






A Graph Pattern Based Approach for Automatic Decomposition of IoT Aware Business Processes

Francisco Martins^{1,2} , Dulce Domingos¹  , and Daniel Vitoriano¹

¹ LASIGE, Faculdade de Ciências, Universidade de Lisboa, Lisbon, Portugal

² University of the Azores, Ponta Delgada, Portugal

`fmartins@acm.org`, `mddomingos@ciencias.ulisboa.pt`,

`daniel.vitoriano@reitoria.ulisboa.pt`

Abstract. The context information that business process can get from the Internet of Things (IoT) can be used as a competitive advantage in terms of optimisation and agility. However, the exchange of messages between central systems and IoT devices come with a price, battery consumption, a scarcely resource of such devices. Despite the literature offers many technical proposals to tackle this problem, we take an approach driven by the process definition perspective. We propose to reduce the number of exchanged messages by decentralising process execution, moving parts of the business processes to IoT devices, and taking advantage of their computational capabilities. The first step for decentralisation is decomposition, i.e., the division of processes into parts and identify those that IoT devices can execute.

In this paper, we present an automatic decomposition solution for IoT aware business processes, described using the Business Process Model and Notation (BPMN). We start from a BPMN definition that follows a centralised approach and apply our decomposition method to transfer to the IoT devices the operations that can be performed there. We use a graph based approach and transform a BPMN definition into a directed graph. Thereafter, we identify cuts that define the parts to be transferred to the IoT devices. This decomposition preserves the control and the data dependencies of the original process, reduces the number of exchanged messages as well as the central processing. The code that IoT devices execute is automatically generated from the BPMN process being decentralised.

Keywords: Internet of things · BPMN · Process decomposition

1 Introduction

Internet of things (IoT) aware business processes put together two technologies, and has gained increased attention in the recent years.

Organisations use business processes to automate and optimise their operations by modelling them as flows of activities enriched with events, data flows, and information about resources and participantes, among other perspectives. Business processes are often defined with graphical notations, such as the Business Process Model and Notation (OMG 2011).

In addition, the IoT makes possible to interconnect everyday physical devices, which can interact with each other and cooperate with their neighbours. These devices are also accessible through the Internet, providing information and functionalities on behalf of physical objects or things. Interconnected small devices pose several challenges, such as energy consumption, the scarcest resource of the IoT; scalability, due to the number of devices that can be quite high; reliability, as network nodes are susceptible to a wide variety of failures; and security, since devices have computational limitations, and, typically, their physical integrity is difficult to assure (Atzori et al. 2010; Lee and Kim 2010; Moreno et al. 2014; Rault et al. 2014; Zorzi et al. 2010).

Business processes can use the IoT to gain competitive advantage in terms of optimisation and agility. Business processes may use the information the IoT provides about what is actually happening in the real world to optimise their execution, and react to new situations in real time (Yousfi et al. 2016). However, in most cases, business processes interact with IoT devices following a request-response or a publish-subscribe scheme to gather information and to trigger actuators. These interaction schemes promote the exchange of messages between IoT devices and the execution engine, which results in a high power consumption profile from the IoT device.

An alternative approach is for business processes to use IoT devices as active participants for executing parts of the business process logic (Haller et al. 2009), taking advantage of their computational and communication capabilities. IoT devices can be used to aggregate and filter data, as well as to make decisions locally, executing local flows, without needing a centralised coordination. Following this approach, it is possible to reduce the number of exchanged messages, increasing battery lifespan, and to promote scalability, by moving parts of the execution of the process from the central engine to IoT devices. We point out that this extra execution consumes power of the IoT device, but the energy consumption of the migrated tasks is much smaller than that used on communication.

Despite the benefits of decentralisation, business processes are still defined following a centralised approach. Our proposal automatically decomposes BPMN business processes, identifying and moving to the IoT parts of the processes that IoT devices can execute, assuring the reduction of the number of exchanged messages, while maintaining data and control dependencies.

Our work is distinct from related work in four main aspects: business process representation; decomposition technique; considered dependencies; and meeting criteria. Some authors represent business processes as generic graphs (Xue et al. 2018), while Nanda et al. (2004) use Web Services Business Process Execution Language (WS-BPEL) (OASIS 2007). Still considering the way these proposals represent business processes, almost all of them only support block structured processes. Xue et al. (2018) supports non-block structured processes, but only deal with processes having regions with one entry point and one exit point (SESE). Decomposition techniques use approaches mainly based on dependency tables (Fdhila et al. 2009) or on generic graphs (Nanda et al. 2004; Xue et al. 2018). While all proposals consider control dependencies, some of them

disregard data dependencies (Nanda et al. 2004). Finally, we can find different criteria for decomposition in the literature such as communication cost, delay cost, or confidentiality (Fdhila et al. 2014; Goettelmann et al. 2013; Hoenisch et al. 2016; Pova et al. 2014).

These proposals cannot be straightforward applied to BPMN business processes. The transformation of a BPMN business process definition to a generic graph poses many challenges, such as the implicit concurrent behaviour introduced by send tasks, and the identification of data dependencies. Furthermore, BPMN business processes can be non-block structured with many entry and exit points.

