of game design all too often prevents development teams from timely achieving the optimal quality.

These challenges urgently require solutions. Our brief discussion indicates that the designer's ability to exert influence on a game's parts is essential. Rules, objectives, and gameplay assumptions are artifacts that require appropriate notations for constructing high-quality digital games. However, game designers lack a common vocabulary for expressing gameplay. Next, we address this need.

2.2 Languages of Games and Play

Languages of games and play are language-centric approaches for tackling challenges and solving problems related to game design and development. We propose studying existing languages and creating new ones. Two central hypotheses drive this study. We formulate a general and a specific hypothesis:

- (1) *Languages, structured notations, patterns, and tools* can offer designers and developers theoretical foundations, systematic techniques, and practical solutions they need to raise their productivity and improve the quality of games and play.
- (2) "Software" languages (and specifically domain-specific languages) can help automate and speed-up game design processes.

Languages of games and play exist in many shapes and forms. The next section describes one specific technical point of view that represents the departure point of this study, which also details and motivates the second more specific hypothesis.

2.3 Domain-Specific Languages

We aim to deliver solutions that automate game design and speed-up game development with so-called Domain-Specific Languages (DSLs), an approach originating in the field of Software Engineering. Van Deursen et al. define the term as follows:

“A Domain-Specific Language is a programming language or executable specification language that offers, through appropriate abstractions and notations, expressive power focussed on, and usually restricted to, a particular problem domain.” [52].

DSLs have several compelling benefits. They have been successfully created and applied to boost the productivity of domain experts and raise the quality of software solutions. For instance, in areas such as carving data in digital forensics [49], engineering financial products [51], and controlling lithography machines [46], to name a few. DSLs divide work and separate concerns by offering domain experts ways to independently evolve and maintain a system’s parts. Typically, DSLs raise the abstraction level and incorporate domain-specific terminology that is more recognizable to its users. Powerful language workbenches enable analyses, optimizations, visualizations [14], and foreground important tradeoffs, e.g., between speed and accuracy in file carving.

Naturally, there are also costs. DSLs are no silver bullet for reducing complexity. Time and effort go into developing the right language with features that are both necessary and sufficient for its users. In addition, a DSL may have a steep learning curve and users require training [52]. While DSLs help users maintain products, DSLs themselves also demand maintenance and must evolve to accommodate new requirements, usage scenarios, restrictions, and laws, such as new legislation on financial transparency or privacy.

2.4 Need for a Mapping Study

There are many compelling reasons to perform a mapping study on languages of games and play. This study can be approached from different research perspectives with distinct research needs and goals. Here, we describe our position and motivation.

We aim to empower game designers with DSLs that automate and speed-up the game design process. We wish to learn how to facilitate the design space exploration and reduce design iteration times. We envision a set of complementary visual languages, techniques, and tools that help designers boost their productivity and raise the quality of games and play. Challenges include providing abstractions and affordances for:

- (1) expressing a game’s parts as source code artifacts, especially interaction-bound game elements, and modifying these at any given moment;
- (2) evolving “games and play” by steering changes in the source code towards new gameplay goals prototyping, play-testing, balancing, fine-tuning, or polishing;
- (3) obtaining immediate and continuous (live) feedback on a game’s quality by continuously play-testing the effect of changes on quantified gameplay hypotheses;
- (4) obtaining feed-forward suggestions that focus creative efforts and assist in exploring alternative design decisions in a targeted way;
- (5) forming better mental models for learning to better predict the outcome on play.

To know where to start automating game design, we need an extensive analysis on existing approaches. However, these efforts are currently not mapped, and opportunities and limitations are not yet well understood. As a result it is unclear which game facets are amenable to DSL development, which features can express game designs, and what the limits of formalism are.

There is no telling if DSLs can deliver, and how the tradeoff between costs and benefits applies to game development.

We perform this mapping study on languages of games and play to obtain evidence to support our hypotheses in general, and suit our own specific research needs by scouting for opportunities for developing DSLs in particular.

3 METHODOLOGY

A systematic mapping study requires a precise description of its scope, research questions, search queries, and databases for accurate and reproducible information extraction, categorization, and comparison [26]. We apply the following methodology:

3.1 Scope

Games have been studied from different perspectives. Language-oriented approaches have been proposed by authors who published in separate fields of research using distinct vocabularies. As a result, language-centric solutions, intended for diverse domain experts and novices solve differently scoped problems related to a game's design, development, and applications. We survey the full breadth of related work.

3.2 Research Questions

The research questions addressed by this study on languages of games and play are:

- RQ1 What are the research areas and publication venues where authors have published, and what does a map of the field look like?
- RQ2 Which languages have been proposed, and how can these solutions be characterized in terms of (1) objectives, scope, and problems addressed; (2) language design decisions, structure, and notable features; (3) applications, showcases, or case studies; and (4) implementation, deployment, and availability?
- RQ3 What are the similarities and differences between approaches, and common research perspectives sharing similar frames and goals, which languages illustrate them, and what are the limitations?
- RQ4 Which developments and trends can be observed in recent work, and what are the challenges and opportunities for future language research and development?

3.3 Sources

We use the meta-repository Google Scholar (GS) to obtain primary sources, because it maps repositories in which we expect to find relevant publications. GS includes traditional sources of publications such as the Association of Computing Machinery (ACM), Institute of Electrical and Electronics Engineers (IEEE), Springer, and Elsevier. In addition, GS includes less-traditional sources such as games conferences that operate independently, influential books, blog posts, and dissertations. Limiting the search to fewer sources would likely make the study more easily reproducible but also reduce its relevance. A wide search over many sources is necessary for answering our research questions, and GS fits this criterion. Note that we exclusively focus on written sources, which excludes games and commercial development products.

3.4 Queries

Starting with a limited view of the field, we begin with a query to find domain-specific languages for game design and game development. We call this our *narrow query*.

```
"domain_specific_language" AND ("game_development" OR "game_design")
```

GS returns approximately 400 results, mainly in the field of software engineering, and although many publications seem relevant, few articles focus on game design. Clearly, the narrow query is biased towards one specific research area and is too restricted for answering our research questions.

Part of a mapping study is identifying the distinct vocabularies experts in separate research areas use for describing similar approaches, and a more general term is "language". We widen the scope accordingly but, unfortunately, we now find many results on subjects that are off-topic. We therefore attempt to filter out irrelevant publications by formulating a *wide query*.

```
language AND ("game_development" OR "game_design")
AND NOT ("sign_language" OR "second_language" OR "language_
acquisition" OR "body_language" OR "game_based_learning" OR "beer_
game" OR gamification OR gamify)
```

GS reports approximately 17.5K results, more than is feasible for us to analyze. We now realize that given the wicked nature of game design and game development, no single query exists that captures all relevant works. We therefore propose a compromise that combines the results of the narrow query with the first 1K results of the wide query.¹ We restrict the language to English. We exclude patents and citations, results for which GS typically has not seen the full source.

3.5 Inclusion and Exclusion Criteria

We select publications according to the following criteria. The inclusion criterion is: The publication describes a structured language-oriented approach for solving problems related to the design or the development of digital games. For instance, we include programming languages, modeling languages, DSLs, pattern languages, ontologies, and structured vocabularies. Digital games (or digital representations) include computer games, videogames, and applied games (a.k.a. serious games), and so on. The exclusion criterion is: Language features with a fixed structure and notation are not described, or the language does not relate directly to games, and as such does not inform the game design process. Therefore, we exclude the mathematics subject of game theory and the general theme of high performance computing. Networking and audio are not excluded *a priori*.

4 REVIEW PROTOCOL

We identify and analyze relevant publications and categorize them according to the review protocol described in the following section. Each section of the protocol addresses a research question. Section 4.1 addresses RQ1, Section 4.2 addresses RQ2. The remaining questions RQ3 and RQ4 are outside the scope of this protocol and are addressed, respectively, in Section 6 and Section 7.

4.1 Research Areas and Publication Venues

For each included publication, we record the available bibliographical information in BibTeX, e.g., about its authors, title, editors, year, publication venue, acronym, publisher, journal, volume, number, ISBN, ISSN, and DOI. When records are incomplete or missing, as is often the case, we insert the information by hand.

In addition, we add a mapping study identifier that denotes its rank in the query results, a number or not found (-), and appears in the narrow (n) or wide (w) query results. For instance, 12n indicates the publication ranked 12 in the GS results of the narrow query, and -w indicates a publication that conforms to the wide query, but is not ranked in the top 1K results. In some

¹ GS limits the number of results to 1K, but one can obtain more when filtering by year.

Table 1. Categories of Research (Adapted from Wieringa et al. [55])

Category	Description
Evaluation research	Investigates a problem or implementation of a technique <i>in practice</i> for gaining <i>empirical</i> knowledge about causal relationships between phenomena or logical relationships among propositions.
Proposal of solution	Proposes a novel solution technique and argues for its relevance without a thorough validation.
Validation research	Investigates properties of a proposed solution that has not yet been used in practice.
Philosophical paper	Sketches a new way of looking at a problem, a new conceptual framework, and so on.
Opinion paper	Provides an author's opinion about what is wrong or good about a topic of interest.
Experience report	Explains steps taken and lessons learned from experiences gained during a project.
Tutorial	Explains and demonstrates how something works, usually by means of illustrative examples.

cases, we include publications for clarity that conform to neither query, stating which keyword is missing, e.g., gd indicates the keywords "game design" and "game development" are both missing.

Each publication is of a certain type: (peer reviewed) paper (or article), thesis, textbook, non-fiction, technical report, manual, blog post, or presentation. In addition, we analyze research categories shown in Table 1, like Petersen et al. [38]. This table extends categories proposed by Wieringa et al. for categorizing peer-reviewed research in requirements engineering with the last two [55]. These are general categories that indicate how reliable and mature a source is, without going into detail.

We construct a visual map of the field by leveraging citation data that relates publications, and categorize publication venues to research areas. First, we extract citation information from the GS research results. Next, we generate a citation graph whose nodes are publications and edges are citations between them. Finally, we visualize this graph using Gephi, an interactive graph visualization framework.² Gephi's force map algorithm draws together publications with citations between them, forming clusters that roughly correspond to research areas.

We count the number of publications in different journals, conferences, workshops, and symposia. We identify research areas by grouping venues according to disciplines and shared topics. We briefly describe each area and zoom in on the related section of the map for illustration. We summarize venues with two or more publications.

4.2 Language Analysis and Summary

We wish to learn how languages compare, what they have in common, what separates them, and what makes them unique. For each included publication, we extract the name of the language or a description in case no name is provided. We summarize each language concisely by analyzing related publications in a style similar to an annotated bibliography. Table 2 highlights the facets we analyze.

²<https://gephi.org> (visited June 6th 2019).

