

Journal of the Association for Information Systems

JAIS 

Special Issue

Artifacts, Actors, and Interactions in the Cross-Project Coordination Practices of Open-Source Communities

Cecil Eng Huang Chua
University of Auckland
aeh.chua@auckland.ac.nz

Adrian Yong Kwang Yeow
Nanyang Technological University
aykyeow@ntu.edu.sg

Abstract

While there has been some research on coordination in FLOSS, such research has focused on coordination within a project or within a group. The area of cross-project coordination, where shared goals are tenuous or non-existent, has been under-researched. This paper explores the question of how multiple projects working on a single piece of existing software in the FLOSS environment can coordinate. Using the Ordering Systems lens, we examine this question via a cross-case analysis of four projects performed on the open source game Jagged Alliance 2 (JA2) in the forum Bear's Pit. Our main findings are that: (1) Ongoing cross-project ordering systems are influenced by the materiality of development artifacts. (2) The emergent trajectory of cross-project ordering systems is influenced by affordances that emerge from the interaction between the goals and desires of the project team building the development artifact, and the materiality of the development artifact. (3) When two parties need to coordinate in the ordering system, all or almost all coordination effort can be borne by a single party. Furthermore, over time, emergent FLOSS projects bear more coordination effort than stable, mature projects.

Keywords: Cross-project coordination, materiality, ordering systems, open-source

* Michael Wade and Kevin Crowston were the accepting senior editors. This article was submitted on 27th September 2009 and went through two revisions.

Volume 11, Special Issue, pp.838-867, December 2010

1. Introduction

Free/Libre Open Source Software (FLOSS) projects represent a new form of organizing where structures are flattened and distributed, and boundaries are fluid and permeable (Kellogg et al., 2006, Ljungberg, 2000). Work in FLOSS projects is distributed, highly modularized, and largely self-organized (Crowston et al., 2007). In many cases, open source software is developed by a small group. The group maintains the “core” of the open source development artifact (i.e., the actual software produced by the open source group), while others create separate projects that develop and submit new features or bug fixes (Fitzgerald, 2006, Scacchi, 2004). New features and bug fixes are typically provided as patches, which modify the development artifact (Crowston, 2008). Thus, in an open source environment, it is common to observe multiple, concurrent projects dependent on other projects for success (Adler et al., 1995, Sydow et al., 2004). In many cases, cross-project coordination is inherent in FLOSS work. For example, the Ubuntu version of Linux is based on the Debian project. Developers of Ubuntu must coordinate with developers of Debian to ensure Ubuntu works. In some instances, open source software development is inherently about coordination. Tikiwiki, for example, relies on a host of other open source projects including the Zend Framework, Smarty, and jQuery for its architecture. Coordination across projects is thus a critical ingredient in the success of much FLOSS software.

Furthermore, this type of work in FLOSS projects parallels that of other IT development projects such as the maintenance of enterprise software where multiple projects rely on shared resources and tools (Grabher, 2004, Whitely, 2006). Although each separate project has its distinct project team with its own goals, teams must coordinate their work across projects because they work on the same development artifact.

Despite the need to understand cross-project coordination, most coordination research in FLOSS and software development, in general, focuses on coordination within a project or group (Crowston et al., 2007, Faraj and Sproull, 2000, Herbsleb et al., 2000). The few studies in the management literature on cross-project coordination are mainly descriptive (Cusumano and Selby, 1995, Sabbagh, 1996). Recently, research has examined cross-project coordination's impact on overall project performance (Hoegl et al., 2004, Kratzer et al., 2008, Li et al., 2008). However, it is not clear from the literature how one manages dependencies across projects, i.e., how cross-project coordination is conducted. This resonates with the urgent call among FLOSS researchers to “grapple with the details of work practices” of such projects to provide a better understanding of this form of distributed work (Crowston and Wade, 2009).

We distinguish cross-project coordination from general coordination by defining it as “the contextualized interactions that manage tasks, activities, and interdependencies across *multiple* projects to realize and achieve *individual* project goals.” In contrast, within-project coordination refers to the contextualized interactions that manage tasks, activities, and interdependencies within a project to realize and achieve *shared* project goals (Faraj and Xiao, 2006). The existence of multiple goals across projects creates notable differences. For example, a single, shared goal implies that work occurs in a definable sequence, ending when the goal is achieved. With multiple goals, practices are emergent as the achievement of a goal by one project may be temporally unrelated to the achievement of another goal by a second project.

The objective of this research is to explore the nature of cross-project coordination in FLOSS communities to highlight additional coordination problems and practices. Given that this particular domain is under-investigated, we conducted an inductive theory-building case study on the cross-project coordination dynamics of an open source computer game known as “Jagged Alliance 2” (“JA2”). Although open source game communities have not been studied in depth in most FLOSS research, a growing number of FLOSS projects focus on the gaming domain (Scacchi, 2004). JA2, like Mozilla, is a release of former proprietary software from an established (now defunct) commercial company and represents part of the ongoing evolution of open source software development (Fitzgerald, 2006, Ljungberg, 2000). The case site is particularly ideal because a large number of

projects (more than 40) occur simultaneously on the site, all connected to the JA2 development artifact.

Using a combination of exploratory cross-case analysis and online ethnography, we studied three representative cross-project cases within the JA2 site. We traced their cross-project coordination interactions from inception to March 2010. Our longitudinal observations of cross-project coordination provide a basis to develop theory about relationships among the material artifact, interactions, and coordination practices.

Our main findings center on how different types of material artifacts – project artifacts, tool artifacts, development artifacts – and within-project practices shape and define coordination practices. A *project* artifact is created as a product of a project. A *tool* artifact is employed by a project to create a project artifact. For example, e-mail and language compilers (tool artifacts) are employed to develop a computer game (project artifact). A *development* artifact is simultaneously a tool and project artifact around which work in the community centers. For example, the Debian operating system is a development artifact in both the Ubuntu and Debian communities. The Debian community builds Debian, while simultaneously leveraging it to build other project artifacts (e.g., user interfaces). The Ubuntu community leverages Debian to build Ubuntu. Finally, a focal project is the project on which an analysis is centered. For example, in an Ubuntu case study, Ubuntu would be the focal project, which coordinates with another project, Debian.

Our study produces three insights: (1) Ongoing cross-project ordering systems are influenced by the materiality of development artifacts; (2) the emergent trajectory of cross-project ordering systems is influenced by affordances that emerge from the interaction between the goals and desires of the project team building the development artifact, and the materiality of the development artifact. Finally, (3) when two parties need to coordinate in the ordering system, all or almost all coordination effort can be borne by a single party. Furthermore, over time, emergent FLOSS projects bear more coordination effort than stable, mature projects.

2. Theoretical Background

2.1 Current research on coordination and FLOSS

The nature of work practices, and in particular, how work is coordinated in FLOSS environments, is still relatively under-researched as compared to the rich coordination literature focused on IT development within commercial entities (Crowston et al., 2005). The few studies conducted within the FLOSS environment focus mainly on coordination within one project, e.g., how programming tasks are assigned within a particular FLOSS project (Crowston et al., 2007). These studies have shown that mechanisms such as standardization; organization; loose coupling; restricted access; the OSS code architecture; incremental release; and the judicious use of tool artifacts like e-mail, scheduling systems, and versioning systems enable coordination (Baldwin and Clark, 2006, Egyedi and de Joode, 2004, Feller et al., 2008, Iannacci, 2005, Jørgensen, 2001).

While these studies shed interesting insights on work within each project, research has shown that work in open-source environments is typically separated into several related projects (German, 2003, Mockus et al., 2002). Although they do not directly study how coordination is enacted across projects, these studies hint at the complexity of such practices. For example, German (2003) described the need for procedures and policies for conflict resolution as well as clear architectural design for interfaces between modules in the GNOME project. Mockus et al. (2002), in turn, discussed how various Mozilla projects that looked at different modules required more formal means of coordination, as there was greater interdependence among them.

Some work outside FLOSS has examined the management of multiple projects. McFarlan (1981) highlights the need for an organization to develop a portfolio of projects to reduce risk. Montealegre and Keil (2000) emphasize the need to consider the Denver Baggage Handling system project in the

context of the multiple construction projects occurring in the Denver International Airport at the time. Hoegl et al.'s (2004) study on European automotive design teams found that cross-project coordination was an important factor impacting project outcomes, while Li et al. (2008) studied the interaction effects between cross-project coordination and project uncertainties on project performance. While these articles suggest the importance of cross-project coordination as a phenomenon to investigate, they either do not directly study the issue or take for granted how cross-project coordination occurs. Hence, there is a need for further research to investigate cross-project coordination.

2.2 Extant coordination theories

Broadly, research on coordination has two perspectives. The first focuses on how tasks are divided and integrated among different organizational units given the degree of uncertainty in the environment and/or within tasks. Examples of research include the Information Processing Model (Galbraith, 1973), Fit and Contingency Model (Lawrence and Lorsch, 1967, Thompson, 1967), and Coordination Theory (Malone, 1987, Malone and Crowston, 1994). This perspective assumes the “environment is predictable enough to characterize existing interdependencies” among these units given the tasks and environment and “that predefined mechanisms can be designed for various contingencies” (Faraj and Xiao, 2006 pg. 1156). The main goal of task-related coordination research is to understand what mode – i.e., the “hows” of coordination – can be applied to specific configuration of tasks, interdependence, and environmental uncertainty (Malone et al., 1999).

However, the cross-project context is characterized by a high level of unpredictability. New projects emerge unexpectedly from bugs or requests, or new open source groups are created to handle a changing environment. These characteristics limit the predictive power of these organizational theories of coordination, directing us toward the second perspective, i.e., the practice perspective of coordination (Faraj and Xiao, 2006, Kellogg et al., 2006).

The practice perspective is more amenable to an analysis of cross-project coordination, as it focuses on coordination practices employed in uncertain environments (Grabher, 2004, Whitely, 2006). Practices are defined as recurrent, materially mediated, and situated social activities (Schatzki et al., 2001). The practice perspective understands coordination as “a temporally unfolding and contextualized process of input regulation and interaction articulation to realize a collective performance” (Faraj and Xiao, 2006 pg. 1157). This stream of work views coordination practice as an *emergent* process that involves *contextual and temporal work practices* organizational actors enact in the course of coordinating their actions, expertise, and roles. Examples of coordination practices include expertise coordination in trauma centers (Faraj and Xiao, 2006) and among emergent disaster relief groups (Majchrzak et al., 2007), generalized procedures of exchange and provisional agreements in online advertising agencies (Kellogg et al., 2006), and boundary spanning practices in distributed project teams (Orlikowski, 2002). Within the practice perspective, we specifically adopt the Ordering Systems Lens for our analysis.

2.3 Ordering Systems Lens

The Ordering Systems Lens focuses on how technological artifacts intertwine with coordination practices. An ordering system is a “unique combination of *coordination artifacts* and *practices* arising from *organizational needs to manage interdependencies that transcend local interactions* to produce a workable degree of order” (Schmidt and Wagner, 2004 pg. 398). In an ordering system, practices are defined in an *emergent* way by the combination of actors, artifacts, and interactions that evolve around the actors and artifacts.

The unit of analysis for the Ordering Systems Lens is, therefore, a practice, defined as an interaction between at least two groups driven by a need to address interdependence. This interaction is situated and derived from “recognition of novel task demands, emergent situations, and the unpredictability of evolving action” (Faraj and Xiao, 2006 pg. 1157). The interaction, in turn, requires artifacts and actors. For example, Schmidt and Wagner (2004) demonstrate how the provision by software of layers in a CAD (computer-aided design) drawing enable separate departments to jointly work on

complex problems by providing modularity. Both FLOSS and organizational research point to the importance of artifacts in facilitating coordination. Studies have examined calendar systems (van den Hooff, 2004), versioning software in FLOSS projects (Scacchi, 2007, Yamauchi et al., 2000), and Intranets (Kellogg et al., 2006).

