Essay Proposal

COMP230-Ethics and Professionalis

Tristan Barlow-Griffin

October 9, 2017

My essay will be on: What are the negative effects industry professionals can experience during "Crunch Time"?

Within the games industry "Crunch-Time" is a well established time in the development cycle of a product. I have found, through personal interactions with ex-industry professionals, that Crunch-Time is often part of the cycle where productivity is highest. In this paper, the author aims to highlight the consequences of Crunch Time on the employees and the final product. The introduction will briefly mention all of the problems Crunch-Time can produce and, provide the essay motivation through opinion pieces, written by industry professionals about Crunch-Time. I will discuss the damage that this kind of work style can have on the moral of the employees. The discussion will also include how the final may be impacted and how this can affect the longevity of the companies involved. By using studies from other industries I will form an argument against this type of work structure, suggesting working methods that remove the necessity that some companies think crunch is. To conclude I will weigh the merits of the strategies currently employed to remove the crunch period and explore methods that can make reduce the impact if a company does have to go into Crunch.

Paper 1

Title: Solving circular dependencies in industrial automation programs

Citation: [1]

Abstract: Prevention of data loss in each scan cycle is of utmost importance in control system programming. For each variable to reflect the latest value, compilers compute the order of execution of control logic objects according to data flow. But this technique for ensuring data integrity fails when a circular dependency or a code loop is found. In this paper, we propose an approach to help solve this issue in 2 steps - by providing an interactive visual representation and providing a list of possible solutions. A tool is implemented based on this approach which can take

either the control logic code or the dump created by the compiler as input. The tool has been validated for effectiveness with industrial use cases from different domains."

Web link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/7819192/

Full text link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=7819192

Comments: Although this paper is aimed at industrial automation programs, applying their methods to ensure data integrity in the design of a game will be beneficial.

Paper 2

Title: Variable dependency analysis of a computer program

Citation: [2]

Abstract: "To keep pace with the advancement of technology, software products are overlooking the chances of soft errors in the program. Program analysis to check program execution flow is an effective way to detect soft errors. High level language computer program execution can be analyzed on the basis of the dependence of the variables used in the program. This paper illustrates a novel method to analyze variable dependencies of program based on automated generation of dependence graph. Dependence graph depicts the connectivity between the program variables where variables works as the vertices and dependence between variables perform as edges. The automated generated dependence graph also discovers the critical variables of a program and these critical variables perform as the key to detect the sequential execution of a program outperforming the existing methods of program analysis."

Web link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/6777891/

Full text link: http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=6777891

Comments:

Paper 3

Title: Predicting defects with program dependencies

Citation: [3]

Abstract: Software development is a complex and error-prone task. An important factor during the development of complex systems is the understanding of the dependencies that exist between different pieces of the code. In this paper, we show

that for Windows Server 2003 dependency data can predict the defect-proneness of software elements. Since most dependencies of a component are already known in the design phase, our prediction models can support design decisions.

Web link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/5316024/`

Full text link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=5316024`

Comments:

Paper 4

Title: Defining and continuous checking of structural program dependencies

Citation: [4]

Abstract: Dependencies between program elements need to be modeled from different perspectives reflecting architectural, design, and implementation level decisions. To avoid erosion of the intended structure of the code, it is necessary to explicitly codify these different perspectives on the permitted dependencies and to detect violations continuously and incrementally as software evolves. We propose an approach that uses declarative queries to group source elements - across programming language module boundaries - into overlapping ensembles. The dependencies between these ensembles are also specified as logic queries. The approach has been integrated into the incremental build process of Eclipse to ensure continuous checking, using an engine for tabled and incremental evaluation of logic queries. Our evaluation shows that our approach is fast enough for day-to-day use along the incremental build process of modern IDEs.

Web link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/4814150/`

Full text link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=4814150`

Comments:

Paper 5

Title: Dependence clusters in source code

Citation: [5]

Abstract: A dependence cluster is a set of program statements, all of which are mutually inter-dependent. This article reports a large scale empirical study of dependence clusters in C program source code. The study reveals that large dependence clusters are surprisingly commonplace. Most of the 45 programs studied have

clusters of dependence that consume more than 10 percent of the whole program. Some even have clusters consuming 80 percent or more. The widespread existence of clusters has implications for source code analyses such as program comprehension, software maintenance, software testing, reverse engineering, reuse, and parallelization.

