

Towards an Ontology of Task Dependence in Organizations

Mena Rizk^a, Mark Fox^b and Daniela Rosu^c

Enterprise Integration Laboratory, University of Toronto, 5 King's College Road, Toronto, Canada

Keywords: Coordination, Enterprise Modelling, Ontology, Organization Model, Task Dependence.

Abstract: In the face of an increasingly dynamic, complex, and uncertain task environment, effective coordination is crucial for organizational success. Based on a real-world case study investigating the nature of coordination challenges in a municipal infrastructure project, we identified shortcomings in the representational frameworks offered by organization studies and enterprise modelling that limit their ability to effectively model task dependence and assist in improving coordination. Starting from existing organizational research literature and domain expertise, we conducted an ontological analysis of task-related concepts, formulated representational requirements, and proposed a formalization. Our approach defines task dependence in terms of the constraints one task imposes on another, underpinned by novel constructs that define how and why a task is constrained. These constructs support the inference of dependencies between tasks, facilitating the discovery of potentially hidden, latent dependencies. We formalized our conceptualization in an ontology, detailed herein using first-order logic. Consistency-verified implementations in Prover9 and OWL are provided. We validated our approach by modelling and solving real-life scenarios provided by our domain-expert collaborators. Our approach lays the groundwork for future extensions that will tackle the modelling of different forms of dependence between agents within an organization.


1 INTRODUCTION


This paper introduces a novel model of task dependence designed to assist organizations in overcoming coordination challenges in complex, dynamic, and unpredictable work environments.


The focus of this paper is on task dependencies — the dependencies between tasks performed by agents, rather than the agents themselves. While acknowledging the existence of other forms of dependence like epistemic, reward, and outcome (Puranam et al., 2012; Raveendran et al., 2020), we intentionally initiate our endeavour at the task level with the aim to robustly represent the inherent circumstances that necessitate coordination, and the tangible consequences of dependencies on task execution. This allows for the modelling of the irreducible dependencies that pertain to the nature of work undertaken by an organization, and the subsequent implications on the integration of work, irrespective of agents' work assignment.

We grounded our efforts in a case study (Section 2) conducted with one of North America's ten most

populous cities, identifying coordination challenges within a large-scale infrastructure project, and analyzing the representational requirements for reasoning about their resolution. In Section 3.1, we present an ontological analysis of task, task structure, and task dependence concepts and introduce definitions for each concept that are informed by both the existing organizational literature and the use cases provided by domain experts. A review of the relevant literature (provided in Section 3.2) discusses the representational limitations of the existing frameworks. Subsequently, we present our working conceptualization of task and task dependence, in Section 4. Its formalization in first-order logic is shown in Section 5. We validated our approach's capacity to represent dependencies and demonstrated its reasoning affordances by applying it to scenarios from a real-life case study (Section 6). We discuss its advantages and limitations in Section 7 and conclude with our future research direction in Section 8.

^a  <https://orcid.org/0009-0008-6095-0100>

^b  <https://orcid.org/0000-0001-7444-6310>

^c  <https://orcid.org/0000-0002-5877-9681>

2 MOTIVATION

2.1 Case Study

The research discussed herein is based on a partnership with the municipal government of one of the ten largest cities in North America (henceforth referred to as “the City”). The objective of this ongoing partnership is to explore the potential for information technologies in aiding the City’s coordination of complex infrastructure projects. With the City-led infrastructure projects growing larger, faster, and more complex, effective coordination across the City’s divisions, stakeholders, projects, and resources has become critical for successful project delivery. A case study of a major current infrastructure project provided initial insights into the current coordination practices and identified areas of potential improvement using information technology.

The undertaking discussed in this section is concerned with the City’s flood risk mitigation efforts within one of its most flood-prone areas. The Riverine Flooding Project (RFP), aimed at reducing riverine flooding risks in the neighbourhood, involves designing and implementing several pieces of infrastructure such as bridges and concrete channels. Concurrent with this, the City’s Urban Flooding Program (UFP) is implementing sewage infrastructure improvements that must be coordinated with the RFP. There are also several planned capital works projects in the area that the RFP must consider in its design. As operating in this complex environment entails intense coordination across various organizations, stakeholders, and tasks, we had access to a wealth of coordination phenomena and challenges to model and help address.

2.2 Challenges

While a detailed report on the case study will be released separately, we outline our findings on the nature of the coordination challenges for the purpose of this paper. We identified three main types of coordination challenges: participation, navigation, and cross-cutting collaboration. *Participation* relates to delays caused by inadequate stakeholder responsiveness. *Navigation* involves difficulties in identifying affected stakeholders and accounting for their needs. *Cross-cutting collaboration* is about synchronizing the efforts of numerous stakeholders across bureaucratic silos to meet interdependent objectives, as opposed to independent pursuits that overlook the broader project context.

Collectively, these challenges present considerable risks to the quality, budget, and timeline of in-

frastructure projects. Indeed, we identified cases of delays, cost overruns, and suboptimal implementations that required rebuilding due to failures in participation, navigation, and cross-cutting collaboration.

To successfully respond to these coordination challenges, it is essential to identify the dependencies necessitating coordination, the causes of coordination failures, and strategies to mitigate the risk of future failures. Therefore, an effective AI-enabled intervention should be grounded in a robust representation of dependence and the entities through which dependencies materialize. In particular, we require the ability to automatically infer instances of dependence, and capture the underlying causes and effects of dependencies. As we will show in Section 3, the existing definitions of dependence and approaches to modelling tasks are insufficient for addressing the observed coordination challenges. We tackled these shortcomings by making explicit the representational requirements for capturing tasks and task dependencies and developing a new approach to modelling them, which we introduce in the next section.

3 LITERATURE REVIEW

3.1 Ontological Analysis of Task Dependence

This section describes the ontological analysis we performed in order to provide a solid foundation for the design and development of our framework. We examined the organizational literature in pursuit of understanding the underlying semantics of the concepts of interest, and their relationships, as well as uncovering any implicit assumptions or inconsistencies that may be present. We established representational requirements for our framework based on the previously outlined challenges and then generated a set of dimensions that are relevant for effectively defining task dependence. These dimensions were subsequently used to evaluate existing definitions and implied conceptualizations in the existing literature, and then propose our own definitions. With these in hand, we reviewed the literature relevant to task modelling (Section 3.2) in order to find frameworks that could help operationalize our definitions and shape our model.

