2. Conceptual Models

The conceptual models that we introduce in this section capture the concepts we need to explain how actions can be performed by, or on behalf of, specific parties, what things are prerequisite for that, and how this can be governed, managed and controlled. Specifically, these concepts can also help provide the means and method by which parties can tell whether or not, or to what extent, work is being done by, or on behalf of, other parties, and determine who these parties are. The ability to answer such questions in a precise way is required for those that want to design and/or implement systems that interact with each other and humans, on behalf of themselves, or others.

This chapter describes a few conceptual models that help to do this. When we define a specific concept, we do so by stating the term we will use to refer to that concept (printed in **bold**), as well as the criterion that can be used to distinguish between what are instances (examples) of that concept, and what are not. The visual representations of the models use commonly used notations and conventions.

2.1. The Party-Driven Actor model

It is common practice to think actions being taken by individuals (people, natural persons) or organizations (enterprises, governments or governmental bodies, enterprises). This practice, however, poses problems when one needs to answer questions as presented in the introductory paragraph of this chapter. One such problem arises from observing that organizations actually cannot do anything. An enterprise cannot hire employees, or sign contracts. They need people that do so on its behalf. Another problem arises when you see someone doing something, e.g., sign a contract. How can one tell whether this is done on behalf of the person itself, or on behalf of some organizations (and if so, what the organization is).

To come to grips with this, we propose a new way of looking at such situations, that is based on somewhat different concepts than that are commonly used, and collectively is called the "Party-Driven Actor model".

Here are some basic definitions (i.e., terms associated with criteria that enable us to distinguish between what is, and what is not, an instance (example) of a particular concept).

An **Entity** is something that is known to exist, e.g., a person, an organization, a computer, an extinct animal, a thought, an idea, a JSON-object – *anything* that anyone can think of as existing. Everything, tangible or not, still existing, extinct or as a future possibility, qualifies as an entity.

An **Actor** is an entity that can act (i.e., actually do things), such as people, or machines. Tables and stones are examples of entities that cannot act. And as we have seen, organizations, too, do not qualify as an actor. In order to continue using the common practice of saying that some organization has done something, we will take that to mean "there is an actor that does this something on behalf of that organization".

An **Action** is something that is actually done: a 'unit of work' that is executed by a single actor, as a single operation, in a specific context. Examples include "drafting a document", "signing a contract", "accepting an order". Business or other processes typically consist of a sequence of actions, where different actions therein may be executed by a single, or multiple actors (people, or machines).

It is typical for actions that they need to be done in particular ways, depending on whom they are done for. If a person is drafting and sending a letter, it matters whether (s)he does so for him/herself,

or for the company (s)he works for. In the latter case, the letter may need to be printed on company paper, and the company may have defined styles (typography, layout, etc.), which the person would do otherwise when drafting and sending a letter on its own behalf.

A **Policy** is a (set of) rules, working-instructions, preferences and other guidance for the execution of one or more kinds of actions, and that (the execution) of such actions can be said to comply with.

Policies are to be created and maintained by those that need such actions to be executed on their behalf. You can say that policies are part of their **knowledge**, i.e., the (intangible) sum of what they know[5].

It is easy to observe that individual people each have and maintain their own, subjective knowledge, which predominantly resides in their minds.[6] However, one can also make the case for organizations to have a 'mind' in which they store 'their knowledge'. This mind would be cabinets with papers, databases, and the like, and the knowledge therein is what is represented by the figures, texts, bits, etc. contained therein.

A **Party** is an entity that sets its objectives, maintains one particular knowledge, and uses that knowledge to pursue its objectives in an autonomous (sovereign) manner, which includes onboarding actors to do (parts of) the actual work. One might say that they have a mind of their own. Typical examples are individuals and organizations. Their minds (subjective knowledge) are what distinguishes one party from another, so every party is 1-1 related to its knowledge (mind).

The relevance of having this concept is that we can now clearly express how the execution of actions work: an action is executed by a single actor (as a single operation, in a single context), on behalf of a single party, which means that it will execute that action in compliance with the policy that this party maintains (within its knowledge). This is illustrated in Figure 1.

