A New Taxonomy of Coordination Models Based on Interaction Types

Li Jiang (College of Computer Science and Technology, Jilin University, Changchun 130012, China)
Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China

Da-you Liu (College of Computer Science and Technology, Jilin University, Changchun 130012, China)
Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China

Abstract - Coordination models can be classified into several classes, such as data-driven vs. control-driven coordination models, dependent vs. emergent coordination models, and hybrid coordination models. All these classifications are based upon implementation details. However, interaction is the kernel aspect of coordination, and coordination is the outcome of interactions between agents, so here, we give another kind of taxonomy based on interaction types, which are coordination models based on direct interactions and indirect interactions. Using this taxonomy, all coordination models are classified into two classes with no intersection, in addition, coordination models with indirect interactions and emergent behaviors are the direction of future work.

Keywords: Multi-agent System, Coordination Model, Data-driven, Control-driven, Emergent Coordination, Interaction

1 Introduction

Massively parallel and distributed systems have opened new horizons for large applications and led to the naissance of coordination models and languages. Almost all of these models share the same intent, namely to provide a framework which enhances modularity, reuse of existing (sequential or even parallel) components, portability and language interoperability and have been classified into data-driven and control-driven models. This is the first stage of coordination models development. Examples are Linda, IWIM, etc.

In the past 20 years, coordination models, and in particular Linda, have proven to be quite successful in tackling the intricacies of medium-to-large-scale open systems. Yet, Linda systems may not scale well with the number of tuple spaces and processes. In the quest for identifying systems that are scalable, researchers of TOTA (Tuples on the Air) [6] and SwarmLinda [7] turned away from standard techniques and looked in the field of Biology, more specifically Zoology where one can find several examples of scalable natural forming multi-agents systems (Swarms). In these systems, interactions involve only indirect but local communications. This is the second stage of coordination models development.

In [9], TOTA and SwarmLinda were classified as emergent coordination models, while Linda and others as dependent models. This is the recent harvest in this coordination model field.

In [2], David Keil and Dina Goldin gave the definitions of direct and indirect interaction. Due to the different interaction mode in these coordination models, we will give another taxonomy based on direct interactions and indirect interactions. This is our work.

The full paper is organized as follows: in the second section, we introduce the data-driven and control-driven coordination models, along with examples and comparison; in the third section, we focus on the dependent and emergent coordination models, along with examples and comparison; in the fourth section, several hybrid models combining data and control-driven methods are presented. Then, the fifth section describes the new division of coordination models based on direct interactions and indirect interactions, along with examples. Last, we draw a conclusion to tell readers what we should do next.

2 Data-driven and control-driven models

2.1 Data-driven coordination models

In data-driven coordination models, the state of a program is defined in terms of both the values of the data being received or sent and the actual configuration of the coordinated components. This means that processes are in charge of handling the data and coordinating themselves with the other processes[1].

Historically, Linda is the first genuine member of the family of coordination models. It provides a simple and elegant way of separating computation from communication concerns. Linda is based on the so-called
generative communication paradigm: if two processes wish to exchange some data, then the sender generates a new data object (referred to as a tuple) and places it in some shared dataspace (known as a tuple space) from which the receiver can retrieve it.

In fact, Linda is not a fully fledged coordination language but a set of some simple coordination primitives, that is, in(), out(), rd(), eval(), inp() and rdp() [1].

Linda has inspired the creation of many other similar languages - some are direct extensions to the basic Linda model but others differ significantly from it. These derivatives are called LINDA family which aim to improve and extend the basic model with multiple tuple spaces, enforcement of security and protection of the data posted to the tuple space, etc. Examples are Bauhaus Linda, Bonita, Law-Governed Linda, Objective Linda, LAURA, Ariadne/HOPLa, Sonia, Jada, SHADE, GAMMA, LO and COOLL, MESSENGERS, Synchronisers, Compositional Programming and Opus.

2.2 Control-driven coordination models

With control-driven coordination models, the state of the computation at any moment in time is defined in terms of only the coordinated patterns that the processes involved in some computation adhere to [1]. The components being coordinated are considered as black boxes that produce and consume data on well-defined interfaces to the external world, usually referred to as ports.

Idealized Worker Idealized Manager (IWIM for short) is a control-driven coordination model. There are two classes of processes: managers and workers. To communicate, a process writes and reads data on its ports. It is not aware to whom information is transmitted. Workers basically perform a computational task. Managers create new processes and regulate the connections between workers by dynamically binding ports of different workers into connections, or disconnecting existing connections.

Other control-driven coordination models are Contextual Coordination model, Toolbus and CoLa [1].

2.3 Summary

To complete this section, we draw the differences between data-driven coordination models and control-driven coordination models as follows:

1) There exists a mixture of coordination and computation code within a process definition in data-driven coordination models, and in control-driven coordination models, coordination and computation can be separated completely.

2) Data-driven coordination models are mostly concerned with data, but in control-driven coordination models, actual values of the data being manipulated by the processes are almost never involved.