3.1.1 Dimensions of Task Dependence

As touched upon in Section 2, a representation of the dependencies that call for coordination is a prerequisite for any reasoning about the observed coordination

challenges. Specifically, we need a model that represents the dependent entities (the subjects in need of coordination), the situations that make them dependent (the necessity for coordination), and the implications that the dependency has for them (the function of coordination). From these reasoning tasks, a distinct set of representational requirements becomes apparent for any model that we wish to utilize to aid our use cases. More precisely, we need a model that captures the organization's work, the dependency relationship's basis between work, and the dependency's effects on the performance of work. Furthermore, we need to capture each of these three requirements at various levels of abstraction in the task structure, such as the goal level and detailed activity level.

Our model must include a representation of the activities to be performed and the desired states to be achieved (i.e., goals). Therefore, our model must distinguish between states, the desire to achieve states, the intention to work towards desired states, and the specific activities performed to work towards desired states. The "structure" of work, often referred to as the task structure, must also be represented, which essentially describes the decomposition of work.

To infer and model dependencies, we need to capture the fundamental "basis" of dependence between two tasks. This can be seen as the core reason one task may depend on another, or the irreducible essence of a dependency relationship between two tasks (Herbsleb and Roberts, 2006). To comprehend why a coordination failure may have occurred or to develop strategies and mechanisms to mitigate them, we need to capture the "effect" of a task dependency on the dependent task. This effect indicates how a dependency influences task execution.

Moreover, to reason about task dependencies in scenarios where the dependency's exact nature cannot be detailed, due to it being unknown or changing too rapidly to warrant a specification, we need the definition of dependence to consider how it can be applied to capture dependencies at various abstraction levels of the work being analyzed. This highlights the dual usage of the term "task" in the literature (achieve a goal versus perform an activity), which we aim to disambiguate by this section's end. For now, we refer to this requirement as the "context" of a dependency.

Therefore, to meet the needs of our representational requirements, any approach for modelling task dependence must explicitly (1) distinguish between agent and task dependence; and allow for the (2) "basis", (3) "effect" and (4) "context" of task dependence to be made explicit. We will refer to these four conditions in Section 3.1.5.

The underlying definitions for task and task struc-

ture must also be accounted for and must be consistent, as these are the entities between which dependencies exist, and the task structure serves as the medium for these dependencies.

3.1.2 Definitions from Organizational Research

Multiple attempts to define task dependence exist in the organizational literature. We focus our analysis on four definitions due to their distinct perspectives, allowing us to examine different elements of their definitions regarding the four dimensions we seek to analyze. For a more comprehensive review of task dependence, we refer readers to (Raveendran et al., 2020).

Thompson (1967) provides one of the earliest definitions of task dependence, grounded in the relationship between tasks' inputs and outputs. He proposed three types of possible task dependencies: *pooled* (both tasks contribute to the same output), *sequential* (one task's output is the input to another), and *reciprocal* (both tasks feed into each other). Crowston (1994), drawing from distributed artificial intelligence literature, asserts that two tasks can only be dependent via a common resource (which includes states). He specifies three possible types of dependencies: *overlapping effects* (two tasks contribute to producing, or causing if the resource is a state, the same resource); *overlapping preconditions* (two tasks require the same resource); and *overlapping effects and preconditions* (the resource produced by one task is required by another). More recently, coming from the emerging microstructural perspective, Puranam (2018) suggests task interdependence is present when the value generated from performing each task is different, depending on whether the other task is performed or not. Finally, Raveendran et al. (2020) consider task dependence's role in informing organization designs, depending on the nature of work being well or ill-understood, further highlighting the context of task dependence.

3.1.3 Analysis of Task

Prior to analyzing the four approaches' interpretation of task dependence, we need to grasp their fundamental assumptions about the nature of tasks and their structure.

Thompson, while not explicitly defining a task, implies that a task is a specific function or activity embedded within a workflow, undertaken by agents with defined inputs and outputs. The concept of "intentionality", or the goal achieved by a task, is absent from this definition. While intentionality may be argued to be implicitly defined through the workflow in which a given task is embedded, it may become increasingly

obscured as the task defined becomes more granular.

Crowston expands on this by considering tasks to encompass both goal achievement and activity performance. He characterizes goals as desired states and activities as actions taken to achieve particular states. However, Crowston's harmonization of "achieving goals" and "performing activities" under the single concept of "task" is problematic. Goals and activities are inherently different; a goal is a desire towards a state, while an activity is an action causing a state. Moreover, Crowston's confounding of resources with states, and outcomes with outputs, further limits the operationalization of this approach for defining task dependence. To illustrate the limitations of this approach, we can reference an upper-level ontology, such as DOLCE (Borgo et al., 2022). Resources and outputs (resources that are produced by an activity) are *endurants* while states and outcomes (states caused by an activity) are *perdurants*.

Puranam provides a more explicit definition, describing a task as a transformation of inputs into outputs in finite time, with an associated value derived from the difference between inputs and outputs. However, the relationship to a task's corresponding goals is not defined. Finally, Raveendran et al. follow Puranam's definition but separately consider goal dependence. They focus on dependencies between agents sharing common goals rather than dependencies between goals themselves. This approach leaves gaps in capturing task dependencies when the particular transformations for achieving these goals are not clearly defined.

3.1.4 Analysis of Task Structure

In Thompson's approach, the conceptualization of task structure does not extend beyond the notion of "workflow" and the categorization of activities into input, technological, and output tasks.

Crowston's view of task structure can be described as a "means-ends" decomposition of intended "effects" (goals) into possible subgoals and primitive activities. This approach attempts to streamline the analysis of tasks by harmonizing goals and activities. However, the confounding of states, intentions, and actions confuses the definition, preventing a clear and unambiguous formal representation of task structure.

Puranam defines a task structure as the most fine-grained means-end decomposition of an organization's goals into its constituent tasks and their interdependencies, operationalized using design structure matrices (Baldwin and Clark, 2000). However, this approach raises a couple of issues. Firstly, the absence of explicit representation of goals, intermediate goals, and their relationship to activities is prob-

lematic. Unlike Crowston, who incorporated the decomposition of goals into actions in his notion of task structure, Puranam's approach lacks this dimension explicitly. Instead, decomposition is implicitly defined through the clustering of interdependent tasks. This absence hinders us from modelling the connection between a given goal and the actions required to realize it, largely limiting our ability to precisely represent the decomposition of high-level tasks into their constituent sub-tasks and corresponding sub-states they are intended to achieve. Secondly, the modelling of decomposition is paradoxically bottom-up rather than top-down. Task structure is defined based on task dependencies, rather than defining task dependence based on task structure. This can be limiting when our goal is to infer dependencies between tasks that are hard to detect, given a representation of the tasks.

