Coordination Analysis: A Method for Deriving Use Cases from Process Dependencies

Xiang (Michelle) Liu
Marymount University
School of Business Administration
Michelle.Liu@Marymount.edu

George M. Wyner
Boston University
School of Management
gwyner@bu.edu

ABSTRACT
Despite the widespread recognition that information technology (IT) and business process are tightly connected, existing system design methods provide limited guidance on how to take business process into account when designing information systems. The primary goal of this paper is to describe a method which helps systems analysts to explore more systematically the potential of IT to change the design of business processes. We use coordination theory to provide a theoretical connection between use cases and the dependencies among activities within a process. By building use cases from dependencies, we are able to consider a wide range of functionality for managing those dependencies and thus incorporate process redesign into the requirements process. We employ a healthcare case as an example to illustrate the proposed method.

Categories and Subject Descriptors
D.2.1 [Software Engineering]: Requirements/Specifications – methodologies.

General Terms
Design

Keywords
Development approach, software design, modeling, coordination, IT Innovation.

1. INTRODUCTION
Despite the widespread recognition that information technology (IT) and business process are tightly connected [1-3], existing system design methods provide limited guidance on how to take business process into account when designing information systems. While use cases [4] are recognized as a powerful method for capturing the functional requirements of a system based on business requirements, use cases require users and systems analysts to define the role that the information system should play in the business process [5]. Analysts often identify use cases by scanning for opportunities to automate existing business processes. The limitation of this approach is that by itself it can lead to “paving the cow paths:” mapping the existing process to its automated version without taking into account the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DESRIST’09, May 7-8, 2009, Malvern, PA, USA.
Copyright 2009 ACM 978-1-60558-408-9/09/05...$5.00.

potential of IT to transform a business process. For this reason, systems analysts must rely on their understanding of both IT and the business process in order to discover opportunities to redesign the business process.

Our belief is that analysts can be more consistently insightful in discovering such opportunities for process redesign if business processes are represented in a way that reveals coordination issues. As coordination is often information intensive, such issues as these are especially likely to be amenable to IT-enabled alternatives. We describe a method based on coordination theory [6] which represents a process as a set of core (“production”) activities and their interdependencies. Our method uses this representation to enable the analyst to explore alternative process designs while taking into account the role an information system might play in enabling and supporting those alternatives.

The remainder of this paper is organized as follows: In section 2 we review the past literature on business process analysis, systems modeling, and existing methods to integrate the two. Section 3 introduces the theoretical background of our proposed method. Section 4 describes our method in detail. In Section 5 we apply our method to analyzing medication processes in a hospital. We use this example as a basis for some preliminary assessment of the ways in which our method might support analysts in arriving at process insights as part of the system design process. The final section presents major contributions, limitations, and future research directions.

2. LITERATURE REVIEW
One approach to capturing software requirements is to develop business process models and derive requirements for the software systems supporting them [7]. Researchers have tried to depict business processes [2, 8-10] and system requirements [11] using various approaches. In this section we will review the past literatures from both the process and system perspectives. We will focus in particular on existing methods that provide explicit support for taking process design into account when carrying out systems analysis and design.

2.1 Business Process Perspective
The Business Process Reengineering (BPR) approach [8, 12] proposes “the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical, contemporary measures of performances such as cost, quality, service, and speed” [10]. Venkatraman [9] identified IT as a central lever for such reconfiguration of business processes. In contrast to the “radical” business process changes of

1 Note: both authors contributed equally to this paper.
reengineering, *process innovation* is an approach based on the core idea that improving business processes should be carried out incrementally [13]. Process innovation emphasizes the concept of continuous change and adaptation.

Barothy et al. [14] identify five major topics within the BPR literature: business transformation, methodology, reengineering principles, success factors, and related research. This extensive literature review indicates that few researchers focus on business processes from the information system perspective or attempt to connect business process models with system requirements. One exception is the work of Gruhn and Wellen [15] who attempted to derive distributed software architecture from business process models. However, they did not provide a procedure or guidance for realizing this proposition.