Table 2. Languages Facets to Summarize and Analyze

Facet	Element	Description
Brief description	Problem	Problem statement, game topic
	Objectives	Goals the authors formulate, challenges addressed
	Solution	Solutions proposed, claims on language application and scope, game genre
	Category	Solution category and application area (Table 3)
Design	Pattern	Language design pattern (Table 4)
	Features	Language features (elements shown in Table 5)
	Examples	Snippets of text, code, diagrams or models
Implementation		How is the language implemented, e.g., interpreter, compiler (these details are usually not described)
Validation	Products	Games, prototypes, and showcases that are constructed using the language
Availability	Web site	URL of a web site providing information on the language, notation, or toolset
	Distribution	URL of a binary distribution or source code repository
	Source license	License under which the source code is available

Table 3. Language Objectives – Solution Scope, Category, and Application Area

Dimension	Category	Description of intent
Scope	Application-specific	Solution is specific for a game or application
	Genre-specific	Solution that is reusable for a specific game genre
	Generic	Generic solution or separated concern
Solution	Framework	Analysis or mental framework for studying, understanding, comparing, categorizing games that does not directly support game development, e.g., ontologies, design patterns, or simulations
	Tool	Authoring tool that facilitates creating a game's parts as models or programs for design or development, e.g., visual environments, programming languages, or DSLs
	Engine	Game engine, reusable building block or software library that integrates models fully into game software
Area	Research	Research vehicle primarily intended for performing research in a specific area
	Educative	Platform primarily intended for teaching a subject to a group of people or example meant to illustrate, educate, or inform
	Practice	Solution primarily intended for practitioners, supporting game design or game development

Claims regarding the scope and applications typically refer to game genres, such as First Person Shooter (FPS), Role Playing Game (RPG) or 2D Platform Game. Although game genre qualifiers are course-grained and not suitable for comparing games in detail [2], they do offer authors ways to indicate the topic of the solution and sketch contours of its scope. In addition, we categorize languages objectives using the categories shown in Table 3.

Table 4. Language Design Patterns (Adapted from Mernik et al. [34])

Dimension	Category	Description
Reuse	Piggyback	Partially uses an existing language, a form of exploitation
	Specialization	Restricts an existing language, a form of exploitation
	Extension	Extends an existing language, a form of exploitation
	Invention	Designs a language from scratch without language reuse
Description	Formal	Formally describes a language using an existing semantics definition method such as attribute grammars, rewrite rules, or abstract state machines
	Informal	Informally explains a language without formal methods

Table 5. Language Features (These Features Are Not Mutually Exclusive)

Dimension	Feature	Description
Notation	Textual	The language has a textual notation
	Visual	The language has a visual notation
Elements	Scopes	Scopes and bounds may be used to separate elements and limit their valid context
	Conditionality	Conditionality features enable or disable other language elements or events
	Recurrence	Recurrence features are elements that can happen again, e.g., in iterations
	Modularity	Modularity features enable composition and/or reuse of language elements
	Domain-specific	Domain-specific language features may be especially created for a purpose that is specific or unique to the subject matter
User Interface	Feedback	Provides a feedback feature enabling understanding
	Mixed-initiative	Provides feedback & feed-forward, alternating between user input and computer-generated alternatives
	Live	Provides immediate and continuous feedback, e.g., a live programming environment

Non-exclusive objectives position languages as: *communication means* for sharing knowledge between experts; *illustration means* for explaining or clarifying problems or solution by example; *maintenance tool* for maintaining and modifying a game's parts over time; *productivity raiser* for increasing the productivity of its users; *quality raiser* for improving a game's quality; and *reuse promotor* for making parts of a game's code or design reusable.

We highlight language design decisions and notable features, including mentions of language reuse and formal semantics. When possible, we use the language design patterns for DSLs proposed by Mernik et al., shown in Table 4 in our description [34]. We analyze language features related to notation, elements, and user interface, described in Table 5. We record how a language is implemented, e.g., as an interpreter or a compiler, and what the host language or formalism is.

Furthermore, we assess applications and availability to form an idea about its status, deployment, and maturity. We list notable applications, showcases and case studies that have been used to validate or evaluate the language in practice. Finally, we report which languages are actually available, and if applicable, we provide links to manuals, teaching materials, source repositories,

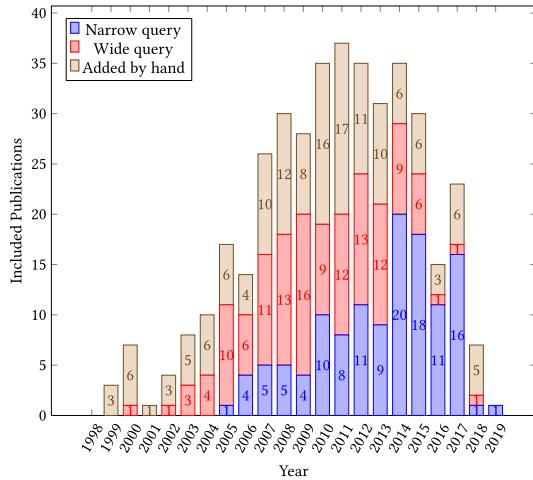


Fig. 2. Amount of publications included in the study per year.

and license agreements. This concludes the protocol. We present the results in the following section.

5 RESEARCH AREAS

Here, we present a systematic map of languages of games and play, which is also available in the form of an accompanying interactive website.³ We performed a search on GS with our narrow and wide queries between the 2nd and 8th of March 2018 and obtained citation data until March 21st. Figure 2 shows a year-by-year count of papers included in this study from both queries and publications we added by hand. Figure 3 shows the citation graph of publications in the search results. Each dot with text represents a publication shown with first author name and year. Publications are included (green nodes) or excluded (red nodes) by applying the criteria from the search protocol. The edges (read clockwise) represent citations. Publications not connected to the graph are omitted in this figure. The aforementioned website adds search, filter, and inspect functionality, integrating the citation graph with the language summaries.

The languages of games and play we have identified originate from the fields of software engineering, artificial intelligence, humanities, social sciences, education, and game studies, with some cross-disciplinary overlap and diffuse areas. Figure 4 illustrates the diversity of publication areas and topics we have selected. The reader is invited to spin the outer wheel of research topics around the publication areas. We describe research areas one by one. We briefly introduce each area, give an overview of venues, and link related research perspectives, which are detailed in Section 6. For conciseness, we only describe venues when the number of identified publications is at least two, as specified by the search protocol.

5.1 Software Engineering and Programming Languages

Software Engineering (SE) researchers study the game domain by developing and applying structured methods, languages, techniques, and tools for engineering better game software. Lämmel covers several subjects of Software Language Engineering (SLE) in his textbook on *Software Languages: Syntax, Semantics, and Metaprogramming* [29]. *Compilers: Principles Techniques and Tools*

³<https://vrozen.github.io/LoGaP/>.

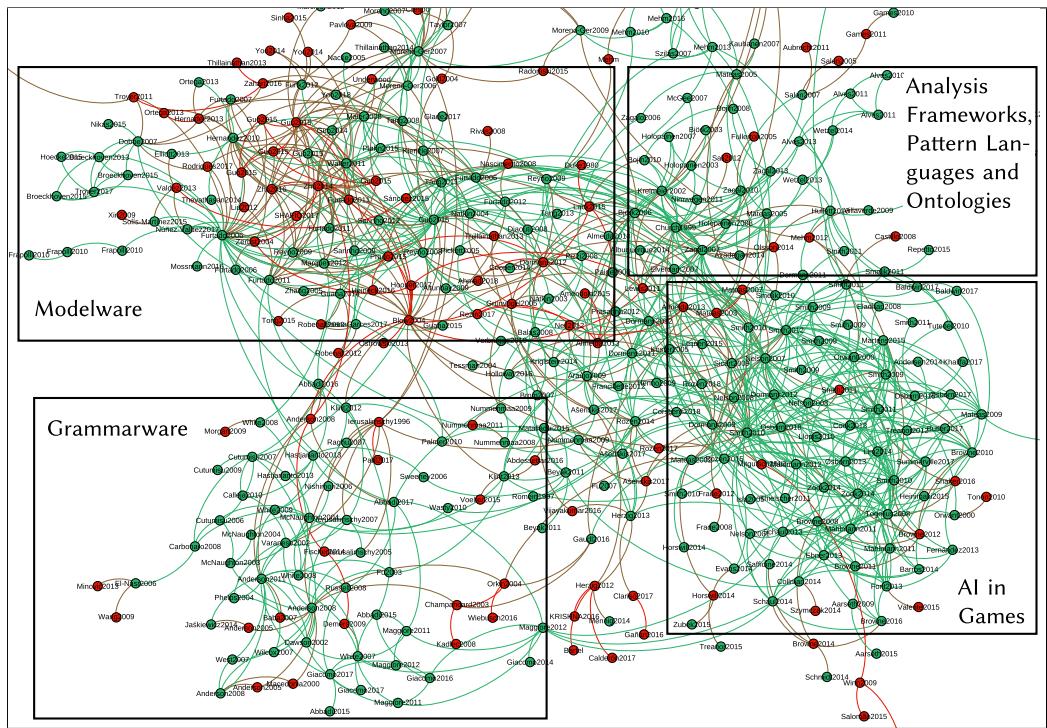


Fig. 3. Systematic map shown as a citation graph. On the left: Software Engineering, Modelware (mainly the top half) and Grammarware (mainly the bottom half). On the right: Analysis frameworks, pattern languages, and ontologies (mainly the top half) and AI in Games (mainly the bottom half).

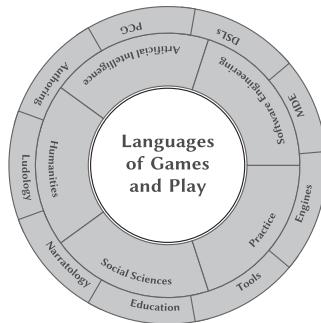


Fig. 4. Language-centric approaches crosscut areas, disciplines, and topics.

(a.k.a. the “*dragon book*”), by Aho et al., first published in 1986, is still regarded as a classic foundational textbook [3].

We identify contributions from Programming Language (PL) research in particular, as shown in Table 6. Figure 3 shows related publications on the left. The ACM Special Interest Group on Programming Languages (SIGPLAN) “explores programming language concepts and tools, focusing on design, implementation, practice, and theory.”

The main source of publications is the International Conference on Systems, Programming, Languages and Applications: Software for Humanity (SPLASH), a large conference “umbrella” of

Table 6. Publication Venues in the Field of Software Engineering and Programming Languages

Venue	Acronym	Years	Ct.
International Conference on Systems, Programming, Languages, and Applications: Software for Humanity (conference umbrella)	SPLASH	1986–	
Conference on Object-Oriented Programming Systems, Languages and Applications	OOPSLA	1986–	1
Symposium on New Ideas in Programming and Reflections on Software	Onward!	2002–	2
International Conference on Generative Programming and Component Engineering	GPCE	2002–	1
International Conference on Software Language Engineering	SLE	2008–	3
Workshop on Domain-Specific Modeling	DSM	2001–	4
International Conference on Software Engineering	ICSE	1975–	2
Workshop on Games and Software Engineering	GAS	2011–2016	2
International Conference on Automated Software Engineering	ASE	1990–	2
Symposium on Principles of Programming Languages	POPL	1973–	2
Science of Computer Programming		2000–	2
ACM Sigplan Notices		1966–	2
Communications of the ACM	CACM	1958–	3
ACM Queue	Queue	2003–	2

colocated events whose names and acronyms are listed in the top part of Table 6. SLE research is traditionally split between modelware and grammarware, which, respectively, revolve around meta-models and grammars [37].

In addition, the search revealed two publications at the International Conference on Software Engineering (ICSE) and two more at the colocated workshop on Games and Software Engineering (GAS), which was organized five times. We also find two invited talks at the Symposium on Principles of Programming languages (POPL) intended to inspire PL research.

Several journals stand out. The monthly *SIGPLAN Notices* includes special issues from associated conferences, including SPLASH, SLE, Onward!, GPCE, and POPL. *Communications of the ACM* is a journal that covers a wider computer science space and articles from *ACM Queue* are included in its practitioners section. In addition, we find two publications in special issues of Elsevier’s *Science of Computer Programming*.

