UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

Descentralização de Computação na Internet das Coisas

Daniel de Almeida Vitoriano

Mestrado em Engenharia Informática

Especialização em Arquitetura, Sistemas e Redes de Computadores

Dissertação orientada por:

Francisco Cipriano da Cunha Martins

Maria Dulce Pedroso Domingos

2017

Agradecimentos

Agradeço o apoio prestado pela FCT – Fundação para a Ciência e a Tecnologia, através do financiamento do projeto DOIT, com a referência PTDC/EEI-ESS/5863/2014 e à Unidade de Investigação LASIGE, com a referência UID/CEC/00408/2013.

Muito obrigado ao professor Francisco e à professora Dulce pela paciência e cuidado com que me ajudaram a construir este trabalho.

Resumo

Quando utilizada nos processos de negócio, a Internet das Coisas (IoT) é geralmente encarada como uma plataforma distribuída de recolha de informação. No entanto, a IoT tem capacidade computacional que pode e deve ser utilizada de forma a maximizar a sua autonomia energética, reduzir a quantidade de informação trocada e executar partes dos processos. De facto, os processos de negócio ainda são definidos seguindo uma abordagem centralizada, dificultando o uso das capacidades destes dispositivos. Nesta dissertação apresentamos uma solução automática de decomposição de processos de negócio BPMN 2.0 (*Business Process Model and Notation*) dependentes da IoT. Partimos de um processo de negócio que segue a abordagem centralizada e, aplicando um método de decomposição, transferimos para os dispositivos da IoT as operações que podem ser aí realizadas, tendo em conta o fluxo de controlo e o fluxo de dados. Para isso, identificámos padrões em modelos BPMN cuja ineficiente utilização de dispositivos IoT é evidente. Começamos por gerar um grafo, a partir de um modelo BPMN, que capture as dependências de fluxo de controlo e de dados das tarefas. Depois, identificamos os caminhos que só contêm elementos BPMN capazes de serem executados em dispositivos IoT e que poderão ser transferidos para esta rede. Em seguida, com base nestes caminhos, verificamos quais se enquadram nos padrões que identificámos e aplicamos o procedimento de transformação correspondente. Por último, redesenhamos o modelo BPMN com as alterações efetuadas. Esta solução permite reduzir o processamento central e o número de mensagens trocadas na rede. Elaborámos e submetemos a nossa solução a diversos modelos BPMN como forma de validação, com a garantia de que o fluxo de execução do processo original nunca é quebrado.

Palavras-chave: Internet das Coisas, Processos de Negócio, BPMN, Decomposição, Descentralização

Abstract

When used in business processes, the Internet of Things (IoT) is generally seen as a distributed information gathering platform. However, IoT has computational capacity that can and should be used in order to maximize its energy autonomy, reduce the amount of information exchanged, and execute portions of the processes. In fact, business processes are still defined following a centralized approach, making it difficult to use the capabilities of these devices. In this dissertation, we present an automatic decomposition solution for Business Process Model and Notation (BPMN 2.0) business processes dependent on IoT. We start from a business process that follows the centralized approach and, by applying a decomposition method, we transfer to the IoT devices the operations that can be performed there, considering the control flow and the data flow of the original process. For this, we have identified patterns in BPMN models whose inefficient use of IoT devices is evident. We begin by generating a graph, from a BPMN model, that captures the control and data flow dependencies of the tasks. Then, we identify paths that only contain BPMN elements that can run on IoT devices and that can be transferred to this network. Then, based on these paths, we check which ones fall into the patterns that we identified and apply the corresponding transformation procedure. Finally, we redraw the BPMN model with the changes made. This solution reduces the central processing and the number of messages exchanged in the network. We have prepared and submitted our solution to several BPMN models as a form of validation, with the guarantee that the execution flow of the original process is never broken.

Keywords: Internet of Things, Business Processes, BPMN, Decomposition, Decentralization

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1
Introdução

Neste capítulo é realizada uma breve motivação ao tema e à problemática que propomos resolver com a nossa solução. São também definidos os objetivos desta dissertação, os seus contributos e a estrutura seguida por este documento.

1.1 Motivação

A IoT tem o potencial para ser um dos catalisadores da nova revolução industrial, motivada pelo rápido desenvolvimento das tecnologias digitais e por uma mudança de paradigma nas metodologias de negócio e no quotidiano das pessoas. É caracterizada pela presença de eletrónica nos objetos, tais como etiquetas RFID (*RadioFrequency IDentification*), sensores, atuadores ou dispositivos móveis, que, através de esquemas de endereçamento único, são capazes de interagir entre si cooperando de modo a atingir um objetivo comum. É um paradigma que torna os objetos que nos rodeiam em atores ativos da Internet, gerando e consumindo informação. A literatura científica tipicamente procura colmatar os seguintes problemas provenientes destes objetos: eficiência energética, que é o recurso mais escasso nestes dispositivos, escalabilidade, visto que o número de dispositivos pode ser bastante elevado, fiabilidade, pois a rede pode ser utilizada para reportar eventos do tipo alarme, e robustez, já que os nós da rede são suscetíveis ao mais variado tipo de falhas [15, 16, 17, 18, 19].

Um modelo de processo é um modelo conceptual de como os negócios realizam as suas operações através da especificação de atividades, eventos, estados, lógica de fluxo de controlo, entre outros fatores. Os modelos de processo são críticos para a otimização e automatização de processos de negócio e são muitas vezes representados numa notação gráfica, como é o caso da norma BPMN [13]. Estes processos podem utilizar as capacidades que os dispositivos IoT possuem para ler e atuar sobre o meio circundante ao processo. Inclusivamente, podem aproveitar a IoT para executar parte do seu fluxo de execução quando devidamente decompostos.

A arquitetura centralizada, muitas vezes encontrada em processos de negócio, potencia a existência destes problemas. Esta arquitetura penaliza o desempenho do sistema pelo número de dispositivos que a rede IoT contém. Isto porque, como grande parte das decisões a efetuar têm que passar pelo sistema central, torna-se incontornável a existência de elevado tráfego na rede, o que, por sua vez, desgasta o tempo útil de vida dos dispositivos, sobrecarrega o sistema central e desincentiva a utilização de grandes quantidades de dispositivos. Estas características limitam a qualidade e fiabilidade destes sistemas. Por outro lado, a descentralização promove escalabilidade, o que reduz a quantidade de mensagens trocadas na rede e reduz o processamento centralizado que é preciso realizar.

A decomposição é um método de dividir um sistema em subsistemas progressivamente mais pequenos que são responsáveis por alguma parte do problema do domínio. A decomposição é também um meio para modelar grandes sistemas e para facilitar a reutilização de modelos parciais, porque reduz o acoplamento, favorece a compactação de especificações e permite que vários níveis de detalhe possam coexistir. Como consequência, os modelos tornam-se mais fáceis de compreender o que, por sua vez, facilita a sua validação, redesenho e otimização [12]

1.2 Objetivos

O objetivo desta dissertação é o de decentralizar e otimizar operações na camada de negócio, de forma automática, que dependam da IoT. Mais concretamente, transformar modelos de processos através da elaboração de um procedimento automático de decomposição de processos de negócio que permite a dispositivos IoT executarem partes da lógica de negócio.

Foram definidos os seguintes objetivos:

1. **Definição de um algoritmo que represente uma possível solução para o problema:** a fim de compreender o problema em causa e definir estratégias para a criação de um algoritmo.

   Desafios:

   - Identificação de padrões em processos de negócio em que a rede de dispositivos IoT está a ser utilizada ineficazmente;
   - Identificação de elementos BPMN que podem ser executados por dispositivos da rede IoT;
   - Definição do algoritmo de decomposição.

2. **Implementação do procedimento automático de decomposição de processos de negócio.**

Desafios:

- Não quebrar o fluxo de execução dos processos de negócio;
- Garantir que o número de passos de comunicação, após a execução da transformação automática é igual ou inferior do que o processo original;

3. **Validação da solução desenvolvida:** para validar e testar a solução desenvolvida de modo a garantir que esta faz o que é esperado e correto.
   Desafios:
   - Identificação de casos de uso que contribuam para a validação da solução;
   - Averiguação de exceções não previstas no desenvolvimento da solução;

1.3 Contribuições

- Desenvolvemos uma solução que decompõe de forma automática processos de negócio de modo a aproveitar os recursos computacionais que os dispositivos IoT dispõem, reduzindo o número de comunicações realizadas;
- Identificámos um conjunto de padrões onde é benéfico remover, reorganizar ou transferir elementos para a rede IoT, aumentando assim o aproveitamento das capacidades computacionais dos dispositivos que a constituem;
- Elaborámos os algoritmos de transformação que compõem a nossa solução, com base nos padrões identificados;
- Publicação do trabalho no INForum 2017.

1.4 Estrutura do documento

Este documento está organizado da seguinte forma:

- Capítulo 2 – Introdução de conceitos e tecnologias relacionadas com a nossa solução e descrição de projetos relacionados com o tema;
- Capítulo 3 – Descrição dos algoritmos e padrões que compõem a nossa solução. Ilustração, através de um caso de uso, de um modelo BPMN que não aproveita da melhor forma os recursos disponíveis da rede de dispositivos IoT e aplicação da nossa solução a este caso de uso;
- Capítulo 4 – Apresentação do protótipo desenvolvido e das tecnologias que o compõem;
- Capítulo 5 – Conclusão e trabalho futuro a realizar.

