# Improving Video Game Development: Facilitating Heterogeneous Team Collaboration through Flexible Software Processes

Juergen Musil, Angelika Schweda, Dietmar Winkler, and Stefan Biffl

Christian Doppler Laboratory for
Software Engineering Integration for Flexible Automation Systems,
Institute of Software Technology and Interactive Systems,
Vienna University of Technology, Vienna, Austria
{jmusil,angelika}@computer.org,
{Dietmar.Winkler,Stefan.Biffl}@tuwien.ac.at
http://cdl.ifs.tuwien.ac.at

**Abstract.** Based on our observations of Austrian video game software development (VGSD) practices we identified a lack of systematic processes/method support and inefficient collaboration between various involved disciplines, i.e. engineers and artists. VGSD includes heterogeneous disciplines, e.g. creative arts, game/content design, and software. Nevertheless, improving team collaboration and process support is an ongoing challenge to enable a comprehensive view on game development projects. Lessons learned from software engineering practices can help game developers to increase game development processes within a heterogeneous environment. Based on a state of the practice survey in the Austrian games industry, this paper presents (a) first results with focus on process/method support and (b) suggests a candidate flexible process approach based on Scrum to improve VGSD and team collaboration. Results showed (a) a trend to highly flexible software processes involving various disciplines and (b) identified the suggested flexible process approach as feasible and useful for project application.

**Keywords:** Video games, Software Process Improvement, Flexible Software Processes, Scrum, Survey.

## 1 Introduction

Similar to traditional software engineering projects, video games include processes for project planning and control, constructive methods for artifact creation, and analytical methods for verification and validation purposes [31]. Thus, lessons learned, best-practices and process improvement approaches, derived from software engineering, are applicable to video game development. Nevertheless, game development teams have to collaborate in a heterogeneous environment to enable high-valuable video games [7]. Although video games as interactive entertainment software are at the first glance perceived as simplistic, game construction activities include a high level of systems complexity because of the involvement of various disciplines. Video game software development (VGSD) is built on the expertise of computer science fields like computer graphics, artificial intelligence, systems architecture and interactive systems. In the last decade VGSD has contributed to the major advancements of high-performance computing in real-time rendering, propagation of motion-control in human-computer interaction and the design, development and maintenance of large-scale cyberworlds. Since video games' purpose is mainly to amuse and appeal, their development differs significantly from the construction of typical utilitarian and engineering-oriented software, i.e. software supporting daily work (database-supported systems).

Besides technological complexity, VGSD is highly cost-intensive[1] and relies on the involvement of multiple heterogeneous disciplines like arts, engineering and design. Due to the inherent complexity and isolated domain-specific methodologies, the games industry suffers from difficulties in workflow integration across disciplines, requirement elicitation, project scheduling and concurrent development for multiple target platforms, i.e. hardware devices. Our previous research [24] indicates that games industry problems like feature creep and crunch time (requirements and scheduling issues) are caused by missing underlying structures like process and applied methodologies. Previous reports, based on a national survey regarding the state of the practice of video game development in Austria [24], also indicate that VGSD trends - similar to traditional software development - to flexible software development approaches [30,28]. Nevertheless, collaboration of heterogeneous teams is still challenging.

This paper focuses on the analysis of the state of the practice in video game development in a heterogeneous environment and presents a promising process approach for supporting and improving video game development projects. Additionally, the process promises to increase the support of heterogeneous teams, frequent changing of requirements, and process and product improvement approaches.

The remainder of this paper is structured as follows: Section 2 discusses related work and research areas on VGSD. Section 3 presents the research issues. We provide a flexible candidate process approach and a first feedback from the Austrian game development community in section 4. Finally, section 5 concludes and illustrates future research work.

## 2 Related Work

This section presents related work in video game software development (VGSD) and identifies research challenges with focus on collaboration in heterogeneous teams in game development projects.

[1] Digital Battle.
`http://www.digitalbattle.com/2010/02/20/top-10-most-expensive-video-games-budgets-ever` (last visited 05/31/2010).

