DESIGNING COORDINATION IN COMPLEX IT PROGRAMS: THE ‘PROGRAM COORDINATION TOOL’

Abstract

In this paper, we present the results of examining an information technology (IT) artefact design project in the context of a complex IT program. This design project was triggered by several IT program-specific challenges, including a large number of interdependencies between multiple involved projects and multiple program hierarchy levels. In addition, a trigger were the assessment results generated at our case organization that available standard tools are not sufficiently effective in enabling the type and degree of coordination needed in such complex IT programs as the one at hand. The IT artefact design process was a multi-year evolutionary process, with several iterations, and resulted in a so-called ‘Program Coordination Tool (PCT)’ that has been successfully adopted at our case organization. The process of artefact design, the outcomes, and insights from artefact implementation and evaluation are discussed in the paper. Implications are drawn for coordinating IT programs.

Keywords: IT program coordination and management, action design reserach, design principles

1 Introduction

A prevalent trend in business is that multiple IT projects are implemented in combination in the context of a larger strategic business initiative, e.g., an organization-wide IT-driven transformation of the entire enterprise architecture. In this case, it is also referred to as an IT program, which consists of multiple, interrelated projects (Pellegrinelli et al., 2007). Managing multiple interrelated IT projects in combination in the context of an IT program involves an even greater degree of coordination and control in order to meet the strategic program goals, making IT programs even more challenging than IT projects. One of the reasons for the increased challenges is the multiple involved interdependencies across individual projects and the multiple levels of the resulting large program organization. In this context, coordination across projects and layers becomes an extremely difficult task, which is not well understood; neither in practice, nor in theory. Examining how to effectively support and enable coordination in such complex IT programs is therefore very relevant.

Prior research on IT project and program management has shed light on the topic of coordination from a behavioural science research perspective. For example, research examining software development projects has explained coordination in terms of distinguishing between horizontal (or lateral) coordination and vertical (or hierarchical) coordination, and arguing for the simultaneous need for both types of mechanisms (e.g., Nidumolu, 1995). As another example, Andres and Zmud (2001) adopt a contingency approach to software project coordination, and, building upon prior organizational design theory (Thompson, 1967; Van de Ven et al., 1976), emphasize the nature of coordination in this context as "facilitating the necessary information exchanges and decisional autonomy needed for effective collaboration and decision-making" (p. 34). However, to the best of our knowledge, there are no studies in the existing literature that have approached the research problem of coordination in IT programs from a design science research perspective. As a result, we have a lack of understanding in the IT project and program management literature about how to effectively design and support coordination in IT programs with the help of IT. This is surprising, given that the task of coordination is inherently linked to information management, as suggested by the above provided definition, for which IT can be used as an enabler, as illustrated by our study.

The program and its results that are presented in this paper started as an internal IT artefact design and implementation process, accompanied by an extensive, multi-year qualitative research project, and ended in a design science research project through a collaboration between industry and academia that emerged in that process. An IT artefact instantiation, i.e., a 'tool' (Orlikowski and Iacono, 2001) called in this paper the 'Program Coordination Tool (PCT),' was designed and implemented over a time period of over two years, resulting in a significant increase in coordination effectiveness and efficiency in an IT program that involves a large-scale transformation of the organization's IT. Lessons and insights are drawn, both from the IT artefact development process itself, as well as from the resulting design product and its embedded use within the organization. In summary, the developed IT artefact represents a form of design knowledge that contributes to the practice and our understanding about coordination in IT programs.

The remainder of this paper is structured as follows. The following section introduces the reader to the theory and literature that guided the research process. Next, the research methodology is explained. The main section of the paper focuses on the analysis results, followed by a discussion of findings and implications.