The limitations associated with Puranam's approach to modelling task structure are also applicable to Raveendran et al.'s approach. They suggest that the level of understanding of a work environment can be expressed in terms of the extent to which the underlying task structure can be modelled in terms of detailed tasks. However, this approach does not solve the issues raised in Puranam's, maintaining the difficulties for our purpose of inferring obscure dependencies.

3.1.5 Analysis of Task Dependence

In this section, we explore how each of the four task dependence approaches meets the four conditions specified at the end of Section 3.1.1.

Separation from Agent Dependence. Thompson defines dependence between workflows across organizational units, implying task dependence is agent dependence caused by the relationship between the tasks of the dependent agents. Conversely, Crowston and Puranam assert task dependence as separate from agents. Furthermore, Puranam shows that task dependence is neither a necessary nor sufficient condition for agents of interdependent tasks to be interdependent, challenging Thompson's perspective. Instead, the relationship between task and agent dependence is influenced by the extent to which the tasks are understood, as noted by Raveendran et al. (2020). In well-understood environments, agent dependencies stem from task dependence, through task allocation decisions. In less clear environments, task dependence emerges from the collective sense-making processes shaped by agent dependence.

Basis of Task Dependence. Thompson views task dependence as arising from workflow directionality, which is defined in terms of overlaps between the inputs and outputs of tasks. In Crowston's approach, the

basis of dependence is the common resource through which two tasks have an overlap in their preconditions and/or effects. Both approaches have limitations. Thompson's approach overlooks dependencies based on information flows that are not necessarily isomorphic with workflow directionality. Crowston's approach, however, captures some of these dependencies, though not without its own limitations, as will be shown shortly. For instance, Malone and Crowston (1994), show that a producer-consumer relationship between two tasks (i.e., the output of one task is the input of another) may be due to either an *inventory* (the second task can only execute after the output of the first is available) or *usability* constraint (the first task must produce its output in a way that is usable by the second task). The former constraint, a sequential dependency in Thompson's language, matches the workflow direction. However, the latter constraint, grounded in information flow, runs in the opposite direction to workflow and would not be definable with Thompson's approach.

While broader than Thompson's basis, a limitation of Crowston's approach is that it is grounded in *whether* a common resource exists between two tasks, without an explicit representation of the underlying reason why the common resource forms a dependency. For instance, the distinction between *inventory* and *usability* constraints is left to a human reasoner applying the model. This distinction is important since the reason why a common resource causes a dependency has implications for how the dependency ought to be managed, as we'll see shortly when discussing "effects".

Puranam and Raveendran et al. propose a different perspective, focusing on the mutual influence between tasks on the value they generate as the basis of dependence. However, the basis here is overly constrained as it only considers if the value of one task varies with *if* another task is performed, rather than *how* it is performed. Additionally, though "value" is intended to be an abstract construct which can be arbitrarily defined, the approach does not allow for an explicit representation of why a differential in the value of a task exists in the first place.

Effect of Task Dependence. Thompson's approach to managing task dependencies involves various coordination mechanisms that vary in "strength" based on the type of task dependence. This suggests effects of task dependencies include interaction intensity and complexity. However, Thompson's framework does not capture the precise ways in which a task can be constrained by dependencies.

Crowston and Puranam also touch on "effects" but don't explicitly address how tasks may vary in

their performance due to dependencies. For each type of dependency, Malone and Crowston (1994) offer a set of coordination mechanisms for managing the dependency, but they do not explicitly model the precise way in which the coordination mechanisms affect how tasks are performed (i.e., why the coordination mechanisms work). For Puranam's approach, the specific dimensions of a task's performance that may change due to its value relation to another task are not explicitly defined. Without a representation of the "effects" of dependence on the performance of a task, we are limited in our ability to reason about the efficacy of alternative coordination strategies for a given dependency.

Context of Task Dependence. Given that the approaches offered by Thompson, Puranam, and Raveendran et al. view tasks as well-defined operations, they are limited in their ability to capture task dependencies in environments where the nature of work is not well understood and tasks cannot be specified beyond the goals they are intended to achieve (i.e., the specific activities are either unknown or cannot be explicitly represented). Crowston's definition of tasks as both the achievement of goals and the performance of activities allows for the capturing of dependencies in ill-understood work environments, where only dependencies between the achievement of goals can be represented. However, due to the limitations specified in Section 3.1.3, operationalizing Crowston's approach is a challenge.

3.1.6 Proposed Definitions

Based on the ontological analysis of tasks, task structure, and task dependence, we have demonstrated that the approaches offered by the organizational literature are neither consistent with one another, nor do they adequately satisfy the conditions that we require to reason about addressing the coordination challenges we found in practice. Below, based on our ontological analysis and required conditions, we propose a set of definitions that form the basis of our model:

- **Task:** The intention of an agent(s) to work towards some goal. A well-understood task can be further specified in terms of activities that when executed cause the desired states, whereas an ill-understood task can only be specified in terms of the goal that is intended to be accomplished.
- **Activity:** An activity can be performed by an agent, and may have inputs (required resources) and outputs (produced resources). An activity causes outcomes, which are the consequences of an activity's performance. Activities may have characteristics which define particular ways in

which the performance of the activity can vary.

- **Task Structure:** A task structure is a decomposition of tasks into the constituent activities that must be performed, and the corresponding states that must hold in order for the goal of the task to be achieved. Activities that cause states may be decomposed into subactivities that cause sub-states, and so forth. In essence, a task structure specifies “how” a task to bring about some goal is achieved in terms of the activities that must be performed and the states that must be true.
- **Task and Activity Dependence:** Task A is dependent on Task B if the way in which the goal of A is achieved is constrained by the way in which the goal of B is achieved. Activity A is dependent on Activity B if the way in which A is performed is constrained by the performance of B.

Here, we explicitly capture that the basis of the dependency is that there are constraints placed on the performance of activity A by the performance of B. Additionally, the concept of “value” is implicitly captured since it can be assumed that the basis of constraint is that in order for the activity to be performed in a way that maximizes its value, the particular way in which it should (or must) be performed may vary with *how* another activity is performed (rather than just *if*). More so, the distinction between task and activity dependence allows us to capture dependencies between work in both well- and ill-understood environments.

