# Software Product Management for Mobile Game Development

David Callele[1] and Krzysztof Wnuk[2]

[1] Department of Computer Science, University of Saskatchewan
Saskatoon, Saskatchewan, Canada
callele@cs.usask.ca
[2] Department of Computer Science, Lund University
Lund, Sweden
Krzysztof.Wnuk@cs.lth.se

**Abstract.** This work describes a case study wherein observations of a greenfield mobile game software product line development effort were gathered to help identify directions for future work. Business and technology requirements were identified by the development team and the consequences of adopting third-party middleware as the basis for a software product line were explored, identifying management of development efforts and product differentiation as areas for future work. The effects of adopting commercially available middleware upon the requirements for mobile game SPLs are discussed and the need for requirements engineering to focus upon the customer experience in this context is identified.
Keywords: Software product line, requirements, mobile games, software product management

## 1 Introduction

Commercial software developer interest in the smartphone application market is significant. The availability of application stores (*a.k.a.* app stores) for product distribution has bypassed much of the traditional retail channel (with the associated negative effects upon the viability of the retail channel). This alternative market access has removed many barriers to new product entry and has quickly been exploited by mobile game developers – it is widely reported that games are the most commonly downloaded smartphone apps.

Customers have embraced the app store model and the variety of purchase options offered. Potential customers learn of the game (in some way), connect to the appropriate server (application store) then download (either for free or for a fee) and install the game. Some games require the payment of a fee before the customer can download the game. Other games adopt a ”try before you buy” format; the customer can start playing for free but after an initial demo they have to buy the game to continue to play. To generate revenue, the game producer must convert the customer during the critical free-play period. Some games adopt the “freemium” model where continued gameplay is free but is greatly facilitated (*e.g.* progress is accelerated) if the customer makes in-game purchases of gameplay related items.

The transactional price of a typical game is now ingrained at $0.99, some fraction of the price of a single cup of coffee in North America – in the same way as a cup of coffee is a discretionary purchase of a mass-market consumable item, so is the purchase of a game. Average revenues of approximately $1,000 per app are reported by Vision Mobile (*Mobile Developer Economics* available at http://developereconomics.com) – far less than that necessary for commercial viability. Mobile game development is, therefore, a financial gamble.

Attempts to mitigate the risk posed by low revenues are likely to put pressure upon development budgets and development timelines. Software product management [6] will be under pressure to minimize investments in process, potentially promoting process evolution in an ever more lightweight direction, placing significant challenges upon the process of defining and scoping mobile games.

We report herein upon the results of a case study [5] directed toward gathering observations of mobile game software development (with a modest emphasis on requirements issues) to more clearly identify directions for future work. The first author, a requirements engineering practitioner and software developer, was embedded within the development team. The second author acted as an independent reviewer of observations and interpreted the results in the context of software product management.

## 2 Related Work

Investigations into the role of requirements in general videogame development [2] have noted that the traditional output of the preproduction process in game development (the Game Design Document (GDD)) was often misused as a (weak) requirements specification. Furtado *et al.* [8] apply Domain Analysis to digital game Software Product Line (SPL) development, advocating the creation of a "game design vision" that guides the Domain Analysis. They explicitly identify the elements that will, and will not, be part of the SPL, drawing these elements from the Game Design Document or its equivalent – noting that traditional Requirements Engineering cannot be applied *as is* to game development [8]. Finally, Furtado *et al.* [8] recognize, as we did [4], that software product management must consider the experience aspects of game requirements.

In complementary work, Blow noted that game development [1], requires an unusually large breadth of engineering knowledge to define functional requirements. However, the technical difficulties stressed by Blow in 2004 were later addressed by experience gained in the development of very-large scale games. On a smaller scale, mobile game development requires meeting the challenges identified by Blow *plus* meeting the market constraints identified in the introduction to this work.

Folmer [7] investigated and reported on the potential for component based game development (*e.g.* rendering engines) as a means of addressing rising development costs. Folmer derived a reference architecture for the game domain, but this work was left behind by market forces with the advent of multi-platform game engine middleware (although they do make note of game engines in their concluding statements, stating that they could be potential "domain frameworks"). Finally, agile development methodologies are often suggested when faced with tight time-to-market constraints. Kanode and Haddad [9] advocate the use of agile methods during preproduction to help en-

sure that the resulting gameplay experience is as intended. However, they report that a thorough preproduction process (the game development analog to a more-than-cursory requirements process) appears to negate expected benefits from the use of agile development methodologies.

