



To Buy or Not to Buy: The Game Developer's Dilemma

Part 1: Understanding Risk

JUNE 2008

The only way to mitigate risk is to concentrate on making something great.

Tim Schafer (*Full Throttle, Grim Fandango, Psychonauts*), D.I.C.E. 2005

Executive Summary

Game development is a risky proposition. Teams must grapple with design risk, execution risk, market risk and financial risk. Perhaps the one factor that can dramatically affect all outcomes, though, is technological risk and the game engine strategy. The decision to build an engine or license an off-the-shelf solution impacts many other aspects of your project, from finances to resources to timelines, and can significantly change your risk profile. Your game's innovation strategy is a key consideration for determining what game components can potentially be licensed. Equally important is the selection of a vendor who will work with you as a true partner. The right game engine and framework can offset certain dimensions of risk and jumpstart your efforts while enabling your team to unleash its creative potential without limitation.

The right relationship with the right vendor providing needed low-innovation core components to your project can dramatically lower your risk profile and free your team up to focus its creativity on the high-innovation aspects to the game that matter the most to ROI.

This paper discusses some of the pros and cons for building and for buying a game engine, and presents a general framework for the decision-making process that can be useful in project planning and risk mitigation. Part 2 of this series focuses on the costs involved in building or buying and inputs to total cost of ownership (TCO) calculations. Taken together, these papers can help you identify scenarios where licensing middleware can greatly accelerate your project, reducing time-to-market and mitigating much of the risk for your title.



One Size Does Not Fit All

Before we begin, let us state that all commercial game engines are not alike, and the vendors who sell them have different philosophies and business practices that can affect the success of their customers' projects. Different game engines come with very different licensing options and standards for customer care, and the strategies driving each vendor's primary business are not the same across the industry. This means that offerings must be evaluated separately.

However, there are enough commonalities to roughly group factors for the "build or buy" question based upon the broad-strokes decision of licensing third-party tech versus developing everything in house. Which strategy is right for you? That depends upon your unique circumstances. This series of whitepapers can help you better understand your situation in order to make the best decision possible and minimize technological risk.

We recognize that you've hired the best and brightest people that you can find, and their opinions matter. The decision framework outlined in this paper is not meant as a replacement for the process of discussion and debate that is common as teams wrestle with these issues. Such decisions sometimes come down to matters of culture more than anything, and no outside source can dictate that. Emergent Game Technologies acknowledges a vested interest in this subject, yet these papers are presented with as much unbiased information as possible in order to help your team make the best decision about your game engine strategy.

The Benefits of Reuse

In other industries, outsourcing and purchasing third-party technologies are accepted standard practices for supplying needed functionality to the business. In game development, where innovation is the presumed Holy Grail and every project feels completely unique, the idea of using a game engine built outside the company can sometimes sound like anathema. For certain types of games and certain types of teams, however, purchasing a game engine can often be an accelerator, specifically due to the benefits of code reusability, a key to risk mitigation in software development.

Some high-profile successes have been reported for use of COTS (commercial off-the-shelf) systems in other industries:

"[T]he National Aeronautics and Space Administration (NASA) successfully employed COTS products in reengineering the Hubble Space Telescope Command and Control system [Pfarr]. The effort incorporated over 30 COTS and GOTS [government off-the-shelf] software components and achieved improved capability while decreasing the use of custom code by 50 percent. The program was able to meet an aggressive schedule, delivering 10 major releases of the system in 48 months."

From *Identifying Commercial Off-the-Shelf Product Risks*, SEI Carnegie Mellon

One mainstream definition of a "reusable" software asset is an artifact that is:

"...well documented, generalized beyond the needs of a single project, thoroughly tested, and ideally has several examples to show how to work with it. Robust assets are much more likely to be reused than items without these characteristics. A *reusable asset* is a robust asset that has been used on three separate projects by three separate project teams (at a minimum). You can claim that something is reusable, but it isn't truly reusable until it's actually been reused; reusability is in the eye of the reuser, not in the eye of the creator."¹

Many commercial game engines pretty much perfectly match this definition.

Of course, the promised benefits of reusability are rarely seen on a single project. Instead, reuse across subsequent projects is where huge advantages are gained – but remember, those previous projects need not be ones delivered by your own studio. A game engine that has already been deployed in many diverse ways is proven to be robust and stable. When licensing an engine that other games have deployed, you are directly benefiting from others' investment into the codebase. The mature codebase provided by commercial middleware can be a distinct advantage for you, not only in time-to-market but also in reduced maintenance costs.

¹ *The Enterprise Unified Process* by Scott W. Ambler

Certainly, reusing the same engine on your own subsequent projects does offer a clear advantage to your studio: your team need not relearn or reinvent software with the next title, and may also be able to re-purpose some content. Tools that are delivered with the engine, as well as other tools that are designed by your team, can usually be repurposed for other projects. Combined, these can add up to significant savings in both time and cost for today's project and into the future.

A middleware vendor can almost be seen as an outsourcing partner, both in providing significant modules of functionality to the team, and also by serving as the primary contact to other companies in the technology supply chain. The discussion below outlines some guidelines for deciding what components are suitable for this psuedo-outsourcing model.