We highlight the following related perspectives: Automated Game design, a multi-disciplinary area that includes SE, in Section 7.3; Applied (or serious) game design, in particular DSLs for expressing subject matter, in Section 6.3; Script and Programming Languages for game development, in Section 6.12; Modeling Languages and model-driven engineering for game development, in Section 6.13; and Metaprogramming, primarily illustrative examples explaining the power of generic language technology, in Section 6.14.

5.2 Artificial Intelligence and Games

The Artificial Intelligence (AI) community has studied how games can benefit from intelligent, usually algorithmic approaches, yielding efficient algorithms, techniques, and tools. In their textbook

Table 7. Publication Venues in the Field of Artificial Intelligence and Games

Venue	Acronym	Years	Ct.
AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment	AIIDE	2005–	14
Workshop on Experimental AI in Games	EXAG	2014–	2
IEEE Conference on Computational Intelligence and Games	CIG	2005–2018	8
IEEE Conference on Games	CoG	2019–	–
International Conference on Computational Creativity	ICCC	2010–	2
EvoStar – The Leading European Event on Bio-Inspired Computation (conference umbrella)	EvoStar	1998–	
International Conference on the Applications of Evolutionary Computation – Games track (EvoGames)	Evo-Applications	2010–	3
IEEE Transactions on Computational Intelligence and AI in Games	TCIAIG	2009–2017	12
IEEE Transactions on Games	T-G	2017–	–

on *AI and Games*, Yannakakis and Togelius describe the theory, use, and application of algorithms and techniques [57].

When languages are created, it is often in the context of intelligent systems (or expert systems), or content generators. Classical AI favored logic programming in Prolog for knowledge engineering, the construction of intelligent systems, which explains why some modern solutions are also based on this paradigm. Notable approaches include logic (Prolog, Answer Set Programming) and Machine Learning. Figure 3 shows related publications, mainly clustered together in the bottom half of the graph. Conferences, symposia, and workshops, shown in Table 7, include the following:

The Association for the Advancement of Artificial Intelligence (AAAI) “aims to promote research in, and responsible use of, artificial intelligence.” This includes the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE) and colocated workshops, such as Experimental AI in Games (EXAG), which are annually organized in North America.

IEEE Conference on Computational Intelligence and Games (CIG) covers “advances in and applications of machine learning, search, reasoning, optimization and other CI/AI methods related to various types of games.” We find contributions related to methods for general game playing such as game description languages and generation of level, strategies, and game rules. The IEEE Conference on Games (CoG) evolved from CIG and widened the scope to cover, among other topics, game technology, game design, and game education.

The prime journal is *IEEE Transactions on Computational Intelligence and AI in Games* (TCIAIG). It recently widened its technological scope and was renamed to *Transactions on Games* (T-G).

We highlight the following related perspectives: Automated Game design, a multi-disciplinary research topic with many contributions from AI and games, in Section 7.3; Game mechanics, frameworks, and systems providing analysis, generation, and explanations of a game’s rules, in Section 6.4; Virtual worlds and game levels, spaces whose structure and composition can be described by languages, tools, and generative techniques, in Section 6.5; Behavior languages, the design of algorithms, tools, and engines for non-player character behavior, in Section 6.6; Narratives and storytelling, in particular technical language-centric approaches, in Section 6.7; Game analytics and metrics, in Section 6.8; and General game playing and Game Description Languages, formalisms for a diverse test-bed for AI and general game playing, in Section 6.11.

Table 8. Articles and Presentations from Game Development Practice

Venue	Acronym	Years	Ct.
Game Developers Conference (UBM Technology Group)	GDC	1988–	4
Game Developer Magazine (UBM Technology Group)		1994–2013	3
Gamasutra (UBM Technology Group)		1997–	3

5.3 Game Development Practice

The Game Developers Conference (GDC), owned by the UBM Technology Group, is a large annual event and not a publishing venue. However, it hosts presentations by practitioners, some of whom share presentation slides identified by this study. The GDCVault⁴ contains audio and video recordings of these presentations and all volumes of *Game Developer Magazine*,⁵ which ran until 2013. Gamasutra⁶ is a web site that continues to publish articles and hosts selected articles from *Game Developer Magazine*. In particular, post-mortems in which developers share experiences about challenges, solutions, decisions (the good and the bad) during a game’s life cycle offer glimpses of game development practice [19]. Given the enormous size of the game industry, the practical accounts we identified in Table 8 are few. These languages likely represent the tip of the iceberg. We reflect on this issue in Section 9.

5.4 Social Sciences and Humanities and Storytelling

Game scholars have extensively studied, analyzed, and critiqued games. They provide insight into how games work, what constitutes play, and how games impact culture and society. In game studies, ludologists have proposed vocabularies, ontologies, and pattern languages aimed at understanding and critiquing games and play in a cultural context.

Aside from studies, game education has yielded textbooks on game design and development practice. Schell introduces the art of game design, offering theory, approaches, and conceptual lenses that help designers think and practice [40]. Fullerton describes a play-centric approach to creating games highlighting the disciplines prototyping and play-testing [17]. Salen and Zimmerman describe game design fundamentals with focus on core concepts, rules, play, and culture [39].

Table 9 shows the publication venues we identified. The International Conference on Interactive Digital Storytelling is the result of merging between its predecessors: Technologies for Interactive Digital Storytelling and Entertainment, and Conference on Virtual Storytelling.

The International Simulation and Gaming Association (ISAGA) has organized an annual conference since 1970. ISAGA can trace its origins back to the now-famous book *Homo Ludens* [23], and aside from game studies also has an education focus. Related is the journal *Games & Simulation*. In addition, we identify articles in the *Journal of Game Studies*.

We highlight the following related perspectives: Ontological approaches and typologies, in Section 6.1; Pattern languages and design patterns, in Section 6.2; Narratives and storytelling, in Section 6.7; Educative languages, in Section 6.9; and Gamification, in Section 6.10.

5.5 Multi-disciplinary Games Research

Researchers and practitioners from different fields meet at multi-disciplinary conferences on games and entertainment computing. They exchange points of view and apply and mix diverse expertise on game design, AI, and language technology, which results in synergy and inter-

⁴<https://gdcvault.com> (visited August 20th 2019).

⁵<https://gdcvault.com/gdmag> (visited August 20th 2019).

⁶<http://www.gamasutra.com> (visited August 20th 2019).

Table 9. Multi-disciplinary Publication Venues on Game Studies, Education, and Storytelling

Venue	Acronym	Years	Ct.
Conference of the International Simulation and Gaming Association	ISAGA	1970–	3
Simulation & Gaming	S&G	1970–	2
Game Studies		2001–	2
European Conference on Games Based Learning	ECGBL	2007–	2
Computers & Education (Elsevier)		2000–	2
International Conference on Interactive Digital Storytelling	ICIDS	2008–	2
Conference on Technologies for Interactive Digital Storytelling and Entertainment	TIDSE	2003, 4, 6	2
International Conference on Virtual Storytelling	ICVS	2001, 3, 5, 7	2

Table 10. Multi-disciplinary Publication Venues on Games and Entertainment Computing

Venue	Acronym	Years	Ct.
International Conference on the Foundations of Digital Games	FDG	2006–	12
Workshop on Procedural Content Generation in Games	PCG	2010–	9
Workshop on Design Patterns in Games	DPG	2012–2015	5
International Conference on Entertainment Computing	ICEC	2002–	5
Workshop on Game Development and Model-Driven Software Development	GD & MDSD	2011, 2012	3
Digital Games Research Association International Conference	DiGRA	2003–	7
International Conference on Intelligent Games and Simulation	GAME-ON	2000–	6
International North American Conference on Intelligent Games and Simulation	GAME-ON-NA	2005–2011+	3
International Conference on Computer Games	CGAMES	2004–2015	4
Brazilian Symposium on Computer Games and Digital Entertainment	SBGames	2002–	4
International Conference on Computers and Games	CG	1998–	2
International Conference on Advances in Computer Entertainment Technology	ACE	2004–2018	5
International Academic Conference on the Future of Game Design and Technology	Future Play	2002–2010	3
ACM Computers in Entertainment	CIE	2003–2018	2

disciplinary advances. Multi-disciplinary venues are the main source of publications included in this study, and perhaps the best source of nuanced approaches. Table 10 shows the publications venues we identified. We briefly describe each one.

The International Conference on the Foundations of Digital Games (FDG) is a multi-disciplinary conference that alternates between Europe and the USA. Two colocated workshops are of special note. The first, Design Patterns in Games (DPG) includes pattern languages and gameplay design

patterns. The second, Procedural Content Generation in Games (PCG) is concerned with generative methods for games, often using AI techniques.

The International Conference on Entertainment Computing (ICEC) is organized annually. The continent of the venue varies. The colocated Workshop on Game Development and Model-Driven Software Development (GD&MDSD) was organized twice. The Digital Games Research Association (DiGRA) organizes its annual DiGRA International Conference, which is a mix of game studies, humanities and technology.

The International Conference on Intelligent Games and Simulation (GAMEON-ON) focuses on structured methods for programming of games that benefit industry and academia. GAME-ON is organized annually in Europe, but also occurs on different continents, e.g., North America (GAME-ON-NA). Another conference branched off from GAME-ON in 2004, as explained by Spronck [44]. It was first known as Computer Games: Conference on Artificial Intelligence, Design and Education (CGAIDE) and later became the International Conference on Computer Games (CGames), which was last organized in 2015.

The International Conference on Computers and Games (CG) is a venue that is not organized every year. CG is associated with the International Computer Games Association (ICGA), which also has its own journal. The Brazilian Symposium on Computer Games and Digital Entertainment (SBGames) is a national event with international visibility and also a source of several DSLs.

Some venues we identified are discontinued. For instance, the Conference on Advances in Computer Entertainment Technology (ACE) was a technology oriented multi-disciplinary conference. In 2018, most members of its steering committee resigned when they could no longer guarantee an impartial review process, and the conference closed down after a community boycott that condemned the conduct of the event's owner. The International Academic Conference on the Future of Game Design and Technology (Future Play) was organized from 2002 until 2010. The journal, *ACM Computers in Entertainment (CIE)* ceased in 2018.

In the next section, we discuss a series of research perspectives. Each of these can be considered a multi-disciplinary point of view.

6 RESEARCH PERSPECTIVES

We present a series of 14 complementary research perspectives on languages of games and play. Each perspective sheds light on what informs the design and construction of good games. Together they form an overview that provides answers to our research questions RQ2 and RQ3. We acknowledge that different decompositions would have been possible, and ours is merely one of many ways to aggregate the results of this study. Our structure represents a best-effort that adheres to the phrasings and research frames of the authors whose works we summarize.

We derive the perspectives from the search results as follows: We group summaries of similar languages. We distill common research frames, highlight challenges, and describe theoretical foundations, systematic techniques, and practical solutions. For conciseness, we only illustrate these views with a selection of representative language summaries.

6.1 Ontologies and Typologies

In search of a common vocabulary for game studies, scholars have defined ontologies that relate written symbols, abstract concepts, and real-world objects. In general, an ontology defines a set of named concepts, their properties, and the relations between them, usually with the goal of categorizing objects of interest in a specific subject area or problem domain for understanding its meaning. In game studies in particular, ontologies and typologies are used to describe, categorize, analyze, understand, and critique games and play.