2 Theoretical Background

A continuing trend in industry is that IT projects are implemented in the context of a larger strategic business initiative due to the increasing strategic importance of IT in businesses. When multiple IT projects are closely coupled and share common goals and objectives, organizations in practice refer to this as an IT program (Pellegrinelli, 2011). Thus, an IT program provides a structural frame within which multiple interrelated IT projects are grouped and executed. IT projects involve a lot of uncertainty (Xia and Lee, 2005), for example due to complex and innovative technologies (i.e., hardware, software) (Zmud, 1980) and requirements that are either unclear or that change over time (Nidumolu, 1996). However, IT programs involve even greater amounts of uncertainty, because they "are constantly subject to influences and developments, emanating from within the organization, from the external environment and from the organization's response to that changing environment" (Pellegrinelli, 2002, p. 230). A related characteristic of IT programs is that they are frequently an instrument to implement an organizational IS strategy (Artto et al., 2008; Ross et al., 2006). They often involve organizational change and in some cases even the entire transformation of the organization's IT. Therefore, they are often also very large and have extensive scope. Finally, IT programs usually involve multiple interdependencies between projects and subprograms that form part of the overall program. Task interdependence, referred to as the extent to which a task requires organizational units to engage in workflow exchanges and the extent to which actions of different work units affect each other (Andres and Zmud, 2001), is frequently a challenge in IT projects, but may play an even bigger role in the IT program context due to the size and scope. Such task interdependencies may result to a large extent from technological interdependencies as the systems to be implemented in one subprogram may be dependent on, or share interfaces with, the systems to be implemented on another subprogram. In summary, due to the above described characteristics, IT programs can be characterized as very complex systems (Amaral and Uzzi, 2007), which require coordinated action.

Coordination, previously defined in IS research as "facilitating the necessary information exchanges and decisional autonomy needed for effective collaboration and decision-making" (Andres and Zmud, 2001, p. 34), is extremely important in IT programs, as indicated by the above explained characteristics. In other words, the characteristics of IT programs, e.g., the involved uncertainty, size, and interdependencies, require coordinated information exchange. Prior IS research on coordination in IT project and programs (e.g., Nidumolu, 1995) has adopted the information processing theory of organizational design (Galbraith, 1973; Galbraith, 1974; Galbraith, 1977), which states that coordination needs to be carefully designed to avoid overloading key decision makers (in traditional organizations also referred to as the 'hierarchy') and to enable an efficient and effective coordination of activities both across vertical and horizontal lines. In the context of an IT program, a strong emphasis lies on the coordination of activities across individual subprograms and projects that are involved in the overall program. This is also referred to in the literature as lateral coordination (Dahlander and O'Mahony, 2011). Galbraith (1973) states that a key contingency factor for the design of coordination in organizations is uncertainty, defined as the difference between the amount of information required and the amount of information already possessed. Galbraith's information processing theory focuses on the key concept of information processing to deal with that uncertainty, suggesting that the amount or capacity of information processing and the design of information processing structures during task execution should be aligned with the level of uncertainty (ibid). In other words, uncertainty facing subprograms and projects in an IT program may determine information processing requirements, and information processing capacities and design structures must be matched with these (Tushman and Nadler, 1978).

3 Research Methodology

The research presented in this paper follows an embedded research design with an emphasis on design science research (DSR) (Hevner et al., 2004), which builds upon Simon's vision of the 'sciences of the artificial' (Simon, 1996). Facilitating this DSR approach, we conducted an extensive amount of qualitative field research at the organization in which the coordination tool was designed and implemented. Thus, the focus lies on examining the design of an IT artefact, i.e., a 'tool' (Orlikowski and Iacono, 2001) in practice, which is called the 'Program Coordination Tool (PCT).' This tool was developed and successfully implemented over a time period of over two years at our case organization, resulting in a significant increase in coordination effectiveness and efficiency in an IT program that involves a large-scale transformation of the organization’s IT. To enhance joint reflection and learning from this tool development and implementation process in an organizational context, a recently proposed method called ‘Action Design Research (ADR)’ (Sein et al., 2011) was used as methodological guidance. ADR specifies a set of principles for conducting design science research that actually has an organizational impact through the ‘action’ of implementing an IT artefact, while not deemphasizing the formalization of learning from that process as well as the design outcomes. In the following, we summarize the application of ADR principles in our research project.

The first stage of ADR involves the formulation of a problem, involving two principles, i.e., (1) practice-inspired research and (2) theory-ingrained artefact. With regards to (1), in our case the presented design research is highly practice-inspired to the extent that the idea for development and implementing a new tool for supporting coordination in IT programs actually emerged in practice, and the collaboration with science emerged later on in the process. Our case organization experienced a number of IT program-specific challenges (discussed in the paper) that prompted the organization to experiment with the design of a new tool that would overcome the limitations of existing tools in the market. With regards to (2), even though the idea for a new tool as well as its development started before the mentioned collaboration between practice and academia, the designed PCT tool was actually informed by previous knowledge that was deeply embedded in the experiences of the designers in practice and, later in the research process, proved to be related to existing theory.