Capítulo 2

Conceitos e trabalho relacionado

Este capítulo descreve as tecnologias e conceitos que estão diretamente relacionados com a solução proposta, bem como os projetos relacionados com o tema.

2.1 Notação para modelação de processos de negócio

A Notação para Modelação de Processos de Negócio (BPMN) é a principal linguagem utilizada no contexto deste projeto. Trata-se de uma das especificações da OMG, que está atualmente na versão 2.0. Ao longo desta dissertação utilizamos simplesmente BPMN para referenciarmos esta versão.

A norma BPMN fornece uma notação que pretende ser compreensível por todos os utilizadores de negócio, desde analistas de negócio que criam os esboços iniciais dos projetos, aos responsáveis por implementar as tecnologias que irão realizar esses processos e às pessoas que irão monitorizar esses processos. Deste modo, a norma BPMN pretende fechar a lacuna existente entre o desenho e execução de processos de negócio e sua a implementação [21].

Para a criação de um modelo de processo utilizando a norma BPMN é necessário ter em atenção as seguintes questões:

- O que dá início (eventos de início) ao processo e quando este termina (eventos de fim)?
- Que passos (tarefas, subprocessos) são necessários?
- Que dependências existem entre as atividades (fluxos de controlo)?
- Que caminhos alternativos existem (*gateways*) e se poderão haver atividades que ocorrem em paralelo?
- Quem é o responsável por executar uma atividade (participantes)?
- Que informações (objetos de dados) são relevantes para um processo?
- Onde ocorrem pontos de interação (fluxos de mensagens) com outros processos?

Existem quatro tipos de diagramas na norma BPMN: processos, colaborações, conversações e coreografias. Um processo descreve uma sequência ou fluxo de atividades numa organização com o objetivo de realizar trabalho. Os processos podem representar pessoas, cargos ou sistemas. Um diagrama do tipo processo contém um único participante (*pool*) e pode ser retratado como um grafo de elementos de fluxos que definem as suas semânticas de execução. O termo elemento de fluxo designa os elementos BPMN de um processo e engloba atividades, *gateways*, eventos, objetos de dados e fluxos de sequência. As atividades representam trabalho a efetuar no processo de negócio. Uma atividade pode ser atómica ou não atómica e pode conter associações de dados. Os tipos de atividades que existem são tarefas, subprocessos e chamadas de atividade. Outro termo que faz sentido introduzir é o de elementos de raiz que é um conceito abstrato que engloba os elementos BPMN que estão contidos no modelo. Exemplos de elementos de raiz incluem colaborações, processos e *data stores*.

As colaborações e coreografias são utilizadas para modelar interações entre processos, ou seja, são constituídas por dois ou mais participantes (processos) e suas interações (fluxos de mensagem). Os diagramas de colaboração focam-se nas comunicações entre participantes detalhando as mensagens por estes trocadas. Os diagramas de conversação são, de um modo geral, uma versão simplificada dos diagramas de colaboração. Representam a troca de informação a um nível mais abstrato, utilizando objetos de conversa em vez de fluxos de mensagens individuais. Estes diagramas fornecem uma visão geral de quais os participantes que cooperam em determinadas tarefas.

As coreografias podem ser vistas como um contrato estabelecido entre duas ou mais organizações. Uma coreografia formaliza o modo como os participantes coordenam as suas interações. O foco não está no trabalho produzido em cada participante, mas sim na troca de informação entre participantes. Os diagramas de coreografia definem a sequência de interações entre participantes e podem ser utilizados para analisar o modo que estes trocam informação de forma a coordenar as suas interações.

Segue-se um conjunto de tabelas que contêm a representação gráfica, e respetivo significado, de elementos BPMN que serão utilizados ao longo desta dissertação.

A Tabela 2.1 ilustra diversos tipos de eventos BPMN. Os eventos ditam o começo e fim de processos (ou atividades) mediante a ocorrência de condições. Exemplos de condições que justificam a utilização de um evento são a chegada de uma mensagem, alarmes, condições relacionadas com uma duração ou instante temporal, sinais, erros, terminação abrupta de processos, entre outros. Quando estas condições ocorrem algures entre o início e o fim de um processo, são utilizados eventos intermédios. Dentro dos três tipos de eventos é possível agrupá-los em duas categorias: eventos que sinalizam condições (incorpora todos os eventos de início e alguns eventos intermédios) e eventos que detetam um resultado (incorpora todos os eventos de

fim e alguns eventos intermédios). Os resultados associados a um evento poderão eventualmente ser detetados por outro evento.

Tabela 2.1 - Tipos de eventos BPMN

Símbolo	Nome	Função
	Evento de início	Representa o início de um processo
	Evento de início condicionado por uma mensagem	Sinaliza que um processo depende de uma mensagem para iniciar a sua execução
	Evento de início condicionado por um instante	Sinaliza que um processo depende de um instante, uma data ou um tempo recorrente para iniciar a sua execução
	Evento intermédio	Indicam que algo ocorre algures entre o início e o fim de um processo
	Evento intermédio condicionado por um instante	A execução do processo é adiada a um ponto no tempo ou durante um período de tempo
	Evento de fim	Assinala o fim de um processo

A Tabela 2.2 contém os três principais tipos de tarefas comummente encontradas em modelos BPMN. As tarefas representam pontos no processo onde é necessário realizar trabalho, sendo por isso elementos executáveis para a norma BPMN. São também conhecidas como atividades atómicas. A norma BPMN especifica três tipos de marcadores que podem ser utilizados em tarefas: marcador de iteração simples, marcador de múltiplas instâncias e marcador de compensação. Uma tarefa pode conter até dois tipos de marcadores. O marcador de iteração simples (representado por uma seta curvada) sinaliza que a tarefa de repetição será iterada enquanto a condição de recursão for verdadeira. Esta condição é avaliada antes de cada iteração. Em adição a esta condição, poderá ser definido um limite numérico, caso se preveja um cenário em que o número máximo de iterações seja atingido. De notar que a condição de repetição pode ser explicitamente definida através de um evento intermédio. O marcador de múltiplas instâncias (representado por três linhas verticais ou horizontais) é utilizado quando são necessárias várias instâncias de uma mesma tarefa. Isto significa que uma tarefa é realizada

várias vezes com conjuntos de dados diferentes. Se a tarefa puder ser executada em paralelo, deverá ser utilizado um marcador de múltiplas instâncias para instâncias paralelas (representado por três riscas verticais). Caso apenas seja possível executar a tarefa sequencialmente, então deverá ser utilizado o marcador de múltiplas instâncias para instâncias sequenciais (representado por três linhas horizontais). Os marcadores de compensação (representados por um símbolo de retrocesso) indicam que a tarefa será executada mediante o cancelamento da execução de outra tarefa. Este tipo de marcador é tipicamente utilizado em processos que retratam a ocorrência de transações.

Tabela 2.2 - Tipos de tarefas BPMN

Símbolo	**Nome**	**Função**
	Tarefa	Representa uma atividade e é também conhecida como tarefa atómica
	Tarefa com os marcadores de iteração simples e compensação	Tarefa de repetição que só pode ser executada quando um processo de transação é cancelado
	Tarefa de envio de mensagens	Envia uma mensagem a um participante externo relativo ao processo. A tarefa termina assim que a mensagem é enviada
	Tarefa de receção de mensagens	Espera pela chegada de uma mensagem de um participante externo ao processo. Após a mensagem ser recebida, a tarefa termina
	Tarefa de serviço com marcador de múltiplas instâncias para instâncias paralelas	Representa uma tarefa que utiliza algum tipo de serviço e será executada, de forma paralela, enquanto a condição de repetição fornecida for verdadeira
	Tarefa de utilizador	Tarefa realizada por um interveniente humano com a assistência de uma aplicação de *software*
	Tarefa manual	Indica uma tarefa que é realizada sem a assistência de aplicações de *software*

	Tarefa de *script*	Representam *scripts* implementados numa linguagem interpretável pelo motor de processo de negócio de modo a serem executados por este

Na Tabela 2.3 podemos encontrar os três tipos de *gateways* mais comuns. Os *gateways* são utilizados para controlar o fluxo de sequência dentro de um processo. Mediante as condições fornecidas, os *gateways* autorizam ou não que o fluxo de execução tenha continuidade ou prossiga outro caminho. Como não representam "trabalho" a ser efetuado, na prática, os *gateways* não representam qualquer impacto nas medidas operacionais do processo que está a ser executado (em termos de custo e tempo, por exemplo).

