AUTHOR(S): HERON, M.J.

TITLE: Ethical and professional complications in the construction of multi-developer hobbyist games.

YEAR: 2016

Publisher citation: HERON, M.J. 2016. Ethical and professional complications in the construction of multi-developer hobbyist games. The computer games journal [online], 5(3), pages 115-129. Available from: https://doi.org/10.1007/s40869-016-0025-0.

OpenAIR citation: HERON, M.J. 2016. Ethical and professional complications in the construction of multi-developer hobbyist games. The computer games journal, 5(3), pages 115-129. Held on OpenAIR [online]. Available from: https://openair.rgu.ac.uk.

Comput Game J (2016) 5:115–129
DOI 10.1007/s40869-016-0025-0

CASE STUDY

Ethical and Professional Complications in the Construction of Multi-Developer Hobbyist Games

Michael James Heron[1]

Received: 13 May 2016 / Accepted: 15 August 2016 / Published online: 24 August 2016

Abstract The modern availability of powerful video game development frameworks has resulted in something of an indie renaissance. Within this blossoming of small scale development are numerous hobbyist developers who build games for their own inherent satisfaction rather than with the expectation of any financial remuneration. While many of these individuals labour in isolation, some have undertaken projects of sufficient scale and complexity to require the recruitment of multiple developers over long periods of time. The lack of direct payment for volunteered efforts in such environments creates numerous interpersonal issues that must be considered—these relate to aspects of intellectual property ownership and the treatment of developers, as well as to the expectations of those players who may invest their time in a hobbyist title. The nature of recruitment for such projects is such that formal experience in software development or games design cannot be assumed, and the management complexities this paradigm introduces are of sufficient complexity that few tools are available to ensure the coherent development of a game. This paper is a reflective analysis of the issues that emerged through the development of one such game. The observations contained within however are applicable to all multi-developer projects where financial compensation for contributions are likely to be non-existent.

Keywords Hobbyist development · Epitaph · Multi developer projects · Collaboration · Video games · Game development

✉ Michael James Heron
m.j.heron1@rgu.ac.uk

1 Robert Gordon University, Aberdeen AB10 7QB, Scotland, UK

1 Introduction

The easy availability of popular game development frameworks along with the rise of digital distribution platforms such as Steam, Xbox Live, the App Store and the Google Play store has resulted in a large expansion in the opportunities for indie and hobbyist game developers (Swain 2009; Martin and Deuze 2009; Guevara-Villalobos 2011; Lipkin 2012). Game development is now a hobby that is within the reach of individuals who may have previously been excluded as a consequence of the scale of the task and the difficulty of securing distribution opportunities. Not since perhaps the era of the bedroom programmer has the production and development of games been quite so accessible to those who have no vocational role in the game industry. Although many games developed by hobbyists are of a necessity restricted in terms of scope and complexity there are still titles that involve many people collaborating on code and content over long periods of time. It is within this context that this paper is situated.

Within this paper too will define hobbyist development as primarily differentiated from indie development by the trait that no member of the team has a full or even part-time vocational role in development—all members of a hobbyist development team are volunteers, and it is this aspect that makes the movement significant. Within this definition, all hobbyists games are indie games, but not all indie games are hobbyist games.

Despite this new renaissance of unremunerated game development, there has not been a corresponding increase in the awareness developers have with regards to the professional and ethical issues that may be associated with their work. The tools have gotten better, but they still work in the same human context. Within the realm of purely hobbyist development, where significant remuneration may not ever materialise, the task of building a volunteer team to collaboratively develop a substantial project introduces considerable interpersonal complexity. While this complexity is true of any team, within a hobbyist project the usual financial levers and incentives are not available to managers.

This social complexity in turn creates a potentially fraught situation—lacking financial remuneration for the effort others might invest in the project, issues of intellectual property and the future prospects of monetization can result in an ethical and legal tangle which few are equipped to unpick. Formal licences for developer contribution can help resolve these issues, but when individuals initially accrete organically and informally around a shared set of goals, such licences may be adopted too late, if ever. The nature of building a development team in hobbyist communities has its own complexities, and there may never be a ‘right’ time to insist on a firm legal and ethical foundation for contribution.

We discuss these issues in relation primarily to the development of the collaborative hobbyist game Epitaph Online (Heron 2015), a text-based MMO developed primarily, but not exclusively, by the author of this paper. We discuss the collaboration regime of the game; the decisions that set its technical framework in place; and the emergent issues of legality and ethics that evolved over its 4 years of closed development. We also discuss the ethical and professional issues that emerge from ‘releasing’ the game in its various open phases. The nature of hobbyist development largely precludes formalised closed testing and this has a corresponding impact on those who invest their recreational time in the game—bugs are inevitable, and the lack of a dedicated customer support team means that recompense, if available, may not be possible to apply consistently. Software development skills cannot be assumed of those who volunteer (Townsend and Heron 2013) and this in turn exacerbates issues of quality control.