3) In data-driven coordination models, the coordination component is usually a set of primitives with predefined functionality which is used in connection with some “host” computational language, while in the control-driven category the coordination component is usually a fully-fledged language.

4) The data-driven category tends to be used mostly for parallelising computational problems. The control-driven category tends to be used primarily for modelling systems.

5) A programmer has more control over the manipulated data in the case of data-driven coordination models, than control-driven ones.

6) The data-driven category tends to coordinate data whereas the latter tends to coordinate entities (which, in addition to ordinary computational processes, can also be devices, system components, etc.).

3 Dependent and emergent coordination models

3.1 Dependent coordination models

Both data-driven and control-driven models rely on coordination rules, schemas and middlewares to provide the same rules to all the processes involved in a distributed task, and where the coordination of all the activities is performed under the auspices of a single coordinator. We call these models dependent coordination models.

Consider the following scenario related to car maintenance [9]:

1. The car indicates to the driver that it needs maintenance due to a faulty part.

2. The car also informs an authorized dealership that a maintenance visit is needed.

3. The dealership checks the inventory to make sure a replacement part is in stock and contacts the supplier in case it is not.

4. The dealership contacts the user to determine a suitable time to perform the scheduled maintenance.

5. The user brings the car to do the scheduled service; the required part is guaranteed to be in stock.
To LINDA, the dealing of this car scenario is as follows:

Tuple spaces can be made available to processes to store tuples representing the message being exchanged. As the car discovers the faulty part it stores a tuple that is available to the user agent as well as the dealership agent. Both agents react to the placement of the tuple by indicating to the user that a problem has occurred and by checking the inventory. Checking of the inventory will involve other agents that can coordinate their tasks in a similar tuple-based fashion. The same applies to the coordination between the user agent and the dealership to set up an appropriate time for performing the maintenance job. Here, the solution to the problem above has to be programmed in advance; reaction rules have to be implemented in the tuple space and the processes have a clear idea of the overall goal [9].

3.2 Emergent coordination models

Different from data-driven and control-driven models, SwarmLinda [7] utilizes self-organization on tuple distribution and tuple search, inspired from ant-foraging techniques. In this model, the coordination between processes does not occur as the ‘central part’ of the interaction. There is no ‘direct correspondence’ between the purpose of local interaction and the global functionality of coordination within a system. Solutions to the general problem emerge from these interactions. Often the local interaction is driven by the self-interest (the individual goals) of the agents, which are not ‘aware’ (do not explicitly model) of the global outcome of the sum of their interactions[9]. This makes SwarmLinda much more suitable for use in open systems due to its scalability and adaptiveness. We call such models as emergent coordination models. A few principles for a more realistic SwarmLinda to consider are [7]:

Simplicity: Swarm individuals are simple creatures that perform simple tasks. They do no deep reasoning and implement a small set of simple rules. The execution of these rules leads to the emergence of complex behavior. Active entities in a SwarmLinda should also obey the principle of simplicity and be “small” in terms of resource usage.

Dynamism: Natural swarms adapt to dynamically changing environments. In open distributed systems, the configuration of running applications and services changes over time. If a tuple is found in a given location, it does not necessarily mean that other similar tuples will exist in the same location in the future.

Locality: Swarm individuals observe their direct neighborhood and take decisions based on their local view. As the key to scalability in SwarmLinda, active entities have to perform only local searches and inquire only to direct neighbors.

3.3 Summary

Differences between dependent and emergent coordination models are:

1) In dependent models, the solution to the coordination of processes is programmed explicitly, while in the latter, solution emerges and is not directly programmed into the processes/agents.

2) In dependent models, the success of the task being performed depends on the individual success of all processes in the system, but in the latter, the emergence of a solution is relatively tolerant to failures in individual processes/agents.

3) In dependent models, the processes depend on the coordination model as a mediator of interaction, but in the latter, agents do not depend directly on each other’s activities.

4) In dependent models, the designer has full control over agents, while in the latter, the design focus is usually on micro-level issues, i.e. referring to the properties of individual agents.

5) In dependent models, each process has a specific and different role in the solution to the coordination problem, but in the latter, agents are less specialized.

4 Hybrid Coordination Models

Both data-driven and control-driven models present several advantages and drawbacks depending on the application domains. Several hybrid models have already been designed by wanting to improve a model of one of the two families or by explicitly confronting and integrating elements from one family to the other.

Examples are Law-Governed Linda and IWIM-Linda that combine control-driven elements with a data-driven model and ECM that integrates shared data-space functionalities in a process-oriented view.

5 Coordination models based on direct interaction and indirect interaction

Definition 5.1[2] Interaction is the ongoing two-way or multi-way exchange of data among computational entities, such that the output of one entity may causally influence the later outputs of the other entities.
**Definition 5.2**[2] Direct interaction is interaction via messages; the identifiers of the recipient are specified in the message.

Direct interaction may be synchronous or asynchronous. In the case of asynchrony in direct interaction, such as email, a computing agent synchronizes with an intermediate shared resource to receive or send a message.