The second stage of ADR, which is intertwined with the first stage, involves the process of building, intervening, and evaluating the outcomes. Principle (3) of reciprocal shaping involves the iterations between the IT artefact and the organizational context. In our case, the artefact design process was a multi-year process with several development cycles that were influenced by changes in the organizational context. Principle (4) of mutually influential roles emphasizes the importance of shared learning and exchange between different project participants. In our case, the research team engaged in extensive interactions with the design team in practice over a multi-year time period. Initially, this researcher-practitioner interaction took the form of a case study research with a focus on explaining reality. Over time, however, this design science research project emerged as a spin-out from this qualitative research project, triggered through the extensive amount of field research at the case study site. Finally, principle (5) deals with the authentic and concurrent evaluation of the design artefact, which in our case was an on-going process as the PCT was being implemented, adopted, and eventually declared as standard for the IT program at our case organization. This on-going process was accompanied by a longitudinal case study at this organization during the same time period. Over a time period of over a year, we conducted an extensive amount of field research at the case organization in which the presented tool was designed, implemented, and currently used. This has resulted in over 50 qualitative face-to-face interviews at the case study site that have been transcribed and coded with the help of the software Atlas.ti (Muhr, 2008). In addition, multiple secondary data sources such as slide sets documenting the purposes and functionality of the PCT have been collected and analysed for triangulation purposes. In addition, a considerable amount of informal exchange of information and ideas has taken place.

The third stage of ADR focuses on reflection and learning with the principle of guided emergence (6), which in our case focused on abstracting the results from artefact design and implementation in terms of meta-requirements and corresponding meta-design components and principles. This abstraction process was facilitated by our qualitative field research, which triggered joint reflection and learning between academia and practice. Finally, principle (7) of the fourth stage of ADR emphasizes the formalization of learning in terms of generalized outcomes, which are the main contribution of this paper and presented herein. The process of formalizing the outcomes has been an iterative process with drafts of the paper being reviewed by different members of the organization, followed by feedback conversations and ‘reality checks.’

The ADR project presented in this paper forms part of a research cooperation that also involves conducting a longitudinal single-case study. In the course of the field research, a key theme of coordination in IT programs emerged from the data analysis and the artefact design project that is the subject of this paper turned out to be of highest practical importance that offered an unique opportunity for generating insights about the IT-based design of coordination in this context. Thus, the DSR project presented in this paper is intertwined with a large, multi-year case study research project. As a result, interviews and secondary materials collected as part of this case study research formed the key basis for the findings presented in this paper. In addition to these case-study related data collection activities, a considerable amount of informal exchange of ideas has contributed to the analysis results that are presented.

4 Analysis: Designing Coordination in Complex IT Programs

4.1 Challenges and Meta-Requirements

The financial organization we studied engaged in a long-term large IT program to transform the organization's IT by creating a new integrated IT platform. In the past, multiple attempts had been made to reengineer parts of the home-grown legacy IT systems platform, but the insight emerged over time that a major platform transformation was needed as opposed to continuous reengineering efforts that did not provide the required leap-change to stay ahead of competitors in the market. In particular, the organization's objective is to significantly increase its cost competitiveness and improve its cost-income-ratio, or CIR. For that reason, the examined IT program was initiated.

Our case organization experienced a number of challenges in the initial phase of IT program ramp-up. First, the scope of the program was enormous, involving multiple business domains and a large number of different requirements, depending on the involved business processes, products, and technology. As a result, the organization was faced with a large degree of heterogeneity in the IT program environment (e.g., a large number of different subprograms, each with a unique goal set, different involved parties, etc.).