3.2 Related Task Frameworks

The concept of a task has been extensively examined across various domains, ranging from psychology (Annett and Duncan, 1967) to organizational studies and information and computer science (Vernadat, 2020; Alam et al., 2015; Guizzardi et al., 2013; Yu and Mylopoulos, 1995; Greenspan et al., 1994), presenting an array of perspectives and methodologies. These span from cognitive theory-based methods focusing on human thought and behaviour, to methods designed to endow robots with autonomy.

An early, and subsequently prominent task modelling framework, the Hierarchical Task Analysis (HTA) (Annett and Duncan, 1967) supported the view that a task is “any piece of work that has to be done” and that task performance “is a goal-directed behaviour”. Their approach, departing from earlier approaches, was to offer functional analysis rather than behavioural descriptions of tasks, initiating the analysis from the task’s goals rather than associated activities. However, despite its name, HTA, does not,

in fact, emphasize tasks, but focuses instead on goals and the operations to achieve them. Due to its ontological commitments, this framework, and its subsequent extensions, do not provide the means to capture and distinguish an activity from the intent to perform it, which is an essential requirement that our representational framework strives to meet.

Successive approaches (Phipps et al., 2011; Stanton, 2006) emerged in organizational sciences as well as computer and information science. Of particular relevance to our work are the frameworks developed from the field of enterprise modelling, which sits at the intersection of IT, information science, and organizational studies. Also relevant, from the computer science literature, are frameworks emerging from the fields of requirements engineering (e.g., i*, KAOS) and AI planning.

Enterprise modelling, defined by Fox and Grüniger (1998) as “a computational representation of the structure, activities, processes, information resources, people, behaviour, goals and constraints of a business, government, or other enterprise”, has generated various complementary frameworks over four decades. These are centred around activities (e.g., IDEF, MERISE), business processes (e.g., ARIS, CIMOSA), and enterprise knowledge (e.g., DEMO by Dietz, 1999; and TOVE by Fox et al., 1995). While these frameworks incorporate elements of agents, goals, tasks, and activities, none distinguish between an activity and the intent to execute it.

AI planning shares similar concerns with our current work, striving for computationally executable plans along with methodologies for social and cognitive plans. Existing methodologies (e.g., Chandrasekaran and Josephson, 1997; and Bermejo-Alonso, 2018) do attempt to capture the activity to perform and how it should be executed as part of a plan. However, they do not account for the intent to perform an activity, a crucial distinction from the goal the activity is designed or expected to achieve.

There are also domain-independent efforts, such as upper-level ontology work (e.g., BFO¹, UFO², DOLCE³, DOLCE-Lite-Plus⁴) that offer a mechanism for modelling agents, goals, tasks, and actions. DOLCE-Lite-Plus, in particular, defines a task as a “course used to sequence activities or other controllable perdurants”, where a course is a “concept that selects (in particular, it sequences) perdurants (processes, events, or states), as a component of some s-

¹<https://basic-formal-ontology.org/>

²<https://nemo.inf.ufes.br/en/projetos/ufo/>

³<http://www.loa.istc.cnr.it/dolce/overview.html>

⁴<https://www.w3.org/2001/sw/BestPractices/WNET/DLP3941.daml.html>

description” (which relates, roles, endurants and contexts). In effect, tasks are (the desired) targets of some role played by an agent and can relate to ground activities or decision-making. In this framework, tasks are explicitly declared to be disjoint from actions and action types, an ontological stance that is compatible with our representational requirements.

4 CONCEPTUAL MODEL

In this section, we introduce our conceptual model, with each subsection covering a different layer. Figure 1 displays the ontology design pattern, highlighting the main concepts and relations discussed.

4.1 Agent, Intentions, and Desires

The foundational concepts in our model are agents, their tasks, and their goals. We define an Agent as an entity capable of possessing goals, having intentions to strive for those goals, and executing actions to attain them. Agents can be individuals, teams, departments, or entire organizations and even temporarily formed entities like committees or task forces. A Task signifies an agent’s intention to exert effort towards a goal. It is defined in terms of the agent with the intent and the goal, which is the target of this intent. A Goal represents a desired “state of affairs”. It is defined in terms of the agent who has the goal and the desired State (detailed later) of the agent. At this stage, we don’t consider the reasons behind an agent’s goal or task. Also worth noting, our model doesn’t incorporate organizational roles since our focus is on capturing dependencies between work, not dependent on specific organizational settings or formal structures.

Tasks may also be defined via Activities through which goals can be achieved. The difference between tasks and activities is important yet subtle. Tasks convey the intent to work towards a goal, while activities are concrete operations that might lead to a state, regardless of whether that state is desirable. Tasks represent the intention to act towards a goal, while activities specify the exact manner in which one can act.

This distinction is useful in contrasting well-defined and ill-defined task structures. For well-defined task structures, where necessary actions to achieve a goal are known, we can understand the specific contribution of each agent. Conversely, in ill-defined task structures, where agents’ specific actions are unpredictable or can’t be predefined, we can still capture their collective intention towards the shared goal. Furthermore, this distinction allows us to ex-

press task dependence in terms of achieving goals rather than merely performing activities. In scenarios where the accomplishment of one goal influences or restricts another, but the exact method of influence can’t be predetermined (as in ill-defined task structures), we model tasks as the intention to work towards a goal. This way, we can articulate the influence of one goal on another through the tasks involved in achieving both goals. Thus, the achievement of a state may be constrained by how another state is achieved, even if the precise “how” (i.e., an activity) is not known or cannot be represented. This gives us a higher level of representation, at the level of “intention to act towards” rather than the specific “act”.

4.2 Activities and Resources

An Activity refers to a well-defined operation that an agent can perform as part of a task, causing an outcome state. It may also be enabled by a state and may require or produce a Resource. An activity can be seen as a production technology (Puranam, 2018) or an IDEF0 function. An activity may have an Activity Decomposition, which breaks down complex activities into constituent sub-activities. This construct captures relationships between activities at different abstraction levels, reminiscent of hierarchical modelling in IDEF0 or the *AggregateActivity* construct in TOVE (Fox et al., 1993). Explicit modelling of the decomposition as a construct allows representation, querying, and visualization of task structures at various abstraction levels. Capturing the specific ways in which an activity’s performance can vary is crucial for activity dependence (the performance of one activity being influenced by another). For this, we use the Activity Characteristic construct, defining a feature of an activity subject to variation. Not every activity characteristic may be part of a dependency. We classify these characteristics into five main categories (not intended to be collectively exhaustive):

- *Input* characteristics: Variations in activity inputs such as *material selection*, *resource consumption*, *cost*, and *information source*.
- *Process* characteristics: Variations in the *method* or *implementation* of an activity.
- *Output* characteristics: Activity performance dimensions that define the output, like *quality*, *quantity*, and *design*.
- *Spatial* characteristics: Activity performance dimensions related to the physical or virtual *location* where an activity occurs.
- *Temporal* characteristics: Temporal features such as *start time*, *end time*, and *duration*.