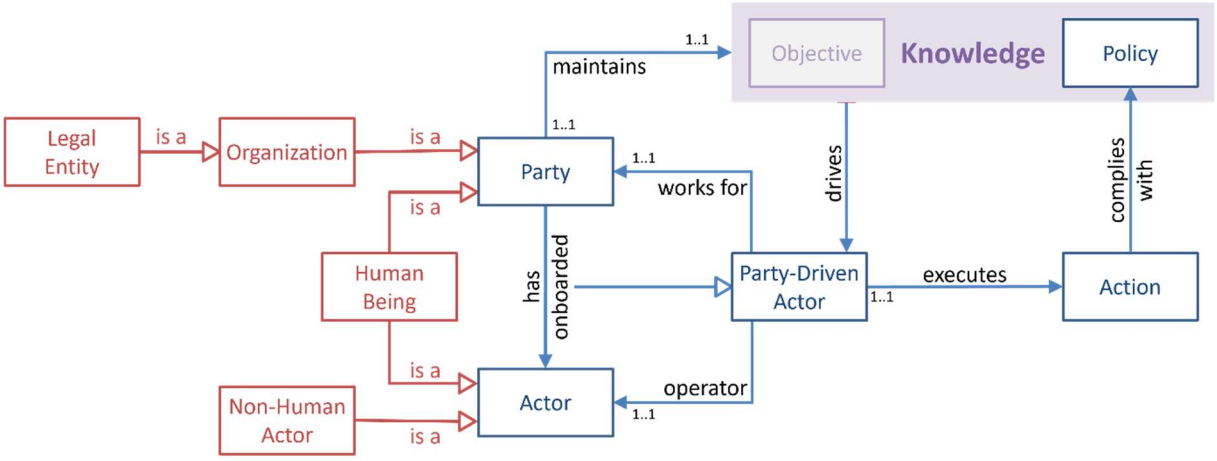

Figure 1: Party-Driven Actor concept diagram

In this figure, the red boxes and arrows represent concepts and relations that are commonly used (including by ourselves) but are not to be considered as part of the Party-Driven Actor (PDA) model.

[5] This includes the familiarity, awareness or understanding of someone or something (WikiPedia). Knowledge includes facts (propositional knowledge), skills (procedural knowledge), or objects (acquaintance knowledge). Knowledge can be acquired in many different ways and from many different sources, including but not limited to experience, reason, memory, testimony, scientific inquiry, education, and practice.

[6] Knowledge also exists in one's body, as shown by the existence of reflexes.

Rather, they explain the relationships between these commonly used concepts and the concepts of the PDA model.

Note that human beings (people) not only qualify as a party, as they have a mind of their own, but they also qualify as an actor, as they can do things. This 'unity', that organizations lack, is often a cause for confusion in the design and of systems. In our way of thinking (i.e., the perspective of this model), it is easy to circumvent such confusions; we will come to this further on.

Party-Driven Actors (PDAs)

Figure 1 also shows the crucial concept of this model, which is the Party-Driven Actor.

Formally, a **Party-Driven Actor (PDA)** is the (conceptualized) relationship between a single party and a single actor that is the result of a party having onboarded the actor (which we will explain a bit later). The purpose of having a PDA is so to say that it executes actions, which must be taken to mean that the actor that is the operator of the PDA executes that action, and the party for which the PDA works maintains the knowledge that **drives** the PDA (hence the term 'party-driven actor'), and that contains the policies that the actor must follow it/the PDA executes actions.

Note that since a PDA is the (conceptualized) relationship between a single party and a single actor, it follows that any pair of PDAs must differ in either the party for whom the PDA works, or the PDA's operator, or both.

Informally, a PDA can be seen as an actor that operates (executes actions) on behalf of a single party, e.g., in their role as employee, or hired workforce. We considered to use the term 'persona', but it didn't quite fit, as a 'persona' is more a particular way of viewing an actor (specifically, a person) than that it establishes an entity of its own, of which the actor is one of the (two) parts. It might work because it reflects that an actor, as it works for different parties, has different 'faces' to show, as if it were different persons.