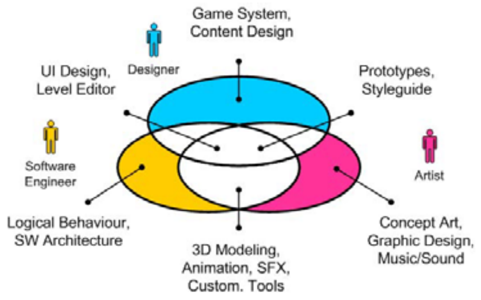
Game System, Content Design
Designer
UI Design, Level Editor
Prototypes, Styleguide
Software Engineer
Artist
Logical Behaviour, SW Architecture
3D Modeling, Animation, SFX, Custom. Tools
Concept Art, Graphic Design, Music/Sound

**Fig. 1.** VGSD as collaborative activity of various heterogeneous disciplines

## 2.1 Game Development

Video games are complex and interactive (software+) systems that require significant research[2] in real-time graphics, artificial intelligence and distributed systems. Further, video games are entertainment software that apply multiple modes (graphics, audio, interaction, story and collaborative play) with the objective to immerse users with interactive experiences by specifically blending these modes [12,16,18]. Figure 1 shows that VGSD relies on various heterogeneous disciplines (arts, engineering, visual design and sound design) in order to create these modes [10].

Land et al. [21] report that existing software engineering standards are applicable for VGSD. Robertson [29] states that the efficacy of disciplines like requirements engineering remains still unclear in highly inventive/creative domains. What makes VGSD further complicated is that the created software does not have a productive purpose in a narrower sense. From an economic view, products can be distinguished in products for value creation (utilitarian) and products for no value creation purposes (non-utilitarian) like movies or music with their major purpose of entertainment and pleasure [14]. Video games as interactive entertainment software can also be regarded as non-utilitarian software products [26] that have affective, experiential qualities [9,16,15] which are neither captured by current non-functional requirements [9,11] nor by any other means [3,13].

Due to the cross-disciplinary complexity of VGSD and the lack of research in major areas like requirements, the video games industry faces various challenges [27].

- *Technology* challenges include first-mover disadvantage in development of new technology, integration of external components and the concurrent development for multiple target platforms [8].
- *Project scope challenges* are caused by a missing knowledge about affective requirements and missing elicitation and validation methods that could efficiently address such requirements [8,9].
- *Scheduling challenges* consist of hard deadlines due to the season-dependent nature of video games (Christmas) and of insufficient workflow integration/ orchestration of the involved production disciplines (arts, engineering) [8,9].

[2] EU GameTools Project. `http://www.gametools.org` (last visited 05/31/2010).

Technological and project scope challenges lead to an effect called *crunch time*, which is a games industry term and describes a significant raise of overtime work over short to medium term periods [10]. The absence of established requirement elicitation techniques often promotes a behavior called *feature creep*. Feature creep is caused by ad-hoc and frequent adding/changing of game functionality and a mismanagement of remaining time, available resources and desired product quality [10].

### 2.2 Collaboration within Heterogeneous Teams

Various disciplines (artists, designers and engineers) are major challenges in video game development projects because they have to collaborate and interact. Video games combine smaller parts into a complex system. Game stories drive the game through individual episodes, visual and audio effects including complex user interaction mechanisms, game engines for game controlling purposes, and hardware components (different target platforms) are the major component of a typical video game [8]. Nevertheless, domain experts elaborate on individual parts of the video game applying various tools and methods according to their role and discipline. A major challenge is to get a common view on the video game, exchange data across disciplines, and enable defect detection across disciplines and tool-borders. Similar to complex automation systems [6,23], where various disciplines have to collaborate to construct software-intensive (software+) systems, e.g. manufacturing systems [4,22,23] involving mechanical, electrical and software engineers [5,6], VGSD has to handle comparable challenges [9,27].

Figure 1 presents three selected important disciplines and the overlapping areas in VGSD. Nevertheless, collaboration requires tool-support for interaction and data exchange, e.g. cross-border versioning systems supporting software code and 2D/3D assets (Avid Alienbrain), production management software (Hansoft), lightweight interaction programming frameworks based on end-user experience prototyping (Adobe Flex, Microsoft XNA), visual scripting systems (Epic UnrealKismet) and interactive live previews (Crytek Sandbox2 editor). These tools support artists, designers and software engineers by point-to-point integration. Nevertheless, there is no comprehensive integration and limitations to process support with respect to supporting VGSD.