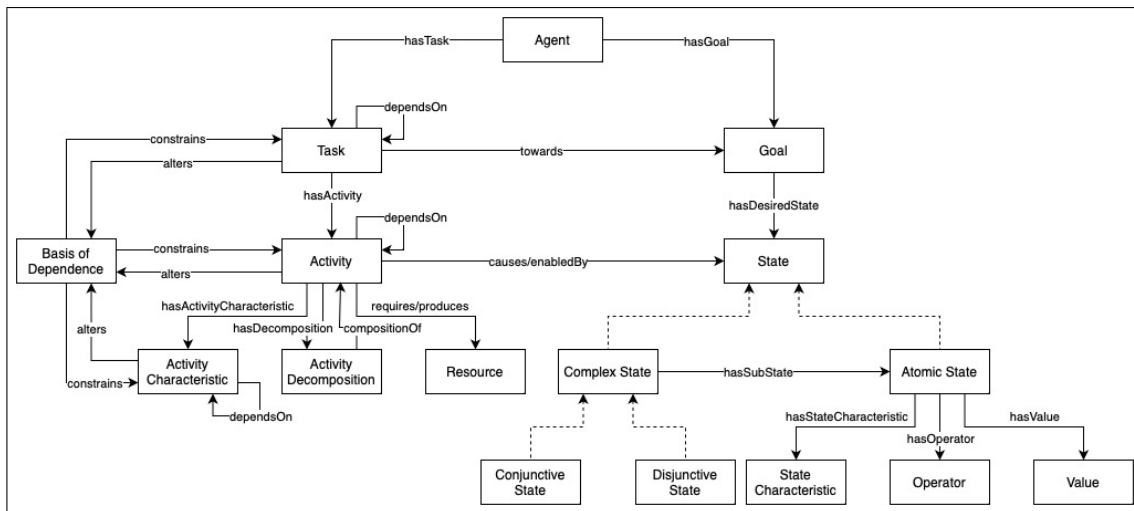


Figure 1: Task dependence ontology design pattern showing main classes and properties (dashed arrows for subclassOf).

4.3 States

A State describes a particular aspect of an object or situation, capturing the idea that the modelled object or situation can have varying conditions over time. States can be complex or atomic. A Complex State can be either a Conjunctive State or a Disjunctive State, representing the conjunction or disjunction of other (sub)states, respectively.

An Atomic State is defined in terms of a state characteristic, operator, and value combination. The State Characteristic is an identifiable property of an object or situation. The Operator defines the relationship between a characteristic and value (e.g., \leq , \geq , $=$, \neq). The Value represents the specific value of the state characteristic defined in an atomic state. It can be based on a unit of measure, an ordinal or nominal scale, or any data type. This flexibility allows for the representation of specific states and reasoning about the achievement of goals that can be satisfied as well as satisfied (Simon, 1947), a feature also seen in i^* (Yu and Mylopoulos, 1995). This is useful when the exact condition for state satisfaction cannot be predetermined (e.g., minimize construction costs) or when a particular state characteristic is hard to quantify (e.g., improve design aesthetic).

4.4 Dependence

A Basis of Dependence captures the underlying reason for one task, activity, or activity characteristic depending on another, establishing the nature of their relationship. This construct is key to our model, as it underpins how task and activity dependencies are inferred. A basis of dependence may constrain or be altered by how a task is carried out, an activity is per-

formed, or an activity characteristic varies. If a task or activity (or one of its characteristics) is constrained by a basis of dependence and that basis is altered by another task or activity, then the former entity depends on the latter. The semantics of the depends on relation vary based on the class of entities being related.

Also crucial to defining a basis of dependence is its dependum – the entity through which a basis of dependence exists. For example, if an activity produces a resource used by another, then the availability of the resource is the basis of dependence and the resource is the dependum. While this is not an exhaustive classification of all possible bases of dependence, we've identified six that collectively allow the representation of a variety of dependencies between organizational tasks and activities:

- *Availability*: The availability of a common resource (either as a common input or intermediary object) may be the basis of dependence between two tasks or activities. Here, the common resource is the dependum.
- *Functional*: Arises when the ability to perform an activity (both dependum and subject of dependence) varies with if or how another activity is performed or task accomplished.
- *Complementarity*: This basis of dependence comes into play when the overall effects of two activities or tasks are either greater or less than the sum of their parts. The non-additive state characteristic is the dependum here. For example, to minimize traffic disruptions, a municipality may seek to bundle several planned maintenance activities (e.g., road repavement, sewer improvements, utility relocation) to occur at the same time.

- *Compatibility*: This basis of dependence arises when two tasks or activities need to coexist, due to some state or activity characteristic condition, but may not functionally influence each other.
- *Uncertainty*: This basis of dependence comes into play when two tasks or activities depend on each other due to the uncertainty of a state or activity characteristic.
- *Complexity*: This basis of dependence applies when the exact nature of the dependency relationship between two tasks or activities is unknown or cannot be explicitly stated.

We don't impose any set relations between these bases of dependence; depending on the domain the ontology is applied to, some bases may be sub-bases of others or disjoint. The complexity basis may serve as a catch-all for bases of dependence that can't be captured by the ones we've defined.

To illustrate the model, consider two activities requiring the same non-shareable, non-consumable, unary-capacity resource. An availability basis of dependence exists between them through the shared resource. Since the start time of either activity affects the resource's availability, both activity characteristics "alter" the basis of dependence. As the resource availability constrains when each activity can start, the basis of dependence "constrains" both activities. Hence, they both depend on each other.

5 FORMALIZATION AND IMPLEMENTATION

In this section, we provide a brief snapshot of our ontology-based formalization of the conceptual model, formulated in first-order logic (FOL). Chosen for its expressivity and the ability to capture advanced rules, FOL equips us with the complex reasoning capabilities necessary for our use cases, such as inferring instances of dependence. Despite its potential lack of decidability that might make alternatives like description logic appealing, we opted to first construct our model "as intended" before translating into a description logic, avoiding upfront constraints on expressivity and inference.