### 2.2 Information System Perspective

Systems analysis and design methods generally emphasize the importance of beginning any design effort by developing an understanding of the business requirements and how they might be supported by information system functions. For example, in his classic text on structured analysis, Yourdon recommends that system requirements be identified by constructing an *essential model* – "a model of what the system must do in order to satisfy the user's requirements, with as little as possible (and ideally *nothing*) said about how the system will be implemented" [16, p. 323]. The principle component for defining business requirements in an essential model is the *environmental model*, which includes a statement of system purpose, a list of business events to which the system must respond, and a description of key information flows between the system and its environment (in particular its users). These artifacts are developed through a series of meetings between analysts and users and are primarily derived from the users' understanding of what the system ought to do. While the meetings and the essential model may lead to discovering new system functionality, there is no systematic support for such explorations. Thus the starting point for Yourdon is the user's understanding of what the system should do, not an understanding of the business process. Insightful users and analysts may take advantage of this design effort as an opportunity to introduce process innovation, but arriving at such insights is not discussed explicitly in the method.

Structured methodologies, such as Yourdon's, are often contrasted with the object-oriented analysis and design techniques which came later. Most current object-oriented (OO) design methodologies are based on the Unified Modeling Language (UML) [17]. UML was developed by Booch, Rumbaugh, and Jacobson, and later standardized by the Object Management Group (OMG) in 1997. Since then it has become the standard modeling language for software development. Unsurprisingly, one of the best known OO design methodologies is the one developed by the original authors of UML: the *Rational Unified Process* (RUP) [5]. RUP is described as "use case driven" in that system requirements are primarily captured in a set of *use cases*, each of which describes how specific system capabilities are used by a business stakeholder to achieve a business goal.

A use case documents a system’s behavior as the system responds to a request from one of its stakeholders. In the Rational Unified Process, a use case is defined as “a sequence of actions a system performs that yields an observable result of value to a particular actor” [5]. The objective of use case modeling is to identify and describe all the use cases that the actors require from the system [18].

For RUP and other use case driven methodologies the question becomes how use cases are identified. The primary answer to this question is quite similar to the structured approach: the analyst meets with users and together they identify use cases based on their understanding of what the system ought to do. Such an analysis may make use of the UML Activity Diagram. This diagram can be used to represent existing or proposed business processes. It shows not only the sequence of activities in a process but also provides a visual representation of how activities are assigned to various actors (roles) in the organization. Activity diagrams can focus attention on points in the process where multiple actors must coordinate their activities and thus can help to identify potential use cases to support coordination at those boundaries. As with Yourdon’s method, process innovation may well emerge from such discussions, but how a process ought to be changed is not explicitly addressed in the method itself.

Some methodologies provide additional support for helping analysts and users to make the best use of their collective understanding of the business in pursuing a use case driven approach: The Joint Application Development (JAD) approach brings all stakeholders together for a series of intense, highly structured, facilitated meetings to identify key system requirements based on a collective understanding of the business situation [19].

TOGAF -- The Open Group Architecture Framework ([http://opengroup.org/architecture/wp/](http://opengroup.org/architecture/wp/)) includes among its techniques a *business scenario workshop*, which provides a framework for capturing all the components of a business process and their relationships. A *business scenario* is used to identify and understand business needs, and thereby to derive the business requirements that the system architecture has to address. The business scenario provides a way to identify specific roles that technology should play.

While TOGAF and JAD do not provide detailed guidance on how to use technology to radically re-invent a process, they provide the means to create an environment which may facilitate arriving at such insight by promoting brainstorming and the inclusion of multiple perspectives on the business situation to be addressed by the system under design.

In addition to the use case driven approach and other techniques based on asking users what they need, some methodologies provide techniques for analyzing business processes themselves. In the RUP *Business Modeling Workflow*, a business process is modeled as a set of one or more *business use cases*. A business use case is similar in spirit to "ordinary" use cases with the difference that the system providing services in a business use case is not an information system but the business itself, which provides service to customers and other stakeholders. Once the business use case model is developed, RUP recommends that the analyst develop a "candidate system use case" for each business use case [5, p. 149]. In other words, use cases are discovered by looking at how an information system might support existing business activities.