In addition, the Ordering Systems Lens requires a longitudinal focus. Interactions are temporally related, as the history of previous interactions enable and/or constrain current interactions (Faraj and Xiao, 2006). Thus, our analysis of coordination dynamics must take into account the “temporally unfolding nature” of these interactions (Faraj and Xiao, 2006 pg. 1157). In summary, in the Ordering Systems Lens, one explores how coordination artifacts shape one or more coordination interactions between actors over time.

We adopt the “Ordering Systems” lens over the others available in the practice perspective (e.g., Bechky, 2006, Faraj and Sambamurthy, 2006) for two reasons. One, the Ordering Systems Lens explicitly accounts for the role of artifacts in coordination practices. Artifacts are important to study, given that the FLOSS context is often at “arm’s length” and mediated by artifacts (Hemetsberger and Reinhardt, 2009). Two, unlike other practice-based studies that focus on one set of central organizational practices, the Ordering Systems Lens views practices as complexly interdependent instead of as “discrete, self-sufficient units” (Schmidt and Wagner, 2004 pg. 398). In exploring cross-project coordination, we wanted to explore the complex interactions between a focal project and other projects the focal project coordinated with, in addition to coordination practices between two projects. This would better match the complex reality of work and coordination in the FLOSS environment.

3. Methodology

3.1 Research Design and Setting

As few works study cross-project coordination in an in-depth manner, we conducted an inductive, theory-building case study (Yin, 2003) of three representative cross-project cases (code mod, data mod, and game mod) within an open-source computer game development community (JA2). Our analysis employed principles of online ethnography and cross-case analysis (Golden-Biddle and Locke, 1993, Hine, 2000, Lee, 1989, Yin, 2003) using the Ordering Systems Lens (Schmidt and Wagner, 2004).

As our case site, we chose the JA2 community hosted in the web forum called the “Bear’s Pit” (www.ja-galaxy-forum.com/board/ubbthreads.php) for two main reasons. First, it provided us open access to multiple projects. The JA2 community has more than 40 separate modding projects surrounding the JA2 game. See Table 1 for representative examples.

A “mod” or modding project involves creating modifications of the original JA2 game. Each project focuses on modifying separate components of the game, e.g., new items, weapons, characters, etc. As the web forum is an open site, it enables us to simultaneously observe and collect data on interactions across multiple concurrent projects over time. The JA2 community also encourages conversation on the forum rather than through private e-mail and archives messages for posterity. JA2’s Bear’s Pit retains most posts made since 2004.

Second, cross-project coordination is an ongoing concern for this community. Specifically, while individual mods can be played as a standalone component with the core game, the common practice is to play multiple mods simultaneously. For example, one may play with the rain and lighting effects from the “WeatherEffects” mod and load bearing equipment from the “New Inventory” mod. Thus, compatibility among different mod projects is important for modders if they want their enhancement played. The site, therefore, presented an opportunity to study the dynamics of cross-project coordination.

Table 1. Partial List of JA2 Mod Projects

Mod Type	Project Name	Description
Code Mod	JA2 1.13	A massive update to the base JA2 game engine. Called JA2 1.13, because the last official release of JA2 was 1.12 Gold.
	Stracciatella	JA2 was developed for Windows in Microsoft Visual C. Stracciatella is a port of JA2 to various Unix platforms.
	New Inventory	Added load bearing equipment (e.g., pocketed vests, backpacks) to the game
	Enhanced Description Box	Added a popup window into the game that better detailed the abilities of each game item.
	100AP	Allows for more fine-grained modeling of human actions in the game
	Many Mercs	Allows the player to recruit up to 32 mercenaries. The original game had an 18 mercenary restriction. Also allows for larger battles involving more enemies and allies.
Data Mod	Early Miguel and Carlos Patch	Allows the player to recruit two mercenaries early in the game.
	DBB	A mod that adds a huge amount of equipment into the game
	PossumDBB	An extension to DBB
	Pink skin	A mod that changes the skin tone of characters in the game
	Thor's Interface	A mod that changes the user interface colors and substitutes movie star faces for those of individuals the player interacts with
	WildFire MODs	3 related mods based on the original WildFire mod. There is a WildFire maps mod, equipment mod, and mercenary mod.
Game mods	Vietnam SOG	A game that uses JA2 to create a story set in the Vietnam era
	Urban Chaos	A game that uses JA2 to create a story set in a more urban environment. The country is called Danubia
	Militia Factions Tracona	A mod of JA2 Unfinished Business in the spirit of the Militia Factions mods. Also adds robots as enemies.
	Renegade Republic	Uses the JA2 1.13 mod to create a new game setting in the Caucasus Mountains.
	Legion	Uses JA2 Unfinished Business to create a new storyline
	Reconquista	JA2 with new weapons and mercenaries

3.2 Data Collection, Case Selection, and Data Analysis

Given our inductive approach, we engaged in several rounds of data collection and analysis where each subsequent round was guided by previous emerging theoretical insights. Thus, we employed theoretical sampling for each round of data collection (Yin, 2003). The iterative process of data collection and analysis was as follows.

Step 1 – General exploration of JA2 projects: We began our exploration of the mod projects in JA2 by compiling all information concerning the forty projects from the online forum – the Bear's Pit. We collected the data according to online ethnographic methods (Hine, 2000), in that one author was not only a player of the JA2 game for nearly a decade, but was also an observer of the main JA2 mod discussion board, Bear's Pit since 2007. Thus, the “embedded author” had a “native” perspective of the community.

We reviewed every post in the “JA2 1.13 Modders HQ” functional heading in the archive from

inception to March 2010.¹ We reviewed this data set to identify emergent themes and concepts related to cross-project coordination. Any sentence or paragraph that could be interpreted as coordination-related was copied and pasted into a separate file. We captured 55 pages of coordination notes in this phase. We further verified and expanded this data set using posts gathered from other sources about which the native author was aware (Klein and Myers, 1999, Miles and Huberman, 1994). See Table 2 for a list of secondary data sources.

These posts provided “multiple vantage points” to triangulate events and perspectives gleaned from the main online forum (Glaser and Strauss, 1967, Klein and Myers, 1999). We also developed a background case describing the software architecture of the JA2 game and mods.

Name/URL	Description
AIM Forum (http://forum.ja2.su/)	Russian forum. Includes discussions on the Night Ops mod
Fadden.com (http://www.fadden.com/techmisc/ja2/)	Summarized statistics of JA2 equipment and characters
Freelancer JA2 site (http://freelancer.ag.ru/ja2/)	Russian JA2 site that contains a map of Arulco, the game setting.
Gamer's Hell (http://www.gamershell.com/news_11045.html)	Contains an article about one of the earliest JA2 mods, Urban Chaos
Gun World (http://www.gun-world.net/ja2mod/ja2.htm)	Chinese gun website. Home of DBB mod.
HAM Wiki (http://ja2v113ham.wikia.com/wiki/Jagged_Alliance_1.13_HAM_Wiki)	A wiki about HAM, a JA2 mod.
Interview webpages (http://jaggedalliance.pl/mfm/en/interview.php http://www.tacticalarcancer.com/content.php?id=43)	Interviews with various JA2 modders
JA Galaxy (http://www.jagalaxy.com/)	Companion to Bear's Pit that presents JA2 news and has mods for download.
JA2 Modding Tools (http://www.esnips.com/web/JA2ModdingTools)	Tools for working with JA2
JA2 Stracciatella site (http://ja2.dragonriders.de)	Includes a forum discussing technical aspects of Stracciatella, a JA2 mod.
JA2 Tortoise Repository (https://81.169.133.124/source/ja2/)	Contains Ja2 1.13 and mod source code
JA2 Wiki (http://ja2vJA2_1.13.pbwiki.com)	Contains various pieces of information about JA2 1.13.
Moddb (http://www.moddb.com/mods/ja2-113-mod)	A database of mods for various games, including JA2
SirTech JA2 site (http://www.jaggedalliance2.com/)	SirTech Canada was the original developer of JA2
Walkthroughs and tips (various websites)	Describes the game rules in detail.
Wikipedia JA2 entry (http://en.wikipedia.org/wiki/Jagged_Alliance_2)	

¹ Bear's Pit has a four-level message display structure. At the top are broad functional headings such as “JA2 & UB Gameplay and Help,” “Game Procedures,” and “JA2 v1.13 Modders HQ”. These in turn are divided into threads, which are subdivided into pages that are further subdivided into postings.

After analyzing this data, we categorized the projects into three main project types: code mod, data mod, and game mod projects (Glaser and Strauss, 1967). This categorization was important, as it highlighted substantively different project types with respect to their work and coordination challenges. Briefly, a code mod project changes the source code of the JA2 game; a data mod project makes changes to data files, and a game mod modifies both source code and data to create a new game. Table 3 describes in detail the different concerns and expertise involved in each type of project.

Table 3. Types of Projects in JA2

Project Type	Definition	Expertise and work involved	Coordination concerns
<i>Code mod</i>	Changes the source code of the JA2 game	<ul style="list-style-type: none"> - Developer must have C/C++ programming skills, and (most typically) access to the Microsoft Visual Studio compiler - Code mods distributed as binaries and changes require compilation of all files for redistribution 	When two code modders independently create changes, the executable files distributed will not contain changes made by the other modder.
<i>Data mod</i>	Makes changes to data files of the JA2 game	<ul style="list-style-type: none"> - Graphics expertise for illustrating maps and item graphics - Some expertise in real-world military equipment required for modeling, e.g., weapons found in the game - Extensive integration of graphics, sound, and mechanical information into XML data files 	Distribution of specific changed data files may conflict, as different mods may alter multiple, often identical, files
<i>Game mod</i>	Modifies both source code and data to create a new game based on JA2	<ul style="list-style-type: none"> - Requires skills associated with both code and data mods - Needs a creative flair and ability to create a storyline that engages the player 	Game mods are continuously enhanced and are often under pressure to add features introduced into JA2 by other modders

Step 2 – Theoretical sampling of case projects: Using the project framework generated from step 1, we theoretically sampled cross-project coordination cases representative of each project type. Our case selection criteria were as follows: (a) the mod project must fit the project definition as stated above, (b) the mod project should have relationships not only with the principal game, JA2 1.13, but with other mods so as to more fully explore cross-project coordination, and (c) the project's mod should be popular, as that implies it would have more user feedback and more incentives to make enhancements. This, in turn, would encourage more cross-project coordination over time. This set of criteria enabled us to select complex and rich examples of cross-project coordination representative of each type of mod project.

While we were interested mainly in three types of mod project, we eventually selected four cases – the JA2 1.13, Enhanced Description Box (EDB), PossumDBB and Renegade Republic projects. We selected the JA2 1.13 mod project not as an analytical case, but as a foundation case because it was the core project for most mod projects within the JA2 community. Most modders employ source code and data from the JA2 1.13 mod project even though they can, in principle, use the original JA2 source code and data. Therefore, the JA2 1.13 mod project wields considerable influence over how coordination is performed on Bear's Pit. However, this influence is not all-powerful. Modders are free to disregard the efforts of the JA2 1.13 team, and some have done so (e.g., the Stracciatella project that focuses on cross-platform functionality). The other three projects formed the core analytical cases and were selected to be representative of each project type. The EDB, PossumDBB, and Renegade Republic projects represented code, data, and game mods, respectively. All three mods

had one principal project leader who performed most of the work. Both DBB and Renegade Republic relied on input from contributors.

