HAL open science

Modding as an Open Source Approach to Extending Computer Game Systems

Walt Scacchi

▶ To cite this version:

Walt Scacchi. Modding as an Open Source Approach to Extending Computer Game Systems. 9th Open Source Software (OSS), Oct 2011, Salvador, Brazil. pp.62-74, 10.1007/978-3-642-24418-6_5 . hal-01570749

HAL Id: hal-01570749

https://inria.hal.science/hal-01570749

Submitted on 31 Jul 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Modding as an Open Source Approach to Extending Computer Game Systems

Walt Scacchi

Institute for Software Research and
Center for Computer Games and Virtual Worlds
University of California, Irvine
wscacchi@ics.uci.edu

Abstract. This paper examines what is known so far about the role of open source software development within the world of game mods and modding practices. Game modding has become a leading method for developing games by customizing or creating OSS extensions to game software in general, and to proprietary closed source software games in particular. What, why, and how OSS and CSS come together within an application system is the subject for this study. The research method is observational and qualitative, so as to highlight current practices and issues that can be associated with software engineering and game studies foundations. Numerous examples of different game mods and modding practices are identified throughout.

1. Introduction

User modified computer games, hereafter *game mods*, are a leading form of user-led innovation in game design and game play experience. But modded games are not standalone systems, as they require the user to have an originally acquired or licensed copy of the unmodded game software.

Modding, the practice and process of developing game mods, is an approach to end-user game software engineering [4] that establishes both social and technical knowledge for how to innovate by resting control over game design from their original developers. At least four types of game mods can be observed: user interface customization; game conversions; machinima; and hacking closed game systems. Each supports different kinds of open source software (OSS) extension to the base game or game run-time environment. Game modding tools and support environments that support the creation of such extensions also merit attention. Furthermore, OSS game extensions are commonly applied to either proprietary, closed source software (CSS) games, or to OSS games, but generally more so to CSS games. Why this is so also merits attention. Subsequently, we conceive of game mods as covering customizations, tailorings, remixes, or reconfigurations of game

embodiments, whether in the form of game content, software, or hardware denoting our space of interest.

The most direct way to become a game mod developer (a game *modder*) is through self-tutoring and self-organizing practices. Modding is a form of learning – learning how to mod, learning to be a game developer, learning to become a game content/software developer, learning computer game science outside or inside an academic setting, and more [5,20]. Modding is also a practice for learning how to work with others, especially on large, complex games/mods. Mod team efforts may also self organize around emergent software development project leaders or "want to be" (W.T.B.) leaders, as seen for example in the *Planeshift* (http://www.planeshift.it/) OSS massively multiplayer online role-playing game (MMORPG) development and modding project [20].

Game mods, modding practices, and modders are in many ways quite similar to their counterparts in the world of OSS development, even though they often seemingly isolated to those unaware of game software development. Modding is increasingly a part of mainstream technology development culture and practice, and especially so for games, but also for hardware-centered activities like automobile or personal computer customization. Modders are players of the games they reconfigure, just as OSS developers are also users of the systems they develop. There is no systematic distinction between developers and users in these communities, other than there are many users/players that may contribute little beyond their usage, word of mouth they share with others, and their demand for more such systems. At OSS portals like SourceForge.net, the domain of "Games" is the second most popular project category with nearly 42K active projects, or 20% of all projects[1]. These projects develop either OSS-based games, game engines, or game development tools/SDKs, and all of the top 50 projects have each logged more than 1M downloads. So the intersection of games and OSS covers a substantial socio-technical plane, as game modding and traditional OSS development are participatory, user-led modes of system development that rely on continual replenishment of new participants joining and migrating through project efforts, as well as new additions or modifications of content, functionality and end-user experience [19,20,21]. Modding and OSS projects are in many ways experiments to prototype alternative visions of what innovative systems might be in the near future, and so both are widely embraced and practiced primarily as a means for learning about new technologies, new system capabilities, new working relationships with potentially unfamiliar teammates from other cultures, and more [cf. 21].