Web link: `http://dl.acm.org.ezproxy.falmouth.ac.uk/citation.cfm?id=1596528&CFID=727945130&CFTOKEN=59144450`

Full text link: `http://dl.acm.org.ezproxy.falmouth.ac.uk/ft_gateway.cfm?id=1596528&ftid=717055&dwn=1&CFID=727945130&CFTOKEN=59144450`

Comments: This paper will identify the implications of having lagre Dependce clusters.

Paper 6

Title: Dependence cluster visualization

Citation: [6]

Abstract: Large clusters of mutual dependence have long been regarded as a problem impeding comprehension, testing, maintenance, and reverse engineering. An effective visualization can aid an engineer in addressing the presence of large clusters. Such a visualization is presented. It allows a program's dependence clusters to be considered from an abstract high level down thru a concrete source-level. At the highest level of abstraction, the visualization uses a heat-map (a color scheme) to efficiently overview the clusters found in an entire system. Other levels include three source code views that allow a user to "zoom" in on the clusters starting from the high-level system view, down through a file view, and then onto the actual source code where each cluster can be studied in detail.

Also presented are two case studies, the first is the open-source calculator bc and the second is the industrial program copia, which performs signal processing. The studies consider qualitative evaluations of the visualization. From the results, it is seen that the visualization reveals high-level structure of programs and interactions between its components. The results also show that the visualization highlights potential candidates (functions/files) for re-factoring in bc and finds dependence pollution in copia.

Web link: `http://dl.acm.org.ezproxy.falmouth.ac.uk/citation.cfm?id=1879227&CFID=727945130&CFTOKEN=59144450`

Full text link: `http://dl.acm.org.ezproxy.falmouth.ac.uk/ft_gateway.cfm?id=1879227&ftid=852741&dwn=1&CFID=727945130&CFTOKEN=59144450`

Comments: This paper will be useful not just for another take on cluster dependence but also the case studies used.

Paper 7

Title: Interprocedural control dependence

Citation: [7]

Abstract: Program-dependence information is useful for a variety of applications, such as software testing and maintenance tasks, and code optimization. Properly defined, control and data dependences can be used to identify semantic dependences. To function effectively on whole programs, tools that utilize dependence information require information about interprocedural dependences: dependences that are identified by analyzing the interactions among procedures. Many techniques for computing interprocedural data dependences exist; however, virtually no attention has been paid to interprocedural control dependence. Analysis techniques that fail to account for interprocedural control dependences can suffer unnecessary imprecision and loss of safety. This article presents a definition of interprocedural control dependence that supports the relationship of control and data dependence to semantic dependence. The article presents two approaches for computing interprocedural control dependences, and empirical results pertaining to teh use of those approaches.

Web link: `http://dl.acm.org.ezproxy.falmouth.ac.uk/citation.cfm?id=367022&CFID=727945130&CFTOKEN=59144450`

Full text link: `http://dl.acm.org.ezproxy.falmouth.ac.uk/ft_gateway.cfm?id=367022&ftid=50938&dwn=1&CFID=727945130&CFTOKEN=59144450`

Comments:

Paper 8

Title: A unified framework for coupling measurement in object-oriented systems

Citation: [8]

Abstract: The increasing importance being placed on software measurement has led to an increased amount of research developing new software measures. Given the importance of object-oriented development techniques, one specific area where this has occurred is coupling measurement in object-oriented systems. However, despite a very interesting and rich body of work, there is little understanding of the motivation and empirical hypotheses behind many of these new measures. It is often difficult to determine how such measures relate to one another and for which application they can be used. As a consequence, it is very difficult for practitioners and researchers to obtain a clear picture of the state of the art in order to select or define measures for object-oriented systems. This situation is addressed and clarified through several different activities. First, a standardized terminology and formalism for expressing measures is provided which ensures that

all measures using it are expressed in a fully consistent and operational manner. Second, to provide a structured synthesis, a review of the existing frameworks and measures for coupling measurement in object-oriented systems takes place. Third, a unified framework, based on the issues discovered in the review, is provided and all existing measures are then classified according to this framework. This paper contributes to an increased understanding of the state-of-the-art.

Web link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/748920/`

Full text link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=748920`

Comments:

Paper 9

Title: Predicting component failures at design time

Citation: [9]

Abstract: How do design decisions impact the quality of the resulting software? In an empirical study of 52 ECLIPSE plug-ins, we found that the software design as well as past failure history, can be used to build models which accurately predict failure-prone components in new programs. Our prediction only requires usage relationships between components, which are typically defined in the design phase; thus, designers can easily explore and assess design alternatives in terms of predicted quality. In the ECLIPSE study, 90 percent of the 5 percent most failure-prone components, as predicted by our model from design data, turned out to actually produce failures later; a random guess would have predicted only 33 percent.