While the issues discussed in this paper cannot ever be fully resolved within the limitations implied by hobbyist development, it is possible that awareness can reduce the likelihood that they will unexpectedly derail a promising project.

2 The Context of Contribution

The nature of hobbyist game development tends to better support solitary, rather than collaborative, development effort. Work must be accomplished during developer free time, which as a currency is often difficult to accumulate and to spend reliably. The highly asynchronous and seasonal nature of development does not permit for organisational traction to be easily generated, and the fitful nature of progress can have a significant dampening effect on momentum. Pragmatics too dictate that a hobbyist game should be, as far as is possible, something that can genuinely be completed and released—this tends to bias hobbyist development away from large, complex games that require multiple developers and towards smaller, more abstract games that focus on mechanics, or procedural generation rather than large amounts of specifically tailored content.

However, neither of these are iron-clad observations without exception. Substantial numbers of hobbyist games do end up being developed either in partnership with another interested individuals or as the result of a consortium of developers who are inspired enough to volunteer their effort without any likely financial reward. The Multiuser Dungeon (MUD) scene which was most vibrant in the early nineties (c.f. Young 1998; Dourish 1998; Heron 2013) is an area where the popularity of the games led to a constant stream of new developers—largely recruited from interested players—joining the creator staff and developing new, on-going content. However, again the nature of voluntary collaboration complicates this—turnover is rapid, and even when an individual volunteers they may end up costing more time in training than they save in development.

In 1998, the first author of this paper started playing one such game—Discworld MUD (Karlsen 2008). After a year of dedicated participation in the player community, which included membership of emergent social structures and de-facto player government structures, he volunteered his services as a developer—or in the parlance of the game, a creator. No financial reward was associated with the position of being a creator, and it conferred no in-game advantages upon player characters. Attempting to influence the game was in fact an offence for which a creator could be dismissed and all impacted players deleted. Such dismissals occurred with some regularity, and the owners of the game put many systems in place to identify creator misconduct and allow for it to be effectively dealt with. After a lengthy tenure as a developer on Discworld MUD, which eventually led to his being promoted to the highest creator rank of Admin, this author resigned his position to begin development of his own game—Epitaph Online.

Contribution as a creator in hobbyist environments is driven primarily by a sense of intrinsic motivation—the desire to contribute to a game and have those contributions experienced and appreciated by other people. In many respects, the motivations for developers in these environments can be understood in the same terms as those contributing to open source projects. Contribution offers opportunities for the perfecting of expertise (Moody 2002; Raymond 2001; Lakhani and Wolf 2003; Von Krogh 2003); the enhancing of personal reputation (Bezroukov 1999; Lakhani and Wolf 2003; Raymond 2001; Von Krogh 2003); and for the inherent fun of building game content (Von Krogh 2003; Lakhani and Wolf 2003; Moglen 1999). Others are motivated by a belief in the game and its community (Hertel et al. 2003; Raymond 2001). Participation in the game development also has the potential to yield real world benefits, with several creators having explicitly acknowledged the role that their contribution had in securing real world employment.

Many too become creators to address what they perceive as fundamental problems with the game they have played such as balance issues or missing functionality. However, the freedom that is required to make sweeping changes to game content is not usually given to new creators. The expectation is that contributors will 'make their bones' by proving themselves trustworthy through sustained, high-quality contribution. Once a track record of authorial leadership (Reagle 2007) has been established, they are usually permitted to agitate for their own personal pet projects and given the freedom to develop them, within constraints.

The key element of this is that individuals contribute for their own personal reasons—they are not, in the main, selling their skills. As such, the relationship they form with the code or content that they produce is not simply that of a disinterested producer creating 'work for hire'—it is an on-going and personal connection. It is also, unless formal contribution criteria are put in place, code that belongs legally and ethically to the person that produced it and not the game for which it was developed. However, even making that assumption about the ownership rights that the game may possess over contributions is additionally frustrated by the widely divergent geographical and legal jurisdiction within which contributors are located.

3 Project Management in Hobbyist Environments

Within environments such as these it is not strictly speaking possible to employ any standard project management techniques. The various tools and levers available to a manager are largely absent. The only real punishment with any force available beyond social approbation is the ultimate sanction of removing a developer from the team. Such an action generates drama, creates resentment, and is ultimately self-defeating—it deprives the project of someone who was, at least at one point, an advocate for its success. There are few sticks available within a totally volunteer environment.

Carrots, where they are available, tend to focus on intrinsic reward—recognising accomplishment with ever-grander developer titles (such as senior creator) and through permitting more freedom to self-direct developments. Developers may also be encouraged to take on more visible leadership roles, allowing for a higher degree of recognition and a greater ability to shape the development of the game over the long term (Reagle 2007). In a large sense, the reward for significant contribution to the game is the ability to make further significant contributions.