Consequently, game modding appears to be (a) emerging as a leading method for developing or customizing game software; (b) primarily reliant of the development and use of OSS extensions the ways and means for game modding; and (c) overlapping a large community of OSS projects that develop computer game software and tools that has had comparatively little study. As such, the research

[1] See http://www.sourceforge.net/softwaremap/index.php, accessed 15 April 2011. The number one category of projects is for "Development" with more than 65K OSS projects, out of 210K projects. So OSS Development and OSS Games together represent half of the projects currently hosted on SourceForge.

questions that follow then are why do these conditions exist, how have they emerged, and how are they put into practice in different game modding efforts.

This paper seeks to examine what is known so far about game mods and modding practices. The research method in this study is observational and qualitative. It seeks to snapshot and highlight current practices that can be associated with software engineering and game studies, as well as how these practice may be applied in CSS versus OSS game modding. Numerous examples of different game mods and modding practices are identified throughout to help establish an empirically grounded baseline of observations, from which further studies can build or refute. Furthermore, the four types of game mods and modding practices identified in this paper have been employed first-hand in game development projects led or produced by the author. Such observation can subsequently serve as a basis for further empirical study and technology development that ties together computer games, OSSD, software engineering, and game studies [19,20,21,22].

2. Related Research

Two domains of research inform the study here: software extension within the field of software engineering, and modding as cultural practice within game studies. Each is addressed in turn.

2.1 Software Extension

Game mods embody different techniques and mechanisms for software extension. However, the description of game mods and modding is often absent of its logical roots or connections back to software engineering. As suggested, mods are extensions to existing game software systems, so it is appropriate to review what we already know about software extensions and extensibility.

Parnas [15] provides an early notion of software extension as an expression of modular software design. Accordingly, modular systems are those whose components can be added, removed, or updated while satisfying the original system functional requirements. Such concepts in turn were integrated into software architectural design language descriptions and configuration management tools [14]. But reliance on explicit software architecture descriptions is not readily found in either conventional game or mod development. Hentonnen and colleagues [8] examine how software plug-ins support architectural extension, while Leveque, et al. [11] investigate how extension mechanisms like views and model-based systems support extension, also at the architectural level. Last, the modern Web architecture is itself designed according to principles of extensibility through open APIs, migration across software versions, network data content/hypertext transfer protocols, and representational state transfer [6]. Mod-friendly networked multi-player games often take advantage of these capabilities.

Elsewhere, Batory and associates [3] describe how domain-specific languages (for scripting) and software product lines support software extension, and now such techniques are used in games that are open for modding. Next, OSS development as a complementary approach to software engineering, relies on OSS code and associated online artifacts that are open for extension through modification and redistribution of their source representations [21]. Finally, other techniques to extend the functionality or operation of an existing CSS system may include unauthorized modifications that might go beyond what the end-user license agreement might allow, and so appear to fall outside of what software engineering might anticipate or encourage. These include extensions via hacking methods like code injection or hooking, whose purpose is to gain/redirect control of normal program flow through overloading or intercepting system function calls, or provide a hidden layer of interpretation, which allow for "man in the middle" interventions. So software extensions and extensibility is a foundational concept in software engineering, as well as foundational to the development of game mods. However, the logical connections and common/uncommon legacy of game modding, OSS development, and software engineering remain under specified, which this paper begins to address.

2.2 Modding as Cultural Practice

Game modding is a practice for user content creation that creates/networks not only game mods but game modders. Within anthropological, behavioral, and sociological studies of computer game play, modding has been studied as an emerging cultural practice that mediates both game play and player interaction with other players (including the game's developers). In some early studies, modding has been designated as a form of "playbour" whereby player actions to create game extensions for use by other players is observed as a form of unpaid (or underpaid) labor that primarily benefits the financial and property interests of game development corporations or hegemonic publishers [10,16,26].