Tabela 2.3 - Tipos de gateways BPMN

Símbolo	**Nome**	**Função**
	*Gateway* exclusivo	Avalia uma condição dada e divide o fluxo do processo em um ou mais fluxos mutualmente exclusivos. O processo apenas poderá seguir um dos caminhos
	*Gateway* paralelo	É utilizado para sinalizar o início da execução de atividades concorrentes
	*Gateway* inclusivo	Divide o fluxo do processo em um ou mais fluxos com base numa condição. O processo segue todos os caminhos cuja condição se verifique
	*Gateway* baseado em eventos	*Gateway cujo* comportamento é determinado por uma configuração de elementos
	*Gateway* complexo	É utilizado em sincronizações que apresentam comportamentos complexos, através de uma expressão de ativação de condição que descreve de forma precisa o comportamento esperado

A Tabela 2.4 ilustra os dois itens de objetos de dados utilizados na norma BPMN. Podem ser utilizadas referências para reutilizar um mesmo objeto de dados ou *data store* num processo. No caso dos objetos de dados, as referências podem especificar diferentes estados de um mesmo objeto em diferentes pontos de um processo. No caso das *data stores*, podem ser utilizadas como origem ou destino de associações de dados. As *data stores* fornecem um mecanismo para as atividades recolherem ou atualizarem informação armazenada que irá persistir além do âmbito do processo.

Tabela 2.4 - Tipos de itens de objetos BPMN

Símbolo	**Nome**	**Função**
	Objeto de dados	Representa informação acessível a um processo, tal como documentos de negócio, e-mails ou cartas
	*Data store*	Representa informação, acessível a um ou mais processos, guardada de forma persistente

Estão presentes na Tabela 2.5 vários tipos de ligações utilizadas na norma BPMN. Estes sinalizam os fluxos de execução de um dado processo.

Tabela 2.5 - Tipos de ligações BPMN

Símbolo	**Nome**	**Função**
	Fluxo de sequência	Liga elementos por ordem sequencial
	Fluxo de mensagem	Representa a troca de mensagens de um participante de um processo para outro
	Associação	É utilizado para ligar informação e artefactos a outros elementos BPMN
	Associação de dados	É utilizado para visualizar entradas e saídas de objetos de dados e *data stores*

A semântica a utilizar para elementos ligados por fluxos de sequência é que um elemento está pronto a ser executado após os seus antecessores estarem concluídos. De notar

que um evento de início não pode conter fluxos de sequência de entrada e um evento de fim não pode conter fluxos de sequência de saída. Podem ser definidas condições em fluxos de sequência de saída, quando estes estão ligados a atividades ou *gateways* inclusivos ou exclusivos.

Os fluxos de mensagem representam a troca de mensagens dentro de um processo. É utilizado para mostrar o fluxo de mensagens entre dois participantes que estão preparados para as enviar ou receber, não podendo, por isso, ser utilizado para ligar elementos dentro de um mesmo participante. Ao contrário dos fluxos de sequência, é possível que eventos de início contenham fluxos de mensagem de entrada e eventos de fim contenham fluxos de mensagem de saída (desde que estejam condicionados por uma mensagem).

As associações de dados indicam o fluxo de informação existente nos processos. Caso sejam unidirecionais, indicam que um objeto de dados pode ser lido no início da execução de uma atividade ou escrito quando a atividade conclui a sua execução. Se forem bidirecionais, então o objeto de dados é modificado (lido e escrito) durante a execução da atividade. As associações de dados possuem uma origem, um destino e uma transformação opcional. O que é copiado (escrito) depende da existência de uma transformação. Caso não haja nenhuma transformação definida ou referenciada, o conteúdo da origem será copiado para o destinatário. As transformações são definidas através de *scripts* elaborados numa linguagem interpretável pela norma BPMN (como Java ou XPath). Em alternativa às associações de dados, existem as associações. Estas podem ser ligadas a fluxos de sequência para indicar que as atividades envolvidas possuem as mesmas relações de entrada e de saída. Quer isto dizer que, no caso de um objeto de dados possuir uma associação ligada diretamente a um fluxo de sequência, a interpretação a fazer é que as atividades de origem e destino desse fluxo são as mesmas que o objeto de dados possui como entrada e saída, respetivamente. Este cenário é exemplificado na Figura 2.1.

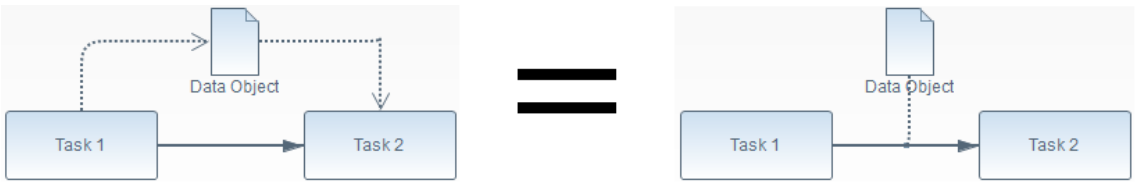

Figura 2.1 - Representação equivalente de fluxo de dados

2.2 Grafos

No contexto desta dissertação, são utilizados grafos direcionados. Um grafo é um quadruplo (N, A, I, F), onde os elementos de N são chamados nós (vértices), os de A são chamados arestas, os de I são os nós iniciais e os de F são os nós finais. Um nó é um par (T, P), onde T corresponde a um elemento BPMN e P ao nome do participante a que o elemento pertence. Uma aresta denota uma dependência entre dois elementos por um fluxo de sequência ou um fluxo de mensagem. Um caminho é uma sequência de nós tal que existem arestas entre nós contíguos do caminho. Um subcaminho corresponde a parte de uma sequência de nós de um caminho maior.

2.3 Trabalho relacionado

O projeto Mentor apresenta uma das primeiras propostas sobre decomposição de processos [10]. Os processos são definidos através de diagramas de estados e a sua decomposição em partições tem em conta o fluxo de controlo e o fluxo de dados.

A utilização crescente da *Web Services Business Process Execution Language* (WS-BPEL) [7] justificou o desenvolvimento de propostas sobre decomposição de processos WS-BPEL. Nanda et al. [6] criam novos subprocessos para cada serviço do processo original, os quais comunicam diretamente, evitando que a sincronização e a troca de mensagens sejam da exclusiva responsabilidade do motor de execução de processos central. No entanto, estes autores não consideram a hipótese de agrupar serviços no mesmo subprocesso. Sadiq et al. [9] e Fdhila et al. [4] decompõem modelos de processos genéricos agrupando várias atividades em subprocessos e distribuindo a execução destes subprocessos por diferentes motores de execução de processos. Sadiq et al. apenas consideram as dependências do fluxo de controlo, enquanto que Fdhila et al. também consideram as dependências de fluxo de dados. Em [3], Fdhila et al. otimizam a composição de subprocessos tendo em conta vários parâmetros de qualidade de serviços, tais como, custo, tempo, fiabilidade e disponibilidade. Yu et al. [11] utilizam programação genética para criar partições (subprocessos) de processos WS-BPEL que utilizam dados de forma intensiva.

Com o objetivo de usufruir das vantagens oferecidas pela nuvem para a execução de partes dos processos de negócio, Duipmans et al. [2] dividem os processos de negócio em duas partes: a que é executada localmente e a que é executada na nuvem. Com esta divisão, os autores pretendem executar na nuvem as tarefas computacionalmente mais intensas e cujos dados não tenham requisitos de confidencialidade. A identificação destas tarefas é feita manualmente. Povoa et al. [8] propõem um mecanismo semiautomático para determinar a localização das atividades e dos respetivos dados baseado em políticas de confidencialidade, custos monetários

e métricas de desempenho. Hoenisch et al. [5] otimizam a distribuição das atividades tendo em conta alguns parâmetros adicionais tais como o custo associado a atrasos na execução de atividades e o tempo não usado, mas pago, de recursos da nuvem. Yousf et al. [14] tiram partido da computação ubíqua e definem processos de negócio ubíquos, como forma de melhorar o desempenho prestado pelos processos de negócio e estendem a norma BPMN para que este suporte requisitos adicionais que são específicos da computação ubíqua, de forma a permitir a modelação de processos de negócio ubíquos.

Em [1] apresentamos uma proposta preliminar ao problema de decomposição de processos de negócio BPMN, de modo a que partes desses processos possam ser executados em dispositivos IoT. A decomposição é baseada em tabelas de dependências tendo em conta apenas o fluxo de controlo. A identificação das partes do processo de negócio que podem ser incluídas em partições a serem executadas pelos dispositivos IoT é feita automaticamente, tendo em conta as capacidades destes dispositivos. O trabalho aqui apresentado considera adicionalmente as restrições derivadas do fluxo de dados. As partições a serem executadas nos dispositivos IoT podem incluir também dados não persistentes do processo. O procedimento de decomposição dos processos prescinde das tabelas de dependências, reduzindo a memória e a computação utilizadas. Os caminhos candidatos a partições são gerados diretamente do grafo criado a partir do modelo definido em BPMN.

Capítulo 3

Decomposição de processos de negócio

Este capítulo descreve a nossa solução de decomposição de processos de negócio com foco na redução de comunicações entre dispositivos IoT e o sistema central. Para ilustrar, utilizamos um processo de negócio de um sistema de rega automática simplificado que serve como caso de uso. Por fim, recorremos a alguns exemplos de modelos BPMN com condições que nos permitem decompor.

3.1 Caso de uso

Uma organização utiliza um sistema de rega automática que controla o nível de água presente em tanques de armazenamento e verifica a necessidade de iniciar uma rega em função da humidade do solo. Neste sistema, sempre que é lido o valor do nível de água dos tanques na rede de dispositivos IoT, é feito um pedido para repor a água em falta. Desta forma, existe a garantia que as irrigações decorrerão com a quantidade máxima de água disponível.