Our previous work proposes a pattern based approach to decompose IoT aware BPMN business processes (Martins et al. 2019; Domingos et al. 2019). We identify the common scenarios in this kind of processes, where is it possible to reduce the number of exchanged messages between the central engine and IoT devices. In addition, we define the transformation rules to move to the IoT parts of the processes that IoT devices can execute, while maintaining control and data dependencies.

The work we present in this paper goes a step further by adopting a more generic approach based on graph concepts. We transform the BPMN definition into a typed directed graph, where we identify cuts that are used to define the parts that can be transferred to the IoT devices. This decomposition preserves the control and the data dependencies of the original process, reduces the number of exchanged messages as well as the central processing. The code that IoT devices execute is automatically generated from the BPMN process being decentralised.

This paper is organised as follows: the next section discusses related work; Sect. 3 presents the use case we resort to illustrate the proposed decomposition procedure and the patterns we identified in our previous work. The decomposition procedure is detailed in Sect. 4, and Sect. 5 overviews the developed prototype. The last section concludes the paper and discusses future work.

2 Related Work

One of the early works on process decomposition is proposed within the Mentor project (Wodtke et al. 1996). The authors define workflow models by using state and activity diagrams and describe how a centralised model can be partitioned and executed in a distributed setting, maintaining the original semantics and taking into account control and data dependencies.

The growing use of the Web Services Business Process Execution Language (WS-BPEL) (OASIS 2007) justified the development of decomposition proposals of such processes. Nanda et al. (2004) transform a BPEL model into a program dependence graph (PDG). Based on the PDG, they propose an algorithm to create partitions by merging portable nodes with fixed nodes, taking into account control and data dependencies. Each partition has exactly one fixed node and zero or more portable nodes. However, this partitioning technique can only be applied to block structured models.

Fdhila et al. (2009) propose a decomposition technique based on dependency tables. They create a direct control dependency table and a direct data dependency table taking into account control and data dependencies. From the dependency tables, they generate transitive dependency tables with the transitive dependencies between activities invoking the same service. This way, each subprocess represents the control flow between activities invoking the same service.

In Domingos et al. (2015), the authors also use dependency tables to decompose IoT aware business processes, considering control flow as well as data flow. In addition, the activities that IoT devices can execute are identified automatically, taking into account the capabilities of these devices.

A subsequent work by Fdhila et al. (2014) decompose processes based on the collocation and separation constraints that designers can define between pairs of activities. Partitions respect the constraints and optimise communication cost and the Quality of Service (QoS) of services assigned to execute activities. Goettelmann et al. (2013) extend this work by adding security constraints to meet the requirements of distributing the executions of some activities into the cloud.

To make use of the advantages offered by the cloud to execute fragments of business processes, Duipmans et al. (2012) divide business processes into two categories: those that run locally and those that can run in the cloud. With this division, the authors intend to perform the most computationally intensive tasks in the cloud, as long as their data is not confidential. The identification of these tasks is performed manually. Povia et al. (2014) propose a semi-automatic mechanism to determine the location of activities and their data based on confidentiality policies, monetary costs, and performance metrics. Hoenisch et al. (2016) optimise the distribution of activities taking into account some additional parameters such as the cost associated with delays in the execution of activities and the unused, but paid, time of cloud resources.

Xue et al. (2018) create partial partitions using a graph based technique for process partitioning. They define the business model through a typed direct graph where edges are typed as control or data, to distinguish control from data dependencies, and vertices are typed as fixed or portable. Based on the direct graph, they identify SESE regions (Johnson et al. 1994) of the graph and generate a process structure graph. After that, they apply a set of transformation rules to group together portable vertices with a fixed vertex. Each group corresponds to a partition. The main difference to previous work is that they support unstructured business models (both control and data dependencies). However, this partition technique disregard the specificities of IoT aware business processes.

Martins et al. (2019) propose a decomposition technique that identifies specific patterns within processes that can be delegated to IoT devices, maintaining the control and the data flows of the original processes. Decomposition follows specific rules according to the identified patterns. This work is further generalised in Domingos et al. (2019), undermining, however, some control flow dependencies.

In this paper we take a step further following a graph-based approach, while assuring data and control dependencies.

3 Use Case Based Decomposition Patterns

This section presents our use case and the patterns that we have identified to reduce the number of communications (message flows) between the central pool and IoT pools (Martins et al. 2019).

3.1 Use Case

We exemplify the application of our proposal through a simplified automatic irrigation system use case. The choice for this use case is justified by the acquired experience in deploying an irrigation controlling system prototype in collaboration with the Lisbon city council. The system controls four electrovalves, managing a total of 40 sprinklers, and is successfully running for almost two years. Furthermore, by selecting a similar use case for illustrating our previous approaches (see, for instance, Martins et al. 2019), it makes it easier to compare them and to showcase our state-of-the-art advances.

This system automatically determines when to irrigate, based on the soil moisture and on the rainfall. The water used for the irrigation comes from tanks, whose water level is also controlled by the system. It is possible to check the water level of the tanks and fill them whenever necessary.

The water level values are stored in a historical record file for future expenses audit. In addition, the system periodically contacts IoT devices to gather rainfall and soil moisture levels. If the levels are below given thresholds, it triggers the irrigation process.

Figure 8 (in the appendix) illustrates the simplified BPMN model of our use case. The two pools define the behaviour of the central system as well as the behaviour of IoT devices (sensors and actuators).