Informally, a PDA can also be seen as a party that does things (executes actions) as it pursues objectives that it has set (and that exist within its knowledge). As we have seen, parties that themselves are incapable of acting need actors to do these things on their behalf, which is precisely what a PDA adds. So, when an organization is said to hire an employee, this means that there is a PDA that works for the organization (which is a party), which implies there is an actor that does the associated work, where the actor is onboarded (we'll get to that) by that party.

According to the PDA model, any text that mentions that either some party or some actor does something, MUST be interpreted to mean that there is a PDA that does this something, where the PDA works for that party, or the actor is the operator of that PDA respectively.

Onboarding

Every PDA represents a relationship between the party that it works for, and the actor that serves as the operator that does the actual, operational work. This relationship, which we call '**onboarding**' exists between a party and an actor during the timeframe in which:

1. The party (continually) ensures[7] that the actor has the capabilities that are required (or handy) for doing the particular kinds of work that the party is tasking it with (which it does to realize its objectives). For example, when an organization wants to hire someone for a specific position, it

[7] That is to say: a particular PDA that 'works for' that party, and hence has an actor that does the associated operational work.

first ensures that this person is suitable for that position. Similarly, when an organization wants to lease or buy software, it will first ensure that this software is fit for the purposes of the organization. Organizations will typically regularly evaluate the fitness of an actor for its tasks, and take actions either to get its capabilities enhanced, or terminate the relationship.

2. The party (continually) ensures[7] that the rights and duties of the actor (towards the party itself, and/or others, as appropriate) are properly specified, maintained, and enforced if necessary. This may take the form of a contract, as is common between organizations and their employees, or organizations and parties from which they hire/lease machines or software.[8] All this is to ensure that the actor, once it is onboarded, can be relied upon to work as the party expects it to (which includes adhering to the policies that the party specifies and maintains for the execution of actions on its behalf).
3. The party (continually) ensures[7] that the circumstances and conditions exist that are needed for the actor to execute the actions that it tasked with. This includes e.g., providing the actor with appropriate roles/permissions, access to the policies that guide the execution of such actions, resources to work with, etc.

The PDA-model's Perspective on Human Beings

Human beings are special in the sense that they qualify both as a party and an actor, which is easy to verify by checking their definitions.

We postulate that every human being (as a party) has onboarded itself (as an actor) for the entire time that (s)he is alive, for any kind of action that it wants to execute itself.[9] This results in the existence of every person having one (and only one) PDA whose party and actor are both this person.

Of course, a person can also onboard other actors, or be onboarded by other parties, which creates new PDAs in which the person is the party or the actor respectively.

It is easy to observe that individuals differ greatly, and not all of them have all capabilities that they might need or would like to have to onboard themselves, or other actors, in such a way that the resulting PDAs would operate to what is beneficial to the individual. The model allows for 'lousy' onboarding practices to exist, thereby acknowledging the autonomy that individuals (and other parties) have to make their own decisions as good, bad, or ugly as they like.[10]

The PDA-model's Perspective on Organizations and Legal Entities

PDAs where the operator and party are the same person are needed to create parties that themselves cannot act, such as enterprises, governments, communities, etc. What it takes is that one or more of these PDAs set objectives, and devise associated policies and other knowledge, where none of these PDAs individually controls them. Rather, they work together to govern and manage such objectives, and to ensure everything gets done to realize them.

Such collaborations can be very informal, e.g., a group of friends that decide to promote social bonding and emotional support, e.g., by sharing hobbies or interests, serving as a support network for its members, celebrating achievements, by group volunteering or community service.

---

[8] Formally, this would be a contract (about the actor) between the party and another party, which is the party that owns or otherwise controls the actor, and hence can 'make' the actor comply with the terms of that contract. The elaboration of this is future work and falls outside the scope of the PDA model.

[9] This isn't shown in Figure 1. However, it is an integral part of the PDA model.

[10] This is also in line with what conceptual models are for, i.e., come to grips with the world around us. Our conceptual models are not intended to model what good or bad judgments are.