Second, this large amount of differentiation that was required to be able to deal with the complexity and scope of the IT program resulted in a large number of interdependencies between subprograms and projects. Interdependencies were of multiple types, the most important type being technological. Program members had to deal with multiple interfaces between systems that formed part of the overall IT landscape. Dealing with these technological interdependencies was complex in itself, but also created task interdependencies as the work activities on one system in an individual project affected or were affected by the work activities on another system in a different project. Thus, activities needed to be coordinated across projects and subprograms to make sure that deliverables from different work streams were aligned and contributed to the goals of creating a single, integrated IT platform.

Third, as a result of the above explained complexity, a multi-level program hierarchy was created that generated work activities on different levels. Our case organization referred to the IT program as the 'overall' program, involving individual subprograms, projects, and sub-projects. This large number of levels was required due to the size and scope of the entire IT platform transformation initiative. However, it also created the challenge of coordinating information exchange across program levels.

Fourth, many internal and external resources worked on the program simultaneously. This posed special challenges to coordination because external resources, in some cases, needed to be treated in a special way in terms of exchanging information and coordinating work activities. A challenge was a high degree of personnel fluctuation as external resources were replaced or the number of external resources had to be reduced at certain time points due to changes or reductions in financial budgets. This created the risk of losing important knowledge and could only be kept internally, at least to some extent, through codification.

As a result of the above, program members in our case were faced with enormous amounts of uncertainty and complexity. Coordination, in this context, turned out to be particularly challenging and IT program managers concerned with the coordination of work activities across work streams and levels created a task force to select or design an appropriate program coordination tool.

4.2 Initiation of a Design Project

The team that received the work order to implement a coordination tool initially screened the market to look for a standard software tool. A member from the program office (PO) that has the role to facilitate coordination in the examined IT program stated:

"There are different tools in the market that support complexity management. For example, we evaluated a tool from [name of provider] as well as another tool from a provider that has experience in the automobile industry, but somehow none of these existing solutions fitted to our problem situation and needs."

Eventually, the design team decided to develop its own solution that would fit the particular requirements explained above. One of the very basic requirements for the tool was to support basic project management processes such as developing a project or program plan, resource management, tracking progress, and budget management. As a result, Microsoft Project (MSP) was adopted as a basis and customized in cooperation with a small external consultancy company. One of the design team leads and basically the chief design architect explained:

"One of the reasons why we adopted MSP was that this tool is currently the commonly accepted standard for project planning and control in the market. I believe that a large majority of large companies use this tool to support IT project and program management … of course I also looked internally what software solutions were currently in use but they all had their limitations for supporting such a large IT program as this one."

With the adoption and customization of MSP, the design project to create an individual solution called the 'Program Coordination Tool (PCT)' (exact name disguised to ensure anonymity) was initiated. The basic idea was to adopt the project management server solution of MSP. This MSP Server solution uses Microsoft SharePoint as a foundation, which was also implemented in the examined IT program. Using MSP Server as a basis, the design team started to customize the solution to the specific needs of the organization. The chief design architect explained:

"We view the MSP Server technology from Microsoft as our operating system and based on that we created what we call the PCT as our App Store, which is customer-specific and offers tailored functionality for repetitive tasks that are conducted by a large number of program members and have to be done."

In another comment, he explained:

"The technological basis was MSP Server with a server-based web user interface, initially used only for planning, an SQL database, and a reporting machine. We combined and integrated these different components to create a tool that complies with our internally defined quality management standards."

Designing the PCT lasted over two years and was an iterative, evolutionary, and incremental process that is explained in the following. As a basis for explaining this process, we first introduce the reader to the defined requirements that guided this process.

4.3 Specific Requirements

A number of specific requirements were identified over time that guided the design of the PCT. For example, one of the key challenges was the high effort to collect and aggregate data from various subprograms and projects. Furthermore, transforming a heterogeneous project portfolio into a homogenous project portfolio was extremely difficult. In addition, there was the need to visualize progress and measure objective achievement on different program levels and it was also a great challenge to achieve transparency and efficiency in the management of the multiple interdependencies.

Above described challenges such as collecting and aggregating data, integrating information to achieve homogeneity in planning, visualizing progress, and achieving transparency are closely related to the task of coordination and the information processing viewpoint of organizational design (see theoretical background section) that emphasizes the need to reduce informational uncertainty and design information processing capacities (in our case the PCT) accordingly. Galbraith (1973) in his treatment of how to design complex organizations mentions the role of information systems and technology in potentially increasing information processing capacity in a complex organization. A key coordination requirement from an information processing perspective is to ensure an efficient support of coordination and a fit between coordination needs and the design of coordination mechanisms. This focus on coordination as well as efficiency and effectiveness guided the chief design architect in our case as explained in the following.