The central pool defines the actions of the central system responsible for executing the irrigation business processes. It contains two execution flows: the top describes tank refilling; the other specifies the irrigation process itself.

The top execution flow is triggered manually (S1 start event) and starts by requesting the tank's water level (SENDT1), sending a message to the IoT pool (SM1). The sensor reads the tank's water level (T11) and forwards this information back to the irrigation process (SENDT12). The Receive Water Level task (RECEIVET2) blocks until a message arrives. Upon message arrival, it stores the information and forwards it (SENDT3) to the actuator (SM2) that triggers the Refill Water Tank task (T17) to set the tank's water level to the top. Meanwhile, the Save Refill Record task (T4) stores this occurrence by writing it to the Historical Record data store (H1).

The bottom flow executes periodically (ST1), and starts by determining if the rainfall and the soil moisture levels provided by the IoT device are within the acceptable range. For this, it requests the rainfall value (SENDT18) to the IoT pool that starts the process (SM6), reads the rainfall (T21), and sends a message with this information (SENDT22) back to the irrigation process. Then, the process checks this value (IF3) and terminates in case it has recently rained. Otherwise, it gets the soil moisture by sending a request to the IoT pool (SENDT5),

which starts process (SM3), reads the soil moisture value (T13), and sends it (SENDT14) back to the irrigation process.

Then, the process checks if it is necessary to start an irrigation cycle. For that, the exclusive gateway Check Moisture Values (IF1) forces the process to follow only one of its paths. If the moisture value is below the defined threshold, it computes the irrigation time based on the moisture level received (T7) and signals the actuators (SENDT8) to start irrigating (T15). The purpose of the irrigation intermediate timer event (ST2) is to wait for the irrigation time before sending a signal (SENDT9) to stop the actuator (T16). Finally, the process records the soil moisture level (T10) into the historical record data store (H2). The purpose of the converging gateway (IF2) is to forward the process to task T10, regardless of the path taken by the process.

This process, despite using IoT devices, takes a centralised approach. In the following sections, we use it to illustrate the various steps of our decomposition procedure.

3.2 Decomposition Patterns

This section presents the patterns that we have identified as ineffective uses of the computational capabilities of IoT devices in Martins et al. (2019). The main concern on the identification of the patterns, and their respective transformations, is to reduce the number of communications between the central process and to preserve the execution flow of the initial business process. This means that the tasks still execute in the original order after the transformations.

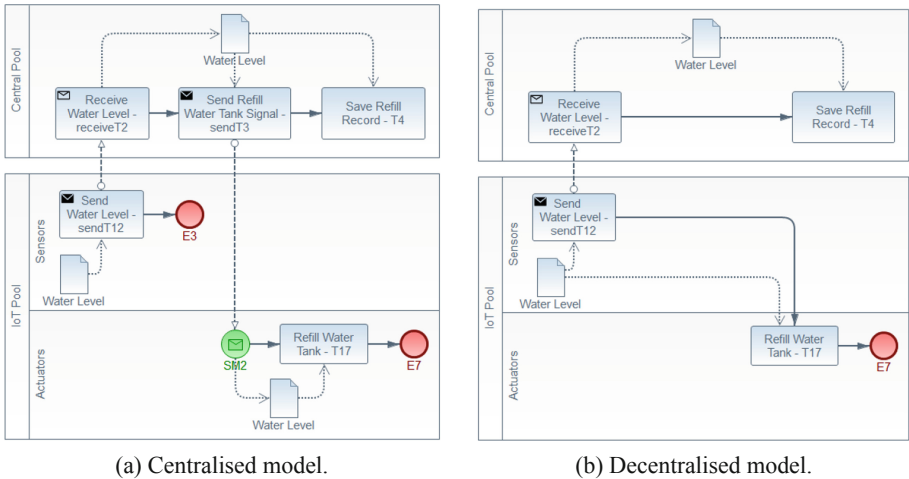


Fig. 1. Pattern 1 example instance taken from the use case - adapted from (Martins et al. 2019).

The first pattern, illustrated in Fig. 1a, is identified by a receive task (RECEIVET2) followed by a send task (SENDT3) in the central pool that are, respectively, preceded and followed by a send (SENDT12) and a receive task or start message event (SM2) belonging to the same IoT pool. This includes an unnecessary communication between the two pools: the transformation eliminates SENDT3 and SM2. To enforce the original control flow, RECEIVET2 is connected to T4 and SENDT12 is connected to T17, as Fig. 1b illustrates.

The second pattern is exemplified in Fig. 2a. The central pool process starts with a timer event (ST1) and is followed by a send task (SENDT18) and a receive task (RECEIVET19), both to and from the same IoT pool. By moving the timer to the IoT pool, we eliminate one communication between SENDT18 and SM6 (typically, IoT devices have timer operations), as Fig. 2b shows.