We systematically focused our data analysis on forum posts related to coordination of the above mod projects. We re-organized our data and expanded it to include any post containing information even remotely associated with coordination. Table 4 summarizes the number of postings centering on EDB, PossumDBB, and Renegade Republic. These posts do not include forum postings where these mods are discussed as ancillary topics. We also did not list the number of posts for the JA2 1.13 project, as the vast majority of analyzed posts (more than 20,000) relate to it. It should be noted that while this study examined more than 20,000 posts, the Bear's Pit has an archive of more than 140,000 postings. Posts not analyzed were not directly relevant to the research and included those about Jagged Alliance (vs. JA2), off-topic political discussions, war stories, and posts about a potential Jagged Alliance movie.

Table 4. Number of Focused Postings For Each Mod

Mod Project	No.focused posts	Start date of posts	End Date
Enhanced Description Box	933	19 July 2008	13 December 2008
PossumDBB (inc. DBB)	1,709	1 January 2007	13 March 2010
Renegade Republic	2,689	27 October 2007	30 May 2009

Besides fitting our criteria of theoretical sampling, we also analyzed how representative these mods were of other projects within the JA2 community in two ways. First, because we selected mods rich in cross-project coordination interactions, we were forced to simultaneously gather data about other mods. We observed that these related mods had similar coordination practices. Second, we performed a smaller analysis on three unrelated mods, the Many Mercs (code), Early Miguel and Carlos (data), and Night Ops (game) mods, and discovered similar findings. Due to space constraints, we do not present the analysis of these latter three mods, but make it available upon request.

Step 3 – Case write-up and analysis: We drew upon the focused postings for each of the four mod projects to write a separate case for each project with regard to its cross-project coordination efforts. The embedded author wrote individual cases about each representative mod, while the other author read and discussed key themes within each case. This differentiated role strategy allowed us to simultaneously serve as involved insider and dispassionate outsider (Adler and Adler, 1988). The goal was to build key themes across the four cases and understand how themes clarify the cross-project coordination dynamics. Each author played the devil's advocate to the other's analysis, and offered complementary insights (Eisenhardt, 1989, Miles and Huberman, 1994).

Step 4 – Ordering Systems Lens: Iterating between extant research and the discussion of the coordination dynamics within and across cases, we identified key analytical concepts and processes. We first began our analysis from the practice perspective of coordination and focused on the cross-project coordination practices occurring within each project over time. It quickly became apparent to us that the role of artifacts was highly salient and deeply intertwined with the coordination practices emerging and evolving over time.

We then adopted the Ordering Systems lens to analyze our data. We reviewed each case to identify the coordination practices between each mod project and other mod projects. For each practice, we identified interactions, the artifact(s) that enabled the interaction, as well as the other projects (actors) coordination took place with. We found that practices could be categorized into three types:

- **Awareness:** the practice by which a focal project obtained and disseminated information.
- **Merger:** the practice by which a focal project combined its development artifact with other projects.
- **Cascade:** the practice by which a focal project handled changes made by other projects it had dependencies with over time.

Step 5 – Respondent Validation: The complete initial draft of this paper was circulated on the Bear's Pit to solicit comments, which were then used to revise the paper. Such respondent validation is useful to ensure authors' interpretations of events correspond to events that actually occur on the case site and, therefore, contributed to the external validity of our study (Mason, 1996). Most responses encouraged authors or centered on incorrect factual statements made. For example, one respondent noted that, "RR is set not in Balkans but in Caucasus mountains, on the territory of former Soviet Union." Others pointed us to other communities associated with JA2. Generally, respondents agreed the paper was a reasonable reflection of the current situation on the Bear's Pit.

Basically, the conclusions are spot on, that's why... we decided to start JA2 Stable Modding Platform. But that's for your next paper, lol. (30 June 2009)

4 Findings

We begin this section with the JA2 1.13 mod as the foundation case, which provides an introduction to the JA2 1.13 mod project and describes how it evolved to become the mod from which most other mods are derived. We then discuss our observations about cross-project coordination practices within the "EDB" mod, the "PossumDBB" mod, and the "Renegade Republic" mod. Given that individual interests and contexts characterize cross-project coordination, we begin each case by describing the interests and situated nature of each mod project. We then describe the awareness, merger, and cascade practices of each mod project.

4.1 Foundation Case: JA2 1.13 – the Dominant Mod

Background: The version of the JA2 code that appeared in the public domain in 2004 was the last version released, JA2 v. 1.12. One group that tinkered with this code was called the Whitehat team. This group changed the way weapons were stored in the game. Originally, weapon statistics were stored in a propriety format. The Whitehat team transformed weapon statistics into an XML file that anyone could edit. While the Whitehat team made some improvements, it disbanded within a year for various reasons. One significant reason was the team could not agree on a compiler to use. JA2 v. 1.12 was written in Microsoft's Visual C, which in 2004 was only available for purchase. Many members of the Whitehat team did not have this compiler and advocated converting the game to a version of C supported by free compilers like Gnu's C compiler (GCC). This proved difficult.

At the beginning I thought to get rid of the comercial [sp] parts is the most important work to do. Microsoft's Visual Studio was needed to build the program. Ergo get rid of MSVC. When gcc comes to compile vobject_blitters.c and stated it don't know nothing about __asm I almost start crying ;-). (5 November 2004)

Another modder, named Madd Mugsy, then took the Whitehat code with the intent to do three things: extend JA2 to make it easy to mod, fix the game-play bugs plaguing the original game, and extend some of the better, but incomplete, ideas from the official JA2 v. 1.12 game. The quote below lists some of Madd Mugsy's first modifications. Because a huge number were made (the original post, if reproduced, would be several pages long), only the most significant are highlighted. These include bug fixes, the externalization of data files, the increase in the game's item limit, and changes to the enemy artificial intelligence. "Externalization" is a conversion of the game's proprietary format data files into XML files that others can modify easily. Externalization makes modding possible for people who cannot understand C. The increase to 5,000 item slots facilitated the inclusion of additional equipment into the game. Finally, changes to enemy artificial intelligence allowed the enemy to use new equipment. Thus, equipment (data) mods could be used not only by players but computer opponents.

I've located and fixed the following bugs in that code so far:... Then I went ahead and added the following features: 1) Weapons from Unfinished Business added... 3) Externalized the following data in xml files... 5) 5000 item slots!... 9) Enemies can and will use new weapons... 19) AI changes: (22 July 2005)

For this research, the main impact of these changes was to create opportunities for non-coders to create data and game mods. Because these and later modifications were so significant, the mod was given the title JA2 1.13 to denote that it was a significant improvement, but kept the basic premise of the original game.

Practices: When JA2 1.13 was introduced, there were no dominant mods – the closest thing to a dominant mod was that of the Whitehat team. Nevertheless, as the quote below demonstrates, coordination with the (then few) other mods was a very real concern. Awareness building, merging with other mods, and maintaining compatibility with other mods over time was a problem.

But what we're going to end up with is about 5 completely different styles of incompatible games and tools that don't work across versions... I'm sure Bearpit would be interested in some of our ideas, and they may or may not incorporate them, but it sure would be nice to meld the two projects into one, then we'd have at minimum 10 people working on one project, instead of 4 people over there and 6 over here. (7 September 2005)

Awareness Building. To coordinate with other mods, the JA2 1.13 developers had to be aware of what other mod developers were doing, and to apprise other mod developers of the direction of JA2 1.13. During these early stages, awareness building was enabled via the Bear's Pit, an online forum (i.e., a tool artifact). The quote below demonstrates awareness. The concern is whether JA2 1.13 can function with a mod that made JA2 viewable in the 800 by 600 screen resolution; the original game only worked in 640 by 480 resolution.

*Original Post: Is the mod JA2 1.13 compatible to JA2 800x600 mod??
Reply: Not presently. (25 July 2005)*

In addition, JA2 1.13 was housed on a publicly accessible versioning system – Tortoise SVN. Tortoise SVN allows different versions of the same program to be housed in separate directories. The official JA2 1.13 version is located on the “trunk” directory. Mods are located on branches away from the trunk. Anyone can download the JA2 1.13 source code, and, therefore, keep themselves apprised of changes to JA2 1.13. However, only a select few individuals can modify the main JA2 1.13 code. The restricted access to the main JA2 1.13 code, in effect, defines projects. Individuals with access to the trunk can be members of the JA2 1.13 project. Other individuals must create separate projects, as they cannot directly contribute to JA2 1.13.

Currently only few coders are allowed to commit changes to the main branch. The changes will be committed only after testing. (JA2 SVN Rules)

Merging. The lack of a common compiler caused coordination failure among the Whitehat team members, because they could not merge code developed by separate modders. In 2005, Microsoft released the Visual C compiler to the public for free. This encouraged work on Windows-based JA2 mods, especially JA2 1.13. Modders now had a common library of C and C++ routines to call on for important low-level activities like drawing pictures on screen.

In 2005, most existing mods were code or game mods of the original JA2 game. Data mods were rare, mainly because, before JA2 1.13, most JA2 data was coded in a JA2-specific proprietary format and, thus, was inaccessible. The game mods that were completed at the time such as Urban Chaos and Vietnam SOG were released and then “abandoned.” These game mods made no attempt to coordinate with other mods.

Thus, in 2005, for all practical intents and purposes, the vast majority of coordination occurred between code mods. Code mods coordinated with each other by publishing their source code. Other modders then compared the source code to the original JA2 1.12 code to identify the changes. The below quote describes how Madd Mugsy introduced the 800 by 600 mod into JA2 1.13.

If it's got the 800x600 enhancement, I can compare the code I've got to it and maybe put that

in too. I'll compare it to the original JA2 gold source too, and see what's been changed. (01 September 2005)

Cascade. As time progressed, the practices employed for merging code mods changed due to two factors. First, the JA2 1.13 source code was made available to the public via the versioning system and could be downloaded. Second, Madd Mugsy had made many changes to JA2 1.13 that players wanted. This meant modders who modified the JA2 1.13 source code would provide their players with not only features of their mod, but also features of JA2 1.13. Over time, as more features were added to JA2 1.13, it became the “default” mod of JA2. Today, almost all mods of JA2 employ JA2 1.13 as the base code, instead of the original JA2 source code.

4.2 Case 1: Code Mod – Enhanced Description Box (EDB)

Project Interests: The EDB project is a modification to JA2 1.13 that allows a player to review the statistics of an in-game item. The original JA2 game had such a description box, but that required enhancements, as JA2 1.13 had made many changes to the way items worked. For example, the original game modeled the firing of a gun as a single number. In JA2 1.13, firing a gun was separated into the cost to lift the gun to a firing position and the cost of pulling the trigger. A comparison of the differences between the original description box and EDB can be found at <http://www.bookgallery.co.il/JA2PublicPosts/descbox.asp>.

The EDB project had to coordinate closely with key source code incorporated into JA2 1.13 such as the new inventory (NIV) and the old inventory management system found in the original JA2. The NIV introduced load-bearing equipment (e.g., pocketed vests and backpacks) into the game. NIV was critical, because EDB wrote its descriptions using windows drawn with NIV. EDB also had to consider other small projects such as the Many Mercs project, a project to allow the player to recruit more mercenaries. The EDB project had to actively coordinate with them so players would be able to combine the mods.

Practices: Table 5 presents the awareness, merging, and cascading coordination practices of EDB, which are described below.