The overall impact of this is that those who are responsible for managing a project's success over the long term must place an unusually large amount of trust in those who have volunteered their time—this is true whether that is a natural inclination on the part of the manager or otherwise. Intrinsic motivation can be fragile (Frey and Jegan 2000) and easily compromised (Schulze and Frank 2003; Bolle and Otto 2010). The role of management then becomes identifying those areas in which someone is most interested in working and creating opportunities there. A competent manager then must work with the developer to collaboratively identify a project that is both attractive to them and valuable for the game. Finally, the manager must make available the resources and remove the obstacles that would stand in the way of its successful completion. Beyond that, a manager's ability to influence the project or the developers that work on it is limited.

Real life obligations frustrate any attempt at coherent scheduling. The inherent ebb and flow of personal motivation along with what may be a complicated geographical dispersal mean that face-time cannot be guaranteed. The presence of both parties in a manager-managed dyad may be erratic on a day by day or week by week basis. It is perfectly possible that the time frames and availability of both may never sync up. Planning and organisation too are made difficult by the fact that much of the decision making must be devolved to those undertaking the work requiring a certain amount of management by motivation (Frey 2002; Roberts et al. 2006). Intrinsic motivation in these environments is bound up in large part with autonomy to shape how a project develops, and this rarely survives being given a fixed specification and being told 'implement that'. A sense of ownership over a project's evolution is required to ensure that the work remains sufficiently interesting to warrant a developer's on-going commitment, however this can fracture a game's thematic consistency and create difficulties in ensuring game balance.

This in turn leads to complications working across a large project—the more autonomy is given, the more inconsistent and incompatible projects will be unless collaboration to a common end is somehow enforced or encouraged. Within Discworld MUD for example there are several implementations of a basic 'faction' system including the crime and legal systems in the city of Genua, the family system within the Counterweight Continent and the City Watch system within Ankh-Morpork. Lacking a common architecture upon which these systems could be built, they all have their own infrastructure and do not integrate with each other. Similarly, the various crafting systems that have been implemented do not represent any coherent or cohesive workflow—the outputs of one part rarely feed as inputs into another no matter how logical that feed-through might be. The systems simply are not designed to work together. They behave differently in equivalent circumstances. Since they share no code, when a new piece of functionality is added to one system it does not rattle through to the others. Each generates its own bugs, and each bug must be dealt with in that specific system—fixes in one are not reflected as fixes in the others. This is all as a result of too much developer autonomy being ceded in an environment where formal expertise in software development is rare. This is to the cost of the development of the game itself. Management then becomes a balancing act between ensuring intrinsic motivation can be harnessed whilst also being respectful of the context and pragmatics of software development (Townsend and Heron 2013).

For the most part, simply providing the right tools in the game engine is enough to resolve these issues, and this is the path that was taken for Epitaph. Epitaph has more factional representation than Discworld, but all factions operate from a common core which ensures that when new functionality is added, or bugs are fixed, it is an improvement for all represented factions. Similarly, all the various flavours of crafting within Epitaph are handled by a common core of code. However, these specific examples are obvious only with the benefit of hindsight—the future problems of code divergence on Epitaph were dealt with early because of experience gained within Discworld MUD. They are also, importantly, things that were put in place before other developers were brought on board so served to anchor expectations as to how game systems were developed. Resolving these problems in the early stages of a project requires considerable foresight or a large body of generalizable experience. It also requires a developer that can invest the time into building an architecture that will work in a multi-developer environment, and doing so at a time when multiple developers may be a distant, even unlikely, dream.

Part of the role of management in environments like this too must then be as an educator—helping those new to the game architecture to learn how things are done, either through the provision of formal training material (c.f. Heron 2010) or through helping locate relevant exemplar assets within the game itself. The nature of the environment requires a peculiar blend of skills that other, more traditional, projects do not necessarily need to emphasise. Discipline is vital in ensuring the effective development of any significant piece of software, but it is precisely this aspect that hobbyist development is least able to leverage.

The nature of collaboration in environments like Epitaph tends, much like the hobbyist development process itself, to progress organically and in fits and starts. Generally speaking, projects are an emergent property of largely unstructured social interactions. Developers discuss game-related topics through whatever social tools are provided. These discussions will lead to further discussions about what can be done to improve the game relative to the topic that has emerged. These discussions will then either be forgotten about when the next topic arises, or enthuse participants sufficiently that they look to collaborate to bring it to fruition. Townsend and Heron (2013) discuss this issues with regards to identifying anything as solid as 'authorship' with regards to these nebulous and organic relationships—essentially, the tightly interrelated nature of software development makes such terms unhelpful for understanding what any individual may contribute to the overall project. This in turns adds complexity to identifying legal authorship or ownership of contributions.