Table 11. Ontological Languages

Nr.	Language	Domain	Notation
1	Game Typology	Game studies	tables and visual diagrams
2	Game Ontology Project	Game studies	tables /pattern-language
3	Pervasive Games Ontology	Pervasive games	class diagram
4	Ontology of Journalism	Journalism	Web Ontology Language

Table 12. Pattern-languages for Analyzing Games and Forming Gameplay Hypotheses

Nr.	Language	Informs the design of
5	Formal Abstract Design Tools	gameplay goals/pattern languages
6	Game Design Patterns	gameplay goals/pattern languages
7	Gameplay Design Pattern	gameplay goals
8	Mechanics Dynamic Aesthetics	digital game systems
11	Collaboration Patterns	collaborative gameplay goals
15	Flow Experience Patterns	flow experience goals
16	Patterns Language for Serious Games Design	applied gameplay goals

Aarseth explains that ontologies are essential for game studies to reach a consensus on the meaning of words, concepts, and relations between them [1]. He scrutinizes different ontological models and describes challenges in their construction. For instance, choosing the level of description, i.e., how fine-grained the ontological elements (or meta-categories) should be, is difficult, because there is no natural halting point in adding nuances, details, or patterns for highlighting differences. In addition, choosing generality or specificity limits applications, since general-purpose ontologies may not be as useful as ontologies constructed for a specific purpose.

There are many languages for constructing ontologies. For instance, the W3C Web Ontology Language (OWL) is an XML-based notation with good tool support for describing semantic relationships.⁷ In some cases languages of games and play use ontologies for describing or guiding parts of the solution. Table 11 lists ontological languages we describe.

6.2 Pattern Languages and Design Patterns

A pattern language describes best practices with empirically proven good results as a reusable solution to commonly recurring problems in a particular area of interest. The approach originates from Alexander et al. who describe a pattern language for towns, buildings, and construction [4]. Often presented in table form or a template, sequential sections highlight different facets of the problem, proposed solution, examples, and related contextual information. In Software Engineering, Object Oriented (OO) design patterns are a well-known means to create, explain, and understand software designs, design decisions, and implementations [18]. In contrast, in game studies and humanities, game design patterns are a means to analyze and explain player experiences, also referred to as gameplay design patterns. The key difference between these kinds of patterns is that the former are prescriptive for structuring software and the latter are analytic regarding gameplay effects. We categorize pattern languages in two categories. The first, shown in Table 12, lists languages for analyzing, categorizing, understanding, and critiquing gameplay. The second, shown in Table 13, lists languages for predicting the effect of changes to a game's design and making informed design decisions, e.g., level design patterns and game mechanics patterns. Programming

⁷<https://www.w3.org/OWL>.

Table 13. Pattern-languages Offering Authorial Affordances over Designing Games and Play

Nr.	Language	Game Artifact	Affects
9	Pattern Language for Sound Design	sound	sound-supported experiences
10	Verbs	abstract player actions	gameplay affordances
12	Pattern Cards for Mixed Reality Games	mixed reality game rules	mixed-reality experience
13	Design Patterns for FPS Games	structure of FPS levels	progression, experiences
14	3D Level Patterns	structure of 3D levels	progression, experiences
17	Operational Logics	depends on concrete representations	inner game workings, external player communication
26	Machinations	game-economic mechanics, feed-back loops	gameplay affordances, strategies, and tradeoffs

Table 14. Languages for Domain Experts to Help Design Applied Games Scenarios

Nr.	Language	Subject matter expert	Objectives
18	StoryTec	educator, non-programmer	educate through stories that integrate learning goals
19	GameDNA	psychologist	assess cognitive processes
20	ATTAC-L	pedagogical expert	educate, prevent cyber bullying
21	EngAGE DSL	educator	improve feedback to learners
22	VR-MED	medical expert	teach family medicine

patterns that directly affect a game's mechanics, narratives, levels, and behaviors are discussed in Sections 6.4, 6.5, 6.6, and 6.7.

6.3 Applied Game Design

Applied (or serious) games have a primary purpose other than entertainment and usually require designs that incorporate subject matter knowledge. Diverse experts from education, psychology, and even medical doctors can help improve game designs, e.g., for learning,⁸ overcoming traumas, and speeding-up recovery. The challenge is integrating domain knowledge in a game's design to achieve specific gameplay goals such that players (for instance, patients or students) learn, reflect, or modify behaviors. Naturally, there are ethical and privacy implications and restrictions of studying player choices, especially if the game also serves as a diagnostic tool. Dörner et al. provide an overview of foundations, concepts, and practice of serious games for prospective developers and users [12]. Applied games may also include physical forms of interaction [35]. Here, we identify languages, mainly DSLs, intended for helping domain experts design and vary scenarios of applied games, e.g., for learning and assessment, as shown in Table 14.

6.4 Game Mechanics

Although most agree that *game mechanics* are rules that affect gameplay, there are many different explanations, theories on how this works, e.g., References [16, 319, 39, 40, 42]. In the previous

⁸Please note that we excluded the term “game based learning” from the wide query due to the large amount of false positives.

Table 15. Game Mechanics Languages

Nr.	Language	Mechanics	Analysis	Generation
23	Petri Nets	game-economic, story	yes	?
24	Game Space Definitions	“stock” logics	yes	
25	BIPED and LUDOCORE	combinatorial	yes	yes
26	Machinations	game-economic	yes	no
27	Micro-Machinations	game-economic	yes	yes
28	Game-o-Matic	combinatorial	yes	yes
29	Mechanic Miner	avatar-centric	yes	yes
30	Gamelan and Modular Critics	combinatorial	yes	?
31	PDDL Mechanics	combinatorial, avatar-centric	yes	yes
32	Sygnus and Gemini	combinatorial	yes	yes

sections, we have described frameworks (Section 6.1) and design patterns (Section 6.2) for understanding, analyzing, and creating game mechanics. This section can be regarded as a *proceduralist view* that applies formalizations and generative techniques to program game mechanics and analyze effects.

Table 15 shows languages, generators, and tools for game mechanics. These works explore the limits of formalism and study to what extent models of mechanics (and players) can be leveraged for predicting and improving a game’s quality. Each language attempts to relate mechanics to aesthetics in different ways. Combinatorial rule spaces expressed with logical notations use constraints for exploring the design space and homing in on desired qualities, for instance using Answer Set-Programming for exploring the design space. Examples include BIPED and LUDOCORE (Language 25) or Gamelan and Modular Critics (Language 30).

Meaningful relationships between real-world subjects, e.g., derived from WordNet and ConceptNet, can be used to instantiate the structure of mechanics, e.g., Game Space Definitions (Language 24). Rhetorical arguments, intended for adding meaning, give structure to the mechanics of news games or micro-games, e.g., to convince players with political or cultural statements. Examples are Game-o-Matic (Language 28), and Sygnus and Gemini (Language 32). Sicart critiques *procedural rhetorics*, and presents opposing arguments [43]. Nelson clarifies a more general position on proceduralism [36], and Treanor and Mateas present an account of proceduralist meaning [48].

Avatar-centric mechanics encoded in ASP, rewrite rules, or Java describe the physics of characters in 2D levels, such as moving, jumping, and bouncing, e.g., Mechanic Miner (Language 29), Planning Domain Description Language Mechanics (Language 31), and PuzzleScript (Language 91).

Game-economic mechanics described in graph notations express how in-game resources flow and which choices players have. They foreground feedback loops that represent investments and tradeoffs. Examples are the well-known Petri Nets (Language 23), the design framework Machinations (Language 26), or its cousin the programming language Micro-Machinations (Language 27).

6.5 Virtual Worlds and Levels

Virtual worlds are spaces in games or simulations that through various integrated audiovisual content and interactive mechanisms support exploration, communication, or play. Game worlds and levels can be populated by player avatars, virtual characters, stories, missions, quests, and so on. Procedural Level Generation is a form of PCG that focuses on generating game levels, spaces that integrate missions, quests, and (lock and key) puzzles, e.g., for dungeon crawlers and platformers. Van der Linden et al. survey Procedural Dungeon Generation [50]. We identify relatively few

Table 16. Tools and Languages for Mixed-initiative Design of Levels and Virtual Worlds

Nr.	Language or Tool	Content
33	Semantic Scene Description Language	classes of concepts and relationships
34	SketchaWorld	3D worlds
35	Tanagra	platform game levels
36	Ludoscope	grammar-based transformation pipelines of 2D levels
37	The Sentient Sketchbook	tile map sketches of 2D levels
38	Evolutionary Dungeon Designer	2D level maps and patterns

language-centric approaches using our protocol, since most authors in this perspective call their solutions “tools.”

Table 16 shows tools and languages for designing levels and virtual worlds. Each of these tools support creating and improving (this is usually called authoring) content, a mixed-initiative style. In *mixed-initiative* approaches, intelligent services (the tool) and the designer collaborate and take turns to achieve the designer’s goals [22]. Typically, designers receive visual computer-generated suggestions, and based on gradually improving insight, make decisions that refine the content iteration by iteration, in a conversational style. However, due to a lack of direct manipulation of generated content, it can be challenging to assure all possible results represent meaningful and high-quality content. In this context, *semantic scenes* are structures for describing meaningful and consistent content that can guide generators. As in other perspectives, authors also apply patterns, constraints, and metrics for generating and analyzing level qualities. Metrics are discussed separately in Section 6.8.

6.6 Behaviors

Defining behaviors of Non Player Characters (NPCs) has been an active topic of technically oriented research. A Behavior Definition Language (BDL) is a programming language that offers a notation that controls powerful AI features for describing believable virtual entities that inhabit game worlds [8]. Many BDLs are implemented as reusable software libraries that complement game engines. According to Anderson, many BDLs are DSLs that also maintain the flexibility of programming languages [8]. Challenges include developing appropriate notations and features, authoring for dramatic realism, improving scalability of parallel behaviors, and raising the fault tolerance. Table 17 shows a limited selection of formalisms including Reactive Planning Languages, Behavior Trees, (Hierarchical) Finite State Machines, and Statecharts. Of course, many languages describe behaviors in one way or another. Our selection represents a wider set of languages.

6.7 Narratives and Storytelling

Storytelling, a field on its own, is concerned with writing, telling, and sharing stories by means of narratives to convey ideas and experiences to an intended audience. Similar to games, stories in a cultural context can be used for entertainment, education, cultural values, and so on. The perspective we describe here is a technical interpretation that envisions programming languages for creating narratives and programming interactive stories using generative techniques. Here, we refer to the creation process as *authoring*, since the users of these languages are first and foremost authors.

Various researchers have focused on Interactive Fiction (IF), interactive drama, and the story components of games, e.g., quests, missions, and stories of adventure games or educational games. Branching narratives can be expressed as graphs, where choices represent alternative sequences

Table 17. Languages for Domain Experts to Help Design Behaviors

Nr.	Language	Description
39	ABL	ABL is a reactive planning language for authoring believable agents with rich personality
40	SEAL	Simple Entity Annotation Language (SEAL) is a C-like script language for describing NPC behaviors
41	BEcool	BEcool is a visual graph language with sensors and actuators for describing expressive virtual agents
42	Behavior Trees	Behavior Trees is a visual language for authoring AI behaviors
43	BTNs	Behavior Transition Networks is a visual notation of hierarchical state machines for describing behaviors
44	POSH#	framework for creating behavior-based AI for robust and intuitive agent development
94	Statecharts	Modeling formalism for describing behaviors
103	RAIL	Reactive AI Language (RAIL) is a metamodel-based DSL for modeling behaviors in adventure games

Table 18. Languages for Authoring, Analyzing, and Generating Narratives, Stories, and Dramas

Nr.	Language	Expresses
39	ABL	Reactive planning language used for interactive drama
46	<e-Game>	Storyboards for educational adventure games, formally analyzed
47	ScriptEase	Patterns of behaviors, quests, and stories, interactive user interface
18	StoryTec	Educational stories and related learning goals, part of a tool set
48	Ceptre	Story worlds and experimental game mechanics, offers logical proofs
49	SAGA	Stories whose compilers target different platforms
50	(P)NFG	Structured computer narratives and IF, playability, and correctness
51	Wander	Number games, represents an early historical account
52	Storyboards	Generating storyboards of game levels, leveraging a planner
53	Versu	Interactive text-based drama, including social conventions
54	Tracery	Stories and art, example of a “casual creator”
107	Fictitious	Demonstrates the use of DSLs for Interactive Fiction

of events or paths. Moreover, emergent stories with generative and dramatic components require integrating social values and knowledge of virtual personas. Challenges include checking the correctness of the paths by analyzing constraints and providing insight with appropriate visualizations and debugging facilities, e.g., into the causality of emergent scenarios. Storyboards are a sequence of images (or illustrations) and text (e.g., dialogues) used for analyzing stories of various kinds of media, including games. Kybartas and Bidarra survey story generation techniques [28]. We identify several languages intended for authoring, generating, and analyzing narratives, summarized in Table 18.