Web link: `http://dl.acm.org.ezproxy.falmouth.ac.uk/citation.cfm?id=1159739&CFID=727945130&CFTOKEN=59144450`

Full text link: `http://dl.acm.org.ezproxy.falmouth.ac.uk/ft_gateway.cfm?id=1159739&ftid=378358&dwn=1&CFID=727945130&CFTOKEN=59144450`

Comments:

Paper 10

Title: Using Software Dependencies and Churn Metrics to Predict Field Failures: An Empirical Case Study

Citation: [10]

Abstract: Commercial software development is a complex task that requires a thorough understanding of the architecture of the software system. We analyze the Windows Server 2003 operating system in order to assess the relationship between its software dependencies, churn measures and post-release failures. Our analysis indicates the ability of software dependencies and churn measures to be efficient predictors of post-release failures. Further, we investigate the relationship between the software dependencies and churn measures and their ability to assess failure-proneness probabilities at statistically significant levels.

Web link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/4343764/`

Full text link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=4343764`

Comments:

Paper 11

Title: Using Software Dependencies and Churn Metrics to Predict Field Failures: An Empirical Case Study

Citation: [10]

Abstract: Commercial software development is a complex task that requires a thorough understanding of the architecture of the software system. We analyze the Windows Server 2003 operating system in order to assess the relationship between its software dependencies, churn measures and post-release failures. Our analysis indicates the ability of software dependencies and churn measures to be efficient predictors of post-release failures. Further, we investigate the relationship between the software dependencies and churn measures and their ability to assess failure-proneness probabilities at statistically significant levels.

Web link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/4343764/`

Full text link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=4343764`

Comments:

Paper 12

Title: Using Software Dependencies and Churn Metrics to Predict Field Failures: An Empirical Case Study

Citation: [10]

Abstract: Commercial software development is a complex task that requires a thorough understanding of the architecture of the software system. We analyze the Windows Server 2003 operating system in order to assess the relationship between its software dependencies, churn measures and post-release failures. Our analysis indicates the ability of software dependencies and churn measures to be efficient predictors of post-release failures. Further, we investigate the relationship between the software dependencies and churn measures and their ability to assess failure-proneness probabilities at statistically significant levels.

Web link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/document/4343764/`

Full text link: `http://ieeexplore.ieee.org.ezproxy.falmouth.ac.uk/stamp/stamp.jsp?arnumber=4343764`

Comments:

References

[1] S. Nair and R. Jetley, "Solving circular dependencies in industrial automation programs," in *Industrial Informatics (INDIN), 2016 IEEE 14th International Conference on*. IEEE, 2016, pp. 397–404.

[2] M. S. Sadi, L. Halder, and S. Saha, "Variable dependency analysis of a computer program," in *Electrical Information and Communication Technology (EICT), 2013 International Conference on*. IEEE, 2014, pp. 1–5.

[3] T. Zimmermann and N. Nagappan, "Predicting defects with program dependencies," in *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, 2009, pp. 435–438.

[4] M. Eichberg, S. Kloppenburg, K. Klose, and M. Mezini, "Defining and continuous checking of structural program dependencies," in *Proceedings of the 30th international conference on Software engineering*. ACM, 2008, pp. 391–400.

[5] M. Harman, D. Binkley, K. Gallagher, N. Gold, and J. Krinke, "Dependence clusters in source code," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 32, no. 1, p. 1, 2009.

[6] S. S. Islam, J. Krinke, and D. Binkley, "Dependence cluster visualization," in *Proceedings of the 5th international symposium on Software visualization*. ACM, 2010, pp. 93–102.

[7] S. Sinha, M. J. Harrold, and G. Rothermel, "Interprocedural control dependence," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 10, no. 2, pp. 209–254, 2001.

[8] L. C. Briand, J. W. Daly, and J. K. Wust, "A unified framework for coupling measurement in object-oriented systems," *IEEE Transactions on software Engineering*, vol. 25, no. 1, pp. 91–121, 1999.

[9] A. Schröter, T. Zimmermann, and A. Zeller, "Predicting component failures at design time," in *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. ACM, 2006, pp. 18–27.

[10] N. Nagappan and T. Ball, "Using software dependencies and churn metrics to predict field failures: An empirical case study," in *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*. IEEE, 2007, pp. 364–373.