Essay Proposal

COMP160 - Software Engineering Essay

Steven Cowie

April 1, 2017

Topic

My essay will be on: Software portability, the issues, benefits and potential solutions.

Paper 1

Title: Game Logic Portability

Citation: [1]

Abstract: Many game engines integrate the game logic with the graphics engine. In this paper we separate the two, thus making the logic portable between game engines. In our architecture the logic is represented as an ontology and a set of rules for a particular application domain. A mediator with an embedded rules-engine links the logic to a suitable game engine.We demonstrate our architecture in two ways. First, we show a traffic accident scenario running on two different game engines, with a separate mediator for each engine. The logic type is smart-terrain logic, with participants triggering events based on interaction and proximity tests. In the second demonstration (a simple first-person shooting game) we show the extensibility and performance of the architecture to control non-player characters quickly manoeuvring using proximity tests and waypoints.

Web link: `http://dl.acm.org.ezproxy.falmouth.ac.uk/citation.cfm?id=1178580`

Full text link: `http://delivery.acm.org.ezproxy.falmouth.ac.uk/10.1145/1180000/1178580/p458-binsubaih.pdf?ip=193.61.64.8&id=1178580&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5%2EEAA225A8AB01C582%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=745864381&CFTOKEN=47977140&__acm__=1491051573_40ef8c07d5784a09e188b7d265e70b5f`

Comments: This article talks about potential solutions to help fix portability between game engines and does experiment to show results.

Paper 2

Title: Title of paper

Citation: [2]

Abstract: The article discusses the advantages and disadvantages of portability in accounting software. It presents a historical background on how portability took hold in the business community. It discusses factors which suggest that many companies will purchase accounting software in the future. It offers information on several operating system platforms for open portability software. It enumerates several dimensions to application portability that a company should investigate.

Web link: `http://web.a.ebscohost.com.ezproxy.falmouth.ac.uk/ehost/detail/detail?sid=beec3153-6569-4ec0-918a-35995e038902%40sessionmgr4010&vid=0&hid=4207&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#AN=28099575&db=bth`

Full text link: http://web.a.ebscohost.com.ezproxy.falmouth.ac.uk/ehost/detail/detail?sid=beec3153-6569

Comments: This paper discusses the pro and cons of portability, it talks about the history and talks about portable software.

Paper 3

Title: Software portability: still an open issue?

Citation: [3]

Abstract: m Portability is widely regarded as a done deal, but, although progress has been made, the problem has not been solved; if anything, it is becoming more complex. The commercial impact of non-portability increases as information systems become more distributed and interoperability becomes a higher priority. This article explores the issues behind the issues and their technical and commercial impact. We also outline some possible solutions being evaluated, particularly within the X/Open community of IT buyers and suppliers. Proposals such as a what-works-withwhat information base and procurement assurance mechanisms are explored.

Web link: `http://dl.acm.org.ezproxy.falmouth.ac.uk/citation.cfm?id=235001`

Full text link: `http://delivery.acm.org.ezproxy.falmouth.ac.uk/10.1145/240000/235001/p88-tanner.pdf?ip=193.61.64.8&id=235001&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5%2EEAA225A8AB01C582%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=745864381&CFTOKEN=47977140&__acm__=1491051975_2084ffcd4119028281a73cdd87a1346a`

Comments: Talks about using middleware to help port between platforms.

Paper 4

Title: Title of paper

Citation: [4]

Abstract: Game assets are portable between games. The games themselves are, however, dependent on the game engine they were developed on. Middleware has attempted to address this by, for instance, separating out the AI from the core game engine. Our work takes this further by separating the game from the game engine, and making it portable between game engines. The game elements that we make portable are the game logic, the object model, and the game state, which represent the game's brain, and which we collectively refer to as the game factor, or G-factor. We achieve this using an architecture based around a service-oriented approach. We present an overview of this architecture and its use in developing games. The evaluation demonstrates that the architecture does not affect performance unduly, adds little development overhead, is scaleable, and supports modifiability.

Web link: `http://dl.acm.org.ezproxy.falmouth.ac.uk/citation.cfm?id=1384913`

Full text link: `http://delivery.acm.org.ezproxy.falmouth.ac.uk/10.1145/1390000/1384913/p3-binsubaih.pdf?ip=193.61.64.8&id=1384913&acc=PUBLIC&key=BF07A2EE685417C5%2EEAA225A8AB01C582%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=745864381&CFTOKEN=47977140&__acm__=1491051712_1e64dfb7f74ac18221b656f0b6205e92`

Comments: Talks about G-factor for portability, talks about the shift into using game engines, talks about portability

Paper 5

Title: Some experience in building portable software

Citation: [5]

Abstract: Several authors have discussed methodology for making software portable, but less has been written about the specific components of programs which are likely to be system-dependent. This paper is based on several years of successful experience in making a major software product (MARK IV) transportable among many operating systems and machines. The product is implemented in assembly language and developed on a single support system for all of the "target" systems. The specific strategies and conclusions presented here are based on more general principles, and should be more or less applicable to systems developed in higher-level languages. The system dependencies are isolated in as few modules as possible. Various techniques are used to include system-dependent code at assembly time, at installation tape creation time and at customer installation time. The system-dependent functions addressed by this paper are: program start and termination, input/output,

primary storage management, interrupt control, checkpointing, module loading, overlay structures and object module format and other installation considerations.

Web link: `http://dl.acm.org.ezproxy.falmouth.ac.uk/citation.cfm?id=803226`

Full text link: `http://delivery.acm.org.ezproxy.falmouth.ac.uk/10.1145/810000/803226/p327-stern.pdf?ip=193.61.64.8&id=803226&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5%2EEAA225A8AB01C582%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=745864381&CFTOKEN=47977140&__acm__=1491052126_40bf57cd03a279766ed737c9e6541da8`

Comments: Talks about separating the system dependencies

References

[1] A. BinSubaih, S. Maddock, and D. Romano, "Game logic portability," in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACM, 2005, pp. 458–461.

[2] K. Garen, "Software portability: Weighing options, making choices," *The CPA Journal*, vol. 77, no. 11, p. 10, 2007.

[3] P. Tanner, "Software portability: still an open issue?" *StandardView*, vol. 4, no. 2, pp. 88–93, 1996.

[4] A. BinSubaih and S. Maddock, "Game portability using a service-oriented approach," *International Journal of Computer Games Technology*, vol. 2008, p. 3, 2008.

[5] M. Stern, "Some experience in building portable software," in *Proceedings of the 3rd international conference on Software engineering*. IEEE Press, 1978, pp. 327–332.