4.4 The Designer’s Knowledge and Background

The entire PCT design team had extensive work experience. For example, the chief design architect brought in valuable experiences and expertise from previous IT projects in more industrialized industries than the financial industry, including the automobile industry. He explained:

“In my prior work experience I have worked for a consultancy that is specialized in supporting large projects or programs. I was involved in the development of software solutions to support program management in the automobile industry, thereby always working very closely at the interface between business and IT. When I came to work for my current organization, I realized that my previous experience from more industrialized contexts was very valuable. What catched my attention right from the start was that I realized many people were working with PowerPoint, focusing on ideal-typical descriptions of plans, etc. that were quickly outdated and entailed a real problem, they did not move anyone to talk with each other on a sustainable basis. However, as I knew from my past experience, you only reduce complexity when people communicate with each other efficiently and effectively.”

Thus, the first key characteristic of the designer’s background is the industrialized industry experience.

A second key background characteristic of the designer was the type of prior software development experience that resembles agile methods. The designer explained:

“What I believe really influenced me in the development of the PCT was my past experience in an IT project at a large automotive company in which our task was to create a process and tool for dealing more effectively and efficiently with problems in the car production process in terms of facilitating adequate information processing to enable timely and targeted responses … we adopted a highly dynamic development approach, we didn’t even define the scope of our project at the outset and just started with little keystones and continued step-by-step and eventually, by creatively finding a way to link the different components, a new information processing platform emerged that is now used at this car company. This outcome was not planned beforehand and was more like a continuous change process. That’s where I learned a lot for how exactly to design the PCT in this program.”

In addition to the designer’s knowledge and background, the personality of individual members of the design team also played a role for facilitating the design process, as illustrated by the following quote:

“Reflecting upon the way in which we designed the tool, I believe that a designer needs to be communicative, willing to make compromises, and resilient, in order to involve many in the process, even though this is sometimes exhausting and anything but easy.”

In summary, the process of designing the PCT that is explained in the following was influenced by the particular professional background of the design team.

4.5 The PCT Design Process

The PCT design process lasted over two years in parallel to the ramp-up and first phases of the examined IT program. Organizationally, the design team was located in the central program office (PO) that was a unit responsible for facilitating cross-project/subprogram and cross-level coordination. A number of key overarching themes emerged from our examination of the design process that are discussed in the following.

One principle for the design process was to focus on integrating and codifying knowledge from diverse internal and external resources that were involved in the IT program. Particularly in the beginning of the examined IT program, a large number of external resources were involved from different consultancy firms that were all engaged in providing consulting services to the firm with regards to how to set-up the IT program organization and how to create the necessary structures, including coordination mechanisms, to run the program successfully. The chief design architect explained:

“External resources came in and out of the program and the development of the PCT helped in capturing a lot of know how because throughout the process, many professionals have looked at the tool, provided feedback, and enabled us to constantly improve the tool and internalize a lot of program management expertise.”

Thus, an importance principle that guided the design process was to integrate and internalize as much relevant program management know how as possible and codify key coordination processes by implementing them into the PCT. As such, the PCT represented a means for knowledge integration, internalization, and codification.

Secondly, the software development process followed a set of principles that resemble the basic idea behind agile software development. One of the important principles was to develop the software tool in small groups with an active involvement of end users, i.e., IT program members, from the beginning onwards. The subprogram, which was one of the subprograms of the overall IT initiative examined in this research project, played a particularly important role in contributing to the functionalities and design of the PCT. The reason for this was that the subprogram started earlier than other involved subprograms and was defined at the outset of the overall IT program as the ‘backbone’ and driver of the entire transformation. Frequently, key end users that contributed to the development of the PCT were leaders or members of the individual program project management offices (PMOs). One of the key users and contributors from the subprogram explained:

“We are faced with enormous challenges in our subprogram and try to confront these challenges by increasing efficiency, particularly trying to take advantage of the possibilities of automation. We have done this the whole time by implementing this PCT for example, we have made significant contributions to its development, from my perspective even the majority of functionalities that are nowadays implemented in that tool are based on ideas that originated in our subprogram. We had the advantage that we were multiple months ahead of the other involved subprograms.”