Game modding also modifies or transforms game play experience, since what is play and what is experience(d) are culturally situated. Examples of this may include single player games being modded into multi-player games. So the experience of single player versus the game environment is transformed into other situations including player versus player, multi-player group play, or team versus team play. Similarly, the modding of games to enable experiences other than expected game play, like using a modded game for storytelling or film-making experiences is also a practice of growing interest, with the emergence of a distinguishable community of gamer-filmmakers who produce *machinima* (described later) as either a literary medium, or an art form [9,12,13].

Other studies have observed that user/modders also benefit from modding as a way to achieve a sense of creative ownership and meaning in the modded games they share and play with others [17,19,20,23], and that game mods and modding practices become central elements in what constitutes play with and through games [24]. Finally, as already observed, OSS project portals like SourceForge host thousands of OSS game development projects that develop and deploy role-playing games (4.3K projects), simulation-based games (2.6K), board games (2.3K), side-scrolling/arcade

games (2K), turn-taking strategy games (1.7K), multi-user dungeons or text-based adventure/virtual worlds (1.6K), first-person shooters (1.6K), MMORPG (0.6K) and more. So development of OSS games and related game development tools can be recognized as a central element in the cultural world of computer games and game development, as well as the world of OSS development [19,20,21].

3. Four Types of Game Mods

At least four types of game mods are realized through OSS development practices. These include (i) user interface customizations and agents, (ii) game conversions, (iii) machinima, and (iv) hacking closed source game systems. Each is examined in turned, and each is facilitated (or prohibited) according to its copyright license.

3.1 User Interface Customizations and Agents

User interfaces to games embody the practice and experience of interfacing users (game players) to both the game system and the play experience designed by the game's developers. Game developers act to constrain and govern what users can do, and what kinds of experience they can realize. Some users in turn seek to achieve a form of competitive advantage during game play by modding the user interface software for their game, when so enabled by game developers. These mods acquire or reveal additional information that users believe will help their play performance and experience. User interface add-ons subsequently act as the medium through which game development studios support game product customization, which is a strategy for increasing end-user satisfaction and thus the likelihood of product success [4].

Three kinds of user interface customizations can be observed. First and most common, is the player's ability to select, attire or accessorize a *player's in-game identity*. Second, is for players to customize *the color palette and representational framing borders* of the their game display within the human-computer interface, much like what can also be done with Web browsers (e.g, Firefox 4 “personas” and “themes”) and other end-user software applications. Third, are *user interface add-on modules* that modify the player's in-game information management dashboard, but do not modify the underlying game play rules or functions. These add-ons provide additional information about game play state that may enhance the game play experience, as well as increasing a player's sense of immersion or omniscience within the game world through perceptual expansion. This in turn enables awareness of game events not visible in the player's pre-existing in-game view. Furthermore, some add-on facilities (e.g., those available with the proprietary *World of Warcraft* MMORPG, scripted in the LUA language) accommodate the creation of automated agent scripts that can read/parse data streamed to the UI within an existing or other add-on dashboard component, and then provide some additional value-added play experience, such as sending out messages or status reports to other players automatically. Such add-on agents thus modify or reconfigure the end-user play

experience, rather than the core functionality or play mechanics available to all other of the game's players. Consequently, the first two kinds of customizations result from meta-data selections within parametric system functions, while the third represents a traditional kind of user-created modular extension; one that does not affect the pre-existing game's functional requirements, nor one included in the operational source code base during subsequent system builds or releases, unless they do alter the software's requirements (e.g., by introducing a new security vulnerability or exploit that must be subsequently prevented).