Steps to a Game Engine Decision

To evaluate your build-vs.-buy options, start with outlining your specific circumstances and needs through the following steps:

Emergent Game Technologies maintains relationships with all the leading hardware manufacturers and best-of-breed component providers and is in a key position to work closely with those vendors in advance of their new product introductions.

This means that you need not allocate resources to the time-consuming tasks of researching and implementing upcoming technologies into your game: Emergent does it for you.

- 1. Assess organizational bias.** If your team is predisposed to building (or buying), that should be acknowledged and the reasons examined. New data that comes in through this assessment process may change opinions – or validate them. In either case, it should be a useful exercise.
- 2. Define your game's core and assess requirements on the Spectrum of Innovation.** Innovation comes in many forms and flavors in the games industry, and the degree of innovation in your game – or an intentional and calculated strategy of low innovation – has a direct bearing on the level of risk. The core of your game may be a completely new gameplay mechanic, it might be a novel twist on a tried-and-true dynamic or a genre-defining invention – or you might be sticking to old standards and doing them better than anyone has ever done, which can be a very profitable strategy. Use the tools provided in this paper to assess your position on the innovation spectrum and better understand your risk profile.

3. **Review available middleware for fit.** Is the candidate engine a good fit today, and tomorrow? Maybe it has what you need for prototyping today and there's functionality on the vendor's future roadmap that you will need, but it's not critical at the outset of the project, in which case the fit may still be complementary. Just as important: Evaluate the vendor itself to be sure that it's the right company to place such an important bet on.
4. **Prepare Risk Assessment/Mitigation Matrix.** By mapping out known risks and identifying mitigation plans, you'll be in a better position to evaluate the different technology options through a realistic and practical lens.
5. **Develop Total Cost of Ownership (TCO) estimates for all best-fit alternatives.** TCO includes purchase price (for commercial products) or R&D expense estimates, as well as cost of supporting the engine, training costs, costs for developing and integrating tools, documentation, and more. Please refer to Part 2 of this series for insights into TCO drivers for your project.

Project Characteristics that Impact Risk

A variety of factors beyond pure technical and functional requirements must be taken into account when evaluating options for your game engine. Important characteristics that can dramatically affect the risk profile for your project include:

- Genre and type of game
- Mix of platforms to be supported
- Generation of chosen platform(s), plus availability and richness of tools
- Development languages to be used
- Tools and tech already procured and available
- Release strategy: all SKUs releasing at once, or staged
- Release date: fixed (i.e., tied to a calendar date such as a movie release or holiday season) or moveable

Emergent updates the technical integration points from the Gamebryo framework to Emergent Tech Connection partner products, alleviating much of the headache in retrofitting and upgrading interrelated tools within the development ecosystem.

- Experience of team working together (is this the first project for this group?)
- Experience of individuals on the team with the specific technology, platform and tools
- Hiring sequence and team assembly and ramp-up strategy; is the entire team in place already?
- Other tools and tech selected (AI, sound, physics, etc.)
- Spectrum of innovation – gameplay, design, technology, etc. – and the degree of innovation in each dimension

Emergent has developed a Game Project Evaluation worksheet to help you analyze these different elements. Additional information on some of these dimensions is provided below, and a further discussion of risk assessment for your individual project can be found towards the end of this paper.

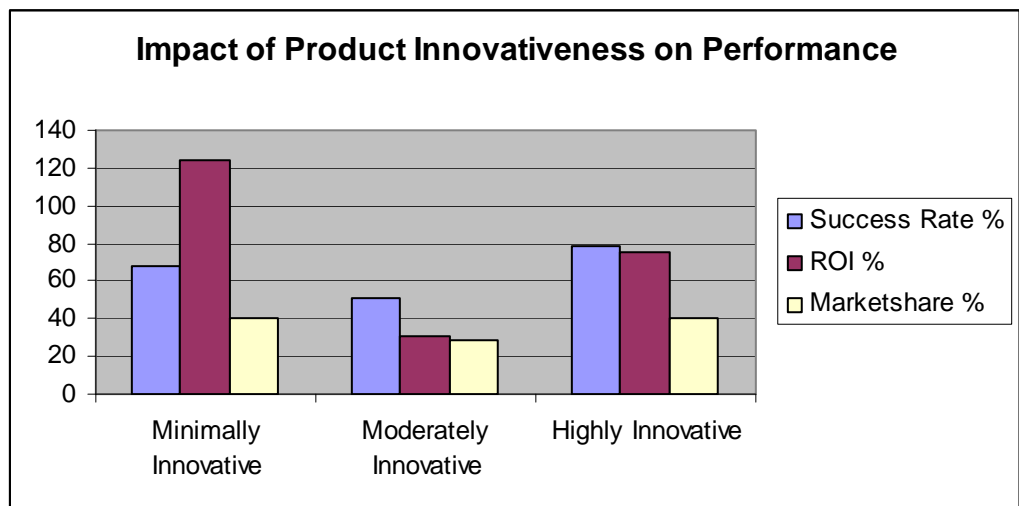
Your Innovation Strategy

Researcher Robert Cooper has done significant work on innovation in product development across industries. The chart on the next page depicts data from a 1991 study² with a somewhat surprising U-shaped curve revealing that moderate innovation in product development is not nearly as profitable as major innovation can be – or no innovation at all. While this study did not specifically include video games, this U-shaped curve can likely be extrapolated to this field.