### 2.3 Game Development Processes

To our knowledge there are limitations of systematic video game development processes. Thus, we conducted an initial web-survey in the Austrian games industry to identify state of the practice and possible future trends regarding process and method support [24]. The study included 20 game development studios (small studios with a predominantly staff size of 4 members: 85%, and large studios with more than 15 staff members: 15%). The response rate of the web survey was 65%. Basically, we identified a trend to unstructured and highly flexible software processes.

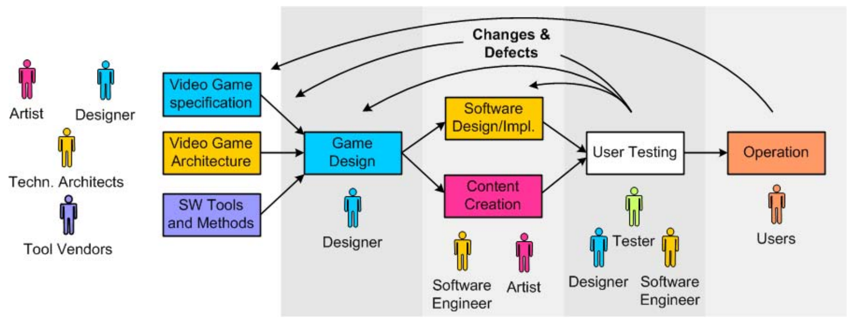

**Fig. 2.** Simple workflow-oriented process for game development

The trend of the survey showed that 23% of the responses do not use any software process, but develop games ad-hoc. Additionally, we observed that 77% of the respondents use some kind of flexible process for project management (Scrum [30]) and software development (XP [2]). An interesting finding was that no studio applied a traditional software process, e.g. V-Modell [3] or RUP [19].

All studios applied some kind of sequential process approach aligned with ad-hoc development or flexible and agile process approaches to address the missing link of heterogeneous disciplines. See figure 2 for the basic components of the process key elements. Note that isolated process steps are executed in an agile way, comparable to Scrum. Nevertheless, Scrum focuses on software development in small iterations (Sprints) with respect to constructing high-quality software products, with limitations to integrating various disciplines, as required by VGSD.

Figure 2 presents a basic waterfall-like workflow, based on a simple process model, to illustrate selected involved stakeholders and the impact of changes, derived from various disciplines. Based on these observations, we see Scrum [30] as a promising approach for game development application. There is a need to include heterogeneous disciplines, e.g.creative arts, game designers and software engineers, in the process model to enable (a) flexible development of video games, (b) improved collaboration between disciplines and (c) an improvement of video game development projects.

## 3 Research Issues

Based on the related work and the identified research challenges, we will focus on two major research questions.

*RQ1: Identification of an agile game development process that fits better to inventive/creative projects with high uncertainties*[4].

Our previous work indicates that Scrum and agile processes are rather popular among game developers [24]. Therefore we use Scrum as a starting point and

[3] `http://www.v-modell-xt.de` (last visited 05/31/2010).

[4] Emerging markets, emerging technology, little existing research in target domain.

enhance the regular Scrum process regarding affective/aesthetic (non-functional) requirements to enable consideration and collaboration of various disciplines. Introducing a domain-tailored process we expect a reduction of feature creep behavior due to more stable requirements and a structured way to better integrate changes in later project phases.

*RQ2: How can heterogeneous teams be integrated in the proposed process?*

A particular problem in game development is the integration of disciplines from non-engineering domains like graphic design, since their unstructured, opportunistic workflow differs considerably from the conservative approaches in engineering. Missing tighter multi-disciplinary workflow integration/orchestration causes workload irregularities (over-/under-production) which leads to effects like crunch time. The process has to be flexible enough to concurrently support different workflow styles and implicitly foster cross-collaborative practice.