3.2 Game Conversions

Game conversion mods are perhaps the most common form of game mods. Most such conversions are partial, in that they add or modify: (a) in-game characters including user-controlled character appearance or capabilities, opponent bots, cheat bots, and non-player characters; (b) play objects like weapons, potions, spells, and other resources; (c) play levels, zones, maps, terrains, or landscapes; (d) game rules; or (e) play mechanics. Some more ambitious modders go as far as to accomplish (f) total conversions that create entirely new games from existing games of a kind not easily determined from the originalgame. For example, one of the most widely distributed and played total game conversions is the *Counter-Strike* (CS) mod of the *Half-Life* (HL) first-person action game from Valve Software. As the success of the CS mod gave rise to millions of players preferring to play the mod over the original HL game, then other modders began to access the CS mod to further convert in part or full, to the point that Valve Software modified its game development and distribution business model to embrace game modding as part of the game play experience that is available to players who acquire a licensed copy of the HL product family. Valve has since marketed a number of CS variants that have sold over 10M copies as of 2008, thus denoting the most successful game conversion mod, as well as the most lucrative in terms of subsequent retail sales derived from a game mod.

Another example is found in games converted to serve a purpose other than entertainment, such as the development and use of games for science, technology, and engineering applications. For instance, the *FabLab* game [22] is a conversion of the *Unreal Tournament 2007* retail game, from a first-person shooter to a simulator for training semiconductor manufacturing technicians in diagnosing and treating potentially hazardous materials spills in a cleanroom environment. This conversion is not readily anticipated by knowledge of the Unreal games or underlying game engine, though it maintains operational compatibility with the Unreal game itself. So game conversions can re-purpose the look, feel, and intent of a game across application domains, while maintaining a common software product line [cf. 3].

Finally, it is common practice that the underlying game engine has one set of license terms and conditions to protect original work (e.g., no redistribution), while game mod can have a different set of terms and conditions as a derived work (e.g., redistribution allowed only for a game mod, but not for sale). In this regard, software licenses embody the business model that the game development studio or publisher seeks to embrace, rather than just a set of property rights and constraints. For example, in *Aion*, an MMORPG from South Korean game studio NCSoft, no user

created mods or user interface add-ons are allowed. Attempting to incorporate such changes would conflict with its EULA and subsequently put such user-modders at risk of losing their access to networked *Aion* multi-player game play. In contrast, the MMORPG *World of Warcraft* allows for UI customization mods and add-ons only, but no other game conversions, no reverse engineering of the game engine, and no activity intended to bypass WoW's encryption mechanisms. And, in one more variation, for games like *Unreal Tournament*, *Half-Life*, *NeverWinterNights*, *Civilization* and many others, the EULAs encourage modding and the free redistribution of mods without fee to others who must have a licensed copy of the proprietary CSS game, but not allowing reverse engineering or redistribution of the CSS game engine required to run the OSS mods. This restriction in turns helps game companies realize the benefit of increased game sales by players who want to play with known mods, rather than with the un-modded game as sold at retail. Mods thus help improve games software sales, revenue, and profits for the game development studio, publisher, and retailer, as well as enable new modes of game play, learning, and skill development for game modders.

3.3 Machinima

Machinima can be viewed as the product of modding efforts that intend to modify the visual replay of game usage sessions. Machinima employ computer games as their creative media, such that these new media are mobilized for some other purpose (e.g., creating online cinema or interactive art exhibition). Machinima focuses attention to playing and replaying a game for the purpose of story telling, movie making, or retelling of daunting or high efficiency game play/usage experience [12,13]. Machinima is a form of modding the experience of playing a specific game, by recording its visual play session history, so as to achieve some other ends beyond the enjoyment (or frustration) of game play. These play-session histories can then be further modded via video editing or remixing with other media (e.g., adding music) to better enable cinematic storytelling or creative performance documentation. Machinima is a kind of play/usage history process re-enactment [cf. 18] whose purpose may be documentary (replaying what the player saw or experienced during a play session) or cinematic (creatively steering a play session so as to manifest observable play process enactments that can be edited and remixed off-line to visually tell a story). Machinima mods are thus a kind of extension of game software use experience that is not bound to the architecture of the underlying game software system, except for how the game facilitates a user's ability to structure and manipulate emergent game play to realize a desired play process enactment history.