Space constraints necessitate us to only include in this paper the axiomatization of a subset of the dependency inferences between tasks, activities, and activity characteristics. Other constructs underpinning axioms (1) to (5) are summarized in Table 1. The current version of the full axiomatization can be found in the

Task Dependence Ontology Github repository⁵. The FOL specification was implemented in Prover9 and verified for consistency. Additionally, we developed a consistency-verified OWL (and SWRL) implementation to allow for further testing and future integration of the ontology into decision-support tools. Both implementations can be found in the repository.

An activity characteristic *dependsOn* another activity characteristic if there is a basis of dependence that constrains the former and is also altered by the latter:

$$\begin{aligned} &\forall b, a_1, a_2, c_1, c_2 (\text{constrains}(b, c_2) \\ &\quad \wedge \text{alteredBy}(b, c_1) \wedge (c_1 \neq c_2) \\ &\quad \wedge \text{hasActivityCharacteristic}(a_1, c_1) \\ &\quad \wedge \text{hasActivityCharacteristic}(a_2, c_2) \\ &\quad \rightarrow \text{dependsOn}(c_2, c_1)) \end{aligned} \quad (1)$$

If an activity characteristic *dependsOn* another activity characteristic, then the activity of the former characteristic is dependent on the activity of the second characteristic:

$$\begin{aligned} &\forall a_1, a_2, c_1, c_2 (\text{Activity}(a_1) \wedge \text{Activity}(a_2) \\ &\quad \wedge \text{hasActivityCharacteristic}(a_1, c_1) \\ &\quad \wedge \text{hasActivityCharacteristic}(a_2, c_2) \\ &\quad \wedge \text{dependsOn}(c_2, c_1) \wedge (a_1 \neq a_2) \\ &\quad \rightarrow \text{dependsOn}(a_2, a_1)) \end{aligned} \quad (2)$$

If an activity *dependsOn* another activity, then the task that the former activity is a part of is dependent on the task that the second activity is a part of:

$$\begin{aligned} &\forall t_1, t_2, a_1, a_2 (\text{Task}(t_1) \wedge \text{Task}(t_2) \\ &\quad \wedge \text{hasActivity}(t_1, a_1) \wedge \text{hasActivity}(t_2, a_2) \\ &\quad \wedge \text{dependsOn}(a_2, a_1) \wedge (t_1 \neq t_2) \\ &\quad \rightarrow \text{dependsOn}(t_2, t_1)) \end{aligned} \quad (3)$$

An activity *dependsOn* another activity if there is a basis of dependence that constrains the former and is also altered by the latter:

$$\begin{aligned} &\forall a_1, a_2, b (\text{Activity}(a_1) \wedge \text{Activity}(a_2) \\ &\quad \wedge \text{constrains}(b, a_1) \wedge \text{alteredBy}(b, a_2) \\ &\quad \wedge (a_1 \neq a_2) \rightarrow \text{dependsOn}(a_1, a_2)) \end{aligned} \quad (4)$$

A task *dependsOn* another task if there is a basis of dependence that constrains the former and is also altered by the latter:

$$\begin{aligned} &\forall t_1, t_2, b (\text{Task}(t_1) \wedge \text{Task}(t_2) \wedge \text{constrains}(b, t_1) \\ &\quad \wedge \text{alteredBy}(b, t_2) \wedge (t_1 \neq t_2) \rightarrow \text{dependsOn}(t_1, t_2)) \end{aligned} \quad (5)$$

6 APPLICATION

In this section, we describe our framework's application to the project introduced in Section 2 to validate its representational effectiveness (visualized in

⁵<https://github.com/rizkmena/Task-Dependence-Ontology>

Table 1: Summary of axiomatization of the task dependence ontology’s core concepts.

Concept	Summary of Axioms
<i>Agent</i>	An entity that has a <i>hasTask</i> relation to a <i>Task</i> and a <i>hasGoal</i> relation to a <i>Goal</i> .
<i>Task</i>	Has a <i>taskOf</i> relation with the <i>Agent</i> that has the <i>Task</i> , and the <i>Goal</i> that the <i>Task</i> is towards. May have <i>Activity</i> (s) associated with it via a <i>hasActivity</i> relation.
<i>Goal</i>	Defined in terms of a <i>goalOf</i> relation with the <i>Agent</i> that has the <i>Goal</i> , and a <i>hasDesiredState</i> relation with the <i>State</i> that the goal is about.
<i>Activity</i>	Can be <i>performedBy</i> an <i>Agent</i> , is an <i>activityOf</i> a <i>Task</i> , and <i>causes</i> a <i>State</i> . Note that the <i>State</i> that is <i>causedBy</i> an <i>Activity</i> is an <i>Outcome</i> . An activity may also be <i>enabledBy</i> a <i>State</i> .
<i>ActivityDecomposition</i>	Is a <i>decompositionOf</i> an <i>Activity</i> and a <i>compositionOf</i> at least two other <i>Activity</i> entities (which are the subactivities of the decomposed activity).
<i>ActivityCharacteristic</i>	An <i>Activity</i> may have a <i>hasActivityCharacteristic</i> relation with an <i>ActivityCharacteristic</i> . <i>TemporalCharacteristic</i> , <i>SpacialCharacteristic</i> , <i>InputCharacteristic</i> , <i>ProcessCharacteristic</i> , and <i>OutputCharacteristic</i> are types of <i>ActivityCharacteristic</i> .
<i>Resource</i>	An entity that is either <i>requiredBy</i> or <i>producedBy</i> an <i>Activity</i> . Can be a <i>Shareable</i> or <i>NonShareableResource</i> , and a <i>Consumable</i> or <i>NonConsumableResource</i> .
<i>State</i>	A <i>State</i> may either be a <i>ComplexState</i> or <i>AtomicState</i> . A <i>ComplexState</i> has at least one <i>hasSubState</i> relation with another <i>State</i> , and may be either a <i>ConjunctiveState</i> or a <i>DisjunctiveState</i> . An <i>AtomicState</i> has a <i>hasStateCharacteristic</i> , <i>hasOperator</i> , and <i>hasValue</i> relations with a <i>StateCharacteristic</i> , <i>Operator</i> , and <i>Value</i> , respectively.
<i>BasisOfDependence</i>	Has a <i>hasDependum</i> relation with either a <i>Task</i> , <i>Activity</i> , <i>ActivityCharacteristic</i> , <i>Resource</i> , or <i>StateCharacteristic</i> . Can have <i>constrains</i> and <i>alteredBy</i> relations to a <i>Task</i> , <i>Activity</i> , or <i>ActivityCharacteristic</i> . <i>Availability</i> , <i>Complementarity</i> , <i>Functionality</i> , <i>Compatibility</i> , <i>Uncertainty</i> , and <i>Complexity</i> are types of <i>BasisOfDependence</i> .