## 4 Proposed Process

In the following section we present a process (figure 3) based on Scrum that is enriched with further elements in order to allow better discipline integration and workflow scheduling. Basically, Scrum consists of three phases: Pre-Game (definition of basic systems architecture and collection of customer requirements and changes within a product backlog), Sprint (iterative development of sprint log items, i.e. a manageable sub set of prioritized product backlog items) and Post-Game (delivery of product increments after sprint completion). See Figure 3.a for a schematic overview of the well-established Scrum process.

The proposed processes is separated into three main areas pre-production, production and project closure. Pre-production is invention-focused and short-term oriented, whereby production is sustainability-oriented and suited for long-term development. Pre-production deals with the prototyping of the key game elements, risk assessment and project requirements. Production covers the actual implementation and is subdivided into three process timelines: vision loop, sprint and validation loop. Project closure covers product distribution and the overall project warp-up.

### 4.1 Pre-production

The main task of pre-production is to identify possible software project candidates, as well as requirement analysis, risk assessment and general project requirements like financing. Identification of software project candidates is mainly achieved by applying pragmatic bottom-up approaches like various forms of end-user oriented prototyping [1] (explorative development, rapid team prototyping [25]). End-user oriented prototyping provides a mocked version of the possible end-product sharing almost the same experiential qualities (interaction, graphics, audio). Well crafted experience prototypes are kind of vertical slice through the intended product that implicitly cover many product subtleties and complexities within comparatively short development time. Combining prototypes with

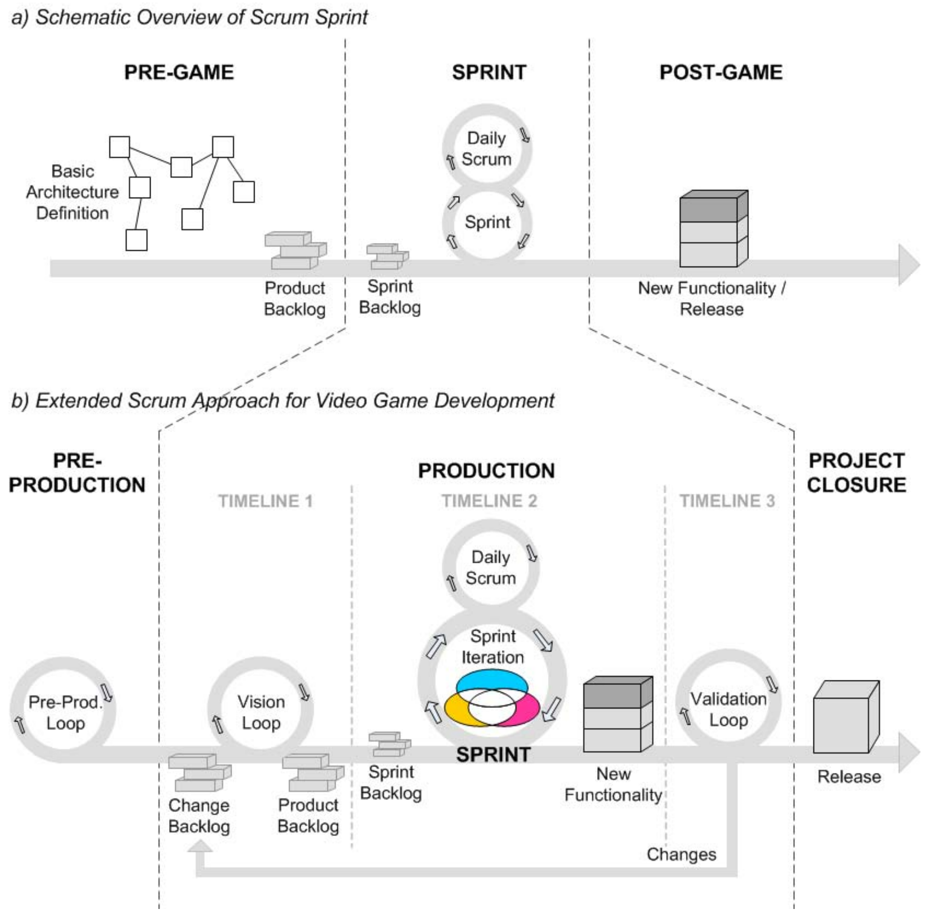

**Fig. 3.** Proposed game development process based on Scrum

formal design and specification methods, e.g. object oriented modeling, can lead to semi-formal *interactive specifications*. Such specifications can be used during implementation to not only cover functional correctness, but also correct end-user experience. Depending on the project, pre-production can last from few weeks up to a year and can require up to 25% of the whole development time [10].