---

2 The authors thank one of the anonymous reviewers for pointing out this use of activity diagrams.
While the approach just described is specific to RUP, it typifies a technique, used in many system design methods, which we will refer to as the "automation approach." In the most general terms, this approach consists of examining the activities in the current business process and identifying ways in which an information system might automate all or part of those activities. The automation approach is not itself a method, but an approach taken by many methods to address the problem of identifying system functionality suited to the business process in question. For example, another method which makes use of the automation approach is an approach developed by Eriksson and Penker [18], who provide a set of extensions to UML for more detailed modeling of business processes. Eriksson and Penker employ an assembly line diagram to identify system use cases based on the information requirements of activities in the existing business process.

The strategy of exploring how the activities in a given process can be automated is a powerful way to identify sources of IT innovation. However, unsurprisingly, a mechanical allegiance to this strategy is potentially counter-productive, since the automation approach takes the structure of the existing process as fixed, whereas users and analysts will typically want to explore the ways in which information technology might lead to fundamental changes in the process itself. For example, if the existing process involves a customer service representative taking an order for a catalog item over the phone, automation might lead to innovations such as an improved phone system which automatically routes calls to representatives, and the development of an order entry system. Analysts and users might also consider use cases based on a fundamental change in the process itself, such as having customers enter their orders directly on a website.

Given an innovative set of requirements or a redesigned process, RUP and its more traditional predecessors are well equipped to support the development of an appropriate information system, but these methods do not provide systematic support for discovering such innovative uses of information technology. What they support extensively is methods for identifying opportunities for automation of a given set of activities and designing systems to provide such automation.

The particular challenge here is that we want to redesign the business process based on new capabilities provided by IT, but these capabilities are best understood in the context of the business process. This circularity is typically resolved by insight: we imagine a process that does not yet exist and which is enabled by a system we have yet to design. This leads to the question of whether it is possible to provide system analysts with tools for exploring process innovation in a more systematic manner.

3. THEORETICAL BACKGROUND

Our approach to the problem of providing support for IT-enabled process innovation can be framed by the question:

How can we represent business processes in a way that allows us to compare and contrast them so as to explore a wider range of potential uses of information technology?

We will argue that the dependency diagram proposed in the coordination theory of Malone and colleagues [6, 20] can serve as such a representation. We will briefly introduce coordination theory and describe how the dependency diagram can play this role.

Building on earlier approaches to categorizing interdependence within organizations [21-23], Malone and colleagues [6, 20, 24] proposed a theory of coordination in which coordination is defined as the act of managing dependencies between activities within a business process.3 Associated with each dependency is a coordination mechanism which consists of those components of the process which manage that dependency. For example, a shared resource dependency might be managed by a "first come first serve" policy or a market-like bidding mechanism or managerial fiat. Coordination theory focuses on the dependencies which arise from resource flows among activities. Three types of dependencies have been identified, corresponding to different patterns of resource flows (as shown in Figure 1 below):

A flow dependency captures the issues which arise when a resource flows from the activity which produces it to the activity which consumes it. Three issues which must be addressed in managing a flow dependency are:

- **Usability**: Does the resource as produced correspond to what is needed by the consuming activity?
- **Synchronization**: Is the resource delivered from producer to consumer at the right time?4
- **Accessibility**: How is the resource made available to the consumer? Three options frequently considered are: move resource to consumer, move consumer to resource, and collocate producer and consumer.

A sharing dependency identifies the issues which arise when a single resource is consumed by multiple activities.

A fit dependency identifies the issues which arise when a single resource is produced through the joint action of multiple activities.

![Figure 1. Dependencies](image)

A principal benefit of this dependency approach is that it can be used to represent the core elements of a business process while

---

3 Portions of this brief description of coordination theory have appeared with minor changes in [25] and in an appendix to [26].

4 This is sometimes referred to in coordination theory as the **prerequisite** aspect of a flow dependency.
abstracting away the coordination activities currently used to organize that process [27]. This representation is known as a dependency diagram.

A dependency diagram is a process model which includes all the production activities associated with a process, where a production activity is one that, in the view of process stakeholders, contributes directly to achieving the business goal of the process. One guideline for identifying such activities is that they are essential to the process and must be included in any future version of the process.

The dependency diagram also includes all the dependencies among the production activities. Each dependency represents a coordination problem that must be solved for the process to function properly. Coordination theory posits that the "non-production" or "coordination" activities (those omitted from the dependency diagram) are present in the process because they serve as coordination mechanisms to manage these dependencies. By representing the underlying dependencies in the process rather than one particular coordination strategy, the dependency diagram is an ideal process representation for exploring alternative process designs, including those which might employ information technology in novel ways to manage dependencies. For example, Crowston [24] uses coordination theory to analyze the software change process at a computer manufacturer and identifies alternative mechanisms for managing dependencies in that process.

