Leveling Up by Design: Games Design + Engineering Communication = Innovative Games and High Functioning Teams

Traci Nathans-Kelly
Cornell University
35 PUBLICATIONS **147** CITATIONS

Leveling Up by Design:
Games Design + Technical/Engineering Communication = Innovative Games and High-Functioning Teams

T. Nathans-Kelly
Senior Lecturer, Engineering Communications Program
College of Engineering, Cornell University
Ithaca, NY USA
E-mail: nathans-kelly@cornell.edu

W. White
Senior Lecturer, Computer Science
College of Engineering, Cornell University
Ithaca, NY USA
E-mail: wmwhite@cs.cornell.edu

Conference Key Areas: Discipline-specific Teaching & Learning, Curriculum Development, Quality Assurance and Accreditation

Keywords: technical communication, engineering education, games course, outcomes, assessment, engineering communication

INTRODUCTION

At Cornell University in the US, two Games Design courses ask student teams to create, design, code, and publish original video games. Our courses combine game theory and design with engineering communication seminars, and they are the only classes that satisfy the Technical Communication Requirement for the Computer Science major in Cornell University's College of Engineering. The instructors' particular mix of scholarly expectations and innovative course design leads to public release of some very playable and successful games while teaching intricacies of team communication. (One 2017 team game developed in the course now has over half a million downloads from Steam.) Addressing how such high-functioning teams evolve, we provide this concept paper which demonstrates how qualitative and quantitative assessments gathered systematically from team assignments, milestone documents, and online assessments from the validated CATME system (Comprehensive Assessment of Team Member Effectiveness) [1].

The limited focus of this paper looks primarily at the Introductory Games course (CS/Info 3152), sophomore level, which uses a managed code environment. By simplifying the programming environment, there is a focus on design and software engineering topics. No more than three or four students on each team are programmers, with at least two designers (including artists, UI, or level designers). Similar good results have also been seen in the Advanced Projects course (CS/Info 4152, a capstone course); students use C++ and are provided coding resources beyond an SDL cross-platform layer. Students can integrate other elements such as graphics, AI, and networking. Both classes are similar in structure and consist solely of a team-based semester-long game design project, graded at many points in the semester, and then exhibited at a public Showcase at the end of the semester.

After a departmental review in 2007, these courses were revised to teach both game design and writing equally to meet the university's Technical Communication Requirement. At first glance, many people assume that a Games class integrates writing for storytelling aspects of game design. However, the documents we assign – and other communication exercises in the courses – are technical/engineering communication pieces and are framed as pre-professional writing for the students.

For the communication aspect, we believe strongly in two mutually beneficial stances. *The first and most important is that the documents and in-class critiques are critical in propelling the game design process forward, not just recording what has already happened*. In this way, our approach differs widely from how many perceive documentation's role in technical work. *The second is that document revision beyond the first draft reflects and supports the iterative and agile design process so valued in game development.*

Thus, these courses incorporate a writing seminar style with at least one discussion section per week to discuss and workshop documents. Two revisions of each major document contribute to a rigor cycle that is at the heart of the course. Teams must communicate to different audiences including players, potential funding partners/bodies, internal teams, managers, app stores reviewers. Importantly, the documentation is tightly integrated with the agile development cycle of the course and is not an add-on activity; this allows the instructors to address external pressures (accreditation bodies and future employer demands), internal pressures (preparing our students for work demands), and self-imposed standards (ensuring that interdisciplinary teams run smoothly while making a viable game).

We begin by outlining the course structure for the Intro course specifically, but both courses use a similar development schedule. We then provide a review of recent writing outcomes for Intro and general team effectiveness (since 2016, using the CATME online team assessment system). We align course outcomes with guidelines from professional/accrediting bodies: 1) IGDA: International Games Development Agency, 2) ACM with IEEE: Joint Task Force on Computing Curricula Association for Computing Machinery and the IEEE Computer Society, and 3) the US's ABET: Accreditation Board for Engineering and Technology.