## 4.2 Production

Production receives the complete project package from pre-production and creates a sellable product within given time, money and quality. The overall production workflow is based on Scrum, whereby it is separated into the three process timelines: vision loop, sprint, and validation loop. Each process has the same sprint iteration duration. Key stakeholder like lead programmer, creative director or executive producer are not limited to one process, but may be involved in all three.

### Vision Loop

The vision loop is comparable to the *product owner* in a regular Scrum and involves technical and creative authorities (lead designer, artists, developers). It maintains product vision over the project life cycle and manages the *product backlog*. The product backlog resembles a regular Scrum backlog with the difference that user stories request experience-prototypes in order to clarify ambiguous affective requirements at early stage. Further, the vision loop receives test results from the validation loop and maintains them in a *change backlog*. Larger product changes are prototyped and carefully evaluated before added to the backlog. Smaller changes are prioritized and added or discarded. At production start up, pre-production hands over the complete project package to the vision loop which evaluates the project package and creates the product backlog and preliminary specifications for the validation loop.

### Sprint

The sprint sub-process executes a regular Scrum sprint and creates a complete product increment at the end of a sprint. Integration of heterogeneous disciplines into a sprint is achieved by inserting discipline-specific *sprint workflows* which are run concurrently during a sprint. An example of a sprint workflow for programmers may be test driven development. Sprint workflows for non-engineering disciplines like graphic design or level design are harder to create and often involve pull systems [28] or Scrum derivates[17] like Scrum-ban[5] [20]. *Daily Scrums* are used for coordination across disciplines by involving sprint team members from all disciplines and to keep the 'big picture' of all team members updated. At sprint end a complete product increment has been created. Depending on the iteration duration of sprint and validation loop, some sprints may pass until the product is forwarded to the validation loop.

### Validation Loop

The validation loop covers product validation, verification and is responsible for overall quality control and greenlights critical product milestones (public beta version, RTM master). Although preliminary testing is already done during sprints, the complexity of games requires time-intensive testing procedures which would not fit into a single sprint. Such tests would include: stress tests, balance testing, user testing, focus group analyses, age rate testing, pre-certification testing, functionality testing, license verification testing. At the end of an iteration, test results are forwarded to the vision loop for further evaluation and review.

## 4.3 Project Closure

Project closure is the last process area and covers the distribution of the final game as well as retrospective analysis (postmortems), processing of created

[5] `http://leansoftwareengineering.com/ksse/scrum-ban` (last visited 05/31/2010).

tools and integration of lessons-learned into the company's knowledge base. Also, *closing kits* are created during this phase. A closing kit is a compilation of final art/code assets, tools and documentation and is used to create a working version of the game without assistance from the original development team [10]. Closing kits are used for patching/modifying an existing game or porting it to another platform.

### 4.4 Expected Benefits and Feedback from Game Developers

The proposed flexible process approach, based on Scrum, has been preliminary evaluated through lively discussions within a national game developer community meeting and a set of informal reviews in the local games industry. The feedback from the meeting and the reviews showed that the proposed process approach was found reasonable and useful and was classified as a promising process for video game development projects. Main expected benefits derived from the community were:

– The solution approach promise to reduce crunch time due to the concept of project timeliness.
– The pre-production phase provides sufficient space to teams to elaborate on potential ideas and takes into account bottom-up approaches like concept-/prototype-driven design. Such approaches are confirmed by game developers, and by our own experience, leading to good end-user focused requirements approximation [1] (thus reducing feature creep) at early project phases [25], but requires seasoned rapid prototypes.
– Game design is not as a single step within a sprint but as a concurrently running process. This process approach enables frequent changes, continuous change inclusion and resembles process structures of industries which have already mastered aesthetics in engineering, like BMW designworksUSA[6]. Concurrent running end-user experience centered game evaluation is already practiced by Microsoft on a large scale for eight years with promising results[7].