## 3 The Case Scenario

This work is based upon practical experience gained and observations made [5] during a mobile game Software Product Management development effort. The development team had five members in addition to the first author, three of which had significant game production and development experience. The long-term business goal for the SPM development effort targets the following relative expenditures for elements of the game development effort: (1) increase the budget for gameplay design from 10% to 20%, (2) reduce the software development budget from approximately 50% to approximately 10% by utilizing SPL techniques for the Core, (3) increase the maximum budget for scripting of media assets from 10% to 40%, and (4) increase the budget for media asset development from 50% to 70%. The team began by identifying the set of business and technology requirements for their effort.

### 3.1 Business Requirements

The development team modeled the business requirements as shown in Figure 1. In the short-term, the development team is focused upon the delivery of a particular game instance. In the long-term, the development team is interested in maintaining and enhancing the SPL assets for the organization: both the software assets (the Core) and the media assets. The three management stakeholders perspectives shown (preproduction, production and corporate) were investigated by the team members using role-playing techniques, drawing upon their personal experience in these roles.

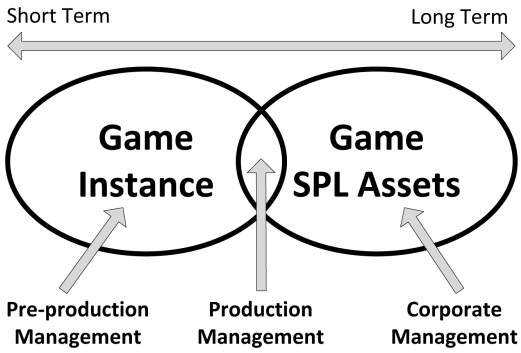


**Fig. 1.** Stakeholder requirements focus

Preproduction management is focused upon the requirements for the specific instance of the game currently under development. They are concerned with defining the game experience (without concern for implementation details) and tend to concentrate upon requirements for the media assets and how they contribute to the look and feel of the game.

Production management is concerned with the current game (the short-term tactics of delivery) *and* with the contribution the game can make to the SPL asset base (maintaining the long-term strategy of media and Core contributions to the SPL assets). Their focus on the need to deliver means that these stakeholders are most likely to address the traditional domain of requirements engineering: the specification of the functional and non-functional requirements for the software development effort.

Corporate management is focused upon the business requirements. In the specific instance of a game development project they are concerned with financing the current project (development and marketing) and maintaining budget control. This implies a set of market requirements (such as game genre and style guidelines) that are not often reported upon in the requirements engineering literature; requirements such as understanding the market for the game are critical precursors to preproduction. Given the market constraints upon these enterprises, understanding which requirements are mandatory and maintaining focus upon addressing *only* those requirements is an important contributor to long-term success.

In the general instance of the game SPL assets, these assets reflect the need to be commercially viable. The higher the proportion of the asset base that has been used (rather than *built but not used* in the release version of the game) or reused, the better the return on investment. Further, these assets may also represent a significant portion of the value of the business entity.

### 3.2 Technology Requirements

The aforementioned market pressures have lead sector participants to investigate any means available to get to market as inexpensively and as quickly as possible. The team considered combinations of one or more of the following approaches for meeting the technology requirements in the face of such market pressure, finally committing to a third-party multi-platform middleware game engine as the basis for the Core.

1. Restrict the Core to one target market (*e.g.* iOS or Android) and build upon the native libraries.
2. Implement a completely proprietary Core for all target markets.
3. Use third-party libraries to address the target markets.
4. Adopt a third-party multi-platform middleware game engine as the basis for the Core.

## 4 Observations

The alignment between the expected implementation of the requirements for the greenfield SPL were sufficiently close to that of a particular third-party middleware game

engine that the decision was made that it was “close enough”. While the middleware does reduce the time to market, it also induces a dependency upon the middleware provider that may be a substantial business risk. Careful management is indicated.

As the capabilities of middleware offerings continue to grow, offering more features (depth) and more supported platforms (breadth), they have become the basis for many mobile game SPLs. They represent the accumulation of significant domain experience and evolve just as the core of any SPL is expected to evolve. Their abstractions define the architecture for any game implemented upon them and their APIs represent the core functional requirements for the software artifact.