6.8 Analytics and Metrics

Data science combines techniques, approaches, and tools from statistics, data mining, data analysis, and machine learning. Data scientists leverage the available data to extract knowledge, gain insights, and predict trends. The game industry increasingly relies on game analytics for developing high-quality games. One technique is using *metrics*, algorithms quantifying system properties,

college students	Squeak (Lang. 62) StarlogoTNG (Lang. 64)		
high school students			
middle school children	Scratch (Lang. 63) Alice (Lang. 59)	AgentSheets and AgentCubes (Lang. 65)	Gamestar Mechanic (Lang. 60)
young children	Kodu (Lang. 61)	computational thinking	programming
			creating & designing games

Fig. 5. Relating languages to education levels and learning goals.

as measures of quality. Designers can use these metrics to test gameplay hypotheses and assess the gameplay quality by studying how metrics evolve over time. For instance, by relating player models or “personas” to how game mechanics are used (Language 55). PlaySpecs can be used to analyze sequences of player actions (Language 57). Launchpad uses metrics to assess platform-level quality (Language 56). In contrast, MAD and SAnR work directly on the engine source code of grammar-based level generators, relating gameplay and software quality (Language 58). Fenton and Bieman describe a rigorous approach for software metrics based on measurement theory [15]. Research challenges include evaluating the quality of content generators [41], identifying suitable metrics for different types of content, and relating metrics to player models and experience.

6.9 Education

Here, we describe educative languages that are end-user solutions aimed at improving learning experiences, e.g., for helping students or children learn programming, computational thinking, and game design in a playful and explorative manner. Figure 5 shows examples and roughly relates languages to educational activities, goals, and (minimum) education levels. Educational languages usually come with an ample amount of study material. In addition, these languages may include learning program and web sites that cater for active online communities.

Languages for learning usually pay extra attention to usability. Of special note are the “block-based languages” that enable users to fit syntactic constructs together like puzzle pieces. These do not permit syntax mistakes that can be especially frustrating to novices, and instead ensure every adjustment is meaningful and educational.

In addition, several languages use a Logo-style positional movement where one can imagine moving around. Logo is an educational programming language that is well-known for its “turtle,” which can be steered using commands for drawing vector graphics.

6.10 Gamification

Gamification aims to apply or retrofit standard game designs to a new or existing system to improve user experiences.⁹ For instance, score, competition, and reward systems have been used in areas like online marketing to stimulate participation with a product or service. We identify languages intended to gamify information systems in general, e.g., GAML (Lang. 66), GLiSMo (Lang. 67), and UAREI (Lang. 68). While these languages are technically reusable, they lack subject matter concepts that help domain experts solve design problems. Recently the term *playification* has been used to describe gamification that facilitates play more effectively by means of tailor-made game designs.

⁹Please note that we excluded the term “gamification” from the wide query due to the large amount of false positives.

Table 19. Game Description Languages (GDLs) for General Game Playing

Nr.	Language	Game domain	Examples of represented games
69	Multigame	mainly board games	Chess, Checkers
70	(Stanford) GDL	combinatorial games	various combinatorial games
71	“Rule Sets”	Pac-man-like games	generated games
72	Ludi GDL	combinatorial games,	e.g., Javalath
73	Strategy GDL	strategy games	Rock Paper Scissors, Dune II
74	Card GDL	card games	Texas hold’em, Blackjack, Uno
75	Video GDL	video games	a variety of classic 2D games
76	RECYCLE	card games	Agram, Pairs, War

6.11 General Game Playing

A key research challenge in AI is developing generally applicable intelligent techniques capable of solving a wide range of complex problems. General game playing, for instance, refers to algorithms that can play many games well.

Game Description Languages (GDLs) are notations with expressive power over restricted game domains intended for evaluating the performance of AI techniques called *general game players* against a wide variety of manually created or generated games. GDLs are meant to cover a varied and representative game space.

The systems that execute GDL programs are used for validating if these techniques are indeed more widely applicable, e.g., by letting them compete. General game players can be search-based [47], e.g., Monte Carlo Tree Search (MCTS) or leverage Machine Learning, e.g., genetic algorithms or neural networks. In a guest editorial of a special issue on “general games,” Browne et al. summarize the state-of-the art, challenges and directions for future research [10]. We show representative GDLs in Table 19 whose domains reflect a shifting research interest of the community over the years. Notably not identified by this study is Zillions of Games.¹⁰

General game playing has the advantage of developing and applying the state-of-the-art AI to digital games, offering advanced tools, simulations, and analyses. GDLs are not necessarily suited for fine-tuning rules and improving gameplay as in automated game design (Section 7.3). Therefore, not all GDLs are *a priori* well-suited for developing games, especially not those intended as research platforms. For instance, the language features of VGDL are coarse-grained and the Stanford GDL is low-level and verbose. GDLs for restricted game domains, such as board games and card games, represent a concise and expressive middle ground.

6.12 Script and Programming

Here, we describe a programming language perspective on game development. For creating a programming language, one usually constructs a *grammar* using the Extended Backus-Naur form (EBNF) or a similar notation, for parsing the textual source code of programs [3]. Here, source code (or textual model) refers to end-user notations called *concrete syntax*. The result of parsing is a parse tree that is often represented using suitable intermediate representations referred to as *abstract syntax*, which usually omits white-space and comment. In the resulting tree, references between definitions and uses must be resolved. This process of *reference resolution* yields a graph that forms an input to analyzers, code generators, and interpreters that further transform or run programs.

¹⁰ <http://www.zillions-of-games.com> (visited April 3rd 2019).

Table 20. Generic Script and Programming Languages Applied to Game Development

Nr.	Language	Application domain
62	Squeak	programming system based on Smalltalk applied in teaching game design
77	Python	programming language, applied for scripting in games
78	Lua	programming language, scripting in games
79	vision on game programming	reflections on features of game programming languages and Haskell
80	DisCo	language and system for creating, executing, and analyzing formal specifications
81	Design by contract	generic approach that uses pre- and post-conditions for checking function calls

Table 21. Domain-specific Languages for Game Development

Nr.	Language	Application domain
82	GameMaker	2D game development with C-like scripting
83	Extensible Graphical Game Generator	game programming
84	Mogemoge	2D games
85	Scalable Games Language	scripting for games
86	Network Scripting Language	scripting and networking
87	4Blocks DSL	DSL for Tetris games (Haskell-based)
88	Casanova	game programming (integrated game engine)
89	Sound scene DSL	sound scene DSL (Haskell-based)
90	MUDDE (historical account)	multi-user dungeon games (MUDs)
91	PuzzleScript	puzzle games

Game programming usually follows a bottom-up approach that composes game systems from reusable parts. Specialized software libraries called *game engines* offer developers reusable Application Programming Interfaces (APIs) for solving challenges in 3D modeling, physics, directional audio, or networking. Many commercial game engines have been developed as reusable platforms for game development, e.g., Unity 3D, Unreal, CryoEngine.¹¹

One approach separates game engines from game-specific source code by using generic interpreters as-is, e.g., Python (Language 77) or Lua (Language 78). Table 20 shows other examples. Another approach leverages general purpose languages by adding domain-specific language extensions, e.g., as an *internal DSL* that reuses the syntax and semantics of the host language. For instance, Scalable Games Language (Language 85) extends SQL, and 4Blocks (Language 87) and Sound Scene DSLs (Language 89) are Haskell-based.

In contrast, purpose-built languages, also known as *external DSLs*, have separate parsers, compilers, and/or interpreters. PuzzleScript (Language 91) and Micro-Machinations (Language 27) are examples of external DSLs. Table 21 shows examples of DSLs for game development. For conciseness, we do not list all textual DSLs that we already describe in other sections. Notably not identified is DarkBasic [20].

¹¹These are not identified by this study.

Table 22. Generic Modeling Languages Applied to Game Design and Development

Nr.	Language	Application domain
92	UML – Metamodeling	generic formalism for meta-modeling, e.g., applied to 2D platform games
93	UML – Class and State Diagrams	generic formalism for object-oriented analysis and design applied in model-driven game development
94	Statecharts	variants of a formalism that describe behaviors as state machines, applied to Game AI and dialogue in games
95	Feature Models	visual variability modeling formalism applied to managing game (and game engine) variability

Table 23. Domain-specific Modeling Languages for Game Development

Nr.	Language	Application domain
96	SharpLudus	RPG games, mobile touch-based games, 2D arcade games
97	Eberos GML2D	2D Games
98	FLEXIBLERULES	digital board game creation and customization
99	PhyDSL	2D mobile physics-based games
100	Pong Designer	Pong-like games
101	Board Game DSL	board games
102	RougeGame Language	visibility Rogue-like dungeon maps
103	Reactive AI Language	behaviors in adventure games

6.13 Model-Driven Engineering

Model-driven game development revolves around abstract *models* that describe game content or work processes, often using visual diagrammatic representations. Modeling languages are based on the principles, techniques, and tools from an area called Model-Driven Engineering (MDE). These models are step-by-step translated, transformed, and combined into a resulting model or source code that integrates with the game software. We identify applications of generic modeling languages in Table 22 and Domain-Specific Modeling Languages in Table 23.

Metamodeling represents the abstract syntax of models as a graph of Unified Modeling Language (UML) classes and references between them. Because metamodeling is often based on the Eclipse Modeling Framework (EMF) and Ecore (the EMF model engine), it enjoys the advantage of generic reusable tools for model transformation, analysis, and productivity, e.g., for adding a textual concrete syntax (Xtext, EMFText), or graphical ones (e.g., using GMF, Graphiti) [37]. In a recent literature review, Zhu and Wang present a model-driven perspective on game development [58].

6.14 Metaprogramming

The metaprogramming perspective considers game development as an application domain for generic language technology. Metaprogramming refers to techniques, tools, and approaches for creating metaprograms that read and transform the source code of other programs, e.g., compilers, interpreters, and integrated development environments. Applying these techniques to game development promises to raise productivity, improve quality, and reduce maintenance costs.