In a dependency diagram, rectangles represent activities. Each oval represents a dependency among the activities linked to it. The direction of the links indicates whether an activity is producing (arrow points to dependency) or consuming (arrow points to activity) the resource associated with a dependency.

For example, Figure 2 is a dependency diagram which provides a simplified view (for illustrative purposes) of the process for completing a homework assignment (e.g. in a college course). The process includes both activities typically carried out by the instructor (Create assignment, Create course content, and Grade assignment) and those carried out by the student (Complete assignment, Receive assignment feedback). The diagram represents the dependencies among these activities. Here we have one fit dependency (completing the assignment depends on making connections between the assignment and the course content) and two flow dependencies (completed assignments must be available before they can be graded and graded assignments must be available before they can provide feedback).

### 3.1 Methods for Creating Dependency Diagrams

To construct a dependency diagram one must identify production activities and the dependencies among them. The starting point for such an analysis is the identification of the activities which comprise a business process. This is often accomplished as part of the initial phase of systems analysis in which analysts seek to understand the existing business process. The results of such analysis may be represented using one or more activity diagrams, a UML diagram used to represent business processes [17], as is done in section 5 below.

Crowston and Osborn [28] describe two methods for identifying dependencies:

- **Dependency-focused analysis** identifies dependencies by looking for resource flows among activities as well as known coordination issues associated with one or more activities.
- **Activity-focused analysis** identifies dependencies by looking at coordination activities and asking what dependencies they coordinate. A coordination activity can be identified by looking at activities which do not appear to be directly related to production, searching for activities which match known coordination mechanisms, or looking for activities carried out by actors who have coordination responsibilities.

Once dependencies have been identified, Wyner [27] describes how a dependency diagram can be produced: all activities are categorized as production or coordination by identifying any activities which are associated with the dependencies identified. A diagram is constructed which includes only the production activities and their associated dependencies.

![Figure 2. Dependency diagram for completing a homework assignment](image)

Crowston and Osborn identify two methods for verifying the diagrams which result from such analysis: looking for gaps and ambiguities in the diagrams, and validating the diagrams with multiple process stakeholders. To these methods we would add several specific questions that might be asked in connection with a given dependency diagram:

1. Are the activities identified during the analysis properly categorized as production or coordination activities?
2. Do the dependencies identified represent coordination issues present in the process?
3. Have dependencies been identified for all coordination activities?
3.2 Dependency Diagrams, Process Design, and System Design

As noted above, the dependency diagram is intended to support process design by allowing an analyst to consider alternative mechanisms for coordinating the production activities of a process. Given that coordination is information intensive, this approach is especially relevant to IT-enabled process redesign: as the cost of IT drops and its capabilities increase, new coordination mechanisms may become feasible [29-31]. The analyst can consider new process designs in which existing coordination mechanisms are replaced by these new IT-enabled coordination mechanisms.

In choosing a coordination mechanism for a dependency, the analyst is modifying the process, since a new coordination mechanism may add, change, or eliminate coordination activities in use in the existing process. At the same time the analyst is also determining the requirements for the information system, since a coordination mechanism specifies the role played by such systems. This suggests that dependency diagrams might play a useful role in facilitating a rethinking of business process as part of an information systems design project. In the next section we describe a method for integrating dependency diagrams into the systems design process by using such diagrams to construct a set of use cases which in turn serve to define the scope and requirements for an information system.

4. THE COORDINATION ANALYSIS METHOD

In this section we describe a method for moving from a dependency diagram to a use case diagram. As discussed above, a dependency diagram depicts a business process, and a use case diagram represents a system to support this process. The former diagram conveys what people are doing; the latter describes what a system does to support these human activities. The keystone of our method is the observation that a dependency diagram can be used to identify two kinds of use cases: those that automate production activities in a process and those which coordinate dependencies. By building uses cases from dependencies, we allow analysts to consider a wide range of functionality for coordinating a process and thus incorporate process redesign into the requirements definition process.