Figure 3a contains an excerpt of our use case that illustrates the third pattern. Typically, IoT devices have sufficient computational capabilities to perform logical and mathematical operations, so it is possible to transfer gateways to the IoT network, as long as the data for making the decision is available. Also script tasks that compute mathematical expressions can be moved to IoT devices as it is the case for task 7. This pattern is characterised by a message flow from the IoT pool to the central process (in this case, from SENDT14 to RECEIVET6), which, afterwards, branches (IF1 gateway) based on the received data. The goal is to transfer as much tasks as possible to the IoT pool, while preserving control and data flow dependencies. Figure 3b illustrates the application of this pattern to our running example. The message flow from the IoT pool to the central process is postponed as long as the central process has BPMN elements that can be moved to the IoT pool. This set of BPMN elements includes exclusive gateways, script tasks that only compute mathematical expressions, timer events, and send tasks targeted at the IoT pool.

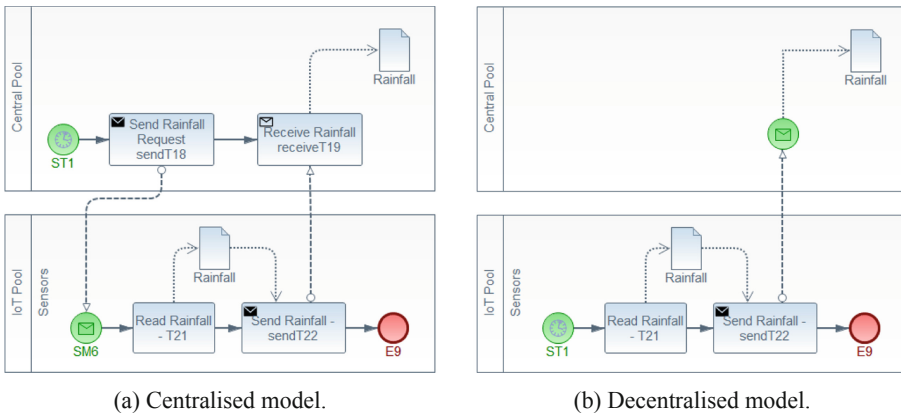


Fig. 2. Pattern 2 example instance taken from the use case - adapted from (Martins et al. 2019).

In Domingos et al. (2019), we generalised the presented patterns making them applicable to additional scenarios, but still relaxing control flow preservation. The result is that the overall meaning of the process remains the same, but some task that might happen in parallel in the original process may happen before others after the transformation, and vice versa. In the work we present in Sect. 4, the decomposed process maintains faithfully both the control flow and the data flow of the original process.

4 Graph Based Decomposition Patterns

Unlike our previous proposals (Martins et al. 2019; Domingos et al. 2019), the work we present here uses a graph based technique to identify which parts of the process IoT devices can execute. Before detailing our technique, this section starts by explaining the transformation from the BPMN business process model to a typed directed graph.

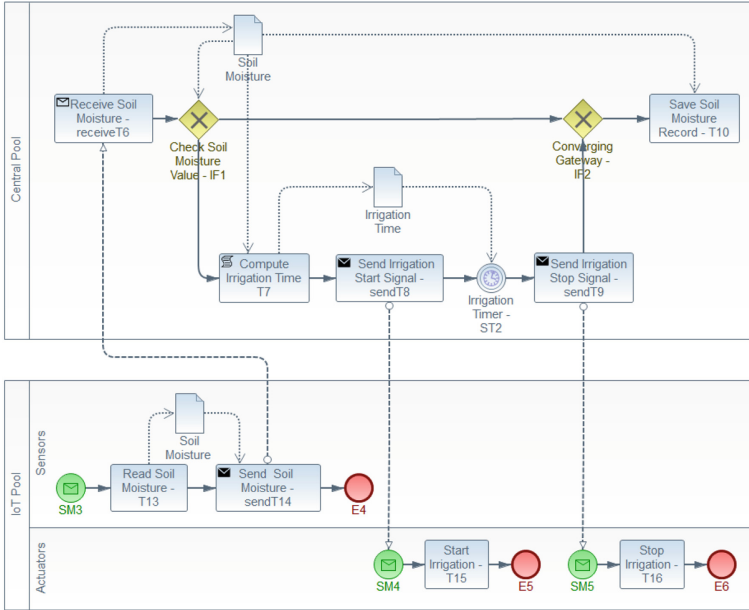
4.1 Defining a BPMN Process as a Typed Directed Graph

The first step to decompose a business process is to represent it as a typed directed graph, where vertices and edges are typed, meaning that we decorate vertices and edges with additional information that is then used by our decomposition algorithm.

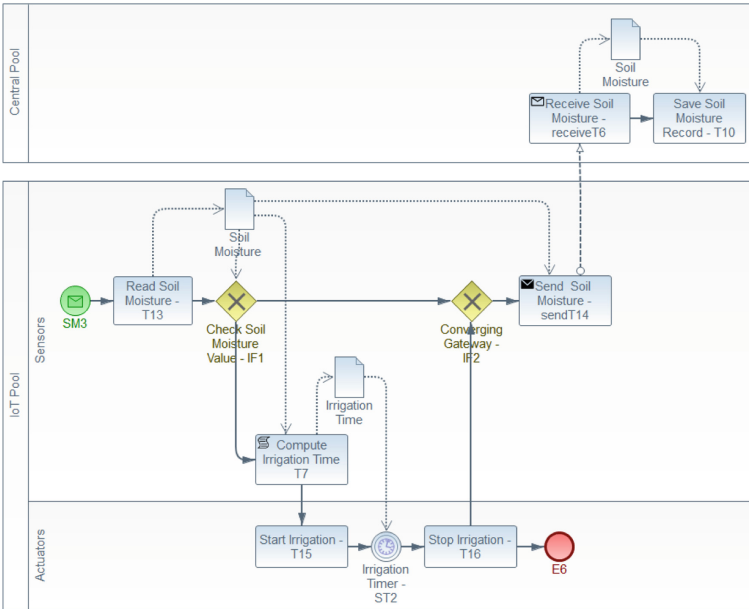
Edges can be of two types: control or data. Sequence flows and message flows are converted to control type edges. Data flow dependencies are derived using *def-use path* dependencies (Ammann and Offutt 2016), where a *def* corresponds to a write into a data object or a data store and a *use* represents an access. To identify data dependencies we exclude send and receive tasks, since these tasks are only used to handle the communication between pools.