Table 24. Applications of Metaprogramming Languages and Language Workbenches

Nr.	Language	Metaprogramming language or workbench	URL
27	Micro-Machinations	Rascal	https://www.rascal-mpl.org
66	GAML	Xtext	https://www.eclipse.org/Xtext
96	SharpLudus	Microsoft DSL tools	https://visualstudio.microsoft.com
99	PhyDSL	Xtext	https://www.eclipse.org/Xtext
104	Whimsy	C++	(various versions exist)
105	Level editors	DiaMeta	http://www2.cs.unibw.de/tools/DiaGen
107	Ficticious	Ginger	(not found)
106	Text aventures	Racket	https://racket-lang.org
108	Dialog Script	Xtext	https://www.eclipse.org/Xtext

Constructing languages and tools by means of metaprogramming requires appropriate meta-tooling. *Language workbenches* are tools that provide high-level mechanisms for the implementation of software languages. Erdweg et al. describe the state-of-the-art in language workbenches [14]. Examples of metaprogramming languages and language workbenches include Epsilon, Gemoc Studio, Meta-Edit+, MPS, Racket, Rascal, Spoofax, and Xtext. Several authors illustrate metaprogramming techniques and apply approaches to example languages. Table 24 shows examples of language of games and play created by means of language workbenches.

The strength of this perspective is the application of state-of-the-art in language engineering and its weakness is that, with some exceptions, many illustrations remain toy examples that are never extensively validated.

7 CHALLENGES AND OPPORTUNITIES

Here, we discuss research trends, synthesize insights, and describe challenges and opportunities for future research and development. First, we discuss the results in general in Section 7.1. Next, we compare and analyze success factors in Section 7.2. Finally, we synthesize one additional perspective on languages of games and play in Section 7.3. Our Automated Game Design perspectives is a specific language-centric discussion on challenges and opportunities for research and development. This section answers research question RQ4.

7.1 General Analysis

Our area of interest “languages of games and play” is a well-studied research topic with a growing number of publications. Figure 2 shows that most papers we included were published after the turn of the millennium. Around this time, roughly between 1998 and 2006, most of the interdisciplinary game publishing venues we identified also came into existence. Since 2005, there is a gradual increase in the term “domain-specific language.” Two factors explain the declining number of publications in this study after 2015. First, the query date limits the search results. Second, GS orders the results of the wide query to show older results first. Therefore, it is likely we missed newer results that could have been included.

Our analysis of the citation graph, shown in Figure 3, reveals a low cohesion between publications. We observe clusters that represent distinct technological spaces, tight-knit communities, fragmented sub-topics, and diffuse areas. Therefore, authors may not find or recognize related work in a wider research context. As a result, relevant literature has gone uncited, efforts and successes have often been one-off, lessons learned have gone overlooked, and several studies and areas

Table 25. Examples of Multi-year, Multi-disciplinary Work on Languages of Games and Play

(AI: Artificial Intelligence, SE: Software Engineering, Edu: Education, G: Games)					
Nr.	Language	Ct.	Years	Areas	Perspectives
65	AgentSheets	6	1995–2012	SE+Edu	visual DSLs, education
39	ABL	5	2002–2008	AI+G+SE	programming, behaviors, interactive drama
7	Gameplay Design Patterns	6	2003–2013	G+Edu	pattern language
47	ScriptEase	8	2003–2013	SE+G	pattern language, visual DSL
96	SharpLudus	6	2006–2012	SE+G	model-driven engineering, visual DSL
46	<e-game>	6	2006–2012	SE+Edu	storyboards, education, visual DSL
25	BIPED and LUDOCORE	9	2008–2012	AI+G	automated game design, mechanics
26	Machinations	6	2009–2012	G	pattern language, automated game design
27	Micro-Machinations	3	2013–2015	SE+G	automated game design, mechanics, programming, visual DSL
88	Casanova	11	2011–2017	SE+G	game programming

Table 26. Highlighting the Differences between Applied and Forgotten Languages

	Alive languages	Dead languages
Language experts	multiple	one
Publication count	multiple	one or two
Publication areas	multiple	one
Validation	applied and validated in practice	not applied, toy examples
Availability	sources or wiki pages are released and maintained up to a point	no source code is available
Examples	tutorials, workshops, and study materials are available	not available

have remained isolated. We view this mapping study as an opportunity to relate relevant primary sources to help frame research problems, reuse available approaches, and benefit from documented experiences. Our map serves to navigate between related perspectives and technological spaces. As time progresses, the map can be extended and reshaped for charting new research trajectories that continue to explore the limits of formalism.

7.2 Success Factors

Our analysis reveals a “grave yard” of dead language prototypes. Few languages ever grow to maturity. Many languages remain solution proposals that are now no longer maintained, available, or in use. This lack of reuse is unfortunate, since creating one language often requires years of research, design, and development. Naturally, this leads to the question of why so many prototypes were abandoned. Here, we discuss observations and insights about shared success factors. Table 25 shows examples of languages that stand out as multi-faceted research with a relatively high publication count. We explain success factors summarized in Table 26.

Languages of games and play represent a considerable effort and a long-term investment. Therefore, success requires a multi-year research trajectory, perhaps spanning multiple research grants and Ph.D. projects. Publishing in different research areas helps answer different related questions

and sheds light on challenges and solutions from different perspectives. Involving multiple researchers, language developers, and practitioners creates co-ownership and continuity.

Traditionally, academia and the game industry have not always seen eye to eye. However, collaborating in applied research projects is essential for validating research in practice. To that end, some research departments include labs and game studios, and host in-house designers. Of course, working with innovative indie game developers or AAA studios on industrial case studies costs time and effort. The main benefit is that case studies can showcase approaches and lead to better and more applicable solutions. Stakeholders can formulate common goals, agree to make research results open source, and protect intellectual property of businesses with Non-Disclosure Agreements (NDAs). As a selling point, students participating in these projects might become employees who bring expertise and help to create new and innovative game products.

Naturally, making solutions available is necessary to apply them. Open source software might include reusable script engines, content generators, visual tools, or programming environments. In addition, for learning to apply solutions effectively, users may require Wiki pages, blogs, example materials, tutorials, and workshops. We found online supplementary materials for roughly half the languages we summarized.

Some languages, in particular educational online languages described in Section 6.9, have thriving user communities that create significant impact. Their users apply solutions in practice, which increases the research visibility and grows a network. Naturally, these benefits come at great cost, e.g., time spent on tools, demos, maintenance, and legacy support. However, when users become stakeholders invested in validation, they also assist in building datasets. Of the 108 language summaries, only 11 are based on at least one publication we categorized as evaluation research. Ultimately, empirical evidence is necessary for scientific research.

7.3 Automated Game Design

Here, we synthesize a unifying perspective on languages of games and play by relating research perspectives from the previous section to challenges and opportunities for future research and development.

Automated Game Design (AGD) aims to speed-up and improve iterative game design processes by automating design processes. Here, we discuss how languages, structured notations, patterns, and tools can help game design experts raise their productivity and improve the quality of games and play. We distill research challenges and relate this perspective to other perspectives on languages of games and play.

Various languages, techniques, and tools have been proposed for creating, generating, analyzing, and improving a game's parts. These design tools offer interactive user interfaces that support prototyping, sketching designs, automating play-tests, and exploring design spaces, usually in a mixed-initiative, conversational, and collaborative manner. Moreover, these tools are often intended to support game design as an explorative, playful, or enjoyable activity. AGD usually shifts design and implementation efforts away from manual, repetitive, time-consuming and error-prone tasks. That way, designers can more efficiently create, improve, and maintain growing amounts of *game content*. We define content as follows:

“Game content refers to every asset of a digital game that can be separately (or together) viewed, understood, modeled, generated, recombined, and improved to affect audio-visual and interactive player experiences.”

Visual content includes textures, models, sprites, imagery, objects that form composite structures such as trees, landscapes, cities, and nature [41]. Audio content includes music, voices, and

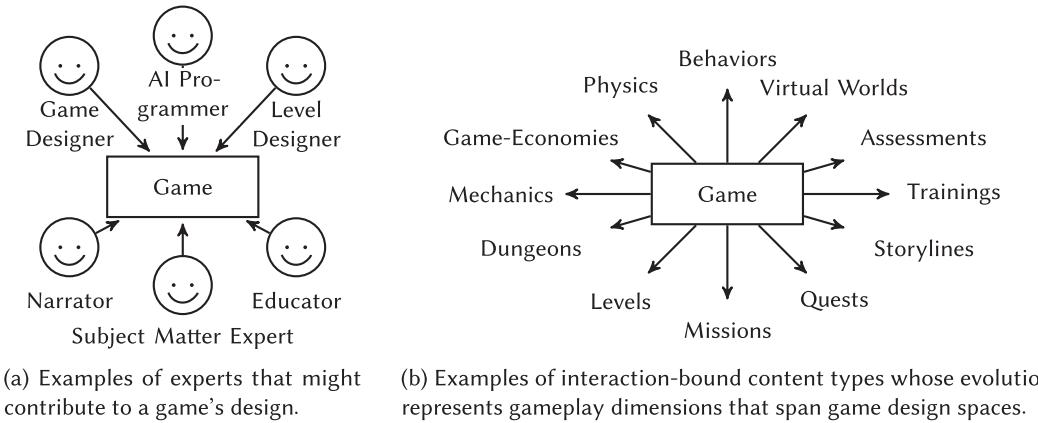


Fig. 6. Game design experts contribute content to evolve games in different dimensions.

effects. Procedural Content Generation (PCG) refers to generative techniques that produce and transform game content [21, 28, 41, 50]. Game engines and software libraries, reusable components for the construction of digital games, are not content [41].

The focus of this study is on “*interaction-bound content*,” content that expresses how players interact and communicate, which especially affects player experiences. Figure 6(a) illustrates the diverse experts that might contribute to a game’s design. Figure 6(b) illustrates types of content (or facets) a game might have. Modifying these content facets evolves a game along multiple related axes or dimensions.

The problem is that these dimensions overlap, often in non-trivial ways, which results in a web of criss-crossing interconnected content dependencies that may differ from game to game. This makes it exceptionally difficult to align a particular interpretation of a dimension with a reusable form of content representation that separates a game design concern. As a result, improving a game’s qualities along one dimension is often difficult without negatively affecting another. Therefore, design experts require *orchestrated* content generators [30] for composing high-quality games from different kinds of inter-connected content, intricately interwoven to support meaningful experiences, in a reusable manner. Next, we relate key challenges of AGD to perspectives on languages of games and play.

Frameworks and pattern catalogues for studying games describe *what* games are. These perspectives inform automating game design by providing context, theory, and structured frameworks for common vocabularies and notations.

- Game designers lack a common *vocabulary*, which hampers specification, communication, and agreement among developers and designers [27]. However, automating game design requires formalizing game concepts, e.g., by performing a *domain analysis* that identifies concepts, names, meanings, and relationships. Useful frameworks and resources are *ontologies and typologies* that help to describe, understand, and characterize games and distinguish what makes games unique. Section 6.1 highlighted this perspective.
- Game designers require design tools, reusable *patterns*, and techniques that help them analyze and predict how modifications to a game’s design will affect the gameplay. Section 6.2 highlighted pattern languages and game design patterns that describe best practices, recurring structures of content, and gameplay, and represent steps towards standardization and reuse.

Other perspectives are content-centric. Related publications typically envision design experts who contribute design artifacts by means of languages and tools. We describe the following perspectives:

- Games may require integrating *subject matter knowledge*. The challenge is providing languages and tools that enable domain experts such as as educators and psychologists to describe scenarios and participate in game design processes. Section 6.3 highlighted a perspective on applied (or serious) game design.
- Many games integrate *game mechanics*, rules that offer playful affordances and bring about interesting player experiences. Game designers require formalisms to express game mechanics, e.g., game economies or avatar physics. Additionally, they require tools for analyzing behaviors, generating rules, balancing strategies, introducing tradeoffs, managing feedback-loops, and automating play-testing. Section 6.4 highlighted a perspective on game mechanics.
- Many games include generated *spaces* such as virtual worlds and game levels. Designers require tools to assist in populating these spaces with various kinds of content, e.g., to create varied and interesting game levels, dungeons, missions, and quests. Section 6.5 gave a perspective on spaces.
- Game designs may integrate *behaviors* of in-game entities such as non-player characters that require dramatic realism or challenge. Designers and AI programmers require formalisms for expressing these behaviors. Section 6.6 illustrated a perspective on behavior languages with different strengths and applications.
- Games designs may integrate *stories* that allow players to progress through stages of a plot. Designers and narrators require languages and tools to express these stories. Section 6.7 illustrated a perspective on techniques that express narratives and story plot using textual notations and graphs.