Awareness Building. The principal way HeadRock, the developer of EDB, maintains awareness of his mod is by making frequent posts about EDB to multiple threads in Bear's Pit. On July 19 2008, a thread announced the start of EDB. Subsequently, regular updates are made to the forum of changes, e.g., “Enhanced Description Box goes ALPHA,” “Enhanced Description Box - Open Beta!,” and “EDB 1.1 Released!” These postings keep others up to date with the progress of EDB and give others a sense of what EDB will do and which code files are being changed.

The forum also provides HeadRock with information on the development of other mods related to project EDB. In the below quote, HeadRock is demonstrating knowledge of the 100 AP mod, and who is involved in its development.

I need to get Chris too [sp] add it to the 100 ap build, so I can test it along with 100ap. (22 August 2008)

In the JA2 version control system (Tortoise SVN), modders can apply to have a branch. Branches break off from the “\branch” directory of Tortoise SVN (i.e., instead of the “\trunk” directory). A branch allows developers to make changes to the source code for their own mods without changing the source code of JA2 1.13, so individual projects can customize the branches as they see fit.

The branches on Tortoise SVN serve two purposes. First, they provide modders with a convenient place to store the changes they make to JA2 1.13. Modders can also employ the Tortoise SVN facilities to obtain different versions of JA2 1.13 to merge with their code. Second, the branches provide modders with an awareness of other mods of JA2 1.13, as every branch is a separate mod. Modders have the capability to download and compare others' mods with JA2 1.13 to determine what other modders have done.

Table 5. Coordination Practices of EDB

Practices	Interactions	Artifacts	Actors
Awareness Building	Make and read posts on Bear's Pit.	Forum	EDB JA2 1.13 Code mod in JA2 1.13 Code mod outside JA2 1.13
	View, download, and examine other mods on versioning system	Versioning System	EDB JA2 1.13 Code mod in JA2 1.13 Code mod outside JA2 1.13
Merging	Download a version of JA2 1.13 and incorporate mod with that source code	Versioning System Development Artifact Forum	EDB JA2 1.13 Code mod in JA2 1.13
	Integrate two mods	Development Artifact Both mods	EDB JA2 1.13 Many Mercs
Cascading	Choose which version of JA2 1.13 to download	Development Artifact	EDB JA2 1.13 Code mod in JA2 1.13
	Integrate with JA2 1.13	Versioning system	EDB JA2 1.13 Code mod in JA2 1.13 Code mod outside JA2 1.13

Merging. With a code mod like EDB, the executable binary only contains the focal code mod and the JA2 1.13 source code the mod was developed on. Two code mods developed from the same source code cannot work together, because only one executable binary can be run at a time. Thus, only code mods in JA2 1.13 can coordinate with a focal code mod project. The below person is complaining that EDB cannot work with Many Mercs.

The problem with all those great mods is, that each of them requires a specific .exe, so there's some sort of regulation which one to choose (see many mercs and EDB!). (16 October 2008)

Therefore, code mods employ two principal practices to coordinate with each other. First, a code mod coordinates by selecting a version of JA2 1.13 to build from. Building from a version of JA2 1.13 means the mod will work with other mods incorporated into JA2 1.13. Typically, these mods are older, tested mods.

Second, a code mod can integrate with a code mod not integrated in JA2 1.13 by downloading the other code mod, identifying the changes made to JA2 1.13, and incorporating those changes into the focal code mod. The below quote describes an early version of the Many Mercs mod that incorporated EDB. The quote illustrates the difficulty in performing this form of coordination, as there is often a delay when a focal code mod incorporates another. In the meantime, the other mod will have been further developed.

That was the one where I had put in the EDB stuff but never quite caught up to the latest that Headrock had done. (20 November 2008)

Cascade. The JA2 1.13 artifact changes at a rapid pace. Even as each code mod is being modified, tested, and released, the JA2 1.13 project is concurrently developing and extending the core code. Therefore, each code mod project performs an iterative process of catch up with JA2 1.13. Usually, each code mod project marks all changes made to the JA2 1.13 version used to easily identify

changes made to the code. To catch up, the project team downloads the latest version of JA2 1.13 and then duplicates their changes to it. This is followed by another round of release and testing. This process of chasing JA2 1.13 can be challenging, as JA2 1.13 is updated very frequently. To illustrate, we observed 137 separate releases of the JA2 1.13 binary between April and early June 2008. Thus, close cross-project coordination is required to keep projects apprised and updated of changes.

Coordination between EDB, JA2 1.13, and the other assorted code mods requires considerable effort. EDB has to integrate the new version of JA2 from the trunk into the branch, test it, and maintain a discussion on the forums. These problems of catching up with JA2 1.13 and identifying bug responsibility can be addressed by having a code mod officially considered part of JA2 1.13. A code mod incorporated into JA2 1.13 automatically becomes part of the base code that other code mod projects have to coordinate with. Such a code mod is migrated from a branch to the trunk, and other code mods automatically download the code mod when they update from Tortoise SVN (Mockus et al., 2002).

To achieve this major state, the JA2 1.13 development team must first accept the mod. Our data suggests no formal governance process to do this. For example, some (frequently small) mods and bug fixes have been incorporated because one member of the JA2 1.13 development team elects to incorporate the mod. However, certain actions can be taken by code mod developers to increase this probability.

First, the code mod project has to be reasonably popular with users, i.e., it must have been played by a number of people. This not only tells the JA2 1.13 team that the mod would generally benefit JA2 1.13, but more importantly, it provides a large user base to identify bugs or unexpected behavior associated with the mod.

Second, certain characteristics of the mod could accelerate the inclusion process. One characteristic is the mod is a single “standalone” feature. This makes it easier for the JA2 1.13 project team to determine what changes were made to the code (see quote below). For example, the New Inventory mod focused entirely on load bearing equipment, while the EDB mod focused on adding details to item descriptions.

The more features you lump together, the less the chance of inclusion into basic JA2 1.13 will be. We do take single features, but not "packages". (27 September 2008)

Another characteristic is that the mod should have the option to be disabled so users that dislike the mod can return to original JA2 behavior. This rule mainly applies to mods by external parties. For example, while there are facilities to turn the EDB, WeatherEffects, New Inventory, and 100 AP mods (see Table 1) on or off, no similar facility exists to merge the cost of aiming and firing a gun in JA2 1.13. In the original JA2, these were a single action.

Apart from these necessary conditions, each code mod may need to fulfill other requirements. In the case of EDB, one key barrier to integration was between its design and one of JA2's original features, the old inventory system. EDB produced a large description box that drew itself on top of the larger inventory panel designed for the New Inventory System, and would not fit in the old inventory panel. When players selected the old inventory system, the original description box would be displayed. The JA2 1.13 developers wanted EDB to work with old inventory.

It means that the majority of those of the JA2 1.13-team, who answered to my question, want the EDB with the Old Inventory, too, not only with the New Inventory, before it would be incorporated into main JA2 1.13. (5 October 2008)

Additional effort and collaboration was required so cross-project coordination through inclusion in the JA2 1.13 code could be achieved. Specifically, a member of the JA2 1.13 team helped add old inventory features into EDB. EDB with old inventory support was released on 29 October 2008, and subsequently incorporated into JA2 1.13.

Once a code mod is incorporated into JA2 1.13, the coordination practices for the focal code mod project are significantly altered. The code mod is treated as being a part of JA2 1.13. New versions of the code mod are released with the new JA2 1.13, and the code mod is incorporated into the SVN trunk. The JA2 1.13 team works directly with the code mod developer, and can elect to work directly on the code mod.

Also, the developer of the included code mod need not expend effort to coordinate with other code mod projects. Instead, other code mod projects download the included code mod along with the rest of the JA2 1.13 code.

4.3 Case 2: Data Mod – PossumDBB

Project Interests: PossumDBB is a mod of “DBB Mod,” which itself is a mod of JA2 1.13. DBB Mod, is maintained by DBoy, a mainland Chinese modder who does not often frequent Bear’s Pit. DBB adds about a thousand items to JA2 1.13. Most of the added equipment is associated with former Communist countries. The PossumDBB project takes these additional items as a base, and adds/changes/removes some items. For example, protective armors introduced by DBB have their protection values scaled down, and PossumDBB also employs a different inventory pocket format from the NIV mod project (included in JA2 1.13).

Practices: Table 6 highlights the coordination practices of PossumDBB.

Table 6. Coordination Practices of PossumDBB			
Practices	Interactions	Artifacts	Actors
Awareness Building	Make and read posts on Bear’s Pit	Forum	PossumDBB JA2 1.13 Other mods
	Boundary spanning	Forum/other forums	PossumDBB DBB JA2 1.13
Merging	Performed manually by players	Development Artifact	PossumDBB JA2 1.13
	Other mods developed to merge several data mods	Development Artifact	PossumDBB Other data mods
Cascading	Work around bugs in JA2 1.13	Development Artifact	PossumDBB JA2 1.13
	Update when DBB changes	Development Artifact	PossumDBB DBB

Awareness Building. Almost all awareness building for PossumDBB occurs on forums such as the Bear’s Pit. Old Possum, the developer of PossumDBB, frequently monitors and posts to the Bear’s Pit, and so is regularly apprised of new developments in various JA2 projects. However, development work on the DBB project is performed on a forum outside the Bear’s Pit. Coordination work between DBB and JA2 1.13, and between the DBB and PossumDBB projects, is enabled by “boundary spanners” who frequent both forums.

umm..i [sp] said i [sp] posted it for Dboy² ..he dosnt [sp] come to bear's pit..lol..i'm [sp] a member of his home site in china..i'm [sp] their token whitey means i [sp] copied and pasted your c omments to his site (11 August 2007)

Besides being an information conduit, these boundary spanners serve as negotiation brokers to help data mod projects develop standards. The quote below explains how the JA2 1.13 and DBB development teams agreed to partial out the number space for inventory items in the game. Note how

² DBoy is the lead developer for the DBB data mod project.

unidirectional the coordination is in the quote. DBB had to adjust its number space to accommodate potential growth in JA2 1.13.

At the moment the highest item number in the basic vJA2 1.13 items.xml is 1030, so the DBB-modmakers shifted their itemnumbers a bit to make room in case I have to add items after 1030 (25 January 2007).

Thus, coordination dynamics for a data mod project involve constant attentiveness to other data mod project developments. This may involve direct or indirect sources of information. Direct sources are those obtained by the developer. These include the Bear's Pit forum the developer frequents, and the developer's attempts to playtest data mod combinations. Indirect sources include users of a data mod and boundary spanners who keep the focal data mod project apprised of changes. In the below quote, a boundary spanner is alerting Old Possum that now would be the best time to revise his mod.

might be a good time to release a new PossumDBB-mod.. seems this is last version of dbb-mod for awhile...907..so you can git to the meat of it without worrying bout it suddenly becoming obsolete (28 January 2009)

PossumDBB does not use Tortoise SVN for awareness building or in any meaningful way. This is a common *non-practice* among data mods. Of the 39 branches of the JA2 1.13 Tortoise SVN server, only one belongs to a data mod, that of WildFire JA2 1.13. One reason is that the material data mods work with can be obtained directly from the distribution copy of JA2 1.13, which comes with XML files, but not source code.

Merging. In most cases, players of data mods perform the merging of the mod. The player downloads the JA2 1.13 artifact and applies data mods on top of it. The quote below illustrates how complicated this process can sometimes be. The user is installing JA2 1.13 version 2085, which is an official release, and layering on version 2124, a potentially buggy, but later version. The user then installs PossumDBB v. 903, and then a patch for 903.