O valor de água reposta é guardado num histórico para futura auditoria de despesas. Além disso, o sistema contacta periodicamente a rede de dispositivos IoT para determinar a humidade do solo. Caso esta esteja abaixo dos valores pré-estabelecidos, é enviado um sinal para a rede de dispositivos IoT que desencadeia o processo de rega. Os valores de humidade do solo são mantidos num histórico de forma a ajustar intervalo de tempo entre comunicações com a rede de dispositivos IoT consoante o período climático.

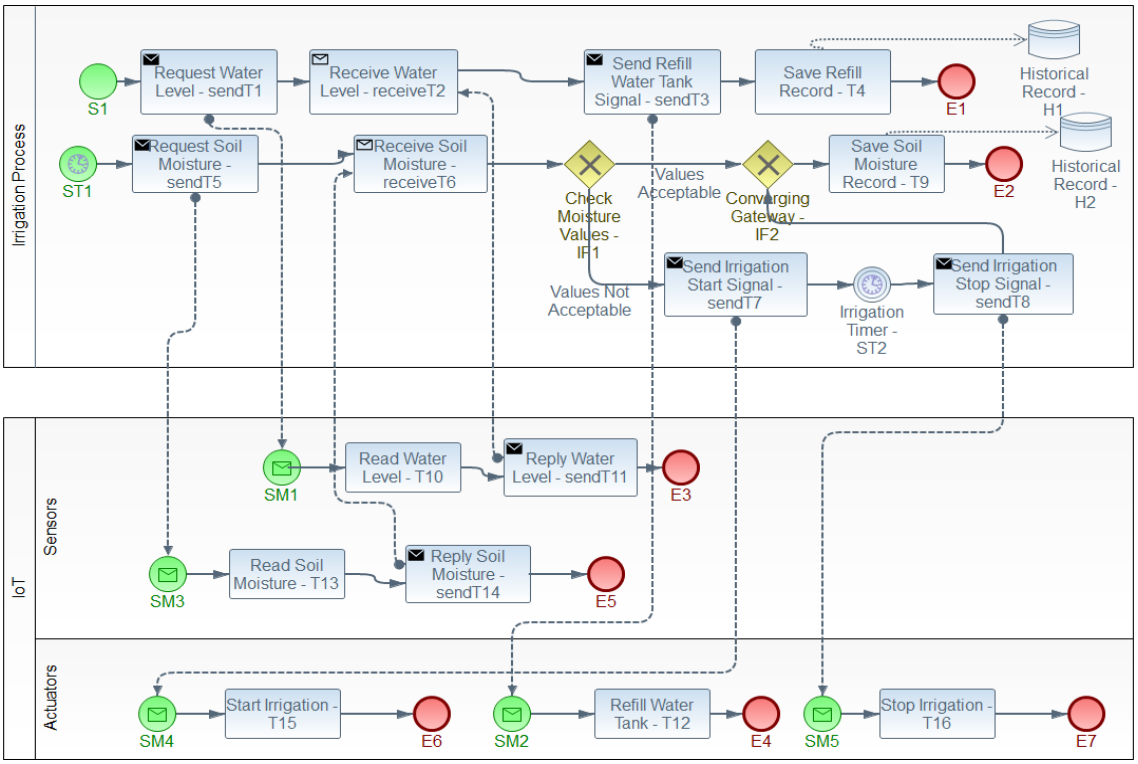

Figura 3.1 - Modelo BPMN de um sistema de rega automática

A Figura 3.1 ilustra o modelo BPMN simplificado deste caso de uso. É composto por dois participantes: o processo de irrigação em cima (*Irrigation Process*) e a rede de dispositivos IoT. O processo de irrigação representa o servidor central responsável pela execução dos processos de negócio. Contém dois fluxos de execução: o de cima descreve o reabastecimento de água num tanque e o outro define o processo de rega. A rede de dispositivos IoT é composta por um conjunto de sensores e atuadores. Os sensores convertem fenómenos físicos em impulsos elétricos que são utilizados para interpretar a leitura de um valor. Os atuadores funcionam de forma inversa aos sensores, ou seja, recebem um impulso elétrico e tornam-no numa ação física. Neste sistema, os sensores são responsáveis pela recolha de informação sobre o nível de água nos tanques e a humidade do solo. Esta informação é encaminhada ao servidor central que toma uma decisão e envia um comando aos atuadores para executarem a ação desejada.

O primeiro fluxo de execução do processo de irrigação começa com o evento de início *S1*. Este desencadeia a tarefa de envio de mensagem *Request Water Level* que contacta os sensores para obter informação sobre o nível de água. Para isso, envia uma mensagem para os sensores (*SM1*) que dá início à execução do processo localmente. Os sensores leem o nível de água que está no tanque (*Read Water Level*) e encaminham esta informação de volta para o processo de irrigação (*Reply Water Level*), terminando em seguida o processo (sinalizado pelo evento de fim *E3*). A tarefa de receção de mensagens *Receive Water Level* aguarda a chegada da mensagem dos sensores. Quando esta chega, desencadeia a tarefa *Send Refil Water Tank Signal* que envia uma mensagem (*SM2*) com a informação recebida para os atuadores. Esta informação é

transmitida à tarefa *Refill Water Tank* para encher o tanque com a quantidade de água em falta, terminando em seguida o processo (com o evento de fim *E4*). Por último, a tarefa *Save Refill Record* regista esta ocorrência escrevendo na *data store Historical Record*, terminando com o evento de fim *E1*.

O segundo fluxo executa periodicamente o evento de início *ST1*. Este fluxo pretende detetar se os níveis de humidade do solo fornecidos pelos sensores são aceitáveis. Para isso, *Request Soil Moisture* contacta os sensores a solicitar esta informação. O evento de início *SM3* começa o processo e a tarefa *Read Soil Moisture* lê dos sensores a informação sobre a humidade do solo. Em seguida, *Reply Soil Moisture* envia uma mensagem com esta informação para o processo de rega e o processo termina. Depois de a mensagem ter sido entregue à tarefa de receção de mensagens *Receive Soil Moisture*, é feita uma análise à informação recebida de forma a verificar se é necessário iniciar uma rega. Para isso, é utilizado o *gateway* exclusivo *Check Moisture Values* que contém a condição sobre os valores considerados aceitáveis e que obriga ao processo a seguir apenas um dos caminhos. Caso os valores não sejam aceitáveis, *Send Irrigation Start Signal* sinaliza os atuadores para darem início à irrigação, através da tarefa *Start Irrigation*. O processo termina com a execução desta tarefa. É assim necessário o envio de outro sinal para que a irrigação em curso termine. O objetivo do evento intermédio de tempo *Irrigation Timer* é esperar uma duração pré-estabelecida de tempo antes de a tarefa de envio de mensagem *Send Irrigation StopSignal* sinalizar os atuadores a terminar a irrigação em curso com a tarefa *Stop Irrigation*. Por fim, é feito um registo pela tarefa *Save Soil Moisture Record* na *data store Historical Record* para assinalar se na sequência deste fluxo ocorreu irrigação. O *gateway* exclusivo *Converging Gateway* tem o propósito de remeter o processo para esta tarefa, independentemente do caminho seguido pelo processo.

Este processo segue uma abordagem centralizada e será utilizado nas secções seguintes para exemplificar os vários passos do procedimento de decomposição.

3.2 Padrões

Esta secção procura descrever e ilustrar com exemplos os padrões que identificámos como casos de utilização ineficaz da capacidade computacional dos dispositivos da rede IoT. A principal preocupação na identificação dos padrões, e das suas respetivas modificações, foi a de não quebrar o fluxo de execução do processo de negócio original. Quer isto dizer que as tarefas executam pela mesma ordem antes e após as modificações. As figuras que se seguem não contêm *lanes* em nenhum dos participantes por motivos de simplificação de entendimento do cenário em causa, visto que a presença de *lanes* não alteraria o resultado em causa. A Figura 3.2 exemplifica o primeiro padrão.