1 COURSE STRUCTURE

The basic structure for Intro and Advanced is shown in Table 1. While the primary grade in these courses is determined by the project completed at the semester's end, there are several intermediate deliverables throughout throughout the entire semester. Full course cycles, assignments, examples, and grading scales can be seen here, as they are too lengthy to cover herein: https://gdiac.cis.cornell.edu/courses/gdiac-courses.php .

While the Engineering Communication Program (ECP) has helped the CS courses with document design and assessment for 11 years, ECP has provided a dedicated instructor since 2014. This instructor helps maintain a writing seminar approach with high enrollments (the Intro course has 72 students spread across 12 teams; the

Advanced course hosts 54 students and 9 teams) with an average of 12 documents; two revisions are allowed for each major document (in bold in Table 1).

Table 1: Basic 16-week Intro and Advanced Course Structure with Deliverables

Week	Task	Deliverables (D), Planning Documents (P), or Reflective Activity (R)	Intended Audience for Documents
1-3	Learning game development technologies	Lab exercises	Instructors
4	Create game idea in team	**Concept Document** (P)	Investors (fictional); instructors
5-6	Demonstrate play for proposed game in non-digital form; present in class	Non-digital Prototype Game (D)	Class, instructors
7	Outline early game specifications	**Gameplay Specification** (P)	Internal team, instructors
8	Playtest in class an early online version of the game	Early Skeleton Playable Game (D)	Class, instructors
9	Game refinements; writing Architecture and Design Specs	**Architecture and Design Specification** (P)	Internal team, instructors
10	Present technical prototype	In-class presentation (D); revisions due (R).	Class, instructors
11	Create and outline level design specification; perform level design critique	**Level Design Document** (P)	Internal team, instructors
12	Present alpha game (early full game); player testing	Playable game (D) + in-class presentation; short 2-week spring report (R)	Internal team, instructors
13	Code walkthrough with instructors and classmates	Code via GitHub; in-class scrum (D)	Internal team, instructors
14	Present beta game (revised full game); player testing	**Game Manual** or **App Store Proposal** (P); short 2-week spring report (R)	Internal team, instructors, public
15	Final game release and in-class presentation	In-class presentation (D); revisions in process (R)	Internal team, instructors
16	Final game refinements for Showcase	Full game (D); **portfolio of all revised documents due** (R)	Internal team, instructors, public
17	Showcase: Public play and vote event	Post-mortem report (R)	Internal team, instructors, public

2 DOCUMENTATION WITHIN AN AGILE GAMES DEVELOPMENT CYCLE

Our game design courses are different in that the communication components are a major portion of the course. We do not frame the communication/professional elements as just another set of skills that employers want or as an add-on skill.

Instead, we believe that the communication components are critical for propelling the development process forward. Professional skills support the management of a large team's complex task set, compel the maturation of game design, ground the way that tests are performed, and help the teams move to game launch. These communication elements are not outside of the games course outcomes. Indeed, they *are* the course outcomes, along with a playable game, and are so assessed.

The nuanced difference in how the communication elements are at the heart of our courses may challenge how CS instruction or engineering schools frame the concepts of "communication skills" or "professional skills" within their programs (we dislike the term “soft skills” as it implies that those skills are easier than the “hard skills” of engineering work). We maintain that heavy communication assets are integral to successful CS and/or engineering work, and within our courses it is always presented thus so that students are not able to silo their developing technical skills sets. There is much to be said about this philosophy within engineering education; one only needs to search for terms like “situated learning” and “active learning” to find a plethora of good work being done by others where cross-functional skills hone the abilities of pre-professionals. *Our core philosophy is this*: documents should always aid and inform the student teams throughout the development process, both in planning for high performance and reflecting for improvement.

3 TRACKING DOCUMENTATION IMPROVEMENTS

When looking at the document grades in the Intro course, we do see some interesting quantitative results over the years. Table 2 documents the average improvement for each revision (each document is allowed two revisions, and all students must contribute to the effort). Note that student improvement has been generally increasing over the years, with significant improvement for the Game Manual and the Architecture Specification, which have changed significantly with our recent approach. But the significant improvement since 2014 can be linked to the addition of the full-time communication instructor in that year working in concert with the CS instructor.