I took a clean install JA2 1.12, 1.13 2085, exchanged exe with 2124, then applied your 903, then 3a patch. Is this correct? (12 August 2008)

The process becomes more prone to error when data mods are combined. In many cases, data mods modify the same files, but no artifact exists to accurately compare and merge changes in XML files across data mods. Changes in data mod projects are, therefore, coordinated via close communication and interactions using the forum. However, such coordination is prone to breakdowns that create incompatibility among the data mod projects. For example, the PossumDBB project is incompatible with the WildFire item mod and WildFire Merc mod projects due to breakdowns in coordination.

I have just found the problem. "MercStartingGear.xml" in WF Merc³ mod has conflict with Dbb items. and the problem when you are trying to debug my save is that you are not using the dbb mod. (10 July 2008)

Various mods provide stopgap data merging artifacts to work around this problem. For example, there are several "one click installers" and a combo mod loader that merge various data mods. All these mods are directed principally at players rather than data modders – the stopgap merges are manually performed by a modder and then distributed as a separate mod. These stopgaps also fail to resolve all the issues.

These combinations are possible using the mods above: HAM + PossumDBB = working properly... HAM + WildFire + PossumDBB = likewise + starting inventory quirks... WildFire + PossumDBB = starting inventory quirks. Due to the volume of work required, I could only do very limited testing using these combinations (28 July 2009)

³ WF Merc is the "WildFire Merc" mod that assigns particular pieces of equipment to mercenaries.

Cascading. One determinant shaping cross-project coordination behavior is that data mods only change data and not code. As such, an inclusion of a data mod automatically turns features incorporated in the data mod on. Features cannot be easily turned off. The quote below is a complaint about the thousands of guns found in DBB.

Question: IMO there's way too many weapons in the mod... and i [sp] basically want to remove all of the guns that I've never heard of from the game

Response: To remove a weapon from the game... go into items.xml, find the weapon, and set its to 0. This will remove it from all shops and make it unselectable by enemies. Of course, if the weapon is placed on any of the maps themselves (or on soldiers that are predetermined in maps), it will still be there. (13 July 2006)

Because data mods cannot be easily disabled, they violate the JA2 1.13 inclusion policy (see code mod case above). Therefore, data mod projects are not integrated into the JA2 1.13 base code. These constraints mean that data mod projects have to continually coordinate with the highly dynamic JA2 1.13 to stay current.

Data mod projects are also highly vulnerable to problems created by the binary that reads the data mod. The projects handle bugs in the binary by developing workarounds. Such workarounds require data mod projects to be up to date with JA2 1.13 changes. When bugs in JA2 1.13 are fixed, the workarounds must be undone. Data mod projects, therefore, must coordinate closely with JA2 1.13 to monitor and update these bug fixes. For example, Old Possum developed a workaround to deal with bugs associated with allocating equipment to particular in-game characters.

Question: Concerning Ira, Dimitri, and Carlos, I am at a lose [sp] as to why they still won't show up with their rifles

Reply: Yes that is a strange bug and it is a bug with JA2 1.13. (13 August 2008)

PossumDBB must address two development artifacts, one being JA2 1.13, the other being DBB. Each time DBB is updated, PossumDBB must be updated as well. A failure to update PossumDBB renders it less desirable, because it becomes incompatible with DBB.

Notable changes(in addition to what I've wrote about previously) are:

The addition of the DBB905 guns,

The national uniforms that were LBE are now armors (A DBB905 change) (18 Aug 2008)

4.4 Case 3: Game Mod – Renegade Republic

Project Interests: Renegade Republic (RR) is developed mainly to explore the ability of the JA2 1.13 engine to develop new game mods. In contrast to the original game's premise, RR is set in a fictional breakaway country in the Caucasus Mountains. As a mod that creates a new game using JA2 1.13 as a basis, RR is fundamentally a data mod. The RR mod changes the game's XML files, characters (including portraits and voices), and maps. However, game mods like RR do not attempt to work within the rules of JA2 1.13, but push JA2 in ways the JA2 developers did not envision. As such, game mods are more vulnerable to code changes than data mods. Thus, while RR makes few changes to the JA2 game engine, RR issues its own executable files. The RR executables are based on older versions of JA2 1.13.

Practices: Table 7 highlights coordination practices of RR.

Table 7. Coordination Practices of Renegade Republic

Practices	Interactions	Artifacts	Actors
Awareness Building	Make and read posts on Bear's Pit	Forum	Renegade Republic JA2 1.13 Other mods
Merging	Code mod merging	Development Artifact	Renegade Republic JA2 1.13 Code mods in JA2 1.13
	Data mod merging	Development artifact	Renegade Republic DBB Other data mods
Cascading	Updating with JA2 1.13 (failure)	Development Artifact	Renegade Republic JA2 1.13

Awareness Building. The RR mod relies on the forum to build awareness and to be apprised of updates of JA2 1.13 and other mods. The *_scorpion*, the key developer in the RR project, is a frequent participant and poster on Bear's Pit, and is regularly apprised of changes in other mods and apprises others of changes to RR.

As a game mod that relies on the JA2 1.13 game engine to function, the installation of RR is more complex than with a typical game. For RR to work, JA2 1.13 must first be installed and then overwritten by RR-modified files. Much of the awareness building on the forums is associated with helping with difficulties associated with these tasks.

I have ja2 + 1.13 installed, extract the renegade "rr" folder into the ja2 folder then dbl click exe and receive a message "cannot start game as fmod.dll file is missing". (04 January 2008)

Merging. Game mods like RR have features of both code and data mods. Like code mods, RR requires a RR-specific binary executable to run. However, what makes RR fun to play is the new setting of RR (Caucasus mountains instead of a South American island), characters, and storyline. All of these are chiefly architected by modifying data.

While RR requires a separate binary executable, like most other game mods, it does not have a branch on Tortoise SVN. When changes to the RR executable are necessary, *The_scorpion*, who was one of the original supporters of creating a mod of JA2 1.13,⁴ uses the forum to leverage his considerable social capital in the JA2 1.13 community to get assistance from JA2 1.13 developers to modify the existing JA2 1.13 executable.

It REALLY is as simple as modifying a single cpp file and can be done in a matter 15-20 minutes. Send me a list of INI options you want disabled, or what you want their values to be and BAM, it's done. (28 August 2006)

The RR project has numerous incentives to incorporate code mods into RR. Most of these code mods incorporate features that enhance RR without substantively impacting the RR storyline. Notably, since the initial release of RR, at least six gameplay enhancing code mods have been developed. These include New Inventory (introduction of load-bearing equipment), EDB, 100AP (finer grained modeling of actions), Many Mercs (able to have more individuals on a map), Spreadpatterns (better shotgun modeling), and HeadRock's Assorted Modifications (HAM – enhanced modeling of suppressive fire. Suppressive fire occurs where one applies a high volume of bullets or artillery fire to psychologically make an enemy less effective).

Is there any new RR version working with 100AP 1.13 exe? (13 March 2009)

⁴ Interview with *The_scorpion* in <http://www.tacticalarcancer.com/content.php?id=43>

The way RR incorporates these mods is to take a version of JA2 1.13 that contains them, make the small modifications to create the executable RR-compatible, and then release a new RR executable. However, because RR “hacks around” existing rules in the code, it is highly vulnerable to bugs introduced in JA2 1.13. Isolating these bugs is challenging, as they often do not affect JA2 1.13, but affect RR. The bug described below causes a hospital in RR to explode when a civilian walks in the wrong place. It does not affect JA2 1.13, because there does not exist a similar scenario in JA2 1.13.

yeah, that's the same bug. Prior to some whack code change in JA2 1.13, civ's couldn't activate such pressure action items. (12 April 2008)

In many cases, these bugs in JA2 1.13 render RR unplayable. In the above example, it is impossible to secure the hospital, since the player cannot control civilian movement, and an explosion can, therefore, occur at any time. Because such bugs can arise, the RR executable is not updated when JA2 1.13 is updated. The last update to the RR executable incorporated New Inventory, but none of the other six mods.

RR coordinates with data mods in much the same way JA2 1.13 coordinates with data mods. JA2 1.13 is installed first, RR is installed on top, and data mods are then layered on top of RR. However, because RR itself modifies data, the introduction of data mods on top of RR can cause some of RR's functionality to disappear. As an example, one can replace the equipment found in RR with the equipment in DBB. This causes all equipment in RR to be overwritten and replaced by DBB equipment.

Cascade. Initially, the RR project coordinated closely with JA2 1.13. Changes to the JA2 1.13 binary were reflected as changes to the RR binary. When bugs were introduced to the JA2 1.13 binary that made RR (but not JA2 1.13) unplayable, RR began updating the executable more circumspectly.

The_scorpion attempted to alert JA2 1.13 developers to the problems with the JA2 engine. However, as earlier mentioned, the JA2 1.13 executable is constantly updated. Generally, JA2 developers prefer to only address bugs in the current version. As a result, it is difficult for The_scorpion to get his issues addressed.

I keep hearing, "just tell us what the problem is" then I spend 5 posts trying to explain what the problem is only to learn that no there has been yet another update I 'should' have been referencing before posting a problem. I mean an every few months moving target is possible to mod. Every few days...not so much. (2 June 2008)

The RR project eventually abandoned all attempts at coordinating with JA2 1.13. Currently, an open beta version of RR is available for play, but randomness in certain events (e.g., interactions with characters in the game), and sudden crashes introduce a certain level of frustration to the gameplay, as the quote below shows.

After a month of playing, I think that I've finished this mod, but I run into a problem. After killing the primary target (olygarch), cutscene outside the palace appears. I listen to the counsel's speech and than the game locks. There is just a stopwatch icon. Is this the end?. (09 October 2008)

At the present time, these issues between RR and JA2 1.13, as well as the internal RR project challenges, are unlikely to be resolved. For one, the JA2 1.13 team has not shown any interest in modifying the JA2 1.13 source to address issues in RR. For another, The_scorpion publicly halted further development on RR, citing flaws in the executable that made further development impossible.

Further work on RR has been postponed, because JA2 1.13 is not developed far enough (read: not moddable enough) at this time. Some important things have still not been externalized in JA2 1.13, so Scorpion might be waiting, if he has not given up already (12 January 2009)

5 Discussion

In this research, we studied how coordination across projects occurs in an open source game modding community. We presented an overview of cross-project coordination processes using three representative projects (EDB, PossumDBB, and RR) to illustrate the emergence and evolution of coordination practices.

The cross-project environment studied involves a central project (JA2 1.13) and a host of other projects working either in tandem or off this central project. As shown in our case studies, all three types of project have strong motivations for cross-project coordination. The main interdependencies across projects are to ensure that each project's output remains compatible with others and that other projects' actions do not create instability in the focal project artifact (Staudenmayer, 1997).

Our analysis suggests three main insights discussed below. The remainder of this section synthesizes the evidence leading to each insight.

5.1 Materiality and Ordering System of Cross-Project Coordination

We found the JA2 artifact was especially critical in shaping coordination interactions in the JA2 community. For example, bugs in the JA2 artifact impacted all three mods. It was easier for EDB to coordinate with other code mods in JA2 than with independent code mods. It was also easier for PossumDBB to coordinate the development of equipment with JA2 than with other equipment data mods, because DBB/PossumDBB modified XML files originating with the JA2 distribution.