² Data from E.J. Kleinschmidt and R.G. Cooper, “The impact of product innovativeness on performance,” *Journal of Product Innovation Management* (1991): 240-251, as cited in *Winning at New Products* by Robert G. Cooper

Take a pass through your requirements to identify the level of innovation – high, medium, or low – of core components of your game.

As part of the entertainment industry, innovation in games is often – though not always – richly rewarded. The level of innovation attempted has a direct impact on the amount of risk your project will carry. As you develop your project requirements and define the scope and mechanics of the game in your planning stages, take an explicit pass through your requirements set to identify the level of innovation – high, medium, or low – of core components of your game. Be sure to list out the low-level innovations that are not necessarily apparent to the player, as these have a definite impact on the risk profile and may be negatively correlated with ROI. By identifying where your project sits on the spectrum of innovation, you can take a realistic approach and gain insights into your risk profile.



A full discussion of innovation in product development as applied to games is beyond the scope of this paper (see references for additional resources). The key is to limit your focus to innovating where it makes a difference: in gameplay and content, but not in the foundational aspects of technology that can be obtained pre-built on the open market.

Resources Should Match Priorities

Once you've laid out your features and requirements and placed them on an innovation scale, you're in a position to ask another useful question: Does your budget reflect the priorities of your project? The areas you've defined for innovation in your game are those where you should put your most precious resources: your people.

"What we're trying to do is get the stuff that makes sense to centralize and be common on that same core platform. For example, there's just some things like physics engines and audio and user interface and all the stuff you've got to do to handle online. There's no reason for every team to have to reinvent that. That's the stuff we want to be core and centralized..."

*Matt Booty,
interim CEO of Midway,
interviewed on
Gamasutra.com*

What parts of your project can be considered, if not commodities, at least standard functional components for which tried-and-true solutions are available from outside sources?

From your list of requirements:

1. First define the heart, or *core*, of your project
2. Then carve out the *context*, which are the ancillary or supporting features and functions that are required but not mission critical.

Now you have two new dimensions of your requirements set to help you assess your strategy for sourcing the functionality, based on the chart below (borrowed from Geoffrey Moore's Cycle of Innovation³ and adapted for game development).

Low Innovation modules are valuable resources that are necessary for the game and are also well understood, almost a commodity. High Innovation components are those precious gems particular to your game, that must be developed specifically by your team of engineers and artists to the unique specifications of your game.

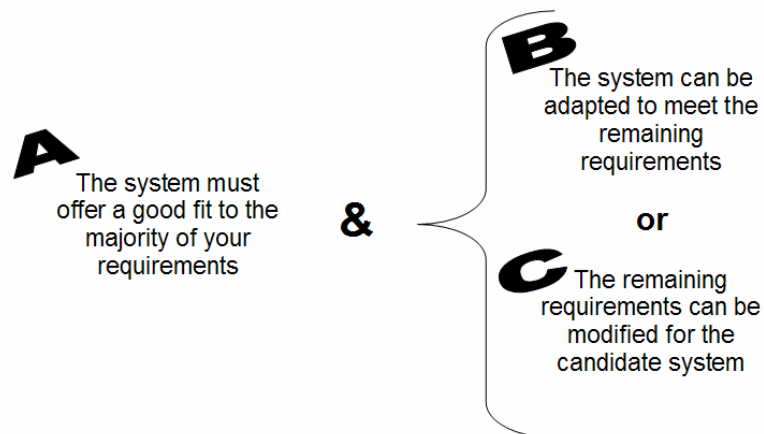
	Low Innovation	High Innovation
Core: Mission-Critical	Buy Commercial Solution	Build In House
Context: Supporting	Contract Development	Partner

³ Geoffrey Moore, *Dealing with Darwin* as presented to Nokia Group, from lecture notes posted at http://3dpeople.blogspot.com/2006_11_01_archive.html

By focusing your team on building the high-innovation heart of your game – the creative game core – and procuring all supporting functionality through contract development, partnering and purchasing, you’ll be in the best position to reduce risk and shorten development cycles and overall costs for your project. The high-innovation core of your game is usually that which is most visible to the player as distinctive and unique. This is where you want to focus your precious resources. All the rest of the game (especially the low-game-innovation but mission-critical components) are prime candidates for the pseudo-outsourcing model when supplied by the right middleware partner. The rest of this paper discusses methods for assessing fit for such purchased components.

Middleware and Supporting Components: Assess the Fit

How do you determine if a particular third-party product is a good fit? A generic heuristic for assessment of middleware includes these distinct conditions:



The architecture of the game engine and its interfaces determine ease of reusability on your project. An engine that is designed appropriately facilitates reuse across titles of very different styles and genres. If the features and interface to the engine are too low-level (for example, only providing rendering), the team will be forced to recreate significant amounts of code with each new title, defeating the goal of reusability. If the engine interface is too abstracted and high-level, it forces too many requirements on the developer (for example, level

editing, or first-person point of view), and all games built on that engine may end up with essentially the same look and feel. An appropriately-designed interface to the engine gives the proper level of abstraction to the feature set while providing all the “heavy lifting” of 2D and 3D graphics, multiplatform console support and the content tools and pipeline requirements for the project.