Thus, end users of the PCT in the individual subprograms were involved very actively in the development and design of the tool, providing them with a very strong motivation to use the tool.

A related process design principle was to enable collaboration between developers and users through the use and creation of common language and understanding. The key idea was to get all involved end users in the IT program agreed on a commonly shared agreement with regards to the activities that were needed and how to support them with a tool. The chief designer explained:

“We focused from the beginning onwards on homogenizing rules of communication to quickly move away from discussions about how to do things and achieve a quick agreement by all participants on the key commonly agreed upon coordination and communication activities … when we started I believe that approx. 40 percent of all communication in the program was about methodological issues,

how to do things, and in the end we were able to reduce that with the help of the tool to about 10 percent. That means that nowadays, individual subprograms and projects have much more time available for the really important things that is getting the real work done."

Another important related process design principle was to develop and implement individual functions for the tool in an iterative, step-by-step approach, while consciously leaving end users enough degrees of freedom and not overwhelming them with a sudden big change. The chief designer explained:

"Reflecting upon the current status of the PCT, if I had specified and implemented all of the functionality in advance, it wouldn't have been the case that all of the program members in the individual subprograms and projects would have adopted the tool. Because the tool evolved over time through iterations that were enabled by end users, everybody could eventually identify with the tool, that's why this tool is such a success. It is a tool that covers to a large extent all of the standard program management and coordination activities, in sum very powerful."

This evolutionary process with active involvement of end users resulted in a strong identification of end users with the resulting PCT, for which reason it has been strongly adopted and is used extensively in the examined IT program.

4.6 Instantiation and Evaluation: The 'Program Coordination Tool (PCT)'

The above explained challenges (4.1), initiation of a design project (4.2), formulation of specific requirements (4.3), designer's knowledge and background (4.4), and the overall PCT design process (4.5) resulted in an IT artefact instantiation, i.e., a tool, that was first developed as a prototype, piloted with the above mentioned subprogram, and then rolled out throughout the entire IT program that is examined in this research project. The tool's capabilities include consistency checks in program and project planning, including top-down roadmaps and bottom-up plans, pro-active and transparent dependency management, easy information gathering and automated reporting, including status, issue, and risk management, enforced standardized program and project coordination, state-of-the-art database solution for an easy data exchange or integration with surrounding systems.

One of the emerging results from the evaluation of the PCT in our case organization is that it is suitable for facilitating information processing and exchange both horizontally and vertically. An IT program member explained:

"The tool is very useful for connecting and linking both the vertical and horizontal lines of communication across layers and subprograms or projects."

Another program member pointed to the usefulness of the tool for dependency management:

"With the help of the PCT we can see very nicely where the dependencies are and you get a good overview that helps for managing them."

The leader of the central PO unit described the usefulness of the tool in the following way:

"The PCT depicts the different layers of the IT program. We have areas in which individual subprograms are fully responsible themselves, these are the lower three levels with subprograms, projects, and subprojects, that is what a program manager from an individual subprogram is allowed to control completely by him/herself. Beyond that we have two additional reporting layers, which present a highly aggregated view and are monitored by the overall program management. That way we have specified clearly who controls which part, which avoids micromanagement and enables program coordination at an aggregated overall level, this is what we agreed upon by implementing this tool."

As can be seen from these illustrative quotes, different users have very different viewpoints on the tool and interact with, or use the tool, in a different way. Thus, individual subprograms and projects are equally supported by the tool as is the overall program management and its supporters such as the centralized PO unit.

4.7 Reflection and Learning: Towards a Meta-Design for IT-Based Coordination Support in Complex IT Programs

Joint reflection and learning in this research project resulted in a number of insights that are summarized in the following. They present a first step towards developing a meta-design for the IT-based support of coordination in complex IT programs.