Different characteristics of the JA2 artifact were especially significant in shaping the modders' use of various tool artifacts for coordination. The main material characteristic was the different aspects of the JA2 artifact, i.e., code, data, or game. A different ordering system emerged for each aspect. This was mainly due to the way the JA2 artifact was designed. JA2 is written in C/C++ rather than a scripting language, and thus, must be compiled, so the code is available separately from the distributed game via the versioning system. To develop a code mod, each project must interact directly with the versioning system. On the other hand, data for JA2 is distributed with the game as (primarily) XML files. Thus, a data or game mod developer can interact sporadically with the versioning system.

Building from that architectural difference in the JA2 artifact, code mods' ordering system naturally revolved around the Tortoise SVN versioning system, as evidenced by the heavy participation by code mods. However, the data and game mods did not participate much in the SVN system and instead relied heavily on forum-based coordination practices such as boundary spanning (Barcellini et al., 2008, Levina and Vaast, 2005, Orlikowski, 2002, Pawlowski and Robey, 2004, Tushman, 1977) that were unique to those mods. The game mod also relied on leveraged social capital to address some of the coordination issues the versioning system would have addressed.

Also, the desire to move to more routine coordination practices made it difficult for code mods to have boundary spanners. The practice that afforded routine coordination was for the code mod to be integrated with JA2 1.13. Integration required that the JA2 1.13 team have an intimate awareness of the characteristics of the code mod. Since only the code mod developer had such awareness and boundary spanners did not, it was necessary for code mod developers to maintain a presence on the Bear's Pit. In contrast, data and game mods could never be integrated with JA2 1.13, and developers of such mods, therefore, had the ability to perform their work in other forums, thereby necessitating boundary spanners to facilitate coordination.

Thus, the key artifact that shapes cross-project coordination interactions in this community is the development artifact, JA2, itself. The development artifact not only directly provides a framework for particular coordination interactions, but it also indirectly influences which tool artifacts a modder employs to coordinate, thereby shaping the emergent ordering system for cross-project coordination. Hence:

- P1a. Materiality of the development artifact shapes cross-project coordination practices, which, in turn, influence the emergent ordering system.*
- P1b. The choice of tool artifacts employed to facilitate coordination is determined, in part, by the materiality of the development artifact.*

This is not to say that tool artifacts do not influence coordination interactions. Indeed, their role was demonstrated both in this paper, and in others (Baldwin and Clark, 2006, Egyedi and de Joode, 2004, Feller et al., 2008, Iannacci, 2005, Jørgensen, 2001). For example, Microsoft's release of the C compiler into the public domain enabled the coordinative practices that eventually led to JA2 1.13. However, when the problem of coordination moves from the within-project to cross-project context, the development artifact becomes salient. Within a project, there is less distinction between the development and project artifact, as the same project team employs both. When one coordinates across projects, one must now consider that the artifact one is developing (i.e., the project artifact) must coordinate with a changing artifact that one is using (i.e., the development artifact), where changes are performed by individuals outside of one's control.

The fact that development artifacts shape coordination interactions leads us to wonder what specific material characteristics of development artifacts shape such interactions. In other words, can we deepen our understanding of the process by which actors, activities, and artifacts are intimately intertwined and, thereby, surface the theoretical role of material artifacts in the work of organizing (Orlikowski, 2010, Orlikowski and Scott, 2008)? While we leave the answer to this question for future research, a partial answer is that in many cases, development artifacts are designed to facilitate coordination. For example, many open source (and traditional) software development projects are designed around modular code architectures (Baldwin and Clark, 2006, Iannacci, 2005). In JA2, the conversion of data from a proprietary to XML-based format similarly facilitates coordination. Definitely, coordination features missing from a development artifact can hinder coordination. The lack of a mechanism in JA2 to merge data from separate XML files, for example, makes coordination among data mods so difficult that some data mods exist purely to address this problem.

Our findings also highlight that for FLOSS, the materiality of the development artifact extends beyond the code. While almost all work on FLOSS projects focuses on coordination among programmers and developers (Crowston et al., 2007, German, 2003, Mockus et al., 2002), our research highlights the importance of coordination with non-developers like artists and subject matter experts in cross-project coordination. These non-programmers focus less on code and more on data. For non-programmers then, a critical factor influencing cross-project coordination patterns is the extent of openness in the development artifact's data. Openness does not just refer to whether the format is accessible to non-programmers, but also the extent to which data across projects can be easily integrated. Hence:

- P1c. Ease of cross-project coordination, especially among non-programmers, is influenced by the extent of openness in the development artifact's data.*

5.2 Emergent trajectories of Ordering System

Our second insight also stems from the materiality of artifacts and speaks to the temporal, emergent aspect of cross-project coordination, or its "trajectory." This is another key contribution, as most studies on coordination study snap-shots of coordination practices – few focus on how coordination evolves over time. Our study shows that material aspects of the development artifact in relation to a set of community norms afford different trajectories of ordering systems, as depicted in Figure 1 below (Markus and Silver, 2008). Specifically, in the code mod project case, we found the ordering system evolved from emergent coordinative practices – where projects manage changes not only in the focal mod, but in mods with which they must coordinate – to coordination that is managed solely within the JA2 1.13 project via the JA2 artifact. This coordination practice is more routine, as it focused only on JA2 1.13 project developments without having to manage cross-project issues emerging from other related projects. In the data mod project, the same emergent ordering system persisted over time. Coordination with the game mod project ended abruptly.

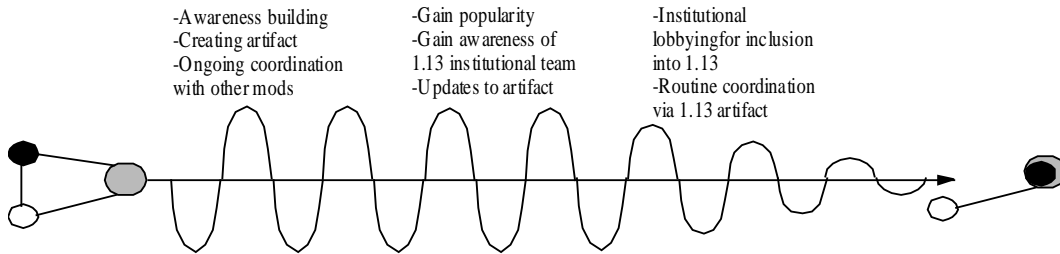


Figure 1a. Routinization of coordination trajectory (code mod case)

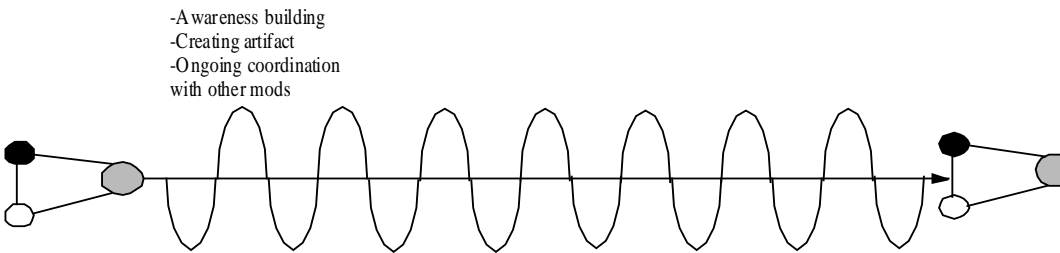


Figure 1b. Ongoing unilateral coordination trajectory (data mod case)

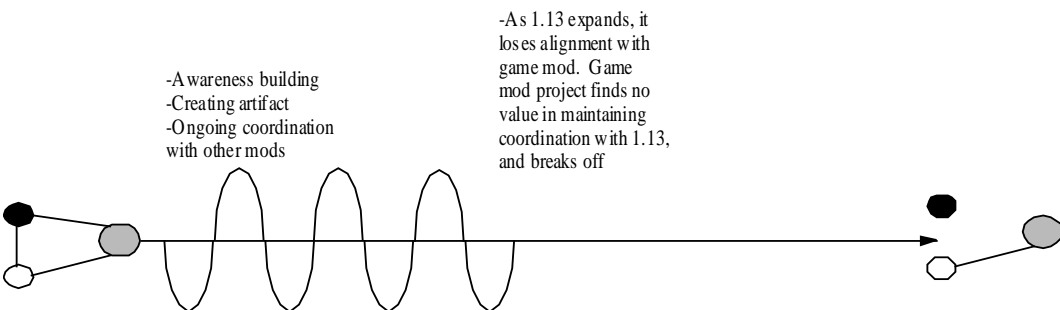



Figure 1c. Termination of coordination (game mod case)

Figure 1. Ordering System Trajectories of Different Kinds of Mods

Legend

Symbol	Description
●	Focal project team e.g., EDB mod team, PossumDBB, or RR mod
●	JA2 1.13 Project Team
○	Other mod projects e.g., Wildfire
—	Coordination links
	Range of coordination activities

With regard to code mod projects, two characteristics of the project artifact in relation to project norms and goals shaped the nature of the ordering system from one that is emergent to one that is more routine. First, in code mod projects, it is possible to physically limit the project to a “standalone” feature. Second, code mod projects can be designed so their features are optional, i.e., users can have a choice whether to turn mod options on or off.

These two characteristics dovetailed with the core project team’s community norms for including code mods into its core project: ease of isolation for testing and bug analysis and features that are compatible with other mods. Code mod projects that follow these community norms can, therefore, be considered for inclusion into the core JA2 1.13 project.

As shown in the EDB project case, once the code mod project is included into JA2 1.13, the ordering system changes from emergent practices to one based mainly on the core 1.13 code. The onus is on other code mod developers making the changes to JA2 1.13 to ensure all changes work with the core 1.13 project.

In contrast, data and game mods did not include either of the material characteristics of “standalone” or “optional” and have no analogous method as version control software to merge XML file changes. Hence, no integration process emerged in their ordering systems to resolve cross-project coordination tensions. Thus, the trajectory of their ordering systems was either a persistence of emergent coordinative practices, or a termination of coordination.

Contrasting the trajectories, one interesting finding is the link between emergent and routine task coordination. Prior research has either examined emergent coordination practices in dynamic situations (Faraj and Sproull, 2000, Faraj and Xiao, 2006, Kellogg et al., 2006, Majchrzak et al., 2007, Schmidt and Simone, 1996) or task-related coordination in stable environments (Argote, 1982, Crowston et al., 2005, Malone and Crowston, 1994, van de Ven et al., 1976). The EDB case potentially demonstrates a pattern where cross-project coordination moves from emergent coordination practices to “routine” coordination as a strategy to mitigate the high level of tension and resources required in emergent coordination (Hoegl et al., 2004).

One question is why this trajectory from emergent to routine coordination only occurred in the code mod, but not the data or game mod cases. The answer obtained from our analysis is that the coordination trajectory was influenced by the affordance emerging from the interaction of two factors: the goals and desires of the project team building the development artifact, and the materiality of the development artifact (Hutchby, 2001, Markus and Silver, 2008, Zammuto et al., 2007).

Interests of the Project Team. JA2 1.13 prospers because it is a game that a large number of JA2 aficionados want to play. The JA2 1.13 team, therefore, has an incentive to integrate JA2 1.13 with popular mods. However, it is not in the JA2 1.13 team’s interest to incorporate a mod that would alienate a significant portion of their player base. Hence, the team enacts a rule that for a mod to be integrated into JA2 1.13, the mod must have the ability to be disabled. This rule strikes a happy compromise between the desires to integrate popular mods and to avoid alienating players who do not wish to play the mod.

There are certain kinds of mods that cannot be disabled—data and game mods. However, given that these mods can be played by overlaying them on top of JA2 1.13, this does not compromise the interests of the JA2 1.13 development team. It compromises the interests of the mod development teams, because in comparison with mods integrated into JA2 1.13, they are unable to coordinate as effectively with other mods.