By breaking down your project into its component parts, you can understand where to focus your efforts in researching appropriate off-the-shelf solutions for the array of functionality your project requires.

Vendor Considerations: Choose Wisely

In evaluating possible off-the-shelf options, the features and attributes of both the candidate middleware *and the vendor who offers it* should be carefully considered. Besides the obvious functional assessment of the software, some vendors will be a better fit for certain projects based upon their operating strategy, culture and breadth of offerings. Your middleware vendor should become a trusted partner for you who is committed to the success of your project. As a starting point, get answers to questions such as the following about your prospective supplier:

Your middleware vendor should become a trusted partner for you who is committed to the success of your project.

- What core business is the vendor in? Do they build software for games, or do they build games?
- What platforms are supported by the engine?
- What technologies are used in the software?
- Is source code available?
- What is the refresh strategy, or the frequency and type of upgrades and patches, plus the mechanisms through which they’re offered and your ability to select the updates required for your specific needs?
- What feature set exists in the current release? What is the roadmap for future functionality, especially for any identified mission-critical features?

- How stable is the vendor? How long have they been around? Are they growing every year?
- What titles have been produced with their tech? What genres?
- What are the support policies, procedures and practices? What reputation does the vendor's support have?
- What types of documentation are available, and how is the quality? Are documents produced for programmers, artists and designers? How comprehensive is it?
- What type of developer training is available, including sample apps, tech demos and/or classroom training?
- What type of training is available for artists and designers, including tutorials, demos and/or on-site sessions?
- What type and quality of community is available of developers and artists using the engine? Are online resources, forums, live update channels and email lists offered and supported by the vendor?

Cost is also an important factor, and the dimensions of project cost must be understood as part of the decision. For more insight into this subject, refer to the second part of this series which covers Total Cost of Ownership and unearthing hidden or often-overlooked costs in the budget and the schedule, to get a better apples-to-apples comparison of your options in technology decisions for your project.

Detailed Risk Assessment and Mitigation

While it is impossible to predict all the issues that can undermine the team's momentum and sabotage a project, an ongoing practice of risk identification and mitigation planning is valuable on any technology endeavor. The Game Project Evaluation Worksheet provided by Emergent helps with more global elements of risk that apply to many types of games. The discussions in this paper may have

also prompted your team to engage in a more in-depth analysis of other risk factors.

A Risk Mitigation Matrix is a useful project management tool to anticipate risks and minimize their impact long before a crisis develops. Several dimensions of each risk should be captured:

- your tolerance for the risk, e.g., low impact, undesirable, or unacceptable;
- the likelihood that it will occur;
- the consequences should the risk materialize; and
- snapshot mitigation, or a quick-and-dirty bulletpoint remediation plan

Risk planning can be as exhaustive as warranted by your project (e.g., if you're building software for a nuclear facility, this would be a cornerstone of project management). For smaller development projects, sometimes a very simple spreadsheet such as the starter sample provided on the next page might be sufficient. More in-depth risk mitigation will be needed for projects higher on the spectrum of innovation.

	Risk Factor	Tolerance	Likelihood	Consequence	Mitigation
1.	Average developer experience on engine/tools is less than 1 year	Undesirable	Medium	Slow ramp-up time, delays in getting to full production	On-site developer training from tools vendor
2.	Vacations and PTO requests	Low impact	High	Delays getting to full production, schedule must be extended; morale problems if policies not clear and consistent	Coordinate schedules, implement advance-notice policies on PTO requests; schedule team-wide shutdown around holidays.
3.	Critical-path tool requires midproject upgrade due to showstopper bug	Undesirable	High	Forces waterfall upgrades of dependent tools across the toolchain	Good vendor relations; use of proven, pre-integrated tools
4.	Lack of standardization among developer tools and workspaces	Unacceptable	Medium	Delays and disruptions in build processes; difficulty in diagnosing issues due to incompatibilities	Development and adherence to developer desktop/tool standards
5.	Engine code is not extensible	Unacceptable	High	Schedule delays and project holdups; rework and wasted effort; missed milestones; morale problems	Proper due diligence and understanding of requirements; assessment of vendor and fit prior to commitment
6.	Codebase is "young"	Undesirable	Medium	Unstable code; longer Q/A phase; less predictable quality	Implementation of rational reuse strategy
7.

Be sure to differentiate between actual risks and results. For example, "Game is released too late for shipment for holiday season" is actually a result, or consequence, of the occurrence of a collection of risks at much lower details of the project, including improper scheduling, unclear definition of scope, lack of resources, unplanned attrition, and much more. These should be mapped out per the nuances of your own circumstances. Depending upon the severity and likelihood of a particular risk factor, development of more in-depth mitigation plans may also be warranted.