The first identified design principle is to effectively *facilitate information processing both horizontally and vertically at the same time*. Horizontally, this meant facilitating information processing across different subprograms and projects to deal with interdependencies between them. To a large extent, interdependencies in our case were of a technological nature, involving for example systems interfaces between systems that were adapted or replaced in different projects and subprograms. Such technological interdependencies also entailed task interdependencies as the activities conducted in one project or subprogram were dependent on the activities in another one. In summary, horizontal information processing rested upon the meta-requirement of dealing with these interdependencies and as a result, dependency management capabilities were built into the PCT. At the same time of facilitating information processing horizontally, it also had to be facilitated vertically across different levels of the program hierarchy to control for consistency of plans in program and project planning across levels. Long-term roadmaps that were provided top-down should be followed at lower levels and plans generated and pursued at lower levels should also be consistent with overall roadmaps. Thus, the capability of performing consistency checks in program and project planning was also built into the PCT. Overall, designing coordination in the IT program to facilitate information processing both horizontally and vertically at the same time was found to be critical for ensuring IT program success.

The second design principle for designing and developing the PCT in our case was to focus on *simultaneously supporting individual projects and subprograms on the one hand and overall program management on the other hand*. Due to the large amounts of uncertainty and complexity experienced in individual projects and subprograms, participants were under enormous pressure to solve problems effectively and deal with unique domain requirements, depending on the respective areas. As a result, any efforts from centralized program units to exercise coordination across individual projects and subprograms initially created resistance to a more or less degree. Frequently, centralized coordination activities would be viewed as overhead costs that did not help in advancing with the achievement of project-related work. On the other hand, due to the overall strategic nature and organizational scope of the IT program, integration and coordination requirements were also high. Thus, an important design principle that was always kept in mind in the process of designing the PCT was to involve and integrate to a large extent the viewpoints and contributions of individual projects and subprograms to motivate them to really use the tool for their own benefit.

5 Discussion and Implications

Discussing the findings of this research project in the light of existing literature yields some interesting insights.

First, our findings illustrate the importance of facilitating coordination and information exchange both horizontally and vertically. Prior research on coordination in software development projects has pointed to the need for both types of coordination (e.g., Nidumolu, 1995). However, in the literature, vertical coordination is characterized as an authoritative, top-down type of mechanism that is exercised by a ‘strong’ entity such as a steering committee and horizontal coordination is characterized very differently, i.e., as a type of coordination focused on rather informal communication and mutual adjustments through personal or group means. In our case study, we do find evidence for the existence of these types of coordination. However, as illustrated by the PCT design project and its outcomes, the nature of coordination may become different in a complex IT program. Specifically, the PCT was able to formalize, standardize, and automate both vertical and horizontal coordination, providing a cross-level and cross-project coordination support, which increased enormously the efficiency in IT program work. Thus, the tool also helped overcome limitations of matrix organizations.

Second, our findings on implementing a tool that simultaneously addressed the needs of individual subprograms and projects, as well as overall program management, relates to coordination and organizational information processing theory that emphasizes the need to manage both differentiation and integration (Galbraith, 1973; Lawrence and Lorsch, 1967). Due to the large amounts of uncertainty and complexity, a large degree of differentiation of program work into individual projects and subprograms is needed. At the same time, with increased levels of differentiation, as well as the large number of interdependencies, a high degree of integration at the overall program level is required. Simultaneously managing both differentiation and integration in organizational contexts and designing coordination accordingly is extremely difficult. This study illustrates the important role of IT-based coordination support toward these ends.

Probably the most interesting overarching finding of this paper is to reveal the necessity of pursuing different things at the same time to effectively coordinate IT programs, namely facilitating information processing both vertically and horizontally simultaneously and supporting individual projects and overall program management at the same time. Prior research has studied this phenomenon, which is also referred to as ‘ambidexterity,’ in the context of software development projects (e.g., Tiwana, 2010). Thereby, the focus has been on examining the ability to simultaneously achieve alignment and adaptability in the software development process, while in our study we identify the ability to simultaneously achieve horizontal and vertical coordination and the ability to simultaneously achieve individual project and overall program success as two new forms of IS ambidexterity.