Materiality of the Development Artifact. The reason these mods cannot fulfill the JA2 1.13 team’s rule is the development artifact does not have functionalities that give rise to such affordances. This constrains the cross-project dynamics (Markus and Silver, 2008, Zammuto et al., 2007). For data mods, the XML files in JA2 1.13 are not segmented so that each data or game mod writes to separate files that can be turned on or off. The JA2 1.13 development artifact, furthermore, is a game, not a game engine. It is designed for a particular scenario, and cannot “switch” between one game and another.⁵ Hence:

P2. Coordination practices are more likely to evolve from emergent to routine practices when features of the artifact in relation to the core project team’s goals and desires afford this trajectory of change.

⁵ Shortly after the initial draft of this paper was written, a new mod called JA2 1.14 Stable Modding Platform was launched to convert JA2 into a game engine. That the community felt the need to create a game engine validates some of the findings here.

5.3 Asymmetrical Interdependence and Unilateral Coordination

The final insight emerging from this research concerns “unilateral coordination.” Prior literature assumes that coordination is a cooperative action involving at least two groups or actors (Faraj and Sproull, 2000, Faraj and Xiao, 2006, Kellogg et al., 2006, Majchrzak et al., 2007). However, coordination does not necessarily presuppose two active partners, given that coordination practice only speaks to the situated social activities involved in the process of managing interactions due to interdependencies between two or more parties (Faraj and Xiao, 2006, Malone and Crowston, 1994). Our study contributes to this gap in the coordination literature to show that coordination can occur when only one party is actively managing the interactions between itself and other groups, i.e., in an asymmetric coordination situation. In the EDB case, for the most part, it was the EDB project that actively coordinated with JA2 1.13. EDB adhered to JA2 1.13’s rules, and regularly updated itself from the SVN. It was only at the final stage, when a member of the JA2 1.13 team offered to help make EDB compatible with the old inventory that there was some form of reciprocity. Similarly, PossumDBB bore all the responsibility for coordinating its project with both DBB and JA2 1.13. Modifications in both JA2 1.13 and DBB were incorporated into PossumDBB. On the other hand, modifications raised by PossumDBB were not addressed by the other projects. Finally, with RR, a member of the JA2 1.13 team helped change the binary’s configuration settings. However, no assistance was provided in identifying or fixing bugs in JA2 1.13 that impacted RR, with the result that all coordination stopped.

In all three cases, the mods coordinating with JA2 1.13 were not entrenched within the community (Staudenmayer, 1997). The lack of embeddedness of those three mods should be contrasted with JA2 1.13 in 2005. In those early days, JA2 1.13 was not the dominant mod. During that time, there were more equal efforts at coordination between JA2 1.13 and other projects like the 800 by 600 mod, where, for example, a JA2 1.13 developer compared the 800 by 600 mod to JA2 1.12, and copied all the changed lines to JA2 1.13. Essentially, in the early stages of a development artifact, no group was dominant, and cross-project coordination efforts were distributed equally among the groups. Over time, as the development artifact stabilized, responsibility for cross-project coordination shifted from an equal distribution among projects to one where relatively new projects bore the brunt. Hence:

P3a. As development artifacts become more stable, the potential for cross-project coordination efforts to become unilateral among projects increases.

P3b. Newer projects focused on a development artifact will potentially bear a greater amount of the responsibility for cross-project coordination.

In the RR project, our findings also demonstrate that in certain situations, unilateral coordination is difficult to sustain in the long run. The RR project’s attempt to change the rules of JA2 1.13 while simultaneously maintaining a level of compatibility with the original game created a high level of interdependence between the two projects that could not be sustained by unilateral coordination.

6. Conclusion

This research is one of the first attempts at trying to understand the problem of coordination across projects in the FLOSS environment. We employed the Ordering Systems Lens, a longitudinal and context-based practice lens that highlights the use of artifacts to perform in-depth case studies of four projects in an open source computer game modding community to understand our research question. Our study provides three salient take-aways.

- **The development artifact shapes and influences cross-project coordination practices.** Given the sensitivity of the ordering systems lens to artifacts, and the specific work context of the field site, our research highlighted the development artifact as particularly critical. Within a project, the only artifact that changes is the one on which the project team works. Artifacts that are inputs to a project (tool artifacts) like compilers, scheduling and project management systems, and email remain relatively constant. Hence, the impact of artifacts on coordination practices in a single project context is less visible. In cross-project coordination, one of the tool artifacts a project employs is the project of another team (development artifact). Changes

that other teams make to the development artifact influence how the focal team works. Thus, coordination practices in cross-project development change over time, because the development artifact is usually dynamic.

- **Coordination practices evolve from emergent to routine coordination when such affordances emerge through the interaction between the materiality of the development artifact and the goals and desires of the project team building the development artifact.** The Ordering Systems Lens also sensitizes the researcher to evolutions of practices over time as a result of actor goals and desires. All project teams managing development artifacts desire to promote their development artifact for others to use. However, each project team promotes its development artifact in a different way. For JA2 1.13, the development team desires to include new features in the development artifact that do not interfere with or compromise existing features. Artifacts, in turn, have features that provide the framework for action – e.g., they allow project groups to do certain things, but do not allow them to do others. When the artifact's features afford specific development team goals and desires, coordination can evolve from emergent to routine practices.
- **Unilateral coordination exists and becomes more common as the development artifact matures.** This was a second longitudinal finding that emerged from our use of the Ordering Systems Lens. While prior literature has demonstrated that coordination requires at least two stakeholders, little work has examined the issue of whether stakeholders share an equal burden in maintaining coordination. Our research highlights that a single stakeholder can bear the burden of coordinating with other parties. Furthermore, unilateral coordination appears more likely to occur as a development artifact stabilizes and matures. New projects attempting to coordinate with stable, mature projects tend to bear the burden of coordination.

Cross-project coordination is an especially relevant phenomenon given the parallel nature of work in open-source environments, where work is typically separated into several related projects. Most FLOSS artifacts have geographically distributed members with a desire to develop their own distinct enhancements to the FLOSS artifact (i.e., projects). These separate projects coordinate via forums and versioning systems, and characteristics of the development artifact are likely to shape how such coordination is conducted. As but one example, many individuals host distinct shape libraries for the FLOSS diagram creator Dia (see for example (<http://dia-installer.de/shapes.html>) and <http://enc.com.au/myscripts/diashapes.html>). The code for Dia, in contrast, is developed and discussed on the Dia Gnome list (<http://mail.gnome.org/mailman/listinfo/dia-list>).

We expect our findings on cross-project coordination to be applicable to these FLOSS projects. For example, the distinction between Dia shapes and code corresponds roughly to the distinction between code and data mods in JA2. Thus, we would expect similar coordination practices to occur in Dia. For example, we would expect to find boundary spanners for shape work, while code work occurs principally on the forum.

Beyond findings relevant to the overall FLOSS community, our study highlights an increasingly common practice in certain FLOSS projects, where the FLOSS artifact is no longer managed by the original developer. Most such projects are associated with computer games. For example, the source code for MechCommander 2 (Microsoft) is in the public domain, while the Temple of Elemental Evil (Atari-bankrupt) has an active modding community. Some FLOSS projects are not computer games. GNU Smalltalk, for example, is not managed by Xerox PARC, the Smalltalk inventor. Similarly, Alexander Larsson, who developed Dia, no longer is involved with its development. Such non-original developer-managed FLOSS environments are likely to have characteristics similar to the Bear's Pit, where separate developers make independent enhancements that must be integrated into the development artifact.

While the coordination trajectories studied are specific to JA2 development and FLOSS projects, our findings are likely to have practical implications for IS project development in general. For example, in

SAP-based systems, there might be coordination differences between core mods, user exits, business reports, and independent work done on Excel output. Core mods that modify the base SAP code may require extra coordination attention when the SAP version is upgraded. A user exit, in contrast, leverages pre-defined points in the SAP structure where one can use another application to process SAP output to be fed back to SAP. User exits are likely more robust to SAP upgrades, because the exit points are more likely to be preserved across upgrades. Thus, the materiality of SAP can potentially shape SAP cross-project coordination both at a point in time, and across time. Furthermore, changes made by SAP in Germany (the focal mature project) will force unilateral coordination on many projects that have no influence on Germany's activities. However, because of differences in the work environment (e.g., more face to face contact in SAP environments), we would expect the specific coordination practices to differ.

Our research represents an early foray into understanding this phenomenon of cross-project coordination, and additional research must be conducted to shed further light on this area. As but one limitation, our work focuses exclusively on cross-project coordination work at a single "site." It is not clear how the idiosyncrasies of that site impact our findings, and as a result, our propositions must be tentative. Further work should be done to explore the nature of cross-project coordination on other projects, FLOSS and otherwise.

Furthermore, our research focused principally on archival data – the projects examined were already mature by the time the research was initiated. We attempted to mitigate this by gathering direct feedback from project participants through presentation of our written case on the forum. However, the perspectives of stakeholders reflecting on a project, and stakeholders who are currently grappling with particular coordination difficulties are likely to differ. Cross-project coordination research would benefit from future work that examines ongoing coordination difficulties.

Finally, this research extends coordination research by acknowledging the role and relevance that materiality of the development artifact plays in cross-project coordination. Our work highlights how both code and data elements of a project shape and afford coordination practices. However, much work remains to theorize further about the issue of "materiality." Our study into how code and data shape coordination practices could perhaps provide the basis to develop further "typologies of material constraints and affordances" (Leonardi and Barley, 2008).