Collaboration then is a product of the social environment, and similarly difficult to direct from the top-down. A will to collaborate must exist or any participation in such activity will be half-hearted at best (Zeiller and Schauer 2011). Given the difficulties though of directing development, collaboration adds a new and challenging dimension to the task of shepherding a hobbyist game to completion. Collaboration may enthuse development in a direction that, while exciting, only exacerbates issues of feature creep (Levesque 2005; Senyard and Michlmayr 2004) while directing effort away from core deliverables. Such ad hoc projects are usually ancillary as far as the game goes—they might eventually end up being core features, but they are rarely in their initial conception so important to the game that their development must be instantly begun—if they were, they'd already be part of the emerging infrastructure of the game itself.

4 Project Scoping and Feature Creep

Runaway Projects (Keil et al. 2000) are often the result of short-lived enthusiasms for a particular development—when the originators are no longer available, or are distracted with other tasks, new developers may be assigned to the project in what becomes an effective and on-going demonstration of the destructive power of the sunk cost effect (Keil et al. 1995). The projects are, in and of themselves, sufficiently interesting to get new developers excited about taking them on—that is part of the problem. If fully developed, the projects may well be as impressive and satisfying as everyone has planned—but these projects will never be completed in their existing form. These new developers, when assigned to the project, spark off their own enthusiastic extensions because few in these environments want to simply implement someone else's creative idea. The project then becomes ever more epic in scope and potential. However, faced with the difficulty of then bringing the original plans plus the new extensions to life, the developer sooner or later loses motivation in the face of the immensity of the task and drifts away. All that is left behind are the plans that they made, and the code representing what relatively meagre amount of progress they were able to achieve.

The cycle then repeats, until a project becomes so mammoth and impossible that all it serves to do is sap away new talent that might have been able to thrive and build confidence with a more manageable project brief. Few new developers have the self-confidence to say 'This project is way too big, let's do a smaller version of it' because that is simultaneously a critique of what may have been years of developer effort, and also an acknowledgement that they are not up to the task as it is presented. That the task presented may be a poisoned chalice is immaterial in this respect. That the project exists and has been started creates a kind of psychological requirement for it to be completed a la the Zeigarnrik Effect (Zeigarnrik 1938; Norton et al. 2011). They are rarely abandoned except grudgingly and when the attrition cost associated with losing promising new developers becomes too high to justify.

These issues emerge when the management of a game cannot simply set the parameters of contribution. In an environment where people are paid for what they do, the act of assigning projects and ensuring their timely completion can be viewed as a managerial transaction. In multi-developer, hobbyist environments it's more about curating an on-going relationship in which those that are being managed perversely have all the genuine power.

Given the social context of this collaboration, a second important issue is raised—that which has been termed 'design by committee' (Henning 2006) as a consequence of a kind of 'concurrent engineering' (Maier 2009, p 3). Much of the social interaction in the development of a game is about the game itself—what would be fun, what is not fun, and what balance issues need to be addressed. The low cost nature of suggestion means that there are lots of ideas and many of these will be high quality and highly relevant. The more people involved in the conversation though, the more likely that discussions will tend incorporate issues of the 'long tail'. Ideas will grow arms and legs, possible abuses will be highlighted and countermeasures designed in. A brainstorming session about a reasonably well constrained scenario will become increasingly blue-sky the longer the discussion persists. An exchange about a relatively obscure scenario in which a particular system has an unusual quirk will eventually, as time goes by, result in someone saying 'Basically we need to rewrite this system from the ground up and add in all these new capabilities'. From a purely theoretical perspective, it's probably even true—working within the context of a budget and deadline, such discussions can be more effectively re-contextualised and limited to what is manageable. Hobbyist developments rarely have the same pressures.

However, when it comes time to implement any of these ideas actual contribution beyond the conceptual is far more difficult to generate. Developer time and enthusiasm are scare resources and must be spent wisely. When the committee design stage has fizzled out, the person that has been tasked with implementing the project finds they are now faced with a largely impossible task because of the well-meaning additions that were suggested during an enthusiastic discussion. What may have been a simple change with a well-defined scope becomes a massive project that will yield, in the end, only incremental benefits over what is already in place. We have a natural tendency as human beings to spend other people's time more freely than we spend our own—when those making suggestions have no 'skin in the game' when it comes to implementation, it is only natural that their reach will exceed the actual developer's grasp.