Technical views on *how* to automate game design with available techniques and approaches originate mainly from the fields of AI and games, software engineering, and education. We relate the following challenges to technical perspectives on languages of games and play:

- Raising the quality of game content requires automated iterative analyses, especially when dealing with generated content. Section 6.8 highlighted a perspective on metrics, which can be used to relate content and player actions to gameplay.
- Usability, understandability, and ease of use are essential qualities for game design tools. Designers require appropriate visual notations and timely feedback to comprehend concepts, master language features, and learn to apply user interfaces effectively. We described a perspective on educational end-user environments in Section 6.9.
- During a digital game's life span, designers may need to make significant changes to its design. Designers require tools that enable modifying a game's design during its entire evolution. Section 6.10 gave a perspective on gamification, which studies techniques for redesigning games and retrofitting game designs.
- Powerful, cutting-edge *AI techniques* are constantly being researched, developed, and improved. The challenge is leveraging these for automated game design, e.g., for analysis, generation, and testing. Section 6.11 described a perspective on a field called *general game playing*, which uses games as a test-bed for AI.
- Developing high-quality games in a time-to-market manner requires *programming and maintaining* growing amounts of source code and rapidly evolving that code towards new gameplay goals. Section 6.12 highlighted a perspective on DSLs, script languages, and

programming environments, which have been created to speed up development and improve the quality and maintainability.

- Developing *visual languages* for game design from scratch is a difficult and time-consuming endeavour. Model-driven engineering offers several reusable formalisms, techniques and approaches. We highlighted a perspective on how design can be automated by means of visual models and tools in Section 6.13.
- Developing and maintaining DSLs, generators, and tools costs a considerable amount of time and effort. Language developers require appropriate *meta-tooling* and metaprogramming techniques to alleviate this effort, which enables rapid prototyping. Several authors advocate the use of generic language technology by demonstrating its power in illustrative examples. We highlighted this perspective in Section 6.14.

We describe two additional open research challenges. First, a game’s quality is limited by the number of game design iterations. Producing high-quality games more quickly requires the duration of game reducing iterations. An open challenge is leveraging *live programming* techniques for providing *live* (immediate and continuous) *feedback* on changes to a program. This may be the key to forming more accurate mental models and better predicting behavioral effects and gameplay outcomes. Second, the composition of content alone does not explain how a game’s simulation is communicated to its players. To fully understand how a game works, designers might require content creation strategies that relate content representations to a set of communication strategies or Operational Logics (Language 17).

8 RELATED WORK

The research perspectives we have described in Section 6 relate work on languages of games and play. We have already mentioned several surveys on more specific topics that align with those perspectives. Here, we briefly discuss more general related surveys, literature reviews, and mapping studies that focused on specific themes and topics. This study poses research questions that were not yet answered and contributes a multifaceted overview of the entire area of interest.

Ampatzoglou et al. perform a systematic review on software engineering research for computer games [7]. They identify topics, research approaches, and empirical research methods.

In addition, two surveys relate to the model-driven-engineering perspective presented in Section 6.13. Tang and Hanneghan examine the state-of-the-art in model-driven game development from a game-based learning perspective [45]. Their overview describes model transformations and several game design languages. Zhu and Wang present a literature review that analyzes 26 model-driven approaches [58]. Their protocol zooms in on target game domains (genres), domain frameworks, modeling languages, tooling, and evaluation methods.

Beckmann et al. perform an exploratory literature study on “live” tooling in the game industry [9]. They analyze articles on Gamasutra and videos of the Game Developers Conference and relate identified tools to degrees of live programming. In contrast, we cover mainly academic sources.

Almeida and da Silva survey game design methods and tools [6] and synthesize requirements from 32 selected publications [5]. They present a map of design frameworks and visual modeling formalisms [6], which overlaps with our perspectives on Ontologies (Section 6.1), Pattern Languages (Section 6.2), and Game Mechanics (Section 6.4).

Several vision papers align with this study. Walter and Masuch discuss how to integrate DSLs into the game development process [54]. Mehm et al. take an authoring tools perspective when discussing research trends [32]. We give an overview of related Ph.D. dissertations in Appendix A.

9 THREATS TO VALIDITY

Systematic mapping studies are intended to create an *unbiased* and *complete* overview of a subject. With that in mind, we have applied methodology guidelines [26] and designed a reproducible and an unambiguous protocol. However, the results of this study are a compromise. By definition the word “game” is a concept whose “essence” cannot be captured in words [56]. Therefore, this study can never be fully complete, unambiguous, and unbiased. Here, we address threats to validity.

Scoping the area of interest. We formulated two queries to obtain evidence for our hypotheses. Our narrow query, aimed at our second hypothesis, is biased towards software engineering where the term “domain-specific language” is common. To obtain a more nuanced and complete overview that also includes other research fields, we formulated a wide query aimed at our first hypothesis, with focus on languages in general. However, more than 16K GS results was more than is feasible for us to analyze. We compromised and chose to limit our analysis to its top 1K results. In addition, we filtered terms that are often, but not always, off-topic. As a result, despite our best efforts, we may have overlooked relevant publications.

Breaking protocol. We cited publications not conforming to either of our queries to clarify the origins, descriptions and applications of a language. In addition, we included several papers that conform to the wide query but did not appear in the top 1K results. We have clearly marked these in the language summaries of Appendix B, and added the bibliographical data in a separate library, as can be seen in Figure 2. While this makes our overview more complete, it also reintroduces the selection bias we wished to avoid in the first place.

Pilot error. We have summarized and related publications from a wide array research fields, areas and topics. Unfortunately, despite our best efforts, we have inevitably overlooked or mischaracterized contributions. This study is intended as an inclusive, constructive, and “living” document that we hope to discuss, improve, and extend over time.

Synthesizing perspectives. We synthesized 15 perspectives on languages of games and play from over 100 language summaries. Our decomposition of the topic of interest is a best-effort interpretation that relies on our personal experience. We acknowledge that it is possible to formulate other research perspectives that extend the collection. Different authors, who have distinct research needs and goals, might want to shed light on a problem from a different angle. They could choose finer granularity to zoom in on an area and reuse different subsets of languages that cross-cut topics in different ways.

Missing in action: game development practice. We have mapped the state-of-the-art in languages of games and play for a wide audience, which in our view, should include practitioners. Unfortunately, because we identified relatively few practical sources, we have not fully delivered on this promise. We acknowledge this is a limitation of our research method, which does not include non-written sources such as games, development kits, engines, and tools. Of course, GS primarily contains academic sources, and the game development industry is not in the business of publishing papers. In addition, fierce competition and time-to-market pressure have led to a degree of industrial secrecy. By not sharing information, many businesses are simply protecting their intellectual property and competitive edge.

Using Google Scholar. Our choice for GS is motivated by its high recall. Using GS, we obtained publications from independent venues we did not know existed. However, Google owns the information records on GS, maps the interests of its users, and does not provide bulk access.¹² This

¹²<https://scholar.google.com/intl/en/scholar/help.html> (visited August 23rd 2019).

complicates systematic studies in general, which require an off-line analysis to “stand on the shoulders of giants.” As a result, it is not straightforward to reproduce this study.

Applying bibliometrics. We obtained citation data from GS and constructed the citation graph using a combination of Python scripts and Gephi. Given the right tools, we could have extracted the citation data directly from the PDFs. In addition, we used Gephi’s built-in layout algorithms to obtain a suitable image. However, the same data can produce different graph layouts as well. Mapping studies are complicated by a lack of tools for bibliographic analysis and bibliometrics. Standardizing and automating mapping studies can help save precious time and improve the quality of literature reviews and surveys in general.

10 CONCLUSION

We have presented an overview of the state-of-the-art in languages of games and play that relates research areas, goals, and applications. We identified and summarized over 100 languages and synthesized 15 research perspectives (or angles) on the topic, each illustrated by selected language summaries. The results show that there is evidence to support both of our research hypotheses. First, languages, structured notations, patterns, and tools can offer designers and developers theoretical foundations that offer experts systematic techniques and practical solutions they need to raise their productivity and improve the quality of games and play. Second, we obtained evidence that DSLs can help in automated game design, and we described illustrative examples that suggest how to achieve this. We also mapped related approaches and described perspectives other than our own departure point with distinct approaches and motivations. Instead of clarifying a single perspective, our map leads in many related research directions, each representing possible departure points for related studies. Ultimately, our map provides a good starting point for anyone who wishes to study and learn more about languages of games and play.

Future Work. We foresee the following future work: First, we plan to maintain and extend the accompanying website. The interactive version of the map serves as a “living document” that enables exploring languages, citation data and summaries, and navigating between them.

Second, we see opportunities for additional mapping studies and literature reviews that intersect with this study and zoom in on specific related areas, such as automated game design, mixed-initiative approaches, procedural content generation, and live programming [9].

ACKNOWLEDGMENTS

We thank the anonymous reviewers, Paul Klint, Tijs van der Storm, and Daria Polak for proofreading and providing valuable feedback that helped improve this article. In addition, we thank the many colleagues who over the years have pointed out relevant languages, venues, and publications.

Last but not least, we thank the authors whose contributions we had the privilege to read, summarize, and relate. We would like to apologize to any peer who feels their work is unfairly treated or incorrectly portrayed.

REFERENCES

- [1] Espen Aarseth. 2011. Define real, moron! Some remarks on game ontologies. In *DIGAREC Keynote-Lectures 2009/10 (DIGAREC)*, Stephan Günzel, Michael Liebe, and Dieter Mersch (Eds.). Potsdam UP, 50–69.
- [2] Espen Aarseth, Solveig Marie Smedstad, and Lise Sunnanå. 2003. A multi-dimensional typology of games. In *Proceedings of the DiGRA International Conference: Level Up*, Marinka Copier and Joost Raessens (Eds.). Utrecht University.
- [3] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. 1986. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley.
- [4] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. 1977. *A Pattern Language—Towns, Buildings, Construction*. Oxford University Press.