Our method consists of the following steps, which we will discuss in further detail below:

1. Draw a dependency diagram of the business process to be supported by the proposed information system.

2. For each activity in the dependency diagram, identify zero or more use cases by considering how the proposed information system could be used to automate or support that activity.

3. For each dependency in the diagram, identify zero or more use cases by considering possible coordination mechanisms for managing that dependency and then considering how the proposed information system could be used to automate or support each coordination mechanism.

4. Draw a use case diagram incorporating the use cases identified in steps 2 and 3 which are to be included in the system scope.

We now consider each of these four steps in a bit more detail:

4.1 Step 1: Draw a dependency diagram

As noted above, Crowston and Osborn [28] and Wyner [27] provide some guidelines on how to construct and validate a dependency diagram for a given business process. The resulting dependency diagram represents the analyst’s understanding of the production activities associated with the business process and the dependencies among those activities.

4.2 Step 2: Identify use cases for each activity

Consider each production activity in the dependency diagram and, as in the automation approach, consider how the system can automate or support that activity. Each service an information system provides to a production activity is identified as a use case.

4.3 Step 3: Identify use cases for each dependency

Consider each dependency shown in the dependency diagram and consider one or more alternative coordination mechanisms. For each coordination mechanism consider what role the information system might play in supporting the coordination (note that the role played by the system will vary, depending on the nature of the particular coordination mechanism). Choose the best of these alternatives. This analysis can be divided into two steps:

1. Consider first the coordination mechanisms which are in use in the current version of the process. Most of these mechanisms can be discovered by considering the role played by the non-production activities (those omitted from the dependency diagram). Ask the question why do we need each non-production activity? In other words, what goal is supported by each of these activities? Since non-production activities serve as coordination mechanisms, when we ask this question for every non-production activity we will typically uncover the coordination mechanisms associated with each dependency in the current process.

2. To identify alternative coordination mechanisms beyond those currently in use, ask the question how else might we manage each dependency?

4.4 Step 4: Draw a use case diagram

We draw a use case diagram to summarize the system functionality identified in steps 2 and 3 above. In many cases drawing this diagram may involve no more than listing all the use cases identified in the preceding steps. However in situations where alternative coordination mechanisms have been identified for a dependency, we may identify alternative use cases which correspond to these alternative process designs. When this occurs the construction of the use case diagram involves determining which of these alternative use cases to include in the system and thereby which coordination mechanisms will be employed in the redesigned process.

Our method begins with a process representation – the dependency diagram created in step 1 – and concludes with a
representation of system functionality – the use case diagram drawn in step 4. This use case diagram can in turn serve as a point of entry into any use case driven system design method. In that sense our method is an additional tool which a systems analyst can use to support the identification of use cases as part of an existing system design method.

5. AN EXAMPLE

In order to assess the potential usefulness of our proposal we will look at how it might be applied to the analysis of a simplified business process. We will use this example to highlight how our method can provide a means for systematically examining potential IT-enabled process innovation and thereby help analysts to supplement their innate capacity to discover innovative uses of IT.

Our example is a stylized version of the medicine administration process in a hospital, Deaconess-Glover Hospital (DGH), as described in a Harvard Business School teaching case [32]. Our interest here is in identifying potential use cases for an information system in the context of this business process.

In our stylized version of the process, a doctor examines a patient and records his or her observations of the patient's condition in the patient's chart. In order to prescribe an appropriate medication, the doctor reviews the patient's chart, including medical history and the results of tests and previous examinations. Based on this information and his or her medical expertise, the doctor prescribes a medication (for example, an antibiotic) to treat the patient's condition. The pharmacist reads the prescription written by the doctor and dispenses an appropriate quantity of medication into a container (cup or bottle) marked with the patient's identification and dosage information. The pharmacist prepares a cart for each hospital floor, containing all the medications for that floor. A member of the hospital staff transports each cart to its floor. A nurse selects the medication container for a patient from the cart, verifies patient identity and dosage information, and administers the medication to the patient. The nurse updates the patient chart to record the dosage given.

A systems analyst might document this process using a description of the kind just given as well as an activity diagram like that shown in Figure 3 below. In this diagram the rounded rectangles represent activities. The placement of activities in vertical “swim lanes” identifies which actor is responsible for carrying out each activity. (For example, the Dispense medication activity is carried out by the pharmacist.) The arrows show the sequence in which activities are carried out.