Figure 2) and demonstrate the type of insights it can generate. Prover9 and OWL implementations of this section are also provided in the Github repository, to allow for independent verification that our framework supports generating the inferences described in this section. Given space constraints, we focus on the elements that illustrate dependencies, rather than a comprehensive task structure. We will explore three instances of dependencies, each at different abstraction levels, to demonstrate the model’s ability to identify dependencies at both well- and ill-defined task structure levels.

The agents are the Riverine Flood Risk Mitigation Team (RM), and the City’s Water Infrastructure Management (WIM) and Transportation Services (TS) departments. The neighbourhood of interest is grappling with substantial flood risks due to both urban (sewer overflows) and riverine (watercourse overflows) sources. RM is designing a bridge as part of its Riverine Flooding Project (RFP). WIM must redesign a sanitary trunk sewer (STS) for capacity expansion and execute the Urban Flooding Program (UFP) to reduce urban flooding. For the STS, WIM must design a new storage pipe and require road access for UFP sewer improvements. TS must repave a road to maintain good road conditions. Three dependencies inferred here are:

Dependency 1. Flooding is a complex issue due to riverine and urban flooding being interconnected sources. If sewer systems are improved to reduce urban flooding, but the region remains vulnerable to a nearby overflow-prone watercourse (riverine risk), flooding will still occur. The flood protection level in a given area is only as strong as the weakest link. Hence, there is a complementary basis of dependence among tasks to reduce various types of flood risk. Particularly, executing the UFP depends on the STS and riverine developments, since the specific sewer improvements are constrained by the existing flood risk altered by these two developments.

Dependency 2. Co-location demands compatibility between the design of the RFP bridge and the STS’s storage pipe. Although they don’t impact each other’s functionality, their coexistence establishes a compatibility-dependent relationship. Since the bridge design precedes the storage pipe design, it alters this dependency and constrains the pipe design, which must align with the bridge design.

Dependency 3. TS’s road paving and WIM’s sewer improvement activities require access to the same road, causing an availability-based dependency. The start times of each activity alter and are constrained by the availability, creating mutual dependence.

Based on the representation of these dependencies, we are afforded the ability to reason about how to address them. The UFP execution’s dependency on the RFP and STS developments informs the considerations WIM must make. For instance, post-development hydrological models could guide the UFP’s activities. The dependency between Activity 1 and Activity 2 necessitates information exchange on the bridge design to facilitate a compatible pipe design. With Activity 3 and Activity 4 mutually dependent due to timing, we see that coordination in terms of scheduling is necessary, positioning a schedule as a coordination mechanism. This awareness can also help WIM and TS to synergistically overlap their activities, reducing public disruptions and setup costs.

In summary, the model effectively encapsulates the task environment at varied specificity levels, identifies task and activity dependencies, uncovers dependency roots, and suggests coordination methods.

7 DISCUSSION

In our discussion, we draw attention to three key, mutually reinforcing benefits of our framework: multi-level abstraction, inferred dependencies, and en-

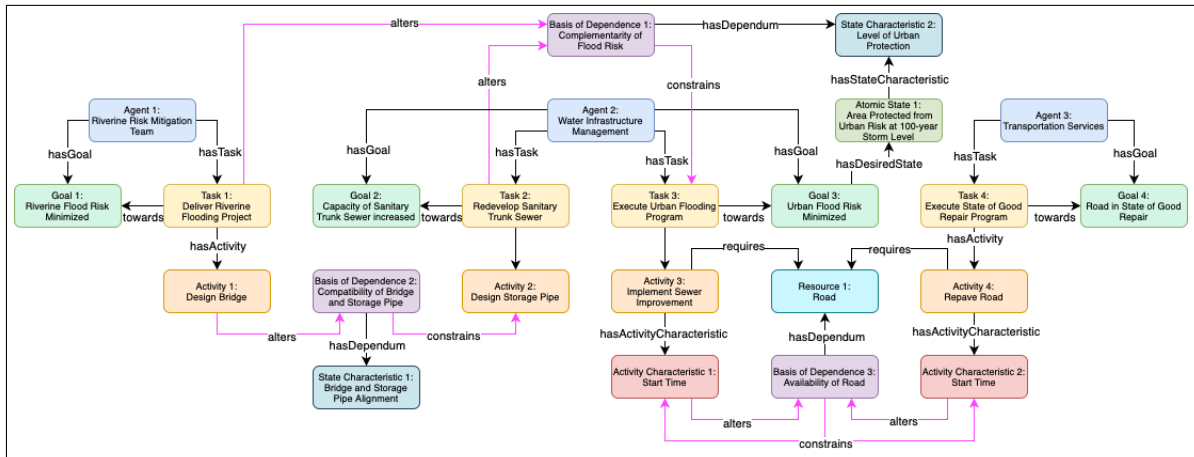


Figure 2: Partial representation of three case study scenarios. Pink relations indicate dependencies that can be inferred.

hanced reasoning about coordination. The precise semantics of our axiomatization, linking tasks and activities through bases of dependence, allows automated identification of critical and potentially latent dependencies. By facilitating the discovery of dependencies that might otherwise go unnoticed, coordination failures can be preempted and opportunities for improved integration can be revealed, as exemplified by the third dependency outlined in Section 6.

As observed in our case study, many project elements required rework due to a lack of dependency awareness. This could be avoided by taking advantage at project design time of decision-support tools that are able to alert decision-makers of potential influences on or from the various project tasks. Our formally axiomatized and implemented framework can serve as the backbone of such tools.

Secondly, by distinguishing the intent to achieve goals from the activities performed to reach them, the model captures dependencies at various task abstraction levels, making it applicable in both well- and ill-understood task environments. This makes our framework practically useful for managing dependencies across strategic, tactical, and operational tiers of policy, program, and project management when embedded in a decision-support tool.

Finally, by expressing dependence through its basis and effect, our model facilitates a deeper understanding of coordination failures’ root causes and reveals potential mitigation strategies. It offers a new perspective on coordination failures by treating them as unsatisfied (or unoptimized) constraints (Herbsleb and Roberts, 2006), with the basis of dependence representing the unsatisfied constraint and the effect revealing the “violating value”. Consequently, our model can be viewed as a higher-level abstraction of constraint networks, providing an explicit frame-

work for reasoning about strategies to “solve” constraint networks based on various activity characteristics. For instance, while the set of *dependsOn* relations between temporal activity characteristics can be seen as forming a temporal constraint network, for which several algorithms exist (Dechter et al., 1991), we can now model constraint networks based on other activity attributes, such as cost, design, and quality networks. With this perspective, we can work towards both human- and computer-based coordination strategies for “solving” the various kinds of networks.