It is clear then that there are multiple layers of what might define contribution in multi-developer environments, and that the ability of individuals to at least partially self-select their own contribution is mandated by the voluntary nature of the system. At one level is the provision of general ideas and suggestions that others will then take on to develop. At the other is the development of fully featured systems and frameworks that will make their way into the game proper. Given the environment, it can be hard to separate out the contributions that an individual has made in terms of their relative value or importance. A good idea presented in the right way may be much more important to a project's eventual success or popularity than weeks of coded effort that duplicates existing but non-generic functionality. While generally speaking it is usually argued that ideas are ten-a-penny (Elaine 2012) and all that matters is the execution (West 2002), the situation is somewhat more nuanced within game development—the right change to the right game system may yield great player satisfaction even if it only ends up being a few lines of code. Ten diligent hours of play testing might result in a value in one system changing from a ten to a nine, with tremendous impact on playability. We cannot exclude the evaluative element of player appreciation from our definitions of contribution, and this in turn creates some of the ethical and legal complexity which we will address.

5 Dealing with Intellectual Property

What we are describing is a complex and socially fragile environment in which to develop software, and everyone involved is generally doing so out of a sense of attachment to the game itself, or the creative outlet it provides (Postigo 2007). However, the loose nature of collaboration and contribution creates a very real difficulty when it comes time to assess rights over intellectual property, and a game is little more than an encapsulation of intellectual property within an externally compelling package.

As a matter of de-facto policy, those charged with the day to day operating and administration of the game require the ability to authorise access to code or to the executed representation that the code presents to players. However, the authority for this is largely simply assumed. The expectation is that the very low stakes involved would disincentive legal challenges in the case that it was seriously contested by another party. Given how these games are operated, usually, for no profit and with no formally valued assets, there is little hope for someone seeking financial compensation for the unlicensed use of their work—enforcing intellectual property rights, after all, is largely the province of the wealthy. Lacking the credible risk of costly legal action, administrators are largely guided instead by their own sense of ethical responsibility to those who have contributed. This in turn is inverted in their dealings with other individuals—in the event that the game's intellectual property is violated, there are few credible opportunities for seeking recompense or restriction. Several times during the life of Discworld MUD, the entirety of the game's gigabytes of code were downloaded by dissatisfied creators and used to start unauthorized 'forks' of the game. Contribution to hobbyist environments requires at least some degree of access to either the underlying code or closed tools that generate content—within Discworld and Epitaph, coders gain increasing access to the code as they prove their ability to contribute. Hobbyist environments are, in large part, trust economies—access to game systems is something that is earned via showing that a developer can be trusted with that access. When that trust is violated the consequences for the integrity and originality of a game system can be significant.

Many environments lack a formal policy on the ownership of contribution. At best there is an informal policy which states generally 'The code that you write here belongs to you, but you grant us the right to use it as we wish'. This informal policy serves reasonably well for most environments, and acts as the foundation tenet of a kind of 'ethical ownership' of the game code. However, it is not a legal ownership in any real sense and its lack of formality means that the arrangement can be revoked at any time by the person who contributed code. Given the tangled nature of what contribution means in these environments, this can be a difficult demand to meet unless the code is easily identified as belonging to a single person, and not too tightly integrated into other game systems.

As part of the normal course of developing a complex game engine, administration teams often make available subsets of the engine for others who want to build their own similar games. These offerings usually contain only structural elements, and rarely if ever include game content beyond that required to demonstrate how the various systems can be used. These are sometimes known as mudlibs or codebases. Discworld MUD has released a number of these over the years, and a variant form of the Discworld mudlib was used to build Epitaph. Epitaph in turn formally forked from the Discworld mudlib in 2012, naming its derivative branch the Epiphany lib. Each version serves to create a basis upon which others can create their own games by using the various building blocks of the engine. They also permit for expanding and changing the game systems to meet a new developer's particular preferences. Other mudlibs and codebases exist, each with their own strengths and weaknesses. The Discworld mudlib is released with the following licence attached (spelling errors preserved as they are in the README file):

> Discworld mudlib distribution - version 20.0
> This is the Discworld mudlib distribution, it is distributed as a publically usable mudlib based off the code running at Discworld mud. There is no warenty for it's use and we cannot be held responsible for anything you do with the code. This code is required to be used in a free way, no commercial use may be made of any of the code contained in this distribution.
> The Discworld administration team (admin@discworld.imaginary.com): Pinkfish, Turrican, Ceres, Sojan and Hobbes

Two features are especially notable about this licence. The first is that it is not a legal licence, and isn't a variation of any of the commonly used open source licences that serve as a baseline for distribution of modern software. This is, in part, a legacy from the original release of the Discworld mudlib which came at a time which issues of intellectual property ownership in software development were not so high profile or so high impact.