We now consider how the four steps of our method might be applied in this situation:

5.1 Step 1: Draw a dependency diagram

As described above, a dependency diagram can be constructed by identifying production activities and the dependencies among them. In this case the analyst would rely heavily on the list of activities identified in the activity diagram. This analysis might result in the dependency diagram depicted in Figure 4 below.

This diagram includes four activities from the original activity diagram which have been identified as production activities because they directly contribute to the goal of providing treatment to the patient in the form of medication:

Examine patient is included because it produces the observations which are a key input to choosing an appropriate course of treatment.

Prescribe medication is included because this is the act of applying medical expertise to identify the actual treatment.

Dispense medication is included because it prepares the medication for administration to the patient.

Administer dosage is included because this is the activity which ultimately achieves the goal of providing treatment to the patient.

To these four activities we have added a fifth production activity: Obtain medication. This activity was added because the Dispense medication activity needs access to medication in order to produce the resource prepared dosage as an output. It would certainly be possible for an analyst to have included this activity in the original diagram, but this reflects our own experience in analyzing this example and illustrates how drawing a dependency diagram can help to refine one’s view of a process.

We have also identified three dependencies:

Flow: observation. The doctor prescribes treatment for the patient based on a diagnosis of the patient's condition, which in turn is based on observations of the patient. A doctor’s diagnosis may also depend on observations made by other caregivers or at other times. For example, the condition of a patient changes over time. If there are no records of previous observations of a patient, a doctor cannot compare a patient's current condition with the patient's previous state and cannot be certain, for example, whether a patient has improved or worsened. Thus a record of patient observations can help a physician choose appropriate medication to prescribe.

Fit: prescription and medication. The medication dispensed to the patient must be consistent with the prescription.

Flow: prepared dosage. The medication prepared in the Dispense medication activity produces the prepared dosage, which will be used by the nurse to administer the medication to the patient.
Doctor  |  Pharmacist  |  Hospital Staff  |  Nurse
---|---|---|---
Examine patient  
Record observations  
Review patient chart  
Prescribe medication

Dispense medication  
Prepare medication cart for floor  
Convey medication cart to floor  
Verify prescription for patient

Administer dosage  
Update patient chart

**Figure 3. Activity Diagram for Administering Medication**

**Figure 4. Dependency Diagram for Administering Medication**
5.2 Step 2: Identify use cases for each activity

In this step the analyst examines each activity in Figure 4 and determines whether it can be entirely or partially automated using information technology. Each such automation will be described as a use case. This is essentially what we have referred to as the automation approach, except that it is restricted to just production activities. Let us consider each activity in turn:

Examine patient. Constrained broadly, this activity can include laboratory testing and imaging procedures which involve information technology, but such automation is tightly coupled with the manufacture of specific medical devices and would not ordinarily be considered in scope by a systems analyst designing a hospital information system. For this reason we would expect the analyst to conclude that this activity is not going to be automated and hence it will not lead to a use case.

Prescribe medication. This activity can be automated by creating an electronic prescription and transmitting it to a pharmacy as is done with a computerized physician order entry (CPOE) application. The new system can support this activity by providing the capability to Order medication, which is thus added as a use case.

Obtain medication. Maintaining a supply of medication on hand in the pharmacy is a function which can be automated, with all the ensuing benefits of inventory management as well as the option to enforce policies concerning what medications constitute the hospital’s formulary (a list of standard medications approved for use in the hospital). For this example, which is focused on the process of prescribing and administering medication to patients, one might argue that this particular function is out of scope (a part of a separate design effort which might focus on the pharmacy’s internal processes). In retrospect this assumption was implicit in our original analysis of the situation: we did not include this activity in the original activity diagram. The dependency diagram (Figure 4) surfaced this issue of system scope, but for purposes of this example we will henceforth consider this activity to be out of scope of the current design effort.

Dispense medication. While the physical act of dispensing medications might be automated someday, this technology is not readily available today and thus our analyst would rule it out of scope. However an information system could be used to deliver prescription information (e.g. using the CPOE application described above) and the inventory status of medications. The use case for this activity can be described as Fill prescription.

Administer dosage. The physical administration of medication to a patient would be considered a manual task.