A review of this paper and its counterpart on costs can be helpful in identifying risk factors that are relevant for your specific project. Risks will change as the project evolves. The Risk Assessment Matrix should be updated on at least a monthly basis with the team leads and published widely. Such a matrix can be especially useful within overall project reporting to different parts of the organization.

We're Here to Help

At Emergent, we recognize that not every team will choose our products. However, we are committed to helping the industry continually raise the bar and want you to make great games, using whatever tools and strategies are right for you. We firmly believe that you will have more success in “making something great,” as Tim Schafer counsels, by concentrating on what you do best – innovative game play, mechanics, design and art – and let us mitigate risk by delivering a killer game engine to your engineers that takes care of the standard headaches that most projects encounter in a cost- and time-effective fashion.

We hope that you find some value in the discussion presented here and invite you to contact us if you have any feedback or would like to begin a conversation around how our products might help you achieve your goals. Also, don't forget to review Part 2 of this paper to understand more about calculating costs and weighing options in the buy-versus-build scenario.

If you would like to learn more about the advantages that Gamebryo can provide to your project, please visit our website, www.emergent.net

REFERENCES

1. Pfarr, T. & Reis J.E. "The Integration of COTS/GOTS Within NASA's HST Command and Control System," 209-221. *Proceedings of the First International Conference on COTS-Based Software Systems* (Lecture Notes in Computer Science, 2255). Orlando, FL, February 4-6, 2002. Berlin, Germany: Springer-Verlag, 2002.
2. *Identifying Commercial Off-the-Shelf Product Risks*, SEI Carnegie Mellon, September 2003;
<http://www.sei.cmu.edu/pub/documents/03.reports/pdf/03tr023.pdf>
3. *The Enterprise Unified Process* by Scott W. Ambler, Prentice-Hall, 2005;
<http://www.enterpriseunifiedprocess.com/essays/strategicReuse.html>
4. *Winning at New Products* by Robert G. Cooper, Perseus Publishing, 2001;
also see great discussion of these concepts as applied to games on Daniel Cook's blog *Lost Garden* at
<http://lostgarden.com/2006/03/never-innovate-halfway.html>



Copyright ©2008 Emergent Game Technologies, Inc. All rights reserved.

Gamebryo, Gamebryo and Floodgate are trademarks and/or registered trademarks of Emergent Game Technologies. Xbox 360 is a trademark of Microsoft Corporation. PLAYSTATION®3 is a trademark of Sony Corporation. Wii is a trademark of Nintendo Corporation. All other trademarks are the property of their respective owners.



To Buy or Not to Buy: The Game Developer's Dilemma

Part 2: Total Cost of Ownership

JUNE 2008

The disadvantage of in-house tools is that they are far more expensive to build and maintain, often costing more than off-the-shelf tools. In many cases it is utterly impossible to build your own tools because of the scope of the application.

Gil Gribb, Tech Lead at Raven Software,
quoted in *Game Engine Anatomy 101* at extremetech.com

Executive Summary

The technology decisions made at the outset of a game development project will fundamentally impact every dimension of the effort, from schedules and quality to team morale. In Part 1 of this series, we outlined a decision-making framework for the game engine strategy based on risk assessment and risk mitigation.

In this second paper, we examine the elements of that strategy that impact costs and can contribute to (or undermine) overall project success, focusing on factors comprising total cost of ownership (TCO) calculations such as support, maintenance and keeping the tool chain up to date. These often-overlooked carrying costs of in-house delivered tech can unfairly burden teams of any experience level. This paper can help producers and managers do a true apples-to-apples comparison in assessing different technology strategies for their next project to come up with the most effective and cost-efficient option to build a great game

"The most radical possible solution for constructing software is not to construct at all."

*Frederick Brooks, Jr.,
The Mythical Man-Month*

The Cost/Benefit Analysis of the Middleware Proposition

Drastic overruns in budgets and schedules are unfortunately common in both software and entertainment, and game projects, being at the nexus of these two dynamic sectors, may seem doomed to this fate. After-the-fact reports of a studio spending \$100MM in three years on core engine development, or a large publisher spending over \$138MM in a single year on its engine strategy, are not unheard of.¹ How do we as an industry start to curb the costs and evolve game development towards more efficient and less painful product cycles? A rational examination of the actual costs of the foundation of the game – the game engine – can help.

As in Part 1 of this series, let us again acknowledge that no two game engines are alike. However, the essential requirements and cost drivers of a graphical game engine and tools framework remain the same, independent of how the code is developed or its origins inside or outside your team.

Emergent Game Technologies strives to provide as much unbiased information as possible in these papers in order to help your team make the best decision on the game engine. We believe that middleware can be a key enabler for this industry's innovation, and we hope your project will be one that can benefit from our products. Our approach to serving game developers is to provide foundational technologies upon which you can build your own unique and market-differentiated game, enabling creativity without compromise through a *buy then build* strategy that can save you money and time so that you can focus on the core of innovation for your game.