Tasks, gateways, and events are converted to vertices. The ones that can be executed in either pools (gateways, timers, end events, and script tasks that only include mathematical operations) as well as send tasks and receive tasks are typed as portable. The others are typed with the name of the pool that represents the participant where they have to be executed.

We point out that the send and receive tasks are represented by fork and join vertices, maintaining the semantics of the concurrent behaviour of the executions flows.



(a) Centralised model.



(b) Decentralised model.

Fig. 3. Pattern 3 example instance taken from the use case - adapted from (Martins et al. 2019).

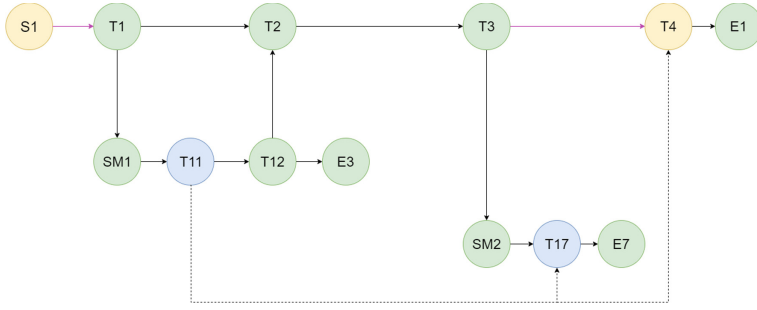


Fig. 4. The digraph of the execution flow that starts with S1. (Color figure online)

Figure 4 presents the typed directed graph that corresponds to the execution flow that begins with start event S1. We use colours to distinguish the type of vertices. Yellow nodes (S1 and T4) represent *Central Pool* vertices, blue nodes (T11 and T17) represent *IoT Pool* vertices, and green nodes represent portable vertices. Dashed and solid lines edges represent data and control types, respectively.

4.2 Decomposition Graph-Based Patterns

To reduce the number of exchanged messages between the central pool and IoT devices, we first identify the scenarios (designed as patterns) where it is possible to reduce messages (from two (or more) to one or from three (or more) to two).

The first scenario occurs when: (1) the graph has a cut, i.e., a partition of the graph into two disjoint subsets; (2) one of the cut sets includes one edge typed as control and zero or more edges typed as data; and (3) one of the subsets includes, at least, two message flows and all its nodes are typed as portable or *IoT Pool*. The rationale behind this scenario is that we can decompose the process, transferring the tasks in the subset satisfying (1), (2), and (3) to the IoT pool. This way, the two or more message flows of the subset are reduced to only one, the one that corresponds to the cut edge that links both subsets.

Figure 5 presents the typed directed graph of the execution flow that starts with ST1 timer event. This graph has four cuts according to the conditions defined for the first scenario. The control cut edge of each of these cuts is identified in pink. When a cut whose subset includes all the other, we select it as it reduces the number of transformations.

To get the final decomposed BPMN process from the graph, fork nodes that represent send tasks in the original process are replaced by diverging parallel gateways, while join nodes that represent receive tasks are replaced by converging parallel gateways. To link both partitions, we add a send task and a receive task (or a message start event) with a message flow. If the cut set also includes data typed edges, this message flow must include the corresponding values.

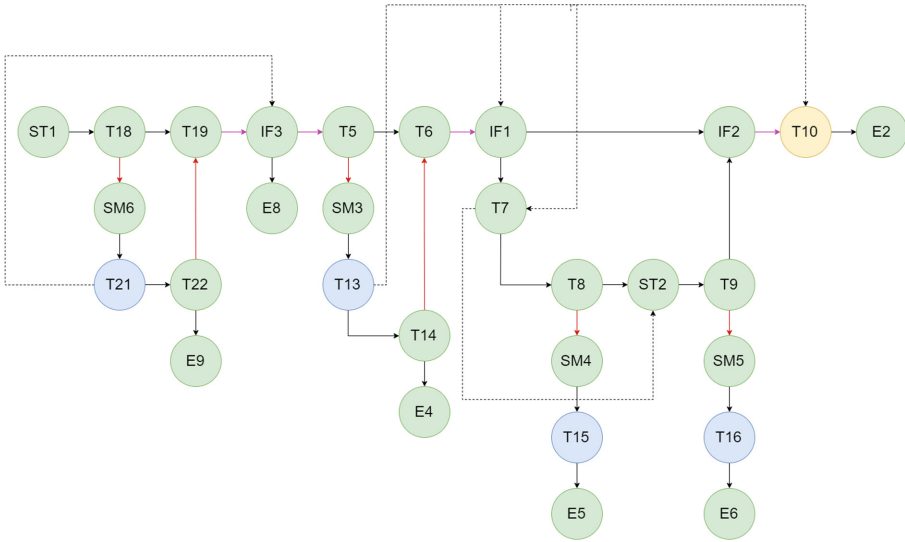


Fig. 5. The digraph of the execution flow that starts with ST1.