However, collaborations can also arrange for complying with rules/laws in some jurisdiction for the purpose of being recognized as a legal entity (i.e.: an entity that is known within some jurisdiction), which typically means that the collaboration gets rights and duties (such as the right to sue and be sued) that are similar to human beings.

Envisaged Practical Benefits of Adopting the PDA Model

1. **Clarity and Precision**: The PDA model offers a clear and precise way of understanding and describing complex relationships between parties, actors, actions, and policies. By providing well-defined definitions and concepts, it eliminates ambiguity and enhances communication and understanding.
2. **Accurate Representation of Reality**: The model addresses the inherent limitations in traditional ways of attributing actions to individuals or organizations. It recognizes that organizations, as entities, cannot directly perform actions but rely on actors to act on their behalf. By acknowledging this distinction, the PDA model provides a more accurate representation of how actions are executed in reality.
3. **Improved Design and System Development**: Adopting the PDA model can lead to more effective design and development of systems, processes, and organizations. By clearly identifying the relationships between parties and actors and their respective roles, it becomes easier to define responsibilities, assign tasks, and align policies and objectives. This clarity can help avoid confusion and miscommunication during system design and implementation.
4. **Enhanced Accountability and Compliance**: The PDA model enables better tracking of accountability and compliance. By explicitly linking actions to parties and their policies, it becomes easier to trace the origin and responsibility for specific actions. This can be valuable in auditing, regulatory compliance, legal proceedings, and ensuring that actions are aligned with organizational objectives and standards.
5. **Efficient Collaboration and Resource Allocation**: Understanding the PDA model can facilitate more efficient collaboration and resource allocation within organizations and across different parties. By clearly defining the relationships between parties and actors, it becomes easier to assign tasks, delegate authority, and optimize resource allocation based on individual capabilities and organizational needs.
6. **Flexibility and Adaptability**: The PDA model accommodates different scenarios and levels of autonomy. It recognizes that individuals can be both parties and actors, allowing for flexibility in self-onboarding and onboarding others. This flexibility supports various organizational structures, such as partnerships, collaborations, and hierarchical relationships, while maintaining the clarity of roles and responsibilities.
7. **Alignment with Legal and Regulatory Frameworks**: The PDA model aligns well with legal and regulatory frameworks, especially when dealing with liability, contractual obligations, and compliance. By explicitly defining the relationships between parties and actors and their associated rights and duties, it provides a solid foundation for legal interpretations and contractual agreements.
8. **Improved Decision-Making and Governance**: By understanding the PDA model, decision-makers can have a more comprehensive view of the parties involved and their knowledge (including policies) when making strategic decisions. This can lead to better-informed governance, improved risk management, and more effective resource allocation.

2.2. The PDA Functions, Roles and Mandates (PDA-FRM) Model

In this chapter, we use (and extend) the PDA model by looking a bit more closely to what a party would need such that PDAs can organize the work that is associated with the realization of their

objectives. This leads to the conceptual model for PDA Functions, Roles and Mandates (PDA FRM), a representation of which is given in Figure 2.