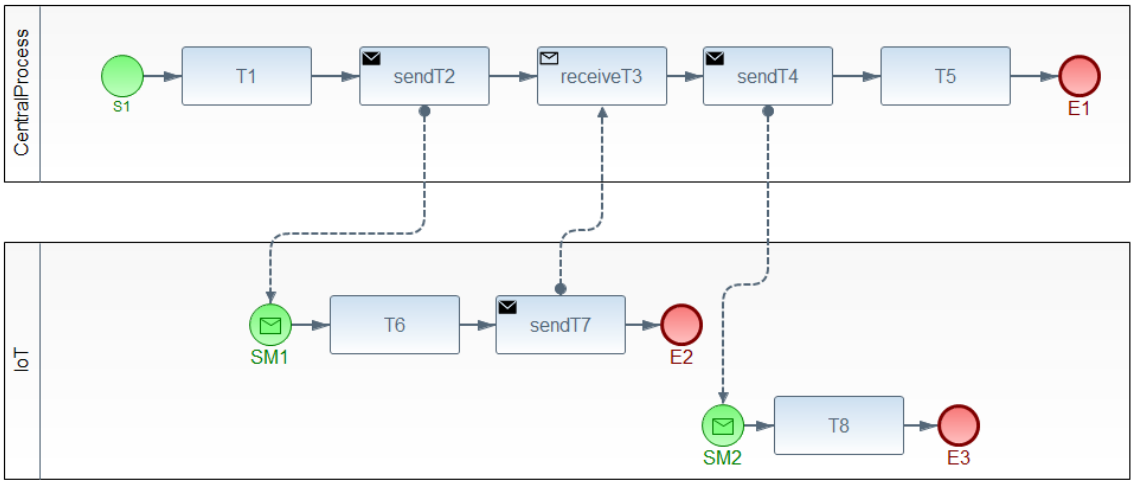

Figura 3.2 - Exemplo onde a aplicação do primeiro padrão é possível

O primeiro padrão é identificado por uma tarefa de receção seguida de uma tarefa de envio de mensagens. Tanto a tarefa de envio que antecede a tarefa de receção como o destinatário da tarefa de envio têm de pertencer ao mesmo participante. Neste exemplo, o padrão é detetado por *receiveT3* e *sendT4*, mas só é possível aplicar modificações a este caso porque *sendT7* e *SM2* pertencem ao mesmo participante (IoT). Este padrão retrata a existência de uma comunicação desnecessária, visto que a rede IoT poderia dar início a *T8* logo após *sendT7* terminar, poupando assim uma comunicação. A decomposição passa por eliminar a tarefa de envio (*sendT4*), o seu destinatário (*SM2*) e a comunicação que os une. A Figura 3.3 ilustra a solução para este exemplo.

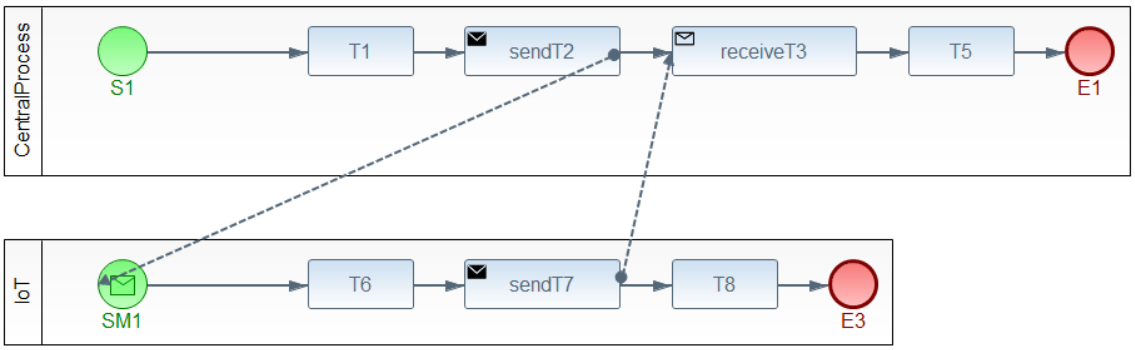

Figura 3.3 - Solução para o exemplo utilizado na demonstração do primeiro padrão

A Figura 3.4 contém um exemplo mais complexo da aplicação do primeiro padrão, desta vez, tendo em conta fluxo de dados.

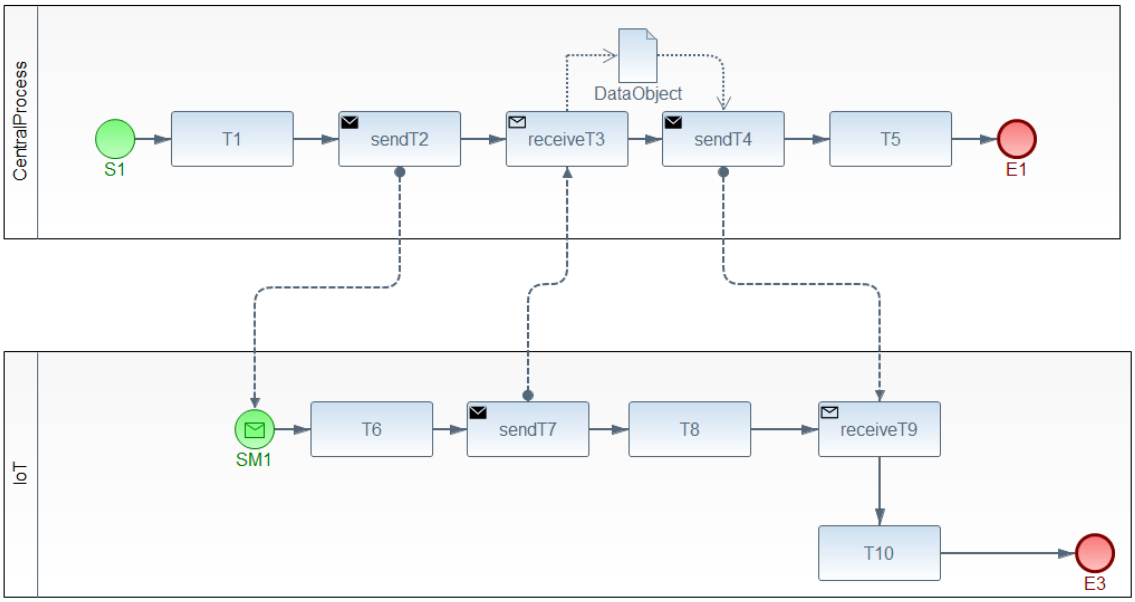

Figura 3.4 - Exemplo da aplicação do primeiro padrão tendo em conta fluxo de dados

O padrão é detetado por *receiveT3* e *sendT4*, mas é possível observar que existe uma tarefa (*T8*) na rede de dispositivos IoT que tem de ser executada antes da receção da mensagem que provém de *sendT4*. De notar que temos uma tarefa de receção (*receiveT9*) ao invés de um evento de início condicionado por uma mensagem, como acontecia no exemplo anterior. *receiveT3* realiza uma operação de escrita sobre um objeto de dados (*DataObject*) e *sendT4* lê deste mesmo objeto, antes de enviar uma mensagem para a rede de dispositivos IoT. Como ambas as tarefas operam sobre o mesmo objeto de dados, é possível aplicarmos a mesma transformação ao processo de negócio e, inclusive, transferir o próprio objeto de dados. A solução para este caso está representada na Figura 3.5.

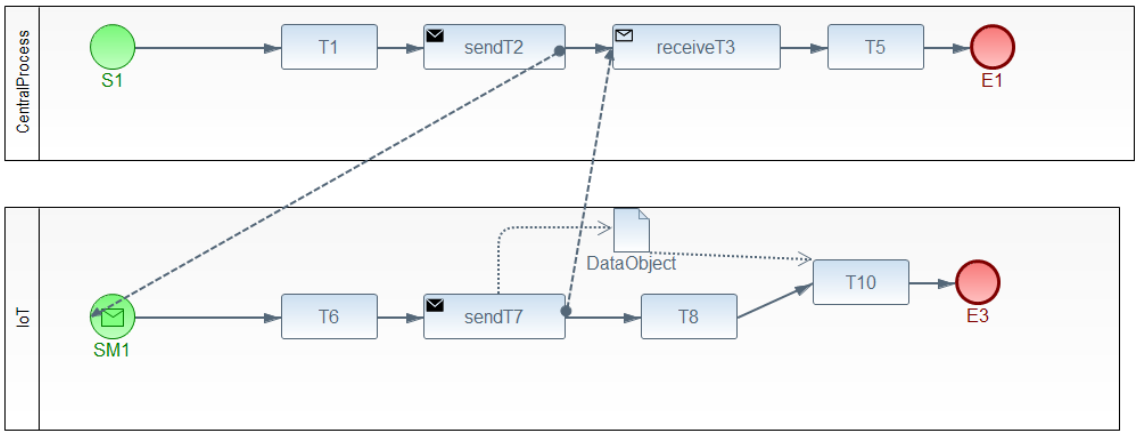

Figura 3.5 - Solução para o exemplo de fluxo de dados onde o primeiro padrão é visível

As tarefas *sendT4* e *receiveT9* foram eliminados. Desta forma, a execução de *T10* está dependente do fim da execução de *T8* e não da chegada da mensagem proveniente do processo central (que em caso de latências elevadas poderia adiar a sua execução por tempo indeterminado). O objeto de dados também pôde ser transferido para a rede de dispositivos IoT, pois o seu conteúdo proveio do resultado da escrita de *receiveT3* sobre a informação que recebeu de *sendT7*, que faz parte desta rede. Assim, é seguro associar este objeto de dados a *T10*, pois este dependia do seu conteúdo para iniciar a sua execução. Por sua vez, a tarefa *sendT7* fica responsável por atualizar os valores do objeto de dados.

O segundo padrão é identificado por um evento de início condicionado por um instante e os dois próximos elementos BPMN deste participante contêm, respetivamente, uma tarefa de envio e uma tarefa de receção de mensagens, como exemplificado na Figura 3.6.