The second thing is that none of the administration team have a firmly established legal right to issue any licence at all. There is no formal transfer of authority from the developer to the administration team, and even if there were that team has undergone many changes from the first years of the game—of the administration team named above, currently at the time of writing only one remains in an active administration role within the game. As such, while there may be an ethical or moral case of ownership that can be exerted by the administration team, it has little or no legal weight behind it. The nature of hobbyist development is such that opportunities for commercialising such a game are limited to non-existent, and so there are few opportunities to violate the licence in any case. However, in the event such a breach occurred there is no mandate for the administration team to negotiate on behalf of all the coders who have contributed to the mudlib over the game's 20+ years of operation—for Discworld that number is a surprisingly large 1000+ developers. The effort they may have invested into the game is also not possible to assess in the aggregate—as in most such projects, there is a very long tail of contribution (Heron et al. 2013).

Strikingly, in this scenario it is those who develop the code, rather than those that exert moral authority, that have all the power. There is an asymmetrical relationship where the administration team can be legally denied (in theory, if not in credible practise) the right to use any code that has been submitted to the game. They in turn cannot deny permission of anyone else unless individual coders permit them a suitable licence. They have only the legal rights that their individual contributions grant them. Strictly speaking, it is likely not legally possible for the administration team of Discworld to release a mudlib at all, largely because there is no formal policy of ownership. It may be a largely academic issue, given how challenges to the admin's team authority to distribute the code of their developers are rare, but in higher-stake environments where real money might be involved it would be an important and problematic situation. This has relevance to Epitaph and all games built using a Discworld mudlib, they are built from a mudlib that the Discworld MUD administration team likely had no legal authority to release. Given the small stakes, everyone involved simply pretends the issues do not exist. This is a luxury that not all hobbyist games would be able to enjoy.

To avoid these problems for Epitaph's future, early on in its development a code ownership policy was put in place to define on what basis contribution was made to the game. It enshrined the right of creators to their own intellectual property and to use it in whatever context they wished outside Epitaph, but it also stressed that code submitted was irrevocably donated as a 'fork' to the game. The full policy can be seen at http://drakkos.co.uk/help/tutorials/creator/code_ownership.html—it is not a binding document, written as it was without the assistance of any legal representation, but it serves as the basis for formalising the agreement that all creators make upon being employed. Formal legal representation, given the lack of a budget to support hobbyist development, is not lightly retained.

To avoid the issue of impermanence of the administration staff roster, a company was set up to act as an overarching owner for Epitaph—this in turn ensures that there is a continuation of ownership should anything happen to the original owner of the game. This company, Imaginary Realities, is not intended to generate a profit—merely ro act as a resolution for issues of ownership. Developers then agree to provide Imaginary Realities with an irrevocable right to use, modify and distribute the code they submit under any licence that the administration team consider appropriate. A restriction is placed in the policy though that this will never be for commercial purposes—while the legal validity of the Discworld licence may be questionable, the Epitaph licence still honours its ethical validity.

In the situation for Epitaph Online, it is further complicated by the fact that the mudlib runs on a second application—a driver called FluffOS. This in turn comes with its own bespoke licence. This in itself is a fork of an earlier application called MudOS, which was dervived from LPMud:

> This game, LPmud, is copyright by Lars Pensjl, 1990, 1991.
> Source code herein refers to the source code, and any executables created from the same source code.
> All rights reserved. Permission is granted to extend and modify the source code provided subject to the restriction that the source code may not be used in any way whatsoever for monetary gain.
> ******
> The name MudOS is copyright 1991–1992 by Erik Kay, Adam Beeman, Stephan Iannce and John Garnett. LPmud copyright restrictions still apply.
> In addition, the entire package is copyright 1995 by Tim Hollebeek.
> ******
> FluffOS is a range of patches to MudOS distributed as the patched source tree for convenience. LPmud and MudOS copyright restrictions still apply.

Both FluffOS and its earlier ancestor MudOS come with a similar issue to the Discworld lib itself, although one that is perhaps more tractable given how the number of contributors over the years is considerably smaller. However, both licences prohibit the use of the code for commercial purposes, and this is not an unusual restriction amongst such games. The popular DIKU codebase for example has been the subject of considerable controversy over the years regarding whether or not derivatives have been changed 'sufficiently' to free developers of its licence, which states, among other things:

You may under no circumstances make profit on *ANY* part of DikuMud in any possible way. You may under no circumstances charge money for distributing any part of dikumud—this includes the usual $5 charge for "sending the disk" or "just for the disk" etc. By breaking these rules you violate the agreement between us and the University, and hence will be sued.

Leaving aside the unlikely outcome that anyone will attempt to sue over a breach of the DIKU licence, it itself exhibits a confused grasp of who is responsible for violating an agreement—the agreement between the creators and the University is not legally binding in itself on those who use the software. Such legally incoherent licences are a feature of the time period in which the code was developed—standard templates such as the GPL or the various Creative Commons licences were either not available, or sufficiently obscure that knowledge of their existence cannot have been assumed. The licence here then represents an agreement unlikely to stand up to much dedicated scrutiny in court, but again exerts an ethical prohibition against profiteering from the work of others.