3.4 Hacking Closed Game Systems

Hacking a closed game system is a practice whose purpose oftentimes seems to be in direct challenge to the authority of commercial game developers that represent large, global corporate interests. Hacking proprietary game software is often focused not so much on how to improve competitive advantage in multi-player game play, but

instead is focused on expanding the range of experiences that users may encounter through use of alternative technologies [7,20]. For example, Huang's [7] study instructs readers in the practice of "reverse engineering" as a hacking strategy to understand both how a game platform was designed and how it operates in fine detail. This in turn enables reconfiguration of new innovative modifications or original platform designs, such as installing and running a Linux operating system (instead of Microsoft's proprietary CSS offering). While many game developers seek to protect their intellectual property (IP) from reverse engineering through end-user license agreements (EULAs) whose terms attempt to prohibit such action under threat of legal action, reverse engineering is not legally prohibited. Consequently, the practice of modding closed game consoles/systems is often less focused on enabling players to achieve competitive advantage when playing retail computer games, but instead may encourage those few so inclined for how to understand and ultimately create computing innovations through reverse engineering or other modifications.

Closed game system modding is a style of software extension by game modders who are willing to forego the "protections" and quality assurances that closed game system developers provide, in order to experience the liberty, skill, knowledge acquisition, conceptual appropriation ("pwned"), and potential to innovate, that mastery of reverse engineering affords. Consequently, players/modders who are willing to take responsibility for their actions (and not seek to defraud game producers due to false product warranty claims or copyright infringement), can enjoy the freedom to learn how their gaming systems work in intimate detail and to potentially learn about game system innovation through discovery and reinvention with the support of others like-minded [cf. 20]. Proprietary game development studios may sometimes allow for such mod-based infringement of their games. For example, the team of modders behind the hacking and conversion of the single-player CSS game, *Grand Theft Auto*, have produced an OSS (now GPL'd) game mod using code injection and hooking cheating methods to realize a networked multi-player variant called *Multi Theft Auto,* that Rockstar Games has chosen not to prosecute for potential EULA violation, but instead to embrace as GTA fan culture [25]. Nonetheless, large corporate interests may assert that their IP rights allow them to install CSS rootkits that collect potentially private information, or that prevent the reactivation of previously available OSS (e.g., the Linux Kernel on the Sony PS3 game console[2]) that game system hackers seek to undo.

Finally, games are one of the most commonly modified types of proprietary CSS that are transformed into "pirated games" that are "illegally downloaded." Such game modding practice is focused on engaging a kind of meta-game that involves hacking into and modding game IP from closed to (more) open. Game piracy has thus become recognized as a collective, decentralized and placeless endeavor (i.e., not a physical organization) that relies on torrent servers as its underground distribution venue for pirated game software. As recent surveys of torrent-based downloads reveals, in 2008 the top 10 pirated games represented about 9M downloads, while in 2009 the top 5 pirated games represent more than 13M downloads, and in 2010 the top 5 pirated games approached 20M, all suggesting a

[2] For details, see http://en.wikipedia.org/wiki/George_Hotz#Hacking_the_PlayStation_3 .

substantial growth in interest in and access to such modded game products[3]. Thus, we should not be surprised by the recent efforts of game system hackers that continue to demonstrate the vulnerabilities of different hardware and software-based techniques to encrypt and secure closed game systems from would be crackers. However, it is also very instructive to learn from these exploits how difficult it is to engineer truly secure software systems, whether such systems are games or some other type of application or package.

4. Game Modding Software Tools and Support

Games are most often modded with tools providing access to unencrypted representations of game software or game platform. Such a representation is accessed and extended via a domain-specific (scripting) language. While it might seem the case that game vendors would seek to discourage users from acquiring such tools, a widespread contrary pattern is observed.