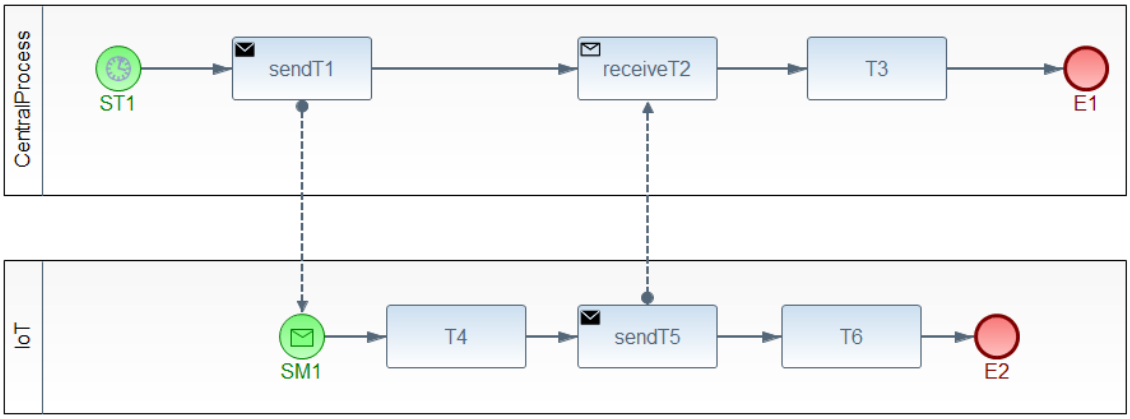

Figura 3.6 - Exemplo onde a aplicação do segundo padrão é possível

Neste exemplo, o segundo padrão é identificado por *ST1*, *sendT1* e *receiveT2*. Neste caso, a tarefa de envio de mensagem, a comunicação e o seu destinatário são eliminados e o evento de início condicionado por um instante é transferido e ligado ao elemento de fluxo que está a seguir ao destinatário. A tarefa de receção é convertida num evento de início condicionado por uma mensagem, de modo a que o processo possa começar.

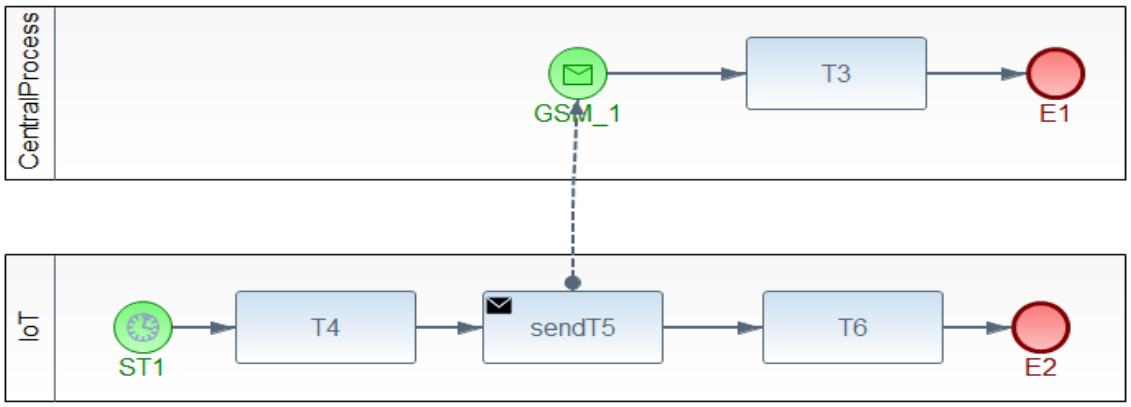

Figura 3.7 - Solução para o exemplo utilizado na demonstração do segundo padrão

A Figura 3.7 ilustra o resultado da aplicação do segundo padrão a este exemplo. *sendT1* e *SM1* são eliminados. *ST1* é transferido para a rede de dispositivos IoT e *receiveT2* é substituído por *GSM_1*. A Figura 3.8 ilustra um caso onde não faz sentido aplicar transformações.

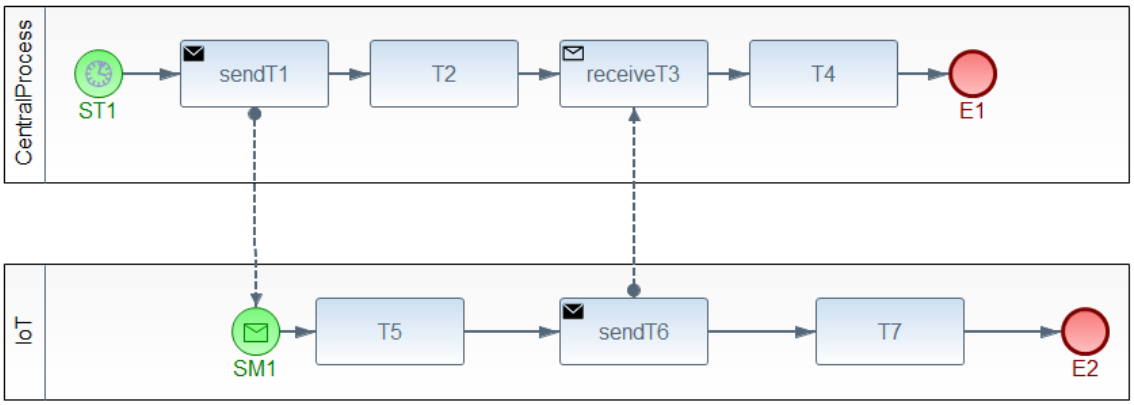

Figura 3.8 - Exemplo onde a aplicação do segundo padrão não é vantajosa

Caso existam tarefas entre a tarefa de envio e a tarefa de receção do processo central, não existe qualquer benefício em transferir o evento de início condicionado por um instante (*ST1* neste caso) para a rede de dispositivos IoT. Se aplicássemos a mesma transformação que no caso anterior, *sendT1* e *SM1* seriam eliminados para que *ST1* pudesse ser transferido para a rede IoT e ligado a *T5*. O problema é que, para manter o fluxo de execução que existia originalmente, seria necessário criar uma tarefa de envio na rede IoT, para que um evento de início condicionado por uma mensagem fosse utilizado no processo central para começar a tarefa *T2*. Estaríamos a alterar o processo de negócio original sem ganho nenhum pois, efetivamente, o número de comunicações manter-se-ia igual.

Por fim, o terceiro padrão é detetado por uma tarefa de envio pertencente à rede IoT e, o elemento BPMN que sucede o destinatário da sua mensagem, é um *gateway*. Tipicamente, os dispositivos IoT têm capacidade computacional suficiente para realizar operações lógicas, por isso faz sentido transferir os *gateways* para esta rede. De modo a manter o fluxo de execução original, as transformações a aplicar quando o *gateway* é transferido dependem dos elementos que o sucedem.

O primeiro passo é colocar o *gateway* na rede de dispositivos IoT. O elemento BPMN que antecedia o *gateway* é substituído por um evento de fim. A tarefa de envio é substituída por um *gateway* paralelo que é ligado ao elemento BPMN que sucede a tarefa de envio e ao *gateway* que foi transferido. Depois, são analisados os ramos do *gateway* transferido e daí poderão ocorrer dois casos:

- O primeiro elemento do ramo é uma tarefa. Neste caso, é criada uma tarefa de envio de mensagens na rede de dispositivos IoT que se ligará a um evento de início condicionado por uma mensagem que, por sua vez, se ligará à tarefa.
- O primeiro elemento do ramo é uma tarefa de envio de mensagens. Neste caso, a tarefa de envio é removida, bem como o seu destinatário, e o *gateway* é ligado ao elemento BPMN que sucede este destinatário. Depois, são verificados os seus elementos sucessores até se encontrar uma tarefa ou um evento de fim. Caso se encontrem mais tarefas de envio, é aplicado novamente o mesmo procedimento. Caso se encontrem tarefas de receção de mensagens, estas serão eliminadas e se existirem eventos intermédios, estes serão transferidos para a rede IoT. Se uma tarefa for encontrada, é aplicado o mesmo procedimento que foi descrito para o primeiro elemento do ramo. Se o antecessor da tarefa encontrada for uma tarefa de envio ou receção, então o evento de início condicionado por uma mensagem é ligado diretamente à tarefa, caso contrário, é ligado ao elemento de fluxo antecessor da tarefa encontrada (tipicamente, um *gateway*).

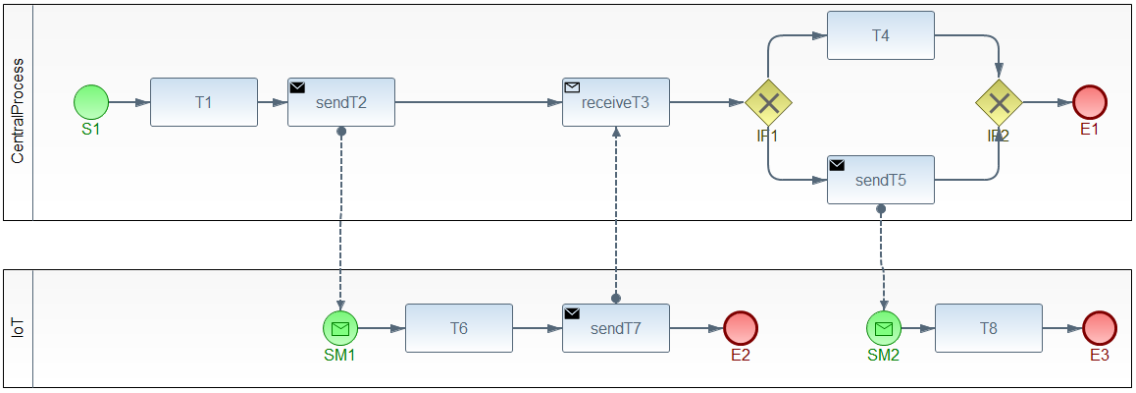

Figura 3.9 - Exemplo onde a aplicação do terceiro padrão é possível

A Figura 3.9 retrata um exemplo onde é possível aplicar o terceiro padrão. *sendT7* envia uma mensagem a *receiveT3* que antecede *IF1*, daí que o padrão é detetado. Também é possível observar que uma tarefa (*T4*) e uma tarefa de envio de mensagens (*sendT5*) são elementos sucessores de *IF1*, por isso, a transformação a aplicar a cada um destes sucessores será diferente. A solução para este exemplo está na Figura 3.10.