Total Cost of Ownership

Games are ever increasing in scope and ambition, and the resulting costs are also going up. Understanding the underlying factors and project expenses is obviously important. When cold hard cash is not expended for a particular component or technology because the plan is to build it in house, it's easy to

¹ Figures collected from annual reports of public companies and interviews in gaming industry press.

minimize or even overlook its actual price tag. The truth is that all components of your game come with overhead and carrying costs, whether you're building – and maintaining – everything yourself (or your publisher is), or you're contracting with the vendor for an annual plan for bug fixes and support.

When modeling expenses, the calculations for total cost of ownership, or TCO, should include everything required to implement the system and keep it functioning smoothly in your environment. TCO captures all the hard costs that can be quantified, including but not limited to:

- the cost of acquisition (e.g., licensing fees), or of development
- project startup and initial integration with existing tech and tools
- retrofitting and future re-integrations when upgrading one or more dependent tools in the chain
- ongoing maintenance (bug fixes and patches)
- enhancements or other extensions of functionality
- training new employees on the system
- developers' support of other developers on their code
- repurposing code and systems for successive hardware generations
- development and maintenance of the documentation

“Gamebryo really simplified our development process. It provided an extremely stable and robust foundation that allowed our team to focus on the tools and content specific to our unique and innovative game.”

*Ryan Seabury,
Creative Director,
NetDevil
(LEGO Universe)*

The latter categories of maintenance, training, documentation and support are sometimes forgotten when calculating cost for in-house efforts, and they can add up to be more than the original R&D investment if not managed. Just the task of keeping all internal tools integrated with the current releases of third-party products can suck time and money out of the project and lead to painful delays – especially when scrambling to recover from the cascade of forced upgrades triggered by a showstopper bug encountered in one such tool. When planning your project, these factors need to be taken into account and weighed into the decision-making around the game engine.

What's in a Game Engine?

Another important point to consider is the full spectrum of functionality that commercial game engines offer. Besides complete graphics libraries and advanced rendering capabilities, commercial engines provide many other important – and costly to develop – features.

A viable game engine must include such critical functionality as:

- scene graph optimizations
- collision detection
- 3D audio
- level editor
- animation tools
- particle, shader and shadowing systems
- terrain editor
- interfaces to various hardware- and platform-specific graphics libraries
- pipeline tools, including robust exporters from modeling packages
- cross-platform capabilities to ease development for multiple target devices

Case in point: The Gamebryo game development environment is a framework that provides not only standard engine functionality, but also a pluggable architecture that is easily extensible with in-house and third-party tools, including those offered by Emergent's Tech Connection certified partners – tool integrations which, by the way, are always kept in sync with the Gamebryo engine.

All engines have differing feature sets which must be evaluated on their own terms², and comparing the capabilities of a candidate engine against a realistic cost/effort estimate of providing that equivalent functionality in house is critical. The remainder of this paper outlines both hard costs and soft benefits available from both sides.

² Please contact the Emergent Sales Team at +1 (512) 356-9882 for more information on Gamebryo's feature set.

Benefits of Acquiring Proven Technologies

Some project dimensions that can be positively impacted by acquiring a previously-deployed game engine include:

Decreased Development Time & Cost: A significant portion of development costs for a given title stems from both the engine and tools for the art pipeline. These costs can range from 40-70% of a project's budget. Projects built on a commercial product often require smaller teams because so much of the game's foundation is already built. Productivity can ramp up fast, especially in the early stages, due to the jumpstart that this foundation provides. Both these factors result in time and cost savings. In fact, use of a commercial game engine can shave from 12 to 36 man-months of R&D effort off the overall project.

Quicker Time to Market: Depending on the number of engineers on the project, the 1-to-3-year reduction in effort usually translates to a shorter development timeline, with decreases of six months to two years and a commensurate improvement in time to market.

Shorter Q&A Cycles: "Young" code, by definition, is not as robust, stable or bug-free as code that has been tested and deployed on multiple platforms and projects. Young code can slow development and lengthen iteration cycles. A commercial game engine brings the advantages of reuse: the vendor's QA team has certified the release, and the bulk of the code has typically been deployed multiple times before. (See Part 1 of this series for a discussion of re-use as a risk mitigation strategy.) When you consider that the test and debug phase can take anywhere from one-tenth to one-third of the entire development cycle, acquiring a significant chunk of your codebase that's already been through those quality processes instantly reduces the timeline by a corresponding factor, typically in the range of one to six months.

"It's much quicker to buy something, to purchase something that's already existing. That's easy, and it's probably less expensive than to create your own."

Taku Murata, Square Enix, in an interview with Gamasutra, April 2008

Increased Content Output: For most games, art and content development consume most of the schedule. By buying rather than building a game engine and tools, the creative team will be able to get a feel for the game play and artwork at an earlier stage in the development cycle, reducing the time required for the create-design-debug loop. Acquiring such tools at the outset of a project serves as an accelerator as the entire team concentrates more energy on content development earlier.

Less Maintenance Burden: An estimated 50 to 70% of lifetime software cost is devoted to maintenance. Licensing a commercial game engine is essentially outsourcing a good chunk of your code maintenance responsibilities, and the overall burden of maintenance is shared across all customers of the engine, reducing the cost for each individual customer.