As an example, Table 2 highlights some of the categories we track across academic years and their spans of improvement for the Intro course. Such tracking of outcomes regarding student work can provide to the instructors some insight into effectiveness (or lack thereof) of course alterations, assignments, and rigor cycles.

Table 2. Percent Average Grade Improvement Per Document Revision Versus Original First Draft in the Intro Course (3152)

Document	2011	2012	2013	2014*	2015	2016
Concept revision	16.48%	25.51%	33.46%	31.01%	22.30%	29.85%
Concept final	16.48%	7.99%	14.41%	14.25%	18.91%	15.66%
Gameplay revision	10.87%	9.82%	23.07%	20.30%	29.71%	33.08%
Gameplay final	18.44%	9.60%	14.37%	18.03%	15.89%	7.55%
Architecture revision	21.10%	48.76%	32.26%	57.06%	73.80%	55.64%
Architecture final	27.02%	11.92%	13.91%	29.83%	26.77%	26.77%
Manual revision	13.10%	3.30%	18.82%	5.93%	12.94%	13.20%
Manual final	11.29%	10.83%	17.40%	12.15%	28.07%	29.87%

Note: *Most first revisions show an initial large jump in improvement*, with final document versions achieving more modest improvements as the work for the final is simple fine tuning.
*Technical/engineering communication instructor added to course in 2014.

Since 2014 when both Games courses added a dedicated communication instructor, student teams have experienced increased acceptance rates and recognition at external game festivals such as Boston Festival of Indie Games (FIG), Casual Connect, and the Independent Game Festival (IGF). Another aim of the structure of these classes is to simulate (with constraints) a more industry-like experience for the students. In 2018, an undergraduate, working now as TA for the Intro games course, reflecting on his experience in the Intro course, said this:

> CS 3152 is the only class in the entire CS department at Cornell that teaches necessary skills for industry, many of which are skills that get ignored in the core curriculum. We work in real-life sized teams where we're responsible for organizing a sizable amount of work, we work in interdisciplinary teams and need to communicate with non-technical peers, and have to document our thought processes and progress. That doesn't happen in other classes, because the assumption seems to be we'll get that experience in industry.

Students nearing graduation, such as this one, come to understand that the rich document cycles and teamwork expectations of the Games courses start them on their journey towards being agile, insightful, and self-assured pre-professionals.

4 TRACKING TEAM MEMBER EFFECTIVENESS

For the past three years, the Comprehensive Assessment of Team Member Effectiveness (CATME) online program helped us track team efforts and issues [1]. As of this writing, CATME has been used by over 100,000 students in over 2,000 institutions, and in over 80 countries [1]. Feedback can be private between students and instructors (our chosen method) or shared between team members. There is a plethora of outputs from CATME for users to explore. For example, Table 3 shows not only the fine-grained types of data that can be collected, but it also demonstrates how we know our annual pedagogical refinements, prompted by CATME feedback, are working. Students, individually, report upon team successes and challenges.

For example, significant assignment changes were made to the Architecture and Design Specifications cycle in spring 2016; the 2018 numbers show significant performance upticks. We interpret improving numbers for the Final Document Portfolio as evidence that our enhanced pedagogical approaches for fostering better teams is working. A closer look at Table 3 reveals interesting trends, too; in the middle of the semester, some teams report slightly higher team cohesiveness than at the end of the term. As instructors, we reviewed their private comments in CATME, and we came to understand that the stress of completing the course and game, along with their other academic obligations, made for a rocky end of semester in terms of team cohesiveness. CATME allows for unique windows into teamwork.