Game system developers are increasingly offering software tools for modifying the games they create or distribute, as a way to increase game sales and market share. Game/domain-specific Software Development Kits (SDKs) provided to users by game development studios represent a contemporary business strategy for engaging users to help lead product innovation from outside the studio. Once Id Software, maker of the *DOOM* and *Quake* game software product line, and also Epic Games, maker of the *Unreal* software game product line, started to provide prospective game players/modders with software tools that would allow them to edit game content, play mechanics, rules, or other functionality, other competing game development studios were pressured to make similar offerings or face a possible competitive disadvantage in the marketplace. However, the CSS versions of these tools do not provide access to the underlying source code that embodies the proprietary game engine—a large software program infrastructure that coordinates computer graphics, user interface controls, networking, game audio, access to middleware libraries for game physics, and so forth. But the complexity and capabilities of such a tool suite mean that any one person, or better said, any game development or modding team, can now access modding tools or SDKs to build commercial quality CSS games through OSS extensions. But mastering these tools appears to be an undertaking likely to be only of interest to highly committed game developers who are self-supported or self-organized.

In contrast to game modding platforms provided by game development studios, there are also alternatives provided by the end-user community. One approach can be seen with facilities provided in meta-mods like *Garry's Mod* or the *AMX Mod X* mod-making package. Modders can use these packages to construct a variety of plug-ins that provide for development of in-game contraptions as game UI agents or user created art works, or to otherwise create comic books, program game

---

[3] For 2008, see http://torrentfreak.com/top-10-most-pirated-games-of-2008-081204/
For 2009, http://torrentfreak.com/the-most-pirated-games-of-2009-091227/
For 2010, http://torrentfreak.com/call-of-duty-black-ops-most-pirated-game-of-2010-101228/

conversions, and produce other kinds of user created content. But both packages require that you own a licensed CSS game like *Counter-Strike: Source*, *Half-Life2* or *Day of Defeat: Source* from Valve Software.

A different approach to end-user game development platforms can be found arising from OSS games and game engines. The *DOOM* and *Quake* games and game engines were released as free software subject to the GPL, once they were seen by Id Software as having reached the end of their retail product cycle. Thousands of games/engines, as already observed, have been developed and released for download. Some started from the OSS that was previously the CSS platform of the original games. However, the content assets (e.g., in-game artwork) for many of these CSS- then-OSS games are not covered by the GPL, and so user-developers must still acquire a licensed copy of the original CSS game if its content is to be reused in some way[4]. Nonetheless, some variants of the user-created GPL'd games now feature their own content that is limited/protected by Creative Commons licenses.

5. Opportunities and Constraints for Modding

Game modding demonstrates the practical value of software extension as a user-friendly approach to customizing software. Such software can extend games open to modding into diverse product lines that flourish through reliance on domain-specific game scripting languages, and integrated SDKs. Modding also demonstrates the success of end-users learning how to extend software to create custom user interface add-ons, system conversions, replayable system usage videos, as well as to discover security vulnerabilities. Game modding therefore represents a viable form of end-user engineering of complex software that may be transferable to other domains.

Modding is a form of OSS-enabled collaboration. It is collaboration at a distance where the collaborators, including the game developers and game users, are distant in space and time from each other, yet they can interact in an open but implicitly coordinated manner through software extensions. Comparatively little explicit coordination arises, except when CSS game developers seek to embrace and encourage the creation of OSS game mods that rely on the proprietary CSS game engine (and also SDK), as a way to grow market share and mid share for the proprietary engine as a viable strategy to entry into the game industry.

However, mods are vulnerable to evolutionary system version updates that can break the functionality or interface on which the mod depends. This can be viewed as the result of inadequate software system design practice, such that existing system modularization did not adequately account for software extensions that end-users seek, or else the original developer wanted to explicitly prohibit end-users from making modifications that transform game play mechanics/rules or unintentionally allow for modification or misappropriation of copy protected code or media assets.