Figure 6 presents the decomposed version of the process for the execution flow that starts with ST1 timer event. In future work, we are going to identify and remove the parallel gateways that neither synchronise two or more control flows nor creates two or more parallel flows.

In the second scenario, it is possible to reduce from three (or more) messages to two messages. This scenario occurs when: (1) the graph has a cut; (2) the cut set includes two edges typed as control and zero or more edges typed as data; and (3) one of the subsets includes, at least, three message flows and all its nodes are typed as portable or as *IoT Pool*. The rationale behind this scenario is that we can decompose the process, transferring all subset task to the IoT pool. This way, the process is going to have only two message flows, the ones that correspond to the edges that belong to the cut set, i.e. that link both subsets.

Figure 4 presents the typed directed graph for the execution flow that starts with start event S1. This graph has one cut in the conditions defined for the second scenario. The control edges that belongs to the cut subset are identified in pink.

As we describe for the first scenario, to get the final decomposed BPMN process from the graph, fork nodes that represent send tasks in the original process are replaced by diverging parallel gateways, while join nodes that represent receive tasks are replaced by converging parallel gateways. To link both partitions, we add two send tasks and two receive tasks (or message start events) with message flows. If the cut subset also includes edges typed as data, these message flows must include the corresponding values.

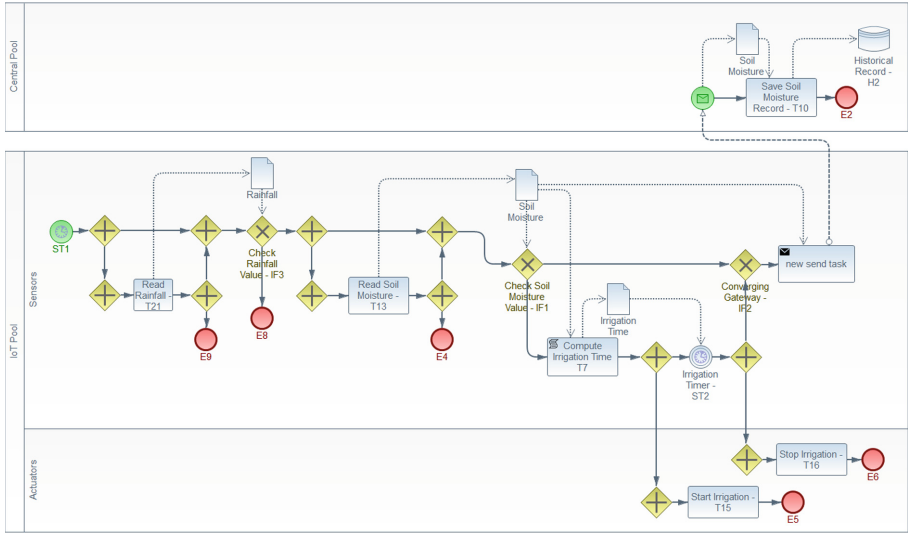


Fig. 6. The decomposed version of the BPMN business process - the execution flow that starts with ST1.

Figure 7 presents the decomposed process for the execution flow that starts with S1 start event.

5 Prototype

We are in the process of conceiving a prototype that performs the decomposition procedure described in the previous section. Our previous prototype tool is available at github (<https://github.com/fcmartins/bpmn-decomposition.git>).

The tool is being developed in Java and makes use of the following tools:

- jBPM (version 6.3.0);
- Eclipse Luna (version 4.4.2) with BPMN2 Modeller and SonarLint plug-ins;
- Graphviz.

The prototype builds on top of previous tools we developed that translate BPMN into CALLAS (a high-level sensor programming language (Lopes and Martins (2016))) and that automatically transform the BPMN model to communicate with the IoT network either using request-response or publish-subscribe architectures (Domingos and Martins 2017a, 2017b).

jBPM implements the BPMN standard and is associated with the Luna version of Eclipse. BPMN2 Modeller is a BPMN graphical visualisation plug-in and SonarLint a plug-in for enforcing source code quality. Graphviz is a tool for drawing graphs specified in the dot language.

In order to execute the prototype, we provide a BPMN file with the model to be decomposed. This prototype also translates the IoT behaviour into Callas bytecode to support the execution of all the BPMN model, as detailed in (Domingos and Martins (2017b)).