However, our framework, in its current form, also has limitations. Primarily, the bases of dependence are asserted, not inferred. Ideally, the existence of all bases of dependence would be automatically inferred based on a robust representation of task structures. For instance, while *availability* can be inferred from the current axiomatization, *complementarity* cannot be, due to the present inability to represent “non-additivity” in state characteristics. Therefore, a more detailed representation of activities and states is needed to facilitate the automated detection of bases of dependence. Moreover, the framework does not support accounting for the relative importance of the various dependencies impacting a task or activity, thus failing to explicitly capture the significance of a constraint amidst large networks of dependencies with inherent trade-offs. These aspects call for refinement of our model in future work.

8 CONCLUSION

In this paper, we present a novel approach to representing tasks, task structures, and task dependencies to help enhance coordination in organizations. Faced with the coordination issues identified in real-life case

studies shared by domain experts, we examined the organizational and enterprise modelling literature for applicable modelling frameworks. We concluded they did not meet the representational requirements we inferred from our use cases. As such, we developed a new model of task dependence, grounded in the notion of constraint, and underpinned by two new constructs that define the basis and effects of a dependency. We discussed a practical application of our framework to a real-world scenario and showed how it enables the inference of (latent) dependencies and the reasoning about coordination strategies to address them. In future work, we plan to extend the framework and its ability to support organization design decisions and facilitate the integration of work by tackling the modelling of inter-agent dependencies.

REFERENCES

- Alam, K. A., Ahmad, R., Akhuzada, A., Nasir, M. H. N. M., and Khan, S. U. (2015). Impact analysis and change propagation in service-oriented enterprises: A systematic review. *Information Systems*, 54:43–73.
- Annett, J. and Duncan, K. D. (1967). Task analysis and training design. *Occupational Psychology*, 42:211–221.
- Baldwin, C. Y. and Clark, K. B. (2000). *Design Rules: The Power of Modularity*. The MIT Press, Cambridge, Mass.
- Bermejo-Alonso, J. (2018). Reviewing task and planning ontologies: An ontology engineering process. In Aveiro, D., Dietz, J. L. G., and Filipe, J., editors, *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2018, Volume 2: KEOD, Seville, Spain, September 18-20, 2018*, pages 181–188.
- Borgo, S., Ferrario, R., Gangemi, A., Guarino, N., Masolo, C., Porello, D., Sanfilippo, E. M., and Vieu, L. (2022). DOLCE: A descriptive ontology for linguistic and cognitive engineering. *Applied Ontology*, 17(1):45–69.
- Chandrasekaran, B. and Josephson, J. R. (1997). The ontology of tasks and methods.
- Crowston, K. (1994). A taxonomy of organizational dependencies and coordination mechanisms. Technical Report 174, Massachusetts Institute of Technology.
- Dechter, R., Meiri, I., and Pearl, J. (1991). Temporal constraint networks. *Artificial intelligence*, 49(1-3):61–95.
- Dietz, J. L. (1999). Understanding and modelling business processes with DEMO. In *Conceptual Modeling—ER'99: 18th International Conference on Conceptual Modeling Paris, France, November 15–18, 1999 Proceedings 18*, pages 188–202.
- Fox, M., Barbuceanu, M., and Grüninger, M. (1995). An organisation ontology for enterprise modelling: preliminary concepts for linking structure and behaviour. In *Proceedings 4th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '95)*, pages 71–81.
- Fox, M. S., Chionglo, J. F., and Fadel, F. G. (1993). A common-sense model of the enterprise. In *Proceedings of Industrial Engineering Research Conference*, pages 178–194.
- Fox, M. S. and Grüninger, M. (1998). Enterprise modeling. *AI Mag.*, 19:109–121.
- Greenspan, S., Mylopoulos, J., and Borgida, A. (1994). On formal requirements modeling languages: RML revisited. In *Proceedings of the 16th International Conference on Software Engineering, ICSE '94*, page 135–147, Washington, DC, USA. IEEE Computer Society Press.
- Guizzardi, R. S. S., Franch, X., Guizzardi, G., and Wieringa, R. J. (2013). Ontological distinctions between means-end and contribution links in the i* framework. In Ng, W., Storey, V. C., and Trujillo, J., editors, *Conceptual Modeling - 32th International Conference, ER 2013, Hong-Kong, China, November 11-13, 2013. Proceedings*, volume 8217 of *Lecture Notes in Computer Science*, pages 463–470.
- Herbsleb, J. D. and Roberts, J. A. (2006). Collaboration in software engineering projects: A theory of coordination. In *Proceedings of the International Conference on Information Systems*, pages 553–568.
- Malone, T. W. and Crowston, K. (1994). The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119.
- Phipps, D. L., Meakin, G. H., and Beatty, P. C. (2011). Extending hierarchical task analysis to identify cognitive demands and information design requirements. *Applied Ergonomics*, 42(5):741–748.
- Puranam, P. (2018). *The Microstructure of Organizations*. Oxford University Press, Oxford, United Kingdom.
- Puranam, P., Raveendran, M., and Knudsen, T. (2012). Organization Design: The Epistemic Interdependence Perspective. *Academy of Management Review*, 37(3):419–440.
- Raveendran, M., Silvestri, L., and Gulati, R. (2020). The Role of Interdependence in the Micro-Foundations of Organization Design: Task, Goal, and Knowledge Interdependence. *Academy of Management Annals*, 14(2):828–868.
- Simon, H. A. (1947). *Administrative Behavior: A Study of Decision-Making Processes in Administrative Organizations*. Macmillan, New York, NY.
- Stanton, N. A. (2006). Hierarchical task analysis: Developments, applications, and extensions. *Applied Ergonomics*, 37(1):55–79. Special Issue: Fundamental Reviews.
- Thompson, J. D. (1967). *Organizations in action: Social science bases of administrative theory*. McGraw-Hill, New York, NY.
- Vernadat, F. (2020). Enterprise modelling: Research review and outlook. *Computers in Industry*, 122:103265.
- Yu, E. S. and Mylopoulos, J. (1995). From E-R to “A-R”—modelling strategic actor relationships for business process reengineering. *International Journal of Cooperative Information Systems*, 4(2):125–144.