Last, one the key constraints on game modding in particular, and software extension in general, are the rights and obligations that are expressed in the original

[4] For example, see http://assault.cubers.net/docs/license.html, accessed 13 April 2011.

software EULA. Mods tend to be licensed using OSS or freeware licenses that allow for access, study, modification, and redistribution, rather than using free software licenses (e.g., GPLv2 or GPLv3). Software extensions that might be subject to a reciprocal GPL style license require that the base/original software system incorporate an explicit software architectural design that requires the propagation of reciprocal rights across an open interface, except through an LGPL software shim [1]. Otherwise, the scope of effectiveness and copyright protections of either free or non-free software (or related media assets) cannot be readily determined, and thus may be subject to copyright infringement or licenses non-compliance allegations. They may also be treated as social transgressions within a community of modders whose perceived ownership of the game mods demands respect and honor of a virtual license that may or may not be legally valid [2]. As the OSS community has long recognized, software rights and freedoms are expressed through IP licenses that insure whether or not a person has the right to access, study, modify, and redistribute the modified software, as long as the obligation to include a free software license is included that restates these rights in unalterable form, is included with the OSS code and its modified distributions.

6. Conclusions

Modding is emerging as a viable approach for mixing proprietary CSS systems with OSS extensions. The result is modded systems that provide the benefits of OSSD to developers of proprietary CSS systems, and to end-users who want additional functionality of their own creation, or from others they trust and seek to interact with through game play.

In contrast, modding is not so good for protecting software and media/content copyrights. Modding tests the limits of software/IP copyright practices. Some modders want to self-determine what copy/modding rights they have or not, and sometimes they act in ways that treat non-free software and related media as if it were free software. Who owns what, and which copy rights or obligations apply to that which is modded, are core socio-technical issues when engaging in modding.

This study helps to demonstrate that game modding is becoming a leading method for developing or customizing game software, whether based on proprietary CSS or OSS game systems. OSS-based software extensions are the leading ways and means for modding game-based user interfaces, converting games from one style/genre to another, for recording game play sessions for cinematic production and replay, and for hacking closed source game systems. Finally, the development of computer game software and tools itself represents a large community of OSS projects that has had comparatively little study, and thus merits further attention as its own cultural world as well as one for OSS development. This last consideration may be important as other empirical studies of OSS development that rely on data from SourceForge will increasingly include OSS game projects within large project samples. This study has therefore begun to address why and how these conditions have they emerged, and how are they put into practice in different game modding efforts. Future study should also

consider whether and how modding might be applied and adopted in other application domains where CSS can be extended through OSS mods.

7. Acknowledgments

The research described in this paper has been supported by grants #0808783 and #1041918 from the National Science Foundation, and grant #N00244-10-1-0077 from the Naval Postgraduate School. No review, approval or endorsement implied. The anonymous reviewers also provided helpful suggestions for improving this paper.

References

1. Alspaugh, T.A., Asuncion, H.A., Scacchi, W.: Intellectual Property Rights Requirements for Heterogeneously Licensed Systems, in *Proc. 17th. Intern. Conf. Requirements Engineering (RE09*), Atlanta, GA, 24-33, September (2009)
2. Alspaugh, T.A., Scacchi, W., Asuncion, H.A.: Software Licenses in Context: The Challenge of Heterogeneously Licensed Systems, *J. Assoc. Information Systems*, 11(11), 730-755, November (2010)
3. Batory, D., Johnson, C., MacDonald, B., von Heeder, D.: Achieving extensibility through product lines and domain specific languages: a case study, *ACM Trans. Software Engineering and Methodology*, 11(2), 191-214 (2002)
4. Burnett, M., Cook, C., Rothermel, G., End-User Software Engineering, *Communications ACM*, 47(9), 53-58 (2004)
5. El-Nasr, M.S., Smith, B.K.: Learning Through Game Modding, *ACM Computers in Entertainment*, 4(1). Article 3B (2006)
6. Fielding, R.T., Taylor, R.N.: Principled Design of the Modern Web Architecture, *ACM Trans. Internet Technology*, 2(2), 115–150 (2002)
7. Huang, A.: *Hacking the Xbox: An Introduction to Reverse Engineering*, No Starch Press, San Francisco, CA. (2003)
8. Henttonen, K., Matinlassi, M., Niemela, E., Kanstren, T.: Integrability and Extensibility Evaluation in Software Architectural Models—A case study, *The Open Software Engineering Journal*, 1(1), 1-20 (2007)
9. Kelland, M.: From Game Mod to Low-Budget Film: The Evolution of Machinima, in H. Lowood and M. Nitsche (Eds.), *The Machinima Reader*, 23-36, MIT Press, Cambridge, MA (2011)
10. Kücklich, J.: Precarious playbour: Modders and the digital games industry, *Fiberculture*, Issue 5, (2005), http://journal.fibreculture.org/issue5/kucklich.html, accessed 13 April 2011.