Table 3: Example CATME Outputs for 3152 course

Architecture and Design Specifications (mid-semester) by Year	C (mean)	I (mean)	K (mean)	H (mean)	% of students responding
2016	4.01	4.02	3.93	4.18	97
2017	3.92	4.15	3.85	3.97	98
2018	4.20	4.32	4.20	4.29	93
Final Document Portfolio by Year	C	I	K	H	% of students responding
2016	3.97	3.94	3.88	4.07	81
2017	3.87	3.97	3.79	3.99	80
2018	4.21	4.27	4.15	4.32	88
Whereas: C = Contributing to the team's work: scale 1 (low)-5 (very high) I = Interacting with teammates: scale 1 (low)-5 (very high) K = Keeping the team on track: scale 1 (low)-5(very high) H = Having relevant knowledge, skills, and abilities: 1 (low)-5(very high)					

The CATME system provides copious statistics for any researcher to investigate for particular classes, well beyond the Table 3 example. Aside from such birds-eye views of a course or project, the CATME system allows for incredible granularity as well as space for long-form student commentary, permitting instructors a window into team problems and triumphs, especially if confidentiality is promised by the instructor.

5 ALIGNMENT WITH ACCREDICATION & ASSESSMENT BODIES

University computer science programs in the US are familiar with accreditation standards for IGDA, ACM/IEEE, and ABET, providing measurable program outcomes frameworks. Accreditation and program review agencies challenge academic programs to think more expansively. As for communication efforts, we agree with The Joint Task Force on Computing Curricula Association for Computing Machinery and the IEEE Computer Society: "Effective professional communication of technical information is rarely an inherited gift, but rather needs to be taught in context throughout the undergraduate curriculum" [2].

There is an oft-stated challenge of providing solid communication skills to undergraduates such that they are prepared well for success in the workforce [1, 3, 4]. Employers want new hires to be well-versed in communication skills and team skills; indeed, beyond problem solving, those are core expectations for new employees in engineering, software development, computer science, and other related fields [5]. Students should be agile in creating strong communication artifacts including project management, proposals, pitches, internal team documents, test plans, user guides, app pages, presentations, etc. We strive to teach our students to work with confidence and speak with authority about their projects and work.

Other academic programs are decidedly working towards communication prowess and team skills for their Computer Science/Software, Engineering, and related degrees in the US [4, 6, 7] and elsewhere [8, 9]. Some have a separate course [7, 8]; others embed communication tasks into their engineering/CS classes [7, 10, 11]. Instructional depth, range, or style isn't mandated by ACM/IEEE, IGDA, or ABET, so the strategies and outcome vary widely. Experiments to incorporate communication and professional skills into engineering/CS include those within MIT [12], Georgia Institute of Technology [13], and Ohio University [14].

We provide below a high-level alignment with ACM/IEEE, IGDA, and ABET guidelines. For tables covering our specific course alignments with these guidelines, see http://chec.engineering.cornell.edu/ecp-research/, which may serve as models for other programs in developing alignments for their programs.

5.1 IGDA Curriculum Framework 2008

The International Game Developers Association posted a resource from its Game Education Special Interest Group in 2008 [3], and states its desire for students to have experience in teams, writing, presenting, and working in cross-discipline teams, noting "Fundamental proficiencies are often absent in graduates, and require special attention" [3]. Our strongest alignment comes with the section titled "3.8: Game Development" where the IGDA addresses workflow in teams, planning, documentation cycles, planning, many of the same game support documents.

5.2 ACM/IEEE 2013 Communication Guidelines

The ACM (Association for Computing Machinery) released the updated "Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science" (CC 2013) in cooperation with IEEE and the IEEE Computing Society [2]. We align with the "Social Issues and Professional Practice" sections, indicated by "SP" therein. Communication ability is considered by the CC 2013 to be a core Knowledge Area for which "mastery experiences" should be provided. CC 2013 has language that covers communication skills including presentations, teamwork, and writing/documentation [9].

5.3 ABET Criterion 3

As with the other guideline sets above, the US's ABET body provides a framework for assessing the teaching of engineering work. ABET's "a-k criteria" [1] show that desired outcomes are more generally outlined than in the IGDA, for example. At the above URL, we provide a mapping that may be useful for CS programs and communication programs when working towards ABET accreditation.