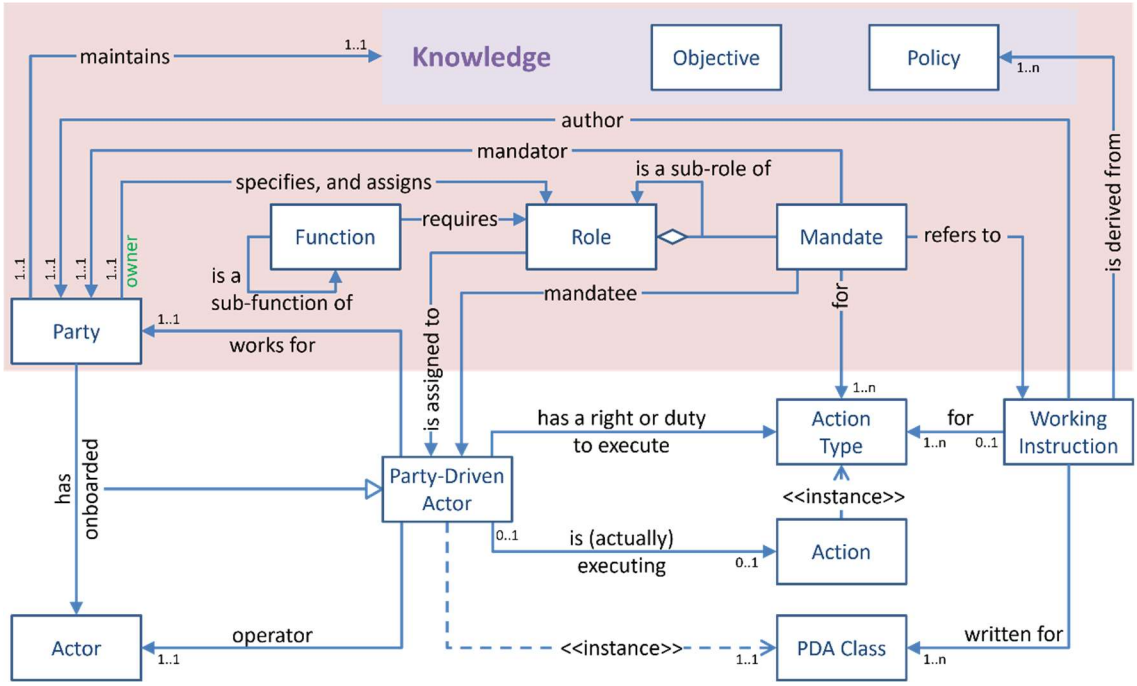

Figure 2: PDA Roles and Mandates concept diagram

The figure shows many concepts from the PDA model, which are there to show how the concepts that are new relate thereto.

Working Instructions

A **working instruction** (for a specific Action Type) provides a set of operational rules and other guidance that (the respective operators of) PDAs must use when they execute actions of such types.

Working instructions are authored by (or on behalf of) the (single) party that has specified the policies according to which actions of such types are to be executed. It almost comes without saying that the working instructions are derived from these policies, so that PDAs that use them as they execute actions of the type for which the working instruction was created, will execute such actions in accordance with how the party wants them to be executed.

For PDAs with human operators, working instructions can come in the form of handbooks, workflows, phone-scripts, etc. Also, such working instructions may be stated in different languages, and a varying level of detail. For PDAs with non-human operators, such as computer programs, working instructions may come in the form of configuration files, scripts, macro's, etc.

It is important that every PDA that has a right or duty to execute specific kinds of actions, can have a working instruction at its disposal that it can actually read and understand, and is written for the class of PDAs of which it is an instance (member)the working instructions for these action types.

Working instructions are typically as short and explicit as possible, so as to be a minimal burden and maximum help for the PDAs that use them. To achieve this, a well-designed classification of PDAs is useful, as authoring a working instruction for a particular class of PDAs may take the specific characteristics, e.g., capabilities that they have, into account. For similar reasons, such a classification of PDAs may facilitate the onboarding of actors.

Mandates

A **mandate**, also as we know this from common practices, is something that

1. is created and maintained by a (single) party, which we refer to as the **mandator** of mandate.
2. specifies the permissions, rights, and duties that are required for (the operator of) a PDA that is allowed to execute a particular kind of action (**action type**) on behalf of the mandator. We use the term **mandatee** (of a mandate) to refer to such PDAs.
3. refer to (any number of[11]) working instructions for these action types.

Every mandatee (of a particular mandate) must be a PDA that works for the mandator (of that mandate). That is because this party remains responsible/accountable for the actions performed under the mandate.[12]

Mandates thus define the boundaries and scope within which a PDA can operate and help to ensure that actions are executed in compliance with the party's policies, as the working instructions are derived therefrom. In practice, mandates can come in various forms, e.g., as part of an employment contract, a (publicly available) list of explicit mandates, etc.