Observations of the trends in the international games industry[8,9] suggest that the proposed process could be a promising way for future work.

## 5 Discussion and Conclusion

The previous section demonstrates a game development tailored-process based on Scrum that can be used to improve workflow integration and collaboration

[6] `http://www.designworksusa.com/vision/process` (last visited 05/31/2010).
[7] `http://www.mgsuserresearch.com/publications` (last visited 05/31/2010).
[8] Stanford University's Entrepreneurship Corner lectures - Electronic Arts. `http://ecorner.stanford.edu/search.html?keywords=EA` (last visited 05/31/2010).
[9] IGDA Leadership Forum. `http://www.igda.org/leadership/` (last visited 05/31/2010, coverage temporarily offline. Videos can be made available by the authors at request).

of heterogeneous game development teams. Separating pre-production and production steps can decouple pragmatic invention-focused bottom-up approaches from a conservative production approach that is suitable for long-term project cycles. Integration of heterogeneous disciplines is achieved by executing multiple discipline-specific workflows during one production sprint iteration that are adjusted by daily Scrums. Such an approach enables each discipline to apply workflows within they are most proficient by concurrently producing in accordance to the pace and demand of all other disciplines. The overall quality of the created product snapshot is validated within a separated process (validation loop), optimized for qualitative and quantitative evaluation of multimodal software. Results are fed back into a synthesis process (vision loop) where the findings are evaluated and probably lead to artistic, technical and interaction design changes in the product backlog. Its is important to be aware, that although there are three concurrent timelines, some key stakeholders (e.g. creative director, art director, executive producers) will be involved in all processes in order to maintain the 'big picture'.

Preliminary evaluation results derived from discussions within a game community meeting and selected reviews of game development experts confirmed feasibility and usefulness of the proposed approach as a promising candidate process approach for video game development practices. These first informal results showed that an extended Scrum approach can increase video game development practice involving stakeholders from different disciplines.

Nevertheless, future research work is required to (a) elaborate on the proposed process approach in more detail, (b) investigate the effects of integrating heterogeneous environments in video game development in a prototype study, (c) evaluate the process approach in a controlled environment and industry to verify/validate expected benefits more systematically, (d) broaden to the scope of the survey in a larger context, (e) examination of different cycle times for vision, sprint and validation loop.

## References

1. Agustin, M., Chuang, G., Delgado, A., Ortega, A., Seaver, J., Buchanan, J.W.: Game sketching. In: Proc. 2nd Int'l Conf. Digital Interactive Media in Entertain. & Arts, pp. 36–43. ACM, Perth (2007)
2. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change, 2nd edn. Addison-Wesley Prof., Reading (2004) 978-0321278654
3. Bentley, T., Johnston, L., Baggo, K.: Putting some emotion into requirements engineering. In: Proc. 7th Australian Workshop on Requirements Engineering, Melbourne, AUS (2002)
4. Biffl, S., Moser, T., Sunindyo, W.D.: Bridging semantic gaps between stakeholders in the production automation domain with ontology areas. In: Proc. 21st Int'l Conf. Software Eng. & Knowledge Eng. (SEKE '09), Boston, MA, USA, pp. 233–239 (2009)
5. Biffl, S., Schatten, A.: A platform for service-oriented integration of software engineering environments. In: Proc. 8th Conf. New Trends in Softw. Method., Tools & Techniques (SoMeT '09), pp. 75–92. IOS Press, Pargue (2009)