Emergent is in a great position to represent your needs, along with our other customers, in our discussions with leading technology players. All our clients benefit from our partnerships and ability to gain early access to the developments in the industry. By building support for new cross-industry innovations directly into our engine's release cycle, our customers' ROI increases and they are better equipped to stay on the leading edge of technology.

Easier Adoption of New Technology: Trying to keep up with the changes in graphics cards and consoles while simultaneously getting a title out the door on schedule is difficult. Adapting programming practices and code to take advantage of multi-core capabilities is a great example. Adding multiple platforms to the mix – especially newer platforms – can further complicate the team's schedule and add pressure on quality. A middleware vendor can smooth your adoption curve for new technologies by incorporating them into the roadmap for the game engine with less of a hit on your own engineering resources. Vendors manage many key relationships within the industry: to console manufacturers, hardware and software companies, and other middleware providers, which frees your team to specialize on the innovations of your game with fewer distractions.

Dedicated Developer Support and Higher Quality Documentation: Whether built in-house or licensed externally, the game engine must be supported, and the programmers who created the engine must be available to answer questions. The true cost of providing this support – and the price the team pays when it's not available – should not be overlooked. Your game engine must have complete documentation, samples and tutorials so that new programmers coming onto the project can quickly come up to speed without incurring a productivity hit to the entire project as new talent is absorbed. Availability of dedicated resources to answer questions and walk through code samples can significantly reduce drag on your entire team and improve output.

Just Because You Can, Doesn't Mean You Should

Engineers like to build stuff, and it's only natural that game engineers often want to build all of the game. This isn't necessarily the best strategy in every situation, however. Game developers are justifiably wary about giving up control of their code through overuse of third-party tech – and the skepticism towards game engines may be warranted when some vendors are also their customers' direct competitors. However, few developers today would even entertain the idea of building their own database, and a game engine is a similar foundational system upon which multiple and varied end-products can be developed. In many situations, licensing a game development framework can provide a head start at the beginning of the project and add up to significant time savings over the project lifecycle – especially when building a cross-platform title.

Emergent Game Technologies takes a different approach to our relationships. Our developers are focused on building useful, practical technology that can accelerate our customers' efforts in producing great games. We incorporate customer feedback directly into our products and design our roadmap around the challenges facing the industry today. Emergent believes that collaborating – with studios and publishers, with other middleware providers through the Emergent Tech Connection, even with universities, professors and students in our Academic Program – is the best way to enable all of us to be more successful. We are committed to delivering quality code and valuable tools to engineers, artists and designers to simplify game development, increase iteration rates, reduce costs and accelerate the evolution of the art.

"It's certainly a wise investment. For the money we invest in Gamebryo, the return is tenfold."

*Rob Denton,
Creative Director/
Co-Founder, EA-Mythic*

When faced with the decision on whether to build everything yourselves or acquire outside technology, alongside your evaluations of the software's functionality, assess the type of partner that your potential vendor will be, and consider that as part of the benefits mix in the equation.

Arguments Against Middleware

Some leaders in the game industry seem diametrically opposed to middleware, and the reasons they cite are actually quite valid for some types of middleware. All middleware isn't created equal, and the arguments against certain types of tools – or more specifically, certain tools vendors – do not apply to all game technologies.

For example, Mike Acton of Insomniac Games participated in a build-or-buy panel at the 2008 D.I.C.E. conference where he was quoted as saying, "At the end of the day, the middleware company makes money in this scenario [of you buying their engine], whether or not your game does well; whether or not their engine works well." The obvious counterpoint to Mr. Acton's comment is that free markets are self-correcting, and inferior products do not stand the test of time. Only game engines that work well will survive, both in the short and long run of this industry. Also, some middleware (Emergent's products being an example) is offered through a variety of licensing models, from standard software licensing agreements to innovative price structures that are tied to a game's revenue stream. This can provide in a balanced risk-sharing arrangement between the parties.

"I don't think we'd be in a big rush to build engine technology from the ground up again. It's just a massive expense."

*David Doak
Free Radical Design
(TimeSplitters,
Haze), interviewed in
GameInformer
Magazine*

In that same D.I.C.E. session, Andy Burke, tool group lead at Insomniac, said, "Third-party tech can reduce your costs, but it will not eliminate your costs. It will almost always take more investment than you think it will." This is absolutely true, and this series of papers strives to illuminate those hidden costs, in order to help you better predict the full investment picture. But this statement overlooks a key point: Third-party tech can reduce or eliminate other costs besides dollars devoted directly to programming. A good third-party solution includes a full suite of developer documentation, training materials, sample apps and tech demos – plus it's supported, which means you can call someone for help when you get stuck. Internally-developed tech comes at a cost, too... *and it will almost always take more investment than you think it will.*