Mandates can be seen as an extension of what traditionally is called a permission. The extension comprises (a) the fact that it may not only be a right, but also a duty to do something, (b) the fact that it may refer to working instructions, and (c) that the party that is responsible for maintaining it is made explicit.

By defining mandates for each role and working instructions that guide the performing of tasks, parties establish a clear framework for authorized actions and behaviours, aligning them with their objectives and obligations. Mandates provide a structured approach to task allocation, facilitating effective collaboration and coordination among multiple PDAs involved in complex interactions.

Additionally, mandates play a crucial role in mitigating risks and ensuring compliance. They serve as a mechanism to regulate access to sensitive information, control the use of resources, and enforce ethical and legal standards. Mandates define the limits of authority and responsibility for each PDA, preventing unauthorized actions and minimizing the potential for misconduct or misuse of power.

Roles and Functions

It is beneficial to create groups of mandates that have the property that they together enable a PDA to perform a complex (set of) task(s). For example, an organization may want all employees to be able to access the organization's premises, log into the organizations network, etc. This would require various mandates (permissions and associated working instructions), which can be grouped (aggregated) into a role called 'employee'. Any PDA that will be assigned this role will then be able to access the organization's premises, log into the organizations network, etc.

[11] A mandate that doesn't refer to a working instruction may also be perceived as a 'permission'. This makes the figure compatible with Role Based Access Control (RBAC), where 'access' is to be taken as a synonym for executing actions of a particular kind.

[12] This is what distinguishes a mandate from delegation. With delegation, the responsibility/accountability is also transferred. Note that with delegation, the delegate (i.e., the analogue of the mandatee) would typically be a party, and not an actor, as entities can only take responsibility or be accountable if they can be sued in court, which requires them to have 'legal personality', which seems to be equivalent to what in our models we refer to as a party. A conceptual model that addresses delegation is future work.

It is also beneficial to aggregate existing roles with additional mandates. For example, the 'employee' role can be aggregated with mandates that are needed to execute actions related to hiring people, thereby creating a new role that might be called 'HRM officer'.

Thus, a **role** is an aggregation of other roles ('sub-roles') and mandates, that is specified by the (single) party (the 'owner' of the role), and that can be assigned to PDAs by (or on behalf of) that party. These role aggregations simplify the management, auditability, and compliance of (assigning and revoking) mandates, while also providing scalability and flexibility as the number of mandates grows. Roles, along with mandates, require periodic review and updates to accommodate the evolving needs of their owners, changing legal regulations, and emerging responsibilities.

Every role is specified, and assigned, by one specific party, which we know must be interpreted as that there is a PDA that works for that party and that has a right or duty to execute the associated kinds of actions, because it is a mandatee of the mandate that provides that right or duty.

A **function** represents a (named) coherent set of activities or purposes of a party, such as 'administration', 'purchasing' or 'relation management'. Performing such actions requires (coherent sets of) mandates and/or roles to be assigned to the PDAs, which provides them with the rights (permissions) and/or duties (responsibilities) to perform these actions on behalf of the mandator.

The ability to aggregate higher-level functions from lower-level functions (sub-functions) benefits parties in several ways. It assists in organizing and categorizing related activities, establishing a structured framework for efficient management. This promotes a clearer understanding of the overall scope and purpose of different areas of work within the party's structure. By facilitating the specification of mandates and roles, it enhances the party's ability to assign responsibilities and authority effectively. Additionally, this aggregation supports strategic planning and resource management, providing a holistic view of the party's operations and enabling informed decision-making.

So, functions help to split up actual work into coherent activities or purposes, where roles (and mandates) provide the rights and duties (authorizations and responsibilities) that PDAs need to perform that work. Together, functions, mandates, and roles form the organizational structure that guides the allocation of tasks and ensures efficient collaboration towards organizational objectives.

A common mistake is to confuse roles or functions with the names they are called by. If two parties own a role that they call 'employee', it is obvious that they would differ because they are aggregations of different roles and mandates. Similarly, it they both have a function called 'administration', the function will be a collection of activities that are not the same for both parties. So, it is important to realize that each role, and each function has a single party that owns it, i.e., that specifies what it is made up of.