6. Biffl, S., Schatten, A., Zoitl, A.: Integration of heterogeneous engineering environments for the automation systems lifecycle. In: 7th Int'l Conf. Industrial Informatics (INDIN '09), Cardiff, UK, pp. 576–581 (2009)
7. Birdwell, K.: The cabal: Valve's design process for creating Half-Life. In: Salen, K., Zimmerman, E. (eds.) The Game Design Reader: A Rules of Play Anthology, 1st edn., pp. 212–225. The MIT Press, Cambridge (2006)
8. Blow, J.: Game development: Harder than you think. ACM Queue 1(10), 28–37 (2004)
9. Callele, D., Neufeld, E., Schneider, K.: Requirements engineering and the creative process in the video game industry. In: Proc. 13th IEEE Int'l Conf. Requirements Eng (RE '05), pp. 240–252. IEEE CS Press, Los Alamitos (2005)
10. Chandler, H.: The Game Production Handbook, 1st edn. Charles River Media, Boston (2006) 978-1584504160
11. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Boston (2000) 978-0792386667
12. Flynt, J.P., Salem, O.: Software Engineering for Game Developers. In: LaMothe, A. (ed.) Game Development Series. Thomson Course Technology PTR, Boston (2005) 978-1592001552
13. Hassenzahl, M., Beu, A., Burmester, M.: Engineering joy. IEEE Softw. 18(1), 70–76 (2001)
14. Hirsch, P.M.: Processing fads and fashions: An organization-set analysis of cultural industry systems. The American Journal of Sociology 77(4), 639–659 (1972)
15. Isbister, K.: Better Game Characters by Design: A psychological Approach, 1st edn. Morgan Kaufmann, San Francisco (2006) 978-1558609211
16. Järvinen, A.: Games without Frontiers: Theories and Methods for Game Studies and Design. PhD thesis, University of Tampere (March 2008), `http://acta.uta.fi/pdf/978-951-44-7252-7.pdf`
17. Keith, C.: Beyond scrum: Lean and kanban for game developers. Gamasutra (2008), `http://www.gamasutra.com/view/feature/3847`
18. Kress, G., Leeuwen, T.V.: Multimodal Discourse, Arnold, London, UK (2001) 978-0340608777
19. Kruchten, P.: The Rational Unified Process: An Introduction, 3rd edn. Addison-Wesley Prof., Reading (2003) 978-0321197702
20. Ladas, C.: Scrumban - Essays on Kanban Systems for Lean Software Development. Modus Cooperandi Press (2009) 978-0578002149
21. Land, S.K., Wilson, M.B.: Using IEEE standards to support America's Army gaming development. Computer 39(11), 105–107 (2006)
22. Merdan, M., Moser, T., Wahyudin, D., Biffl, S., Vrba, P.: Simulation of workflow scheduling strategies using the MAST test management system. In: 10th Int'l Conf. Control, Automation, Robotics & Vision (ICARCV 08), Hanoi, VNM, pp. 1172–1177 (2008)
23. Moser, T., Biffl, S., Sunindyo, W.D., Winkler, D.: Integrating production automation expert knowledge across engineering stakeholder domains. In: Int'l Conf. Complex, Intelligent & Softw. Intensive Systems (CISIS '10), Krakow, POL (2010)
24. Musil, J., Schweda, A., Winkler, D., Biffl, S.: A Survey on the State of the Practice in Video Game Software Development. Technical report, QSE-IFS-10/04, TU Wien (2010) `http://qse.ifs.tuwien.ac.at/publication/IFS-QSE-10-04.pdf`
25. Musil, J., Schweda, A., Winkler, D., Biffl, S.: Synthesized Essence: What Game Jams Teach About Prototyping of New Software Products. In: Proc. 32nd Int'l Conf. Software Engineering (ICSE'10) - New Ideas and Emerging Results, pp. 183–186. ACM, Cape Town (2010)

26. Peltoniemi, M.: Life-cycle of the games industry: the specificities of creative industries. In: Proc. 12th Int'l Conf. Entertain. & Media in the Ubiquitous Era, pp. 54–58. ACM, Tampere (2008)
27. Petrillo, F., Pimenta, M., Trindade, F., Dietrich, C.: What went wrong? A survey of problems in game development. Comput. Entertain. 7(1), 1–22 (2009)
28. Poppendieck, M.: Lean Software Development: An Agile Toolkit. Addison-Wesley, Reading (2003)
29. Robertson, J., Heitmeyer, C.: Point/counterpoint. IEEE Softw. 22(1), 48–51 (2005)
30. Schwaber, K.: Agile Project Management With Scrum. Microsoft Press, Redmond (2004)
31. Sommerville, I.: Software Engineering, 8th edn. Addison Wesley, Reading (2006) 978-0321313799