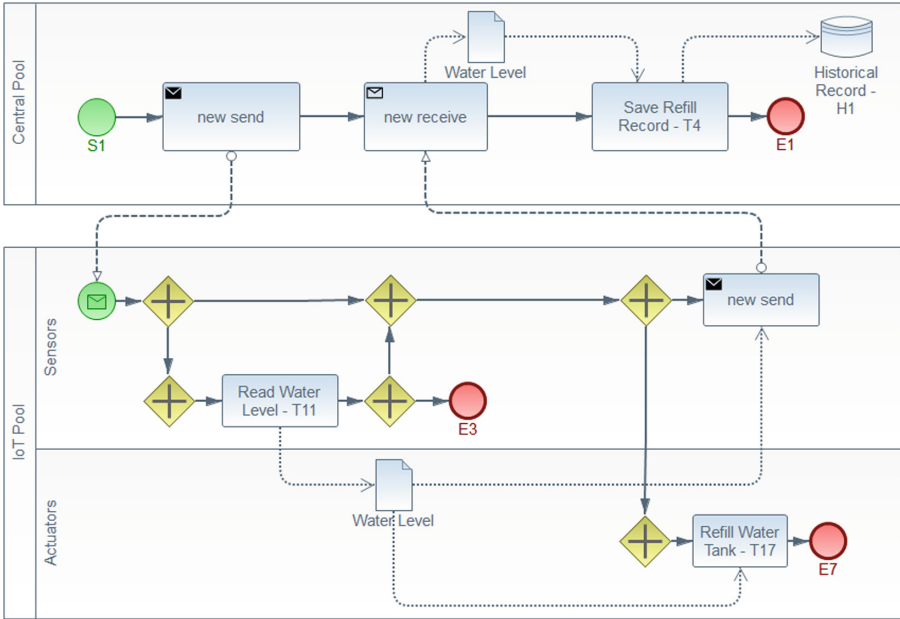


Fig. 7. The decomposed BPMN business process—the execution flow that starts with S1.

6 Conclusions and Future Work

Business processes are increasingly using information made available by IoT devices to provide timely responses according to their context. This is a challenge for IoT devices as they are limited by battery lifespan.

The work we propose in this paper mitigates this problem taking an approach from the business process perspective. The automatic decomposition technique we developed takes advantage of the computational capabilities of IoT devices, identifying the parts of process definitions that IoT devices can execute and transfers them from the central systems. This way, we reduce the number of exchanged messages, increasing the energy autonomy of these devices.

The evaluation of the completeness of our approach as well as its results is not straightforward. As stated before, the results have to be checked manually, being a very time-consuming task.

As for future work, we intend to formalise our approach in order to be able to prove the results. In addition, we plan to evaluate this work by comparing it with an heuristic based approach and using a larger set of processes generated by our process generator tool.

Acknowledgements. This work is partially supported by FCT funding through LASIGE Research Unit, ref. UID/CEC/00408/2018, and by project DoIT, ref. PTDC/EEI/ESS/5863/2014.

Appendix

A Simplified Irrigation System BPMN Process

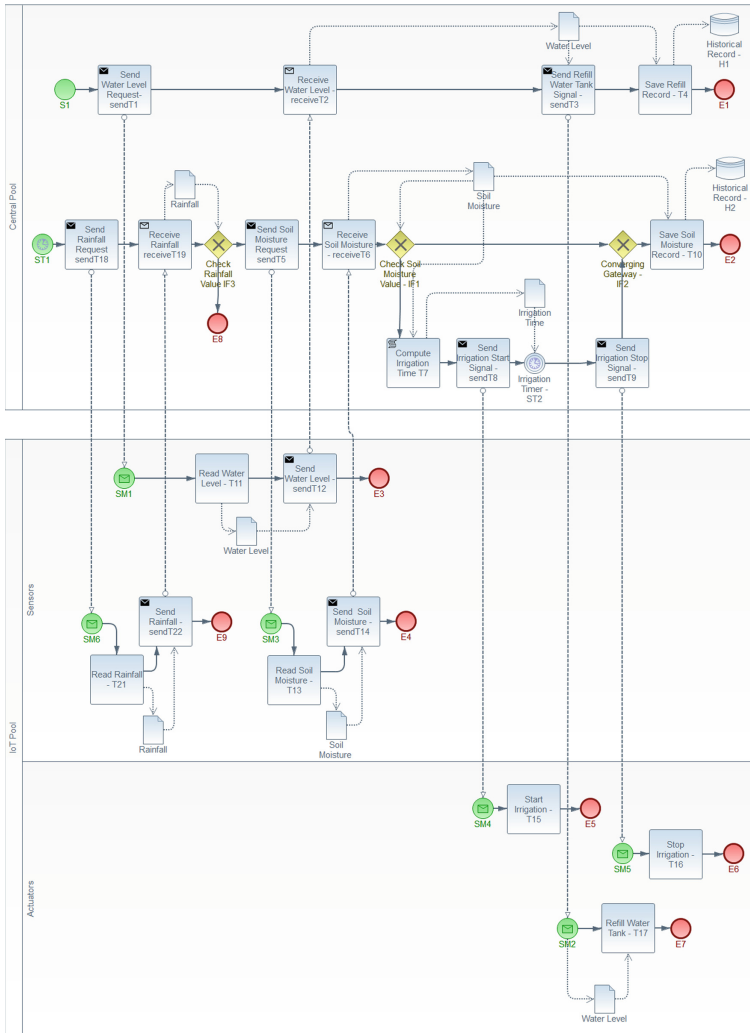


Fig. 8. BPMN model of an automatic irrigation system - a case study.