There is little then to credibly stop someone who wishes to make a profit from their game from doing so even when the mudlib or the codebase they use insists on non-commercialisation. Lacking much in the way of legal architecture or precedent, the community has instead solidified around a culture of vocal condemnation for those that violate licences. Zen (2003) discusses one particular case—that of the mud called Medievia:

> It is important to understand from the license that it was never meant to be a legal barrier for developers. The lack of legal counsel in writing the licensing agreement is clear when examining the misspellings, non-legal terminology,

> and loopholes in the agreement. Thus, the licensing agreement was really a request from the DIKU authors to acknowledge their efforts if other developers used the DIKU code.

The reaction of the community is similarly discussed:

> Consequently, despite the large amount of support shown for Medievia, backlash from the developers within the MUD community was harsh. Many crusaded against Medievia by trying to inform gamers about the illegality of the Medievia or, in the case of the cracker who stole Medievia code, directly attack Medievia. Even some Medievia players decided to stop playing because of the ethics involved while others felt that because of the DIKU license issue, the game had stopped focusing on its players and had declined in recent years. Two of Medievia's coders, Cestus (Kurt Schwind) and Thranz (Keith Hudson) resigned from Medievia in 2000 due to ethical objections to the game code. Additionally, the same year, two of the community's major resources on the Web, Top Mud Sites and the MudCenter, removed Medievia from their MUD listings. Clearly, many in the development community and those in charge of resource centers believed that Medievia benefited greatly from the DIKU code, and they felt that Medievia was illegally generating revenues. By omitting Medievia from their databases, Top Mud Sites and MudCenter greatly altered the geography of the MUD community. As major resources within the community, this omission is a signal to others that to the people in the MUD community, Medievia should not exist.

The effectiveness of such reactions are questionable, given how Medievia survives to this day while other less morally dubious games have withered away. However, what is shown in this short discussion is the extent to which the community itself will act to police what it sees as licence violations. Such controversies rarely make themselves known to players however, as developers tend to congregate on sites where those individuals are correspondingly rare.

6 Conclusion

In this paper we have addressed some of the legal and ethical issues that developers of hobbyist games will face, especially in multi-developer environments. We've done this primarily with reference to the text game Epitaph Online and its ancestor Discworld MUD, as this is where the largest bulk of these issues have been encountered. While Epitaph Online is a game that has a very specific style and niche, the lessons are relevant to all titles and genres.

It's incumbent on those responsible for administering the evolution of a game that they understand the implications of intellectual property obligations. When working just within a small group of friends, concepts like code ownership policies may seem abstract and potentially divisive. However, the long term implications of not agreeing upon these issues early are considerable—it's entirely possible that someone can lose all legal right to do anything at all with their game by not being mindful of the consequences.

Similarly, outside of the legal context of licensing there is an ethical and professional context that revolves around being a productive member of a larger community. Even in cases where the legal sanctions that might be applied are unlikely to have much weight, community responses can be striking in their righteous indignation. There are already many more games in the world than any one person can possibly play, and the last thing a developer needs is a whole community of otherwise like-minded souls actively briefing against their offering.

It is incumbent then on those responsible for the management of mulit-developer, hobbyist environments to proactively consider the implications of intellectual property before they become an issue. This may often seem like putting the cart before the horse, worrying about the legal ownership of a product that does not exist in any meaningful form. As we have seen in this paper though, leaving these considerations aside until there is a concrete need has the inevitable consequence of rendering future negotiations all but impossible to fairly conclude.

References

Bezroukov, N. (1999). Open source software development as a special type of academic research: Critique of vulgar Raymondism. *First Monday*, *4*(10). doi:10.5210/fm.v4i10.696.

Bolle, F., & Otto, P. E. (2010). A price is a signal: On intrinsic motivation, crowding-out, and crowding-in. *Kyklos, 63*(1), 9–22.

Dourish, P. (1998). Introduction: The state of play. *Computer Supported Cooperative Work (CSCW), 7*(1), 1–7.

Elaine, J. (2012). Ideas are easy. *Storyline, 32*, 67.

Frey, B. S. (2002). "How does pay influence motivation?" In *Successful management by motivation* (pp. 55–88). Berlin, Heidelberg: Springer.

Frey, B. S., & Jegen, R. (2000). Motivation crowding theory: A survey of empirical evidence. *Journal of Economic Surveys, 15*(5), 589–611.

Guevara-Villalobos, O. (2011). "Cultures of independent game production: Examining the relationship between community and labour." In Proceedings of DiGRA 2011 Conference: Think design play.

Henning, M. (2006). "The rise and fall of CORBA". *Queue, 4*(5), 28–34.