Development efforts based upon such middleware must depend more heavily upon the playing experience than technological capabilities to achieve market differentiation. The middleware can also impose constraints upon game designs: If a game concept requires a feature not available in the middleware, it may be cut due to budget and/or time constraints.

Once the middleware platform was adopted, it was our experience that game SPL assets evolved, as shown in Figure 2, to become: (1) extensions to the middleware (build upon the middleware to increase the *depth* of possibilities), (2) additions to the middleware (build beside the middleware to increase the *breadth* of possibilities), and (3) proprietary SPL Assets including reusable software and media assets of all types.

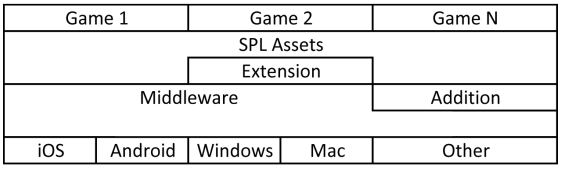


**Fig. 2.** Architectural Model

## 5 Conclusions and Future Work

Effective software product management is a key success factor in mobile game development. We have observed the process of introducing SPL planning into a game development effort and note the following for further consideration in future work.

- The adoption of a game engine as the core of the SPL creates a need for a software product management model for middleware based mobile game software product lines.
- The ROI for adopting middleware as the core of a SPL should be extensively evaluated as there may exist special considerations due to the relative size of investments for middleware *vs.* extensions, additions, and other SPL assets.
- Management decision support (beyond ROI) is needed for evaluating the effect of extensions *vs.* additions to middleware based SPL when meeting market pressures.

– Mechanisms to identify and control product differentiation in middleware based SPL are needed as the scope for technology-based product differentiation is reduced by this dependency.

The items discussed so far in this work lead to the conclusion that requirements engineering and software product management in mobile game development should focus upon "what is possible within the budget" rather than on "things that we would like to have". There is little margin for error and there is no guarantee that, even if the requirements are correct, the game will be a market success. Moreover, adopting third-party middleware and executing on third-party hardware gives a scant opportunity for technology-based product differentiation. New software features will remain exclusive for only a single product cycle – competing on this basis is very difficult and places developers in a constant state of technological warfare with their competition.

If requirements engineering is only about determining the functional and non-functional requirements for the software artifact, then it appears that our contributions to this domain are destined to shrink, perhaps even to the point of near irrelevancy as software development consumes an ever smaller portion of the development budget. However, if we understand that the customer experience is the differentiator, and we adapt and extend requirements engineering to support this *modus operandi* with techniques such as experience requirements [4], our ability to contribute remains strong.

## References


1. Jonathan Blow. Game development: Harder than you think. *Queue*, 1:28–37, February 2004.
2. David Callele, Eric Neufeld, and Kevin Schneider. Requirements engineering and the creative process in the video game industry. In *RE '05: Proceedings of the 2005 13th IEEE International Requirements Engineering Conference*, pages 240–250, Paris, France, 2005. IEEE Computer Society.
3. David Callele, Eric Neufeld, and Kevin Schneider. Introducing experience requirements. In *RE '10: Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, Sydney, Australia, 2010. IEEE Computer Society.
4. Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. Selecting empirical methods for software engineering research. In Forrest Shull, Janice Singer, and Dag .. Sjøberg, editors, *Guide to Advanced Empirical Software Engineering*, chapter 11, pages 285–311. Springer London, London, 2008.
5. Christof Ebert. The impacts of software product management. *J. Syst. Softw.*, 80:850–861, June 2007.
6. Eelke Folmer. Component based game development: A solution to escalating costs and expanding deadlines? In Heinz Schmidt, Ivica Crnkovic, George Heineman, and Judith Stafford, editors, *Component-Based Software Engineering*, volume 4608 of *Lecture Notes in Computer Science*, pages 66–73. Springer Berlin / Heidelberg, 2007.
7. Andre Furtado, Andre Santos, and Geber Ramalho. Streamlining domain analysis for digital games product lines. In Jan Bosch and Jaejoon Lee, editors, *Software Product Lines: Going Beyond*, volume 6287 of *Lecture Notes in Computer Science*, pages 316–330. Springer Berlin / Heidelberg, 2010.
8. C.M. Kanode and H.M. Haddad. Software engineering challenges in game development. In *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on*, pages 260 –265, april 2009.