References

- Ammann, P., Offutt, J.: Introduction to Software Testing, 2nd edn. Cambridge University Press, New York (2016)
- Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. *Comput. Netw.* **54**(15), 2787–2805 (2010)
- Domingos, D., Martins, F.: Modelling iot behaviour within BPMN business processes. *Procedia Comput. Sci.* **121**, 1014–1022 (2017a)
- Domingos, D., Martins, F.: Using BPMN to model internet of things behavior within business process. *IJISPM-Int. J. Inf. Syst. Project Manag.* **5**(4), 39–51 (2017b)
- Domingos, D., Martins, F., Caiola, L.: Decentralising Internet of Things aware BPMN business processes. In: Kanjo, E., Trossen, D. (eds.) S-CUBE 2014. LNICST, vol. 143, pp. 110–119. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17136-4_12
- Domingos, D., Respicio, A., Martins, F., and Melo, B.: Automatic decomposition of IoT aware business processes - a pattern approach. *Procedia Comput. Sci.* 164, 313–320 (2019). CENTERIS 2019 - International Conference on ENTERprise Information Systems
- Duipmans, E.F., Pires, L.F., da-Silva Santos, L.O.B.: Towards a BPM cloud architecture with data and activity distribution. In: Proceedings of the 2012 IEEE 16th International Enterprise Distributed Object Computing Conference Workshops, pp. 165–171. IEEE (2012)
- Fdhila, W., Dumas, M., Godart, C., García-Bañuelos, L.: Heuristics for composite web service decentralization. *Softw. Syst. Model.* **13**(2), 599–619 (2014)
- Fdhila, W., Yildiz, U., Godart, C.: A flexible approach for automatic process decentralization using dependency tables. In: Proceedings of the 2009 IEEE International Conference on Web Services, (ICWS), pp. 847–855. IEEE (2009)
- Goettelmann, E., Fdhila, W., Godart, C.: Partitioning and cloud deployment of composite web services under security constraints. In: 2013 IEEE International Conference on Cloud Engineering (IC2E), pp. 193–200. IEEE (2013)
- Haller, S., Karnouskos, S., Schroth, C.: The Internet of Things in an enterprise context. In: Domingue, J., Fensel, D., Traverso, P. (eds.) FIS 2008. LNCS, vol. 5468, pp. 14–28. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00985-3_2
- Hoenisch, P., Schuller, D., Schulte, S., Hochreiner, C., Dustdar, S.: Optimization of complex elastic processes. *IEEE Trans. Serv. Comput.* **9**(5), 700–713 (2016)
- Johnson, R., Pearson, D., Pingali, K.: The program structure tree: computing control regions in linear time. *SIGPLAN Not.* **29**(6), 171–185 (1994)
- Lee, G.M., Kim, J.Y.: The Internet of Things problem statement. In: Proceedings of the 2010 International Conference on Information and Communication Technology Convergence (ICTC), pp. 517–518. IEEE (2010)
- Lopes, L., Martins, F.: A safe-by-design programming language for wireless sensor networks. *J. Syst. Architect.* **63**, 16–32 (2016)
- Martins, F., Domingos, D., Vitoriano, D.: Automatic decomposition of IoT aware business processes with data and control flow distribution. In: Proceedings of the 21st International Conference on Enterprise Information Systems, ICEIS 2019, Heraklion, Crete, Greece, 3–5 May 2019, vol. 2, pp. 516–524 (2019). <https://doi.org/10.5220/0007766405160524>
- Moreno, M., Úbeda, B., Skarmeta, A.F., Zamora, M.A.: How can we tackle energy efficiency in IoT based smart buildings? *Sensors* **14**(6), 9582–9614 (2014)

- Nanda, M.G., Chandra, S., Sarkar, V.: Decentralizing execution of composite web services. *SIGPLAN Not.* **39**(10), 170–187 (2004)
- OASIS: Web services business process execution language version 2.0 (2007). <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
- OMG: Business Process Model and Notation (BPMN), Version 2.0, January 2011. <http://www.omg.org/spec/BPMN/2.0>
- Povoa, L.V., de Souza, W.L., Pires, L.F., do Prado, A.F.: An approach to the decomposition of business processes for execution in the cloud. In: Proceedings of the 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA), pp. 470–477. IEEE (2014)
- Rault, T., Bouabdallah, A., Challal, Y.: Energy efficiency in wireless sensor networks: a top-down survey. *Comput. Netw.* **67**, 104–122 (2014)
- Wodtke, D., Weißenfels, J., Weikum, G., Dittrich, A.K.: The mentor project: Steps towards enterprise-wide workflow management. In: Proceedings of the Twelfth International Conference on Data Engineering, pp. 556–565. IEEE (1996)
- Xue, G., Liu, J., Wu, L., Yao, S.: A graph based technique of process partitioning. *J. Web Eng.* **17**(1&2), 121–140 (2018)
- Yousfi, A., de Freitas, A., Dey, A.K., Saidi, R.: The use of ubiquitous computing for business process improvement. *IEEE Trans. Serv. Comput.* **9**(4), 621–632 (2016)
- Zorzi, M., Gluhak, A., Lange, S., Bassi, A.: From today's Intranet of Things to a future Internet of Things: a wireless and mobility-related view. *IEEE Wirel. Commun.* **17**(6), 44–51 (2010)