Heron, M. J. (2010). "The epitaph survival guide". Retrieved June 10 2013. Available Online at http://www.scribd.com/doc/135219005/Epitaph-Survival-Guide.

Heron, M. J. (2013). "Likely to be eaten by a Grue"—the relevance of text games in the modern era. *Computer Games Journal, 2*(1), 55–67.

Heron, M. J. (2015). "A case study into the accessibility of text-parser based interaction." In Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (pp. 74–83). ACM.

Heron, M., Hanson, V. L., & Ricketts, I. (2013). "Open source and accessibility: Advantages and limitations". *Journal of Interaction Science, 1*(1), 1–10.

Hertel, G., Niedner, S., & Herrmann, S. (2003). "Motivation of software developers in open source projects: An internet-based survey of contributors to the linux kernel". *Research Policy, 32*(7), 1159–1177.

Karlsen, F. (2008). “Quests in context: A comparative analysis of discworld and world of warcraft”. *Game Studies, 8*(1), 1–18.

Keil, M., Mann, J., & Rai, A. (2000). Why software projects escalate: An empirical analysis and test of four theoretical models. *Mis Quarterly*, *24*(4), 631–664.

Keil, M., Truex, D. P., & Mixon, R. (1995). “The effects of sunk cost and project completion on information technology project escalation”. *IEEE Transactions on Engineering Management, 42*(4), 372–381.

Lakhani, K. R., & Wolf, R. G. (2003). Why hackers do what they do: Understanding motivation effort in free/open source software projects. Working Paper 4425-03, Sloan School of Management, MIT, Cambridge, MA.

Levesque, M. (2005). “Fundamental issues with open source software development (originally published in Volume 9, Number 4, April 2004).” First Monday.

Lipkin, N. (2012). Examining Indie’s independence: The meaning of “Indie” games, the politics of production, and mainstream cooptation. *Loading*, *7*(11), 8–24.

Maier, M. W. (2009). *The art of systems architecting*. Boca Raton: CRC Press.

Martin, C. B., & Deuze, M. (2009). The independent production of culture: A digital games case study. *Games and culture, 4*(3), 276–295.

Moglen, E. (1999). Anarchism triumphant: Free software and the death of copyright. *First Monday*, *4*(8). doi:10.5210/fm.v4i8.684.

Moody, G. (2002). *Rebel code: The inside story of Linux and the open source revolution*. New York City: Basic Books.

Norton, M. I., Mochon, D. & Ariely, D. (2011). “The ‘IKEA effect’: When labor leads to love”. Harvard Business School Marketing Unit Working Paper 11-091.

Postigo, Hector. (2007). Of mods and modders chasing down the value of fan-based digital game modifications. *Games and Culture, 2*(4), 300–313.

Raymond, E. S. (2001). *The cathedral and the bazaar: Musings on linux and open source by an accidental revolutionary*. Sebastopol: O’Reilly Media Inc.

Reagle, J. M. Jr (2007). “Do as I do: Authorial leadership in wikipedia.” In Proceedings of the 2007 international symposium on Wikis (pp. 143–156). ACM.

Roberts, Jeffrey A., Hann, Il-Horn, & Slaughter, Sandra A. (2006). Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the Apache projects. *Management Science, 52*(7), 984–999.

Schulze, G. G., & Frank, B. (2003). Deterrence versus intrinsic motivation: Experimental evidence on the determinants of corruptibility. *Economics of governance, 4*(2), 143–160.

Senyard, A., & Michlmayr, M. (2004). “How to have a successful free software project.” In IEEE Software Engineering Conference. 11th Asia-Pacific (pp. 84–91).

Swain, C. (2009). Who needs a publisher… or a retailer or a marketer? *Computer, 42*(2), 103–105.

Townsend, J., & Heron, M. J. (2013). Authorship and Auteurship in the collaborative development process of text-based games. In *Chercher le text: Locating the text in electronic literature conference, Paris, France*.

Von Krogh, G. (2003). Open-source software development. *MIT Sloan Management Review, 44*(3), 14–18.

West, Michael A. (2002). Ideas are ten a penny: It’s team implementation not idea generation that counts. *Applied Psychology, 51*(3), 411–424.

Young, Kimberly S. (1998). Internet addiction: The emergence of a new clinical disorder. *Cyber Psychology & Behavior, 1*(3), 237–244.

Zeigarnik, B. (1938). On finished and unfinished tasks. *A source book of Gestalt psychology*, *1*, 300–314.

Zeiller, M., & Schauer, B. (2011). Adoption, motivation and success factors of social media for team collaboration in SMEs. In Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies (p. 4). ACM.

Zen, L. (2003). The impacts of medievia and medthievia. Available Online at http://www.stanford.edu/group/htgg/cgi-bin/drupal/sites/default/files2/lmzen_2003_1.pdf.