- [5] M. S. Almeida and F. S. da Silva. 2013. Requirements for game design tools: A systematic survey. In *Proceedings of the 12th Brazilian Symposium on Games and Digital Entertainment*.
- [6] Marcos Silvano Orita Almeida and Flávio Soares Corrêa da Silva. 2013. A systematic review of game design methods and tools. In *Proceedings of the 12th International Conference on Entertainment Computing, (LNCS)*, Vol. 8215, Juilia C. Anacleto, Esteban W. G. Clua, Flavio S. Correa da Silva, Sidney Fels, and Hyun S. Yang (Eds.). Springer, 17–29. DOI : https://doi.org/10.1007/978-3-642-41106-9_3
- [7] Apostolos Ampatzoglou and Ioannis Stamelos. 2010. Software engineering research for computer games: A systematic review. *Inf. Softw. Technol.* 52, 9 (2010), 888–901. DOI : <https://doi.org/10.1016/j.infsof.2010.05.004>
- [8] E. F. Anderson. 2008. *On the Definition of Non-player Character Behaviour for Real-time Simulated Virtual Environments*. Ph.D. Dissertation. Bournemouth University.
- [9] Tom Beckmann, Christian Flach, Eva Krebs, Stefan Ramson, Patrick Rein, and Robert Hirschfeld. 2019. An exploratory literature study on live-tooling in the game industry. In *Proceedings of the Workshop on Live Programming*.
- [10] C. Browne, J. Togelius, and N. Sturtevant. 2014. Guest editorial: General games. *IEEE Trans. Computat. Intell. AI Games* 6, 4 (Dec. 2014), 317–319. DOI : <https://doi.org/10.1109/TCIAIG.2014.2369009>
- [11] Richard Coyne. 2005. Wicked problems revisited. *Des. Stud.* 26, 1 (2005), 5–17.
- [12] Ralf Dörner, Stefan Göbel, Wolfgang Effelsberg, and Josef Wiemeyer. 2016. *Serious Games*. Springer. DOI : <https://doi.org/10.1007/978-3-319-40612-1>
- [13] Mirjam P. Eladhari and Elina M. I. Ollila. 2012. Design for research results: Experimental prototyping and play testing. *Simul. Gaming* 43, 3 (2012), 391–412. DOI : <https://doi.org/10.1177/1046878111434255>
- [14] Sebastian Erdweg, Tijs Van Der Storm, Markus Völter, Meinte Boersma, Remi Bosman, William R. Cook, Albert Gerritsen, Angelo Hulshout, Steven Kelly, Alex Loh, et al. 2013. The state of the art in language workbenches—Conclusions from the language workbench challenge. In *Proceedings of the 6th International Conference on Software Language Engineering, (LNCS)*, Vol. 8225, Martin Erwig, Richard F. Paige, and Eric Van Wyk (Eds.). Springer, 197–217. DOI : https://doi.org/10.1007/978-3-319-02654-1_11
- [15] Norman Fenton and James Bieman. 2014. *Software Metrics: A Rigorous and Practical Approach* (3rd ed.). CRC Press. DOI : <https://doi.org/10.1201/b17461>
- [16] Tracy Fullerton, Christopher Swain, and Steven Hoffman. 2004. *Game Design Workshop: Designing, Prototyping, and Playtesting Games*. CMP Books.
- [17] Tracy Fullerton, Christopher Swain, and Steven Hoffman. 2008. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games* (2nd ed.). Morgan Kaufmann.
- [18] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- [19] Austin Grossman (Ed.). 2003. *Postmortems from GameDeveloper*. CMP Books.
- [20] Jonathan S. Harbour and Joshua R. Smith. 2007. *DarkBASIC Pro Game Programming* (2nd ed.). Thomson Course Technology.
- [21] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. 2013. Procedural content generation for games: A survey. *ACM Trans. Multim. Comput., Commun. Applic.* 9, 1 (Feb. 2013), 1–22. DOI : <https://doi.org/10.1145/2422956.2422957>
- [22] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceeding of the CHI'99 Conference on Human Factors in Computing Systems*, Marian G. Williams and Mark W. Altom (Eds.). ACM, 159–166. DOI : <https://doi.org/10.1145/302979.303030>
- [23] Johan Huizinga. 1938. *Homo Ludens: Proeve Ener Bepaling van het Spelelement der Cultuur*. Wolters-Noordhoff.
- [24] Robin Hunnicke, Marc 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 Artificial Intelligence*. AAAAI, 1–5.
- [25] Jesper Juul. 2011. *Half-real: Video Games between Real Rules and Fictional Worlds*. The MIT Press.
- [26] Barbara Kitchenham and Stuart Charters. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Technical Report. Keele University and Durham University Joint Report. EBSE-2007-01.
- [27] Paul Klint and Riemer van Rozen. 2013. Micro-machinations: A DSL for game economies. In *Proceedings of the 6th International Conference on Software Language Engineering, (LNCS)*, Vol. 8225, Martin Erwig, Richard F. Paige, and Eric Van Wyk (Eds.). Springer, 36–55. DOI : https://doi.org/10.1007/978-3-319-02654-1_3
- [28] B. Kybartas and R. Bidarra. 2017. A survey on story generation techniques for authoring computational narratives. *IEEE Trans. Computat. Intell. AI Games* 9, 3 (Sept. 2017), 239–253. DOI : <https://doi.org/10.1109/TCIAIG.2016.2546063>
- [29] Ralf Lämmel. 2018. *Software Languages: Syntax, Semantics, and Metaprogramming*. Springer. DOI : <https://doi.org/10.1007/978-3-319-90800-7>
- [30] Antonios Liapis, Georgios N. Yannakakis, Mark J. Nelson, Mike Preuss, and Rafael Bidarra. 2019. Orchestrating game generation. *IEEE Trans. Games* 11, 1 (2019), 48–68. DOI : <https://doi.org/10.1109/TG.2018.2870876>

- [31] Michael Mateas and Andrew Stern. 2005. Build it to understand it: Ludology meets narratology in game design space. In *Proceedings of the DiGRA International Conference*. Digital Games Research Association.
- [32] Florian Mehm, Christian Reuter, Stefan Göbel, and Ralf Steinmetz. 2012. Future trends in game authoring tools. In *Proceedings of the 11th International Conference on Entertainment Computing, part of the 2nd Workshop on Game Development and Model-Driven Software Development*, Marc Herrlich, Rainer Malaka, and Maic Masuch (Eds.). Springer, 536–541. DOI :https://doi.org/10.1007/978-3-642-33542-6_70
- [33] Tom Mens. 2008. Introduction and roadmap: History and challenges of software evolution. In *Software Evolution*. Springer, 1–11. DOI :https://doi.org/10.1007/978-3-540-76440-3_1
- [34] Marian Mernik, Jan Heering, and Anthony M. Sloane. 2005. When and how to develop domain-specific languages. *Comput. Surv.* 37, 4 (Dec. 2005), 316–344. DOI :<https://doi.org/10.1145/1118890.1118892>
- [35] Florian Müller, Rohit Ashok Khot, Kathrin Gerling, and Regan L. Mandryk. 2016. Exertion games. *Found. Trends Hum. Comput. Interact.* 10, 1 (2016), 1–86. DOI :<https://doi.org/10.1561/1100000041>
- [36] Mark J. Nelson. 2012. Sicart’s “Against Procedurality”—A reply. *Kmjn.org*. (May 2012). Retrieved from http://www.kmjn.org/notes/sicart_against_procedurality.html.
- [37] Richard F. Paige, Dimitrios S. Kolovos, and Fiona A. C. Polack. 2013. Metamodelling for grammarware researchers. In *Software Language Engineering, Proceedings of the 5th International Conference on Software Language Engineering, (LNCS)*, Vol. 7745, Krzysztof Czarnecki and Görel Hedin (Eds.). Springer, 64–82. DOI :https://doi.org/10.1007/978-3-642-36089-3_5
- [38] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, Giuseppe Visaggio, Maria Teresa Baldassarre, Stephen G. Linkman, and Mark Turner (Eds.). BCS.
- [39] Katie Salen and Eric Zimmerman. 2003. *Rules of Play—Game Design Fundamentals*. The MIT Press.
- [40] Jesse Schell. 2014. *The Art of Game Design: A Book of Lenses*. AK Peters/CRC Press.
- [41] Noor Shaker, Julian Togelius, and Mark J. Nelson. 2016. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer. DOI :<https://doi.org/10.1007/978-3-319-42716-4>
- [42] Miguel Sicart. 2008. Defining game mechanics. *Game Stud.* 8, 2 (Dec. 2008).
- [43] Miguel Sicart. 2011. Against procedurality. *Game Stud.* 11, 3 (Dec. 2011).
- [44] Pieter Spronck. 2004. CGAIDE AND GAME-ON 2004. *ICGA J.* 27, 4 (Dec. 2004), 241–241. DOI :<https://doi.org/10.3233/ICG-2004-27410>
- [45] Stephen Tang and Martin Hanneghan. 2011. State-of-the-art model driven game development: A survey of technological solutions for game-based learning. *J. Interact. Learn. Res.* 22, 4 (Dec. 2011), 551–605.
- [46] Ulyana Tikhonova, Maarten Manders, Mark van den Brand, Suzana Andova, and Tom Verhoeff. 2013. Applying model transformation and event-B for specifying an industrial DSL. In *Proceedings of the 10th International Workshop on Model Driven Engineering, Verification and Validation, co-located with 16th International Conference on Model Driven Engineering Languages and Systems, (CEUR Workshop Proceedings)*, Vol. 1069, Frédéric Boulanger, Michalis Famelis, and Daniel Ratiu (Eds.). CEUR-WS.org, 41–50.
- [47] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Trans. Computat. Intell. AI Games* 3, 3 (2011), 172–186. DOI :<https://doi.org/10.1109/TCIAIG.2011.2148116>
- [48] Mike Treanor and Michael Mateas. 2013. An account of proceduralist meaning. In *Proceedings of the DiGRA International Conference*, Celia Pearce, John Sharp, and Helen W. Kennedy (Eds.). Digital Games Research Association.
- [49] Jeroen van den Bos and Tijs van der Storm. 2011. Bringing domain-specific languages to digital forensics. In *Proceedings of the 33rd International Conference on Software Engineering*, Richard N. Taylor, Harald C. Gall, and Nenad Medvidovic (Eds.). ACM, 671–680. DOI :<https://doi.org/10.1145/1985793.1985887>
- [50] Roland van der Linden, Ricardo Lopes, and Rafael Bidarra. 2014. Procedural generation of dungeons. *IEEE Trans. Computat. Intell. AI Games* 6, 1 (Mar. 2014), 78–89. DOI :<https://doi.org/10.1109/TCIAIG.2013.2290371>
- [51] A. van Deursen. 1997. Domain-specific languages versus object-oriented frameworks: A financial engineering case study. In *Proceedings of the Conference on Smalltalk and Java in Industry and Academia*. Ilmenau Technical University, 35–39.
- [52] Arie van Deursen, Paul Klint, and Joost Visser. 2000. Domain-specific languages: An annotated bibliography. *ACM SIGPLAN Not.* 35 (2000), 26–36.
- [53] Christel van Grinsven, Max Otten, and Olaf Koops. 2019. *Games Monitor the Netherlands 2018 – Full Report*. Technical Report. Dutch Game Garden. <https://www.dutchgamegarden.nl>.
- [54] Robert Walter and Maic Masuch. 2011. How to integrate domain-specific languages into the game development process. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology, ACE 2011, Lisbon, Portugal, November 8–11, 2011*. ACM, 1–8. DOI :<https://doi.org/10.1145/2071423.2071475>

- [55] Roel Wieringa, Neil A. M. Maiden, Nancy R. Mead, and Colette Rolland. 2006. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requirements Engineering* 11, 1 (March 2006), 102–107. DOI : <https://doi.org/10.1007/s00766-005-0021-6>
- [56] Ludwig Wittgenstein. 1953. *Philosophical Investigations*. Blackwell. Translated by G.E.M. Anscombe.
- [57] Georgios N. Yannakakis and Julian Togelius. 2018. *Artificial Intelligence and Games*. Springer. DOI : <https://doi.org/10.1007/978-3-319-63519-4>
- [58] Meng Zhu and Alf Inge Wang. 2019. Model-driven game development: A literature review. *Comput. Surveys* 52, 6 (Nov. 2019), 1–32. DOI : <https://doi.org/10.1145/3365000>

Received February 2020; revised July 2020; accepted July 2020