5.3 Step 3: Identify use cases for each dependency

The analyst refers to each dependency in Figure 4 and considers how it is currently managed as well as alternative approaches to managing the dependency, including especially those which might be enabled by increased use of information technology. For each coordination mechanism which the analyst considers a viable alternative, he or she identifies a use case that describes the role the information system would play in supporting that coordination mechanism. Let us consider each dependency in turn:

Flow: observation. This dependency is currently managed by recording observations of the patient and reviewing those observations prior to prescribing medication. We might simply automate these activities using an electronic medical record with use cases Record observations and View medical record. How else might this dependency be managed?

This flow dependency requires that information about the patient's condition be made available to the doctor in carrying out the Prescribe medication activity. This suggests the possibility that the medical record might actually be made available during the use case which supports this production step. That is, we might add the functionality for viewing a medical record to the Order medication use case. Similarly, the recording of observations could also be incorporated into this same use case. An interesting possibility for providing further support to the physician would be for the system to use the observation entered by the physician to suggest the most likely prescriptions. At a minimum this might save the physician time by streamlining the prescription process, but it also might remind the physician of the full range of available medications.

Fit: prescription and medication. This dependency corresponds to the requirement that the drug dispensed by the pharmacist match the drug specified in the prescription. While not stated explicitly in our analysis so far, the production activity Dispense medication would include such cross-checking, which is a core component of a pharmacist's skill and practice. Alternatives in this context would not likely involve replacing this reliance on the pharmacist's competency but rather supplementing it. For example, a potential fit issue is the use of different names for the same drug, in particular in the case of generic medication. The system might address this issue by displaying all alternative names for a medication or, in the event that the CPOE system is integrated with the inventory system, displaying the name used for the dosage currently on hand in the pharmacy. These alternatives do not lead to a new use case but rather to additional functionality to be provided in the Fill prescription use case.

Flow: prepared dosage. It is striking that of the ten activities in the activity diagram (Figure 3 above) four are part of the management of this flow dependency: Of these, Prepare medication cart for floor and Convey medication cart to floor are both concerned with making medications physically accessible. Verify prescription for patient contributes to the management of the usability of the medication by ensuring that the right medication has been provided for the patient. Update patient chart helps to manage the synchronization of this flow by helping to identify any instance where the medication was not provided to the patient. While automating the physical transport of medications from pharmacy to patient may be feasible, doing so would require extensive investment in specialized technologies (e.g. robotics or automated transport). For this reason and to keep our focus on more typical IS functionality, we will consider it to be out of scope in the current analysis. That leaves two activities to consider: Verify prescription for patient and Update patient chart. One might choose to automate each of these activities as a separate use case. However, observe that both of these activities are associated with the same dependency, which raises the question of whether they might be supported by a single use case. Rather than two separate interactions with the system, the nurse might bring up the patient record to verify that the correct dosage has been provided and then, after administering the dosage, update the record to record the dosage. This would result in a single use case Verify and record dosage. Finally, given that
updating the patient chart is done in part to identify a missed dosage, one might add functionality to the system to notify the physician of any prescription that is not recorded as administered within a certain timeframe.

5.4 Step 4: Draw a use case diagram

In the preceding steps we identified use cases associated with production activities and with dependencies. Where alternative coordination mechanisms arose we chose one to focus on for each dependency. In some cases we also determined certain use cases to be out of scope. We now capture the results of our analysis in the following use case diagram (Figure 5):

![Use Case Diagram](image)

Figure 5. Use Cases Identified by the Coordination Analysis Method

5.5 Assessment of the Coordination Analysis Method

Although the above example is meant only to illustrate how our approach might be applied in practice it does suggest how and to what extent our approach might be useful to analysts, who would ordinarily rely on process redesign based on insight, followed by application of some variant of the automation approach. The potential value of our method is its ability to identify the coordination mechanisms employed in the existing process and thus lead to a consideration of IT-enabled alternatives to these mechanisms. In the current example we see several instances where the dependency analysis pointed to such possibilities:

First, our analysis of dependencies led to combining use cases and to some extent activities in the business process. The activities Record observations, Review patient chart, and Prescribe medication are now combined into a single activity supported by the Order medication use case. The activities Verify prescription for patient and Update patient chart are supported by a single use case Verify and record dosage. In both these cases a combined use case allows the actor to review and update information on a single screen and in a single step.