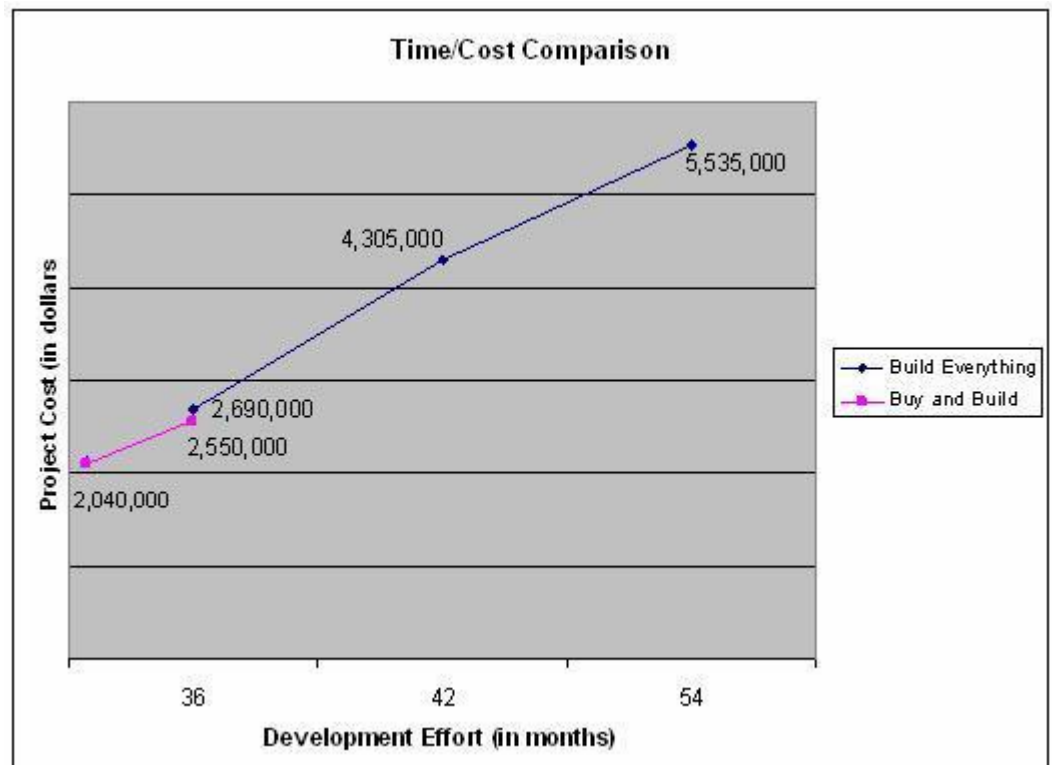
Buy *and* Build

Good software is flexible, and the same is true for good software teams. Perhaps our industry and the technologies available have all evolved to the point where a best-of-both-worlds solution is truly feasible. A realistic assessment may

reveal that the opportunity cost of building everything in house is an unacceptable price to pay in both hard dollars and delays, as time spent building what could otherwise be purchased means time *not* spent applying your creative prowess to building a great game. One very small project was modeled to compare two different engine development options:

- a buy-and-build strategy would likely require 10 people and 30 months – though it could possibly take as few as 24 months
- a build-everything approach would require at least two more developers (12 or more) and could potentially stretch out from a minimum of 36 to 52 months or more.

The results are depicted in the following chart.



The best game engines are architected to make them extensible, with reusability at the very core. In many cases, a development team can take the tried and true components of a commercial game framework and build a high-quality game on top of it much more easily, quickly and cost effectively than they ever could when starting from scratch. With the right partner as middleware vendor, a strategy of *buy and build* often becomes the best option for a good team to deliver the most amazing results.

We're Here to Help

At Emergent, we recognize that not every team will choose our products. However, we are committed to helping the industry continually raise the bar and want you to make great games, using whatever tools and strategies are right for you. We firmly believe that you will have more success in “making something great,” as Tim Schafer admonishes, by concentrating on what you do best – innovative game play, mechanics, design and art – and let us mitigate risk by delivering a killer game engine to your engineers that takes care of the standard headaches that most projects encounter in a cost- and time-effective fashion.

We hope that you find some value in the discussion presented here and invite you to contact us if you have any feedback or would like to begin a conversation around how our products might help you achieve your goals.

If you would like to learn more about the advantages that Gamebryo can provide to your project, please visit our website, www.emergent.net.

REFERENCES

1. Interim CEO Matthew Booty in Midway's Q1 2008 earnings call with investors:

My major goals are simple. First, get our 2008 games set up for success. Secondly, put Midway back on the path to profitability. In terms of these two goals, I firmly believe that our core strategy is sound, and after spending three years and over \$100 million to develop a centralized engine, tool set and outsourcing infrastructure, I believe that we have a foundation that will allow Midway to lower the development cost and time to market for future titles, while also allowing us to focus on quality.

2. And from an interview on *Gamasutra*, 5/27/08:

It's several years, and well over a hundred million dollars invested in our common technology [shared tools and tech]. Getting that to work is one of those things that's easy to say, but difficult to really pull off at a deep, cultural level.



Copyright ©2008 Emergent Game Technologies, Inc. All rights reserved.

Gamebryo, Gamebryo and Floodgate are trademarks and/or registered trademarks of Emergent Game Technologies. Xbox 360 is a trademark of Microsoft Corporation. PLAYSTATION®3 is a trademark of Sony Corporation. Wii is a trademark of Nintendo Corporation. All other trademarks are the property of their respective owners.