**Definition 5.3**[2] Indirect interaction is interaction via persistent, observable state changes; recipients are any computing entities that will observe these changes.

In direct interaction a message's destination is fixed by the identifiers of the recipients at the time of sending, while in indirect interaction the identity of the receivers depends on dynamically generated events such as the entry of agents into the vicinity of data emitted. Indirect interaction is therefore a natural model for systems of mobile agents whose ability to perceive and act upon their environment is localized, in contrast to systems with no spatial constraints on communication.

**Example:** Ant colonies solve the problem of efficiently foraging for food sources efficiently by a multi-agent interaction in which each ant deposits pheromones (evaporating scent chemicals) as it walks, and each ant follows pheromone trails. Heavily traveled (hence strong, hence attractive) pheromone trails correspond to short paths to food. As food is exhausted at a site, the trails to it evaporate. Without a plan, the ants find a set of paths to the food that tends toward optimality. The communication among the ants is indirect, via pheromone deposits that change the state of the environment, rather than via message passing.

From the above definitions, we can find that some of the above dependent (data-driven and control-driven) coordination models use direct interaction (Actors and CoLa), and others use indirect interaction (Linda family and IWIM, Toolbus). But to emergent coordination models, there are only indirect interactions (TOTA and SwarmLinda). Let’s see this fact in detail.

For most data-driven coordination models like Linda, the Shared Dataspace is a common, content-addressable data structure. All processes involved in some computation can communicate among themselves only indirectly via this medium. Particularly, they can post or broadcast information into the medium and also they can retrieve information from the medium either by actually removing this information out of the shared medium or merely taking a copy of it. Some processes can send their data into the medium and then carry on doing other things or even terminate execution while other processes asynchronously retrieve this data; a producer need not know the identity of a consumer (and vice versa) or, indeed, whether the data it has posted into the medium has been retrieved or read by anyone. Apparently, this belongs to indirect interaction.

With control-driven coordination models like CoLa[1], processes communicate directly via message passing. In CoLa, there is no global name space: each process has a locally temporary view of the system (locality). This is achieved using for each process two abstractions: i) a Range of Vision defines the set of processes a process can communicate with at a specific moment of its life; and ii) a Point of View defines a group of processes in a particular Range of Vision, allowing to abstract communication topologies such as rings or hypercubes. In order to communicate, CoLa extends the message passing paradigm by adding to a message: (i) a high-level descriptor indicating the sender, the set of receivers, and a set of methods used by a receiver to treat the body of the message; and (ii) a high-level protocol describing the communication protocol used to deliver the message. This is direct interaction.

To emergent coordination models like TOTA and SwarmLinda, interaction is based on swarm intelligence like ant foraging which belongs to indirect interaction. So, we can say, indirect interaction is essential for emergent coordination models. Here we give a new classification of coordination models in the following table.

**Table 1. The new classification of coordination models.**

<table>
<thead>
<tr>
<th>Dependent</th>
<th>Emergent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-driven</td>
<td>Control-driven</td>
</tr>
<tr>
<td>Direct interaction</td>
<td>CoLa, Actors</td>
</tr>
<tr>
<td>Indirect interaction</td>
<td>Linda family, IWIM, Toolbus</td>
</tr>
<tr>
<td>TOTA, SwarmLinda</td>
<td></td>
</tr>
</tbody>
</table>

**6 Conclusions**

In this paper, we have investigated the important research field of multi-agent systems-coordination models. Complementing to the existing divisions (data-driven vs. control-driven coordination models and dependent vs. emergent coordination models), we give another division based on the definitions of direct and indirect interaction, i.e. coordination models based on direct and indirect interaction, involving both dependent vs. emergent coordination models and data-driven vs. control-driven coordination models.

To multi-agent systems, coordination is an important research area. To coordination, models are of great
The current situation is, systems are more and more complex, dynamic and open, old methodologies and models no longer fit the current and future demand, but little attention has been paid to new methodologies and models.

As surveyed in this paper, emergent coordination models based on indirect interaction are much more suitable for use in open systems due to its scalability, adaptiveness and fault tolerance, so next, we should mine more and more emergent coordination strategies and models to satisfy the current need in MAS development.

Current emergent strategies through indirect interaction includes: swarm intelligence (e.g. ant foraging) [7,10], cooperative state-changing rules[12] and game theory[4]. Existing models are: Arthur’s El Farol Bar problem [13], Minority Games[4] (MG), TOTA and SwarmLinda. We hope to find more and more new emergent strategies and models in the future.

Additionally, combination of dependent and emergent coordination, direct and indirect interaction can also facilitate the arrival of new models and strategies.

7 Acknowledgements

This paper is supported by NSFC Major Research Program 60496321, Basic Theory and Core Techniques of Non Canonical Knowledge; National Natural Science Foundation of China under Grant Nos. 60373098, 60573073, the National High-Tech Research and Development Plan of China under Grant No.2003AA118020, the Major Program of Science and Technology Development Plan of Jilin Province under Grant No. 20020303, the Science and Technology Development Plan of Jilin Province under Grant No. 20030523.

References