Second, we identified additional capabilities for managing dependencies that go beyond the functionality of the existing process: In the Record observations use case the system will suggest medications known to be useful given the observation entered by the doctor, thus saving time and possibly providing some relevant clinical information. In the Fill prescription use case the system will display all alternative names for the prescribed medication and thus facilitate matching the prescription to medications on hand in the pharmacy.

It is important to observe that such insights are not the exclusive province of the kind of dependency analysis which we have carried out using our method. Indeed, a competent systems analyst might well arrive at many and perhaps all of these insights without using such an approach.

As noted in section 2 above, an analyst can get insight from activity diagrams such as Figure 3, which clearly indicate boundaries within a process. For example, Figure 3 clearly indicates that the transition from Prescribe medication to Dispense medication involves a hand off from Doctor to Pharmacist. By focusing on such hand offs an analyst may identify new system functionality to support coordination across these boundaries.

However, while some process insights can readily be obtained from activity diagrams, dependency diagrams are able to surface additional coordination issues. For example, the dependency diagram in Figure 4 identifies the dependency between Examine patient andPrescribe medication. Such a dependency is not shown explicitly in Figure 3 because both activities involved fall within the same swim lane. Furthermore, dependency diagrams explicitly represent dependency diagrams and abstract away the current choice of coordination mechanisms, thus inviting alternatives to be considered.

Thus we hypothesize that our method provides significantly more direct support for scanning for insights into the process and how it might be improved, where existing approaches rely more heavily on the analyst to arrive at such insights by informal and less systematic means. While a rigorous test of this hypothesis is beyond the scope of this paper we do claim that the example just given provides some preliminary support for the plausibility of this hypothesis by illustrating how the proposed method is used to arrive at such insights.

6. Conclusion

The goal of this paper is to propose a method, coordination analysis, which employs business process analysis to identify a system's requirements as a set of use cases. We applied this method to a simple example and suggested how this approach might support analysts in their quest for insight about process design. As we discussed before, the typical medication administration process in a hospital involves not only actors but also resource flows among those actors. Therefore, it is not easy to capture system requirements from process descriptions directly. Instead, our method employs an analysis of dependencies to identify potential uses of IT for managing those dependencies.

Our strategy is to represent a process using a dependency diagram and thereby leave open how the process is coordinated and the
role the system plays in this. Thus the variable (non-core) parts of a process are defined in the context of information system design. Therefore, the method becomes a set of guidelines for the transition from a dependency diagram to a use case diagram. Each resulting use case represents either a coordination mechanism or a production activity in the dependency diagram.

Our example, while admittedly simple, suggests two potential benefits of our method. First, our approach promotes the consideration of alternative coordination mechanisms for managing each dependency and thus leads to a consideration of alternative process designs informed by consideration of "a rich library of generic coordination mechanisms" [20]. Second, our method focuses more attention on coordination activities, which are especially amenable to redesign using information technology.

The intended use of our method is as a supplement to existing systems analysis and design methodologies. Where an existing methodology might call for an analyst to move directly from a map of the existing process (e.g. Figure 3 above) to a set of use cases (e.g. Figure 5), the coordination analysis method offers an intervening step to bridge between the two. Thus the activity diagram in Figure 3 is used to construct a dependency diagram (Figure 4) and this dependency diagram is then used to analyze activities and dependencies in order to explore potential changes to the process as well as system functionality, resulting in a set of use cases (Figure 5).

6.1 Limitations and Future Research Directions

Our method is still in the preliminary stage and has not been applied in a full-scale business environment. To date we have applied this method to several small examples such as the one described in this paper. We need to assess this method in actual systems design settings and obtain feedback from practitioners, including systems analysts and IT project managers. In addition, we do not yet have empirical results as to how our method affects the process and outcomes of system development. Given the difficulty of a rigorous comparison of design methods in the field, a possible next step might be to assess the relative strength of our method in a laboratory setting.

7. ACKNOWLEDGMENTS

We gratefully acknowledge research support for this study provided by Boston University School of Management’s Institute for Leading in a Dynamic Economy (BUILD) and System Research Center (SRC).

8. REFERENCES