6 Conclusion

This paper presented the results of examining an IT artefact design project in the context of a complex IT program. Insights about the requirements and solution components and design principles for effectively designing coordination support were presented that contribute to our understanding about coordination in IT programs, about which there is currently not much literature. Future research may build upon these contributions to develop a design theory of coordination in IT programs. Furthermore, our theoretical understanding about IS ambidexterity is enhanced by identifying and examining two different design principles that entail the need to pursue two distinct things at the same time.

References

Amaral, L.A.N. and Uzzi, B. (2007). Complex Systems—A New Paradigm for the Integrative Study of Management, Physical, and Technological Systems. Management Science, 53 (7), 1033-1035.

Andres, H.P. and Zmud, R.W. (2001). A Contingency Approach to Software Project Coordination. Journal of Management Information Systems, 18 (3), 41-70.

Artto, K., Kujala, J., Dietrich, P. and Martinsuo, M. (2008). What is project strategy? International Journal of Project Management, 26 (1), 4-12.

Dahlander, L. and O'Mahony, S. (2011). Progressing to the Center: Coordinating Project Work. Organization Science, 22 (4), 961-979.

Galbraith, J.R. (1973). Designing Complex Organizations, New York, Addison-Wesley Publishing Company.

Galbraith, J.R. (1974). Organization Design: An Information Processing View. Interfaces, 4 (3), 28-36.

Galbraith, J.R. (1977). Organization Design, London, Addison-Wesley Publishing Company.

Hevner, A.R., March, S.T., Jinsoo, P. and Ram, S. (2004). Design Science in Information Systems Research. MIS Quarterly, 28 (1), 75-105.

Lawrence, P.R. and Lorsch, J.W. (1967). Differentiation and Integration in Complex Organizations. Administrative Science Quarterly, 12 (1), 1-47.

Muhr, T. (2008). ATLAS.ti - The Knowledge Workbench, Berlin, Germany, Scientific Software Development.

Nidumolu, S.R. (1995). The Effect of Coordination and Uncertainty on Software Project Performance: Residual Performance Risk as an Intervening Variable. Information Systems Research, 6 (3), 191-219.

Nidumolu, S.R. (1996). A Comparison of the Structural Contingency and Risk-Based Perspectives on Coordination in Software-Development Projects. Journal of Management Information Systems, 13 (2), 77-113.

Orlikowski, W.J. and Iacono, C.S. (2001). Research Commentary: Desperately Seeking the "IT" in IT Research - A Call to Theorizing the IT Artifact. Information Systems Research, 12 (2), 121-134.

Pellegrinelli, S. (2002). Shaping context: the role and challenge for programmes. International Journal of Project Management, 20 (3), 229-233.

Pellegrinelli, S. (2011). What’s in a name: Project or programme? International Journal of Project Management, 29 (2), 232-240.

Pellegrinelli, S., Partington, D., Hemingway, C., Mohdzain, Z. and Shah, M. (2007). The importance of context in programme management: An empirical review of programme practices. International Journal of Project Management, 25, 41-55.

Ross, J.W., Weill, P. and Robertson, D. (2006). Enterprise architecture as strategy: Creating a foundation for business execution, Boston, Harvard Business Press.

Sein, M.K., Henfridsson, O., Purao, S., Rossi, M. and Lindgren, R. (2011). Action Design Research. MIS Quarterly, 35 (1), 37-56.

Simon, H.A. (1996). The Sciences of the Artificial, Cambridge, MA, MIT Press.

Thompson, J.D. (1967). Organizations in Action, New York, McGraw-Hill.

Tiwana, A. (2010). Systems Development Ambidexterity: Explaining the Complementary and Substitutive Roles of Formal and Informal Controls. Journal of Management Information Systems, 27 (2), 87-126.

Tushman, M.L. and Nadler, D.A. (1978). Information Processing as an Integrating Concept in Organizational Design. The Academy of Management Review, 3 (3), 613-624.

Van de Ven, A.H., Delbecq, A.L. and Koenig, R., Jr. (1976). Determinants of Coordination Modes within Organizations. American Sociological Review, 41 (2), 322-338.

Xia, W. and Lee, G. (2005). Complexity of Information Systems Development Projects: Conceptualization and Measurement Development. Journal of Management Information Systems, 22 (1), 45-83.

Zmud, R.W. (1980). Management of Large Software Development Efforts. MIS Quarterly, 4 (2), 45-55.