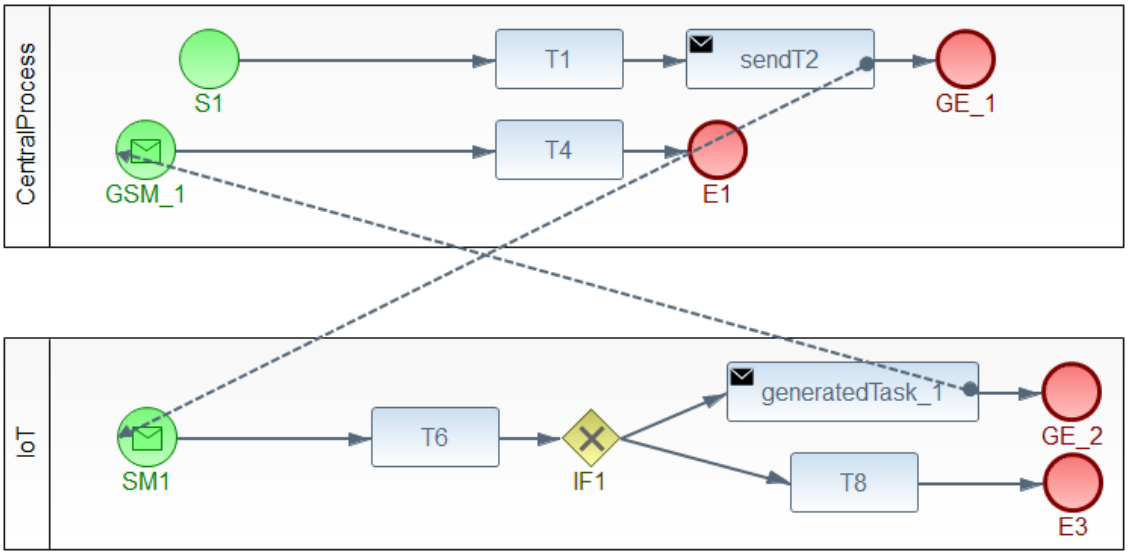

Figura 3.10 - Solução para o exemplo utilizado na demonstração do terceiro padrão

A solução obtida demonstra que foi possível reduzir uma comunicação. Primeiro, *IF1* é transferido para a rede de dispositivos IoT e *sendT7* é eliminado. É criado um *gateway* paralelo na rede IoT e ligado a *E2* e *IF1*, mas é posteriormente removido porque não é necessário para este caso. Isto acontece porque o objetivo do *gateway* paralelo é o de manter o fluxo de execução do processo original e, para os casos em que este é diretamente ligado a um evento de fim, a sua existência é desprezável. Em vez disso, *IF1* é ligado a *T6* e *receiveT3* é substituído por um evento de fim (*GE_1*). Em seguida, são visitados os sucessores de *IF1* e é aplicada uma transformação que mantém o fluxo de execução do processo original. Para o caso de *T4*, é necessário haver um evento de início para que este possa começar. Para isso, é criado um evento de início condicionado por uma mensagem (GSM_1) e ligado a *T4* no processo central, enquanto que na rede IoT é criada a correspondente tarefa de envio (*generatedTask_1*) e ligada a *IF1*. No caso de *sendT5*, como este comunica com a rede IoT e *IF1* já faz parte desta rede, é removido assim como *SM2* e *IF1* é ligado diretamente a T8. No processo central, *IF2* é removido por já não se justificar a sua utilização, contudo estes detalhes serão melhor explicados na secção do algoritmo.

É também possível realizar controlo de dados sobre este padrão, desde que o objeto de dados ligado ao *gateway* que se pretende transferir não esteja a ser também utilizado por uma tarefa. A Figura 3.11 mostra um cenário mais complexo do terceiro padrão, em que existe dependência de dados por parte de um *gateway* e existem várias comunicações em um dos ramos provenientes desse *gateway*.

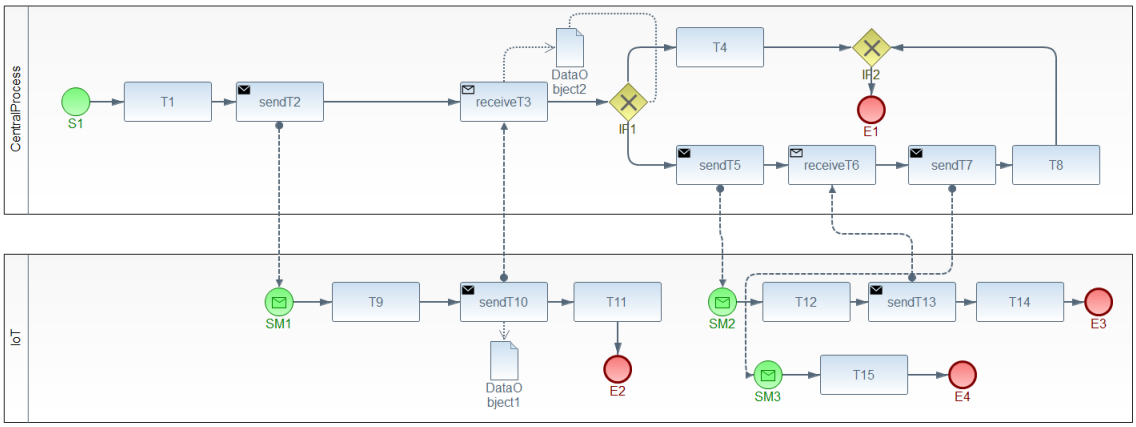

Figura 3.11 - Exemplo da aplicação do terceiro padrão tendo em conta fluxo de dados

O padrão é detetado por *sendT10*, *receiveT3* e *IF1*. É seguro transferir o *gateway* IF1 para a rede de dispositivos IoT, porque *DataObject2* é escrito por uma tarefa de envio, dependente da informação de *DataObject1* que faz parte da rede IoT. Caso *T1* realizasse operações de escrita sobre *DataObject2* e *IF1* dependesse deste mesmo objeto de dados, então não seria possível realizar qualquer modificação ao modelo BPMN, pois perder-se-ia uma destas dependências com a transferência de *IF1* para a rede IoT, quebrando assim o fluxo de execução do processo original. A Figura 3.12 apresenta a solução para este exemplo.

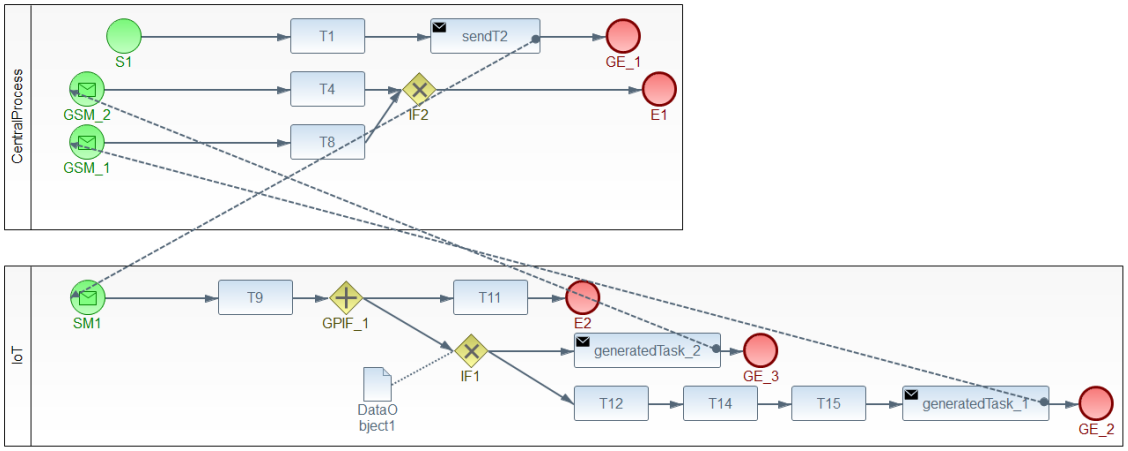

Figura 3.12 - Solução para o exemplo de fluxo de dados onde o terceiro padrão é visível

A tarefa *sendT10* é eliminada e, *receiveT3* é transformado num evento de fim (*GE_1*). *IF1* é transferido para a rede de dispositivos IoT e ligado a *DataObject1*, por conter a informação que deu origem a *DataObject2*. Por esta razão, *DataObject2* é eliminado, bem como todas as suas ligações. É criado um *gateway* paralelo (GPIF_1) que é ligado ao elemento BPMN sucessor da tarefa de envio apagada (*T11*) e a *IF1*. Como neste caso o *gateway* paralelo está ligado a uma tarefa, este não é removido para que o fluxo de execução original se mantenha. Para o ramo inferior de *IF1*, *sendT5*, *SM2*, *sendT13*, *receiveT6*, *sendT7*, *SM3* são eliminados. As tarefas que fazem parte do caminho onde ocorriam estas comunicações são devidamente ligadas

entre si, respeitando o fluxo de execução original. Deste modo, além de cortamos bastantes comunicações desnecessárias, o modelo torna-se mais compreensível. Assim que é detetada a tarefa *T8*, cria-se uma tarefa de envio (*generatedTask*_1), um evento de início condicionado por uma mensagem (*GSM*_1) e ligam-se ambos. *GSM*_1 é também ligado a *T8* e cria-se um evento de fim (GE_2) para terminar o fluxo de execução deste ramo. Como o ramo superior de *IF1* começa logo por uma tarefa, é realizado o mesmo procedimento de criação de tarefa de envio (*generatedTask_2*) na rede IoT e correspondente evento de início condicionado por uma mensagem *(GSM_2*) no processo central e dão-se por concluídas as modificações a efetuar neste exemplo.