11. Leveque, T., Estublier, J., Vega, G.: Extensibility and Modularity for Model-Driven Engineering Environments, *16th IEEE Conf. On Engineering Computer-Based Systems* (ECBS 2009), 305-314, (2009)

12. Lowood, H., Nitsche, M. (Eds.): *The Machinima Reader*, MIT Press, Cambridge, MA (2011)

13. Marino, P.: *3D Game-Based Filmmaking: The Art of Machinima*. Paraglyph Press, Scottsdale, AZ, (2004)

14. Narayanaswamy, K., Scacchi, W.: Maintaining Evolving Configurations of Large Software Systems, *IEEE Trans. Software Engineering*, SE-13(3), 324-334 (1987)

15. Parnas, D.L.: Designing Software for Ease of Extension and Contraction, *IEEE Trans. Software Engineering*, SE-5(2), 128-138 (1979)

16. Postigo, H.: Of mods and modders: Chasing down the value of fan–based digital game modifications, *Games and Culture*, 2(4), 300–313 (2007)

17. Postigo, H.: Video Game Appropriation through Modifications: Attitudes Concerning Intellectual Property among Modders and Fans. *Convergence*, 14(1), 59-74 (2008)

18. Scacchi, W.: Modeling, Integrating, and Enacting Complex Organizational Processes, in K. Carley, L. Gasser, and M. Prietula (Eds.), *Simulating Organizations: Computational Models of Institutions and Groups*, 153-168, MIT Press (1998)

19. Scacchi, W.: Understanding the Requirements for Developing Open Source Software, *IEE Proceedings—Software Engineering*, 149(1), 24-39, February 2002. Revised version in K, Lyytinen, P. Loucopoulos, J. Mylopoulos, and W. Robinson (Eds.), *Design Requirements Engineering: A Ten-Year Perspective*, LNBIP 14, Springer-Verlag, 467-494 (2009)

20. Scacchi, W.: Free/Open Source Software Development Practices in the Game Community, *IEEE Software*, 21(1), 59-67, January/February (2004)

21. Scacchi, W.: Free/Open Source Software Development: Recent Research Results and Emerging Opportunities, *Proc. European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Dubrovnik, Croatia, 459-468, September (2007)

22. Scacchi, W.: Game-Based Virtual Worlds as Decentralized Virtual Activity Systems, in W.S. Bainbridge (Ed.), Online Worlds: Convergence of the Real and the Virtual, Springer, New York, 225-236 (2010)

23. Sotamaa, O.: When the Game Is Not Enough: Motivations and Practices Among Computer Game Modding Culture, *Games and Culture*, 5(3), 239-255 (2010)

24. Taylor, T.L.: The Assemblage of Play, *Games and Culture*, 4(4), 331-339 (2009)

25. Wen, H.: Multi Theft Auto: Hacking Multi-Player Into Grand Theft Auto With Open Source, *OSDir*, 25 May (2005), http://osdir.com/Article4775.phtml. Also see http://www.mtavc.com/ and http://en.wikipedia.org/wiki/Multi_Theft_Auto. All accessed 1 June 2011.

26. Yee, N.: The Labor of Fun: How Video Games Blur the Boundaries of Work and Play, *Games and Culture*, 1(1), 68-71, 2006.