References

- Adler, P. A. and P. Adler (1988) "Intense Loyalty in Organizations: A Case Study of College Athletes," *Administrative Science Quarterly* (33) 3, pp. 401-417.
- Adler, P. S., A. Mandelbaum, V. Nguyen, and E. Schwerer (1995) "From Project to Process Management: An Empirically-Based Framework for Analyzing Product Development Time," *Management Science* (41) 3, pp. 458-484.
- Argote, L. (1982) "Input Uncertainty and Organizational Coordination in Hospital Emergency Units," *Administrative Science Quarterly* (27) 3, pp. 420-434.
- Baldwin, C. Y. and K. B. Clark (2006) "The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model," *Management Science* (52) 7, pp. 1116-1127.
- Barcellini, F., F. Detienne, and J.-M. Burkhardt (2008) "User and Developer Mediation in an Open Source Software Community: Boundary Spanning Through Cross-Participation in Online Discussions," *International Journal of Human-Computer Studies* (66) 7, pp. 558-570.
- Bechky, B. A. (2006) "Gaffers, Gofers, and Grips: Role-Based Coordination in Temporary Organizations " *Organization Science* (17) 1, pp. 3-21.
- Crowston, K. (2008) The Bug Fixing Process in Proprietary and Free/Libre Open Source Software: A Coordination Theory Analysis, in V. Grover and M. L. Markus (Eds.) *Business Process Transformation* Armonk, NY: M.E. Sharpe, pp. 69-100.
- Crowston, K., Q. Li, K. Wei, U. Y. Eseryel et al. (2007) "Self-Organization of Teams for Free/Libre Open Source Software Development," *Information and Software Technology* (49) 6, pp. 664-575.
- Crowston, K. and M. Wade (2009) "Call for papers: Empirical Research on Free/Libre Open Source Software," http://aisel.aisnet.org/jais/jais_floss_callforpapers.pdf.
- Crowston, K., K. Wei, Q. Li, U. Y. Eseryel et al. (2005) Coordination of Free/Libre Open Source Software Development. *Twenty-Sixth International Conference on Information Systems, 2005*, pp. 11-23.
- Cusumano, M. A. and R. W. Selby (1995) *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People*. New York, NY: The Free Press.
- Egyedi, T. M. and R. v. W. de Joode (2004) "Standardization and Other Coordination Mechanisms in Open Source Software," *Journal of IT Standardization Research* (2) 2, pp. 1-17.
- Eisenhardt, K. M. (1989) "Building Theories from Case Study Research," *Academy of Management Review* (14) 4, pp. 532-550.
- Faraj, S. and V. Sambamurthy (2006) "Leadership of Information Systems Development Projects," *IEEE Transactions on Engineering Management* (53) 2, pp. 238-349.
- Faraj, S. and L. Sproull (2000) "Coordinating Expertise in Software Development Teams," *Management Science* (46) 12, pp. 1554-1568.
- Faraj, S. and Y. Xiao (2006) "Coordination in Fast-Response Organizations," *Management Science* (52) 8, pp. 1155-1169.
- Feller, J., P. Finnegan, B. Fitzgerald, and J. Hayes (2008) "From Peer Production to Productization: A Study of Socially Enabled Business Exchanges in Open Source Service Networks," *Information Systems Research* (19) 4, pp. 475-493.
- Fitzgerald, B. (2006) "The Transformation of Open Source Software," *MIS Quarterly* (30) 3, pp. 587-598.
- Galbraith, J. (1973) *Designing Complex Organizations*. Reading, MA: Addison-Wesley.
- German, D. M. (2003) "The GNOME Project: a Case Study of Open Source, Global Software Development," *Software Process: Improvement and Practice* (8) 4, pp. 201-215.
- Glaser, B. G. and A. L. Strauss (1967) *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Chicago: Aldine Publishing.
- Golden-Biddle, K. and K. Locke (1993) "Appealing Work: An Investigation of How Ethnographic Texts Convince," *Organization Science* (4) 4, pp. 595-616.
- Grabher, G. (2004) "Temporary Architectures of Learning: Knowledge Governance in Project Ecologies," *Organization Studies* (25) 9, pp. 1491-1514.
- Hemetsberger, A. and C. Reinhardt (2009) "Collective Development in Open-Source Communities: An

- Activity Theoretical Perspective on Successful Online Collaboration," *Organization Studies* (30) 9, pp. 987-1008.
- Herbsleb, J. D., A. Mockus, T. A. Finholt, and R. E. Grinter. (2000) Distance, Dependencies, and Delay in a Global Collaboration. *Proceedings of the ACM conference on Computer Supported Cooperative Work, Philadelphia, PA, 2000*, pp. 319-328.
- Hine, C. M. (2000) *Virtual Ethnography*. London: Sage Publications.
- Hoegl, M., K. Weinkauff, and H. G. Gemuenden (2004) "Interteam Coordination, Project Commitment, and Teamwork in Multiteam R&D Projects: A Longitudinal Study," *Organization Science* (15) 1, pp. 38-55.
- Hutchby, I. (2001) "Technologies, Texts and Affordances," *Sociology* (35) 2, pp. 441-456.
- Iannacci, F. (2005) "Coordination Processes in Open Source Software Development: The Linux Case Study," *Emergence: Complexity and Organization* (7) 2, pp. 21-31.
- Jørgensen, N. (2001) "Putting It All In The Trunk: Incremental Software Development in the FreeBSD Open Source Project," *Information Systems Journal* (11) 4, pp. 321-336.
- Kellogg, K. C., W. J. Orlikowski, and J. Yates (2006) "Life in the Trading Zone: Structuring Coordination Across Boundaries in Postbureaucratic Organizations," *Organization Science* (17) 1, pp. 22-44.
- Klein, H. K. and M. D. Myers (1999) "A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems," *MIS Quarterly* (23) 1, pp. 67-94.
- Kratzer, J., H. G. Gemuenden, and C. Lettl (2008) "Balancing Creativity and Time Efficiency in Multi-Team R&D Projects: the Alignment of Formal and Informal Networks," *R&D Management* (38) 5, pp. 538-549.
- Lawrence, P. R. and J. W. Lorsch (1967) *Organization and Environment: Managing Differentiation and Integration*. Boston, MA: Harvard Business School Press.
- Lee, A. S. (1989) "A Scientific Methodology for MIS Case Studies," *MIS Quarterly* (13) 1, pp. 33-50.
- Leonardi, P. M. and S. R. Barley (2008) "Materiality and Change: Challenges to Building Better Theory About Technology and Organizing," *Information & Organization* (18) 3, pp. 159-176.
- Levina, N. and E. Vaast (2005) "The Emergence of Boundary Spanning Competence in Practice: Implications for Implementation and Use of Information Systems," *MIS Quarterly* (29) 2, pp. 335-363.
- Li, Y., T. Lie, J. J. Jiang, and G. Klein. (2008) Coordinating Multiple Interdependent Projects in Innovative Product Development Programs. *International Research Workshop on IT Project Management, 2008*.
- Ljungberg, J. (2000) "Open Source Movements as a Model of Organising," *European Journal of Information Systems* (9) 4, pp. 208-216.
- Majchrzak, A., S. L. Jarvenpaa, and A. B. Hollingshead (2007) "Coordinating Expertise Among Emergent Groups Responding to Disasters," *Organization Science* (18) 1, pp. 147-161.
- Malone, T. W. (1987) "Modeling Coordination in Organizations and Markets," *Management Science* (33) 10, pp. 1317-1332.
- Malone, T. W. and K. Crowston (1994) "The Interdisciplinary Study of Coordination," *ACM Computing Surveys* (26) 1, pp. 88-119.
- Malone, T. W., K. Crowston, J. Lee, B. Pentland et al. (1999) "Tools for Inventing Organizations: Toward a Handbook of Organizational Processes," *Management Science* (45) 3, pp. 425-443.
- Markus, M. L. and M. S. Silver (2008) "A Foundation for the Study of IT Effects: A New Look at DeSanctis and Poole's Concepts of Structural Features and Spirit," *Journal of the Association for Information Systems* (9) 10, pp. 609-632.
- Mason, J. (1996) *Qualitative Researching*. Thousand Oaks, CA: Sage Publications.
- McFarlan, F. W. (1981) "Portfolio Approach to Information Systems," *Harvard Business Review* (59) 5, pp. 142-150.
- Miles, M. B. and A. M. Huberman (1994) *Qualitative Data Analysis: An Expanded Sourcebook*. Thousand Oaks, CA: Sage Publications.
- Mockus, A. T., R. O. Y. Fielding, and J. D. Herbsleb (2002) "Two Case Studies of Open Source Software Development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology* (11) 3, pp. 309-346.
- Montealegre, R. and M. Keil (2000) "De-escalating Information Technology Projects: Lessons from the Denver International Airport," *MIS Quarterly* (24) 3, pp. 417-447.

- Orlikowski, W. J. (2002) "Knowing in Practice: Enacting a Collective Capability in Distributed Organizing," *Organization Science* (13) 3, pp. 249-273.
- Orlikowski, W. J. (2010) "The Sociomateriality of Organisational Life: Considering Technology in Management Research," *Cambridge Journal of Economics* (34) 1, pp. 125-141.
- Orlikowski, W. J. and S. V. Scott (2008) "Sociomateriality: Challenging the Separation of Technology, Work and Organization," *Academy of Management Annals* (2) 1, pp. 433-474.
- Pawlowski, S. D. and D. Robey (2004) "Bridging User Organizations: Knowledge Brokering and the Work of Information Technology Professionals," *MIS Quarterly* (28) 4, pp. 645-672.
- Sabbagh, K. (1996) *21st-Century Jet: The Making and Marketing of the Boeing 777*. New York, NY: Scribner.
- Scacchi, W. (2004) "Free and Open Source Development Practices in the Game Community," *IEEE Software* (21) 1, pp. 59-66.
- Scacchi, W. (2007) Free/Open Source Software Development: Recent Research Results and Emerging Opportunities. *6th Joint Meeting on European Software Engineering Conference, Dubrovnik, Croatia, 2007*, pp. 459-468.
- Schatzki, T., K. K. Cetina, and E. Von Savigny (2001) *The Practice Turn in Contemporary Theory*. London: Routledge.
- Schmidt, K. and C. Simone (1996) "Coordination Mechanisms: Towards a Conceptual Foundation of CSCW Design," *Computer Supported Cooperative Work: The Journal of Collaborative Computing* (5) 2/3, pp. 155-200.
- Schmidt, K. and I. Wagner (2004) "Ordering Systems: Coordinative Practices and Artifacts in Architectural Design and Planning," *Computer Supported Cooperative Work: The Journal of Collaborative Computing* (13) 5/6, pp. 349-408.
- Staudenmayer, N. A. (1997) *Managing Multiple Interdependencies in Large Scale Software Development Projects*, Massachusetts Institute of Technology.
- Sydow, J., L. Lindkvist, and R. DeFillippi (2004) "Project-Based Organizations, Embeddedness and Repositories of Knowledge," *Organization Studies* (25) 9, pp. 1475-1488.
- Thompson, J. D. (1967) *Organizations in Action*. New York, NY: McGraw-Hill.
- Tushman, M. L. (1977) "Special Boundary Roles in the Innovation Process," *Administrative Science Quarterly* (22) 4, pp. 587-605.
- van de Ven, A. H., A. L. Delbecq, and R. Koenig (1976) "Determinants of Coordination Modes Within Organizations," *American Sociological Review* (41) 2, pp. 322-338.
- van den Hooff, B. (2004) "Electronic Coordination and Collective Action: Use and Effects of Electronic Calendaring and Scheduling," *Information and Management* (42) 1, pp. 103-114.
- Whitely, R. (2006) "Project-Based Firms: New Organizational Form or Variations on a Theme?," *Industrial and Corporate Change* (15) 1, pp. 77-99.
- Yamauchi, Y., M. Yokozawa, T. Shinohara, and T. Ishida. (2000) Collaboration With Lean Media: How Open-Source Software Succeeds. *Proceedings of the ACM conference on Computer Supported Cooperative Work, Philadelphia, PA, 2000*, pp. 329-338.
- Yin, R. K. (2003) *Case Study Research: Design and Methods*. Beverly Hills, CA: Sage Publications.
- Zammuto, R. F., T. Griffith, A. Majchrzak, D. J. Dougherty et al. (2007) "Information Technology and the Changing Fabric of Organization," *Organization Science* (18) 5, pp. 749-762.

About the Authors

Cecil Eng Huang CHUA is an Associate Professor at the University of Auckland. He received a PhD in Information Systems from Georgia State University, a Masters of Business by Research from Nanyang Technological University and both a Bachelor of Business Administration in Computer Information Systems and Economics and a Masters Certificate in Telecommunications Management from the University of Miami. Cecil's research interests are in control and coordination of information systems, and information systems modeling. Cecil has publications in such journals as Communication Monographs, Decision Support Systems, Journal of the AIS, MIS Quarterly and the VLDB Journal.

Adrian YEOW (Ph.D. University of Maryland, College Park) is an Assistant Professor in the Division of Information Technology and Operations Management at Nanyang Business School, Nanyang Technological University. Adrian's research focuses on how information technologies, institutions, and organizations interrelate with each other in organizational processes such as complex IT implementations and ongoing daily coordination practices. His works have been published in Information & Organization as well as in top refereed conferences such as International Conference of Information Systems (ICIS), Academy of Management and INFORMS' Conference on Information Systems & Technology (CIST). Prior to entering academia, Adrian was a Manager of Product Development for a major Singapore communications company and has extensive product development and project management experience in telecommunications and software development.