6 SUMMARY

Spring semester of 2018 marks the eleventh anniversary of our integration of writing with game development. Agile and continuous development require that we (as instructors) revisit and revise the courses each year. The core feature of our approach is to make sure that the documentation always moves the development process forward, and CATME assists in our continuous assessment efforts. For all documents in the process, we identify for our students exactly why it is important to the agile cycles; we cut documents that don't contribute to the process. By applying these simple rules, we believe that instructors can create highly effective and professional teams that are better prepared to meet demanding early careers in CS or related fields.

References

[1] Loughry, ML, & Ohland, M (2018), CATME: Smarter Teamwork, About. http://info.catme.org/about/ .

[2] International Game Developers Association (2008), *IGDA Curriculum Framework: The Study of Games and Game Development*. Technical Report v 3.2beta. IGDA, http://www.igda.org/?page=resources.

[3] Association for Computing Machinery (ACM) and IEEE. 2013. Computer Science Curricula 2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. Technical Report. http://www.acm.org/education/CS2013-final-report.pdf.

[4] ABET. 2014. ABET Criteria for Accrediting Engineering Programs. Technical Report. http://www.abet.org/wp- content/uploads/2015/05/E001-15-16-EAC-Criteria-03-10-15.pdf.

[5] Burge, JE, Anderson, PV, Carter, M, Gannod, GC, & Vouk, MA (2011), Integrating communication instruction throughout computer science and software engineering curricula, Proceedings of the 2011 American Society for Engineering Education Annual Conference and Exposition, Vancouver, BC.

[6] United States Department of Labor (2015), Occupational Outlook Handbook. (December 2015). Retrieved April 5, 2017 from https://www.bls.gov/ooh/computer-and-information-technology/ computer-and-information-research-scientists.htm

[7] Etlinger, HA (2006), A framework in which to teach (technical) communication to computer science majors, Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, 2006, ACM, New York, NY, USA, pp. 122–126.

[8] Kaczmarczyk, L, Kruse, G, Lopez, DR, & Kumar, D (2004), Incorporating writing into the CS curriculum, Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education 2004, New York, NY, USA, pp. 179–180.

[9] Blume, L, Baecker, R, Collins, C, & Donohue, A (2009), A ‘Communication Skills for Computer Scientists" course, Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education 2009, ACM, New York, NY, USA, pp. 65–69.

[10] Cajander, Å, Daniels, M, McDermott, R, & von Konsky, BR (2011), Assessing professional skills in engineering education, Proceedings of the Thirteenth Australasian Computing Education Conference 2014, Vol. 114, Australian Computer Society, Darlinghurst, Australia, pp. 145–154.

[11] Hartman, JD (1989), Writing to learn and communicate in a data structures course, Proceedings of the Twentieth SIGCSE Technical Symposium on Computer Science Education 1989, ACM, New York, NY, USA, pp. 32–36.

[12] Mirel, B, Prakash, A, Olsen, LA, & Soloway, E (1997), Improving quality in software engineering through emphasis on communication, Proceedings of the 1997 American Society for Engineering Education Annual Conference and Exposition 1997, ASEE, Milwaukee, WI, USA.

[13] Stickgold-Sarah, J, & Thorndike-Breeze, R (2016), Disciplinary specificity in engineering communication: Rhetorical instruction in an undergraduate engineering research class, Proceedings of the 2016 American Society for Engineering Education Annual Conference and Exposition, ASEE, New Orleans, LA, USA.

[14] McNair, L, Miller, B, & Norback, J (2005), Integrating discipline specific communication instruction based on workforce data into technical communication courses, Proceedings of the 2005 American Society for Engineering Education Annual Conference and Exposition, ASEE, Portland, OR, USA.

[15] Liu, C, Sandell, K, & Welch, L (2005), Teaching communication skills in software engineering courses, Proceedings of the 2005 American Society for Engineering Education Annual Conference and Exposition, ASEE, Portland, OR, USA.