3.3 Procedimento de decomposição

A solução desenvolvida decompõe os processos de negócio transferindo parte do processo para a rede de dispositivos IoT com o objetivo de aproveitar o seu poder computacional e reduzir o número de comunicações. Para isso, realizamos os seguintes passos:

1. Gerar um grafo, a partir de um modelo de processos de negócio BPMN, que capture as dependências de fluxo de controlo das tarefas – Gerar grafo.
2. Gerar uma lista de subcaminhos que podem ser movidos para a rede de sensores sem fios – Caminhos candidatos.
3. Redefinir os participantes com base nos caminhos obtidos – Redefinir participantes.
4. Criar o novo modelo BPMN – Desenhar colaboração.

3.4 Algoritmos

Nome: Gerar grafo

Descrição: Construímos um grafo com os elementos BPMN de cada processo do modelo. O grafo diferencia fluxos de sequência de fluxos de mensagem em dois tipos de arestas para que haja distinção entre arestas que representam a ocorrência de uma comunicação das demais.

Objetivo: extrair a informação referente a elementos BPMN e fluxos de controlo de um dado modelo de processos de negócio BPMN e colocá-la num grafo direcionado.

Entrada:

- Modelo de processos de negócio BPMN.

Saída:

- Lista de processos;
- Lista de associações;
- Colaboração;

- Lista de *data stores*;
- Grafo direcionado.

Passos:

1. Iterar sobre os elementos de raiz do modelo e inicializar a lista de processos, a colaboração e a lista de *data stores* com os elementos encontrados.

   1.1. Por cada processo, criar um par constituído pelo nome do processo e uma associação e adicionar à lista de associações até todas as associações do processo terem sido adicionadas.

2. Iterar sobre a lista de processos e adicionar ao grafo, sob a forma de nó ou aresta, os elementos BPMN correspondentes. As arestas são criadas a partir de fluxos de sequência ou fluxos de mensagem e os restantes elementos BPMN são utilizados na criação dos nós. Caso estes sejam eventos de início ou eventos de fim, serão também adicionados à lista de nós iniciais ou finais do grafo, respetivamente.

3. Por cada fluxo de mensagem da colaboração, adicionar uma aresta ao grafo com esta dependência.

Aplicação ao caso de uso (Figura 3.1):

- Lista de processos: [*Irrigation Process*, IoT].
- Lista de associações: []
- Colaboração:
  - Lista de participantes: [*Irrigation Process*, IoT].
  - Lista de fluxos de mensagem: [sendT1 → SM1, sendT11 → receiveT2, sendT3 → SM2, sendT5 → SM3, sendT14 → receiveT6, sendT7 → SM4, sendT8 → SM5].
- Lista de *data stores*: [H1, H2].

- Grafo direcionado:

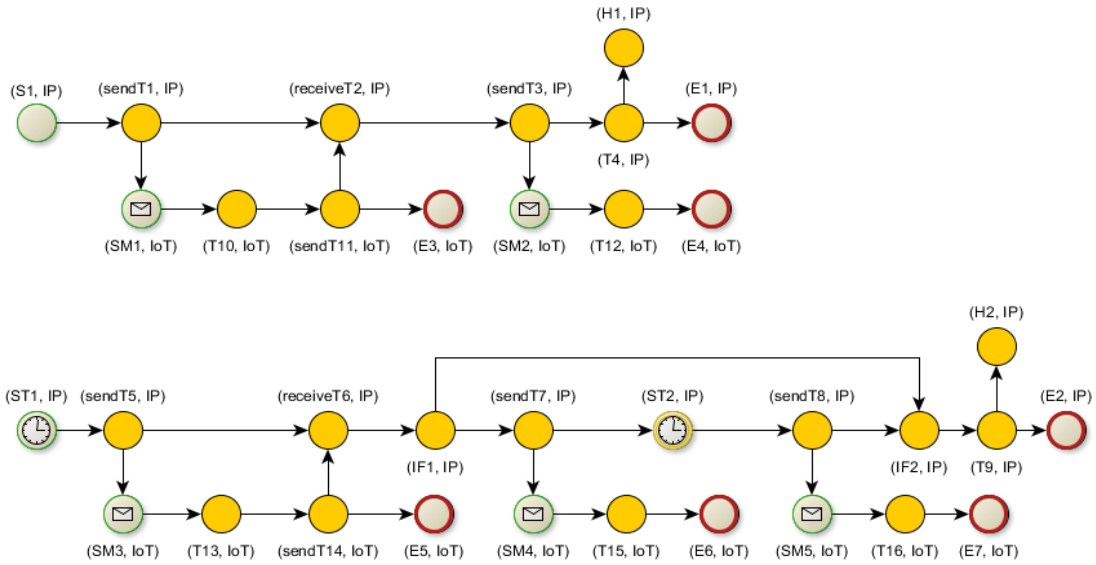

Figura 3.13 - Grafo resultante do modelo BPMN de rega automática

Nome: Caminhos candidatos

Descrição: são obtidos caminhos compostos exclusivamente por elementos BPMN que podem ser executados na rede de dispositivos IoT. O resultado terá tantos caminhos quantos fluxos de execução que contenham comunicações entre o servidor central e a rede IoT existirem.

Objetivo: descobrir os caminhos que comunicam com a rede de dispositivos IoT que poderão ser alvo de transformações.

Entrada:

- Grafo direcionado.

Saída:

- Lista de caminhos candidatos a sofrerem alterações.

Passos:

1. Para cada nó inicial do grafo, fazer uma travessia no grafo enumerando todos os caminhos a partir deste nó. A condição de paragem da travessia é encontrar um nó que pertença à lista de nós finais do grafo. Contudo, só serão aceites caminhos que contenham arestas de comunicação.
2. Filtram-se os nós que contenham elementos BPMN que fazem parte da rede IoT ou que contenham elementos BPMN (eventos condicionados por um instante, eventos condicionados por uma mensagem, tarefas de envio e receção de mensagens e *gateways*) que podem ser executados nos dispositivos IoT.
3. Remover caminhos cujos nós estejam contidos em caminhos de maior dimensão.

Aplicação ao caso de uso:

Passos intermédios (1 e 2 do algoritmo):

Caminhos do grafo que comunicam com a rede de dispositivos IoT:

[S1, sendT1, SM1, T10, sendT11, receiveT2, sendT3, SM2, T12, E4];

[S1, sendT1, SM1, T10, sendT11, receiveT2, sendT3, T4, E1];

[S1, sendT1, SM1, T10, sendT11, E3];

[S1, sendT1, receiveT2, sendT3, SM2, T12, E4];

[ST1, sendT5, SM3, T13, sendT14, receiveT6, IF1, IF2, T9, E2];

[ST1, sendT5, SM3, T13, sendT14, receiveT6, IF1, sendT7, SM4, T15, E6];

[ST1, sendT5, SM3, T13, sendT14, receiveT6, IF1, sendT7, ST2, sendT8, SM5, T16, E7];

[ST1, sendT5, SM3, T13, sendT14, receiveT6, IF1, sendT7, ST2, sendT8, IF2, T9, E2];

[ST1, sendT5, SM3, T13, sendT14, E5];

[ST1, sendT5, receiveT6, IF1, sendT7, SM4, T15, E6];

[ST1, sendT5, receiveT6, IF1, sendT7, ST2, sendT8, SM5, T16, E7].

Explicação: Este passo contém todos os caminhos que satisfaçam as seguintes propriedades:

- O início do caminho é o primeiro nó de cada fluxo de execução (*S1* e *ST1* neste caso);
- Existe pelo menos uma aresta de comunicação com a rede de dispositivos IoT e, tanto o elemento de origem como o elemento de destino de todas as arestas de comunicação presentes no caminho, estão também presentes;
- O caminho termina com um evento de fim.

A Figura 3.14 ilustra o caso de um caminho do grafo que não respeita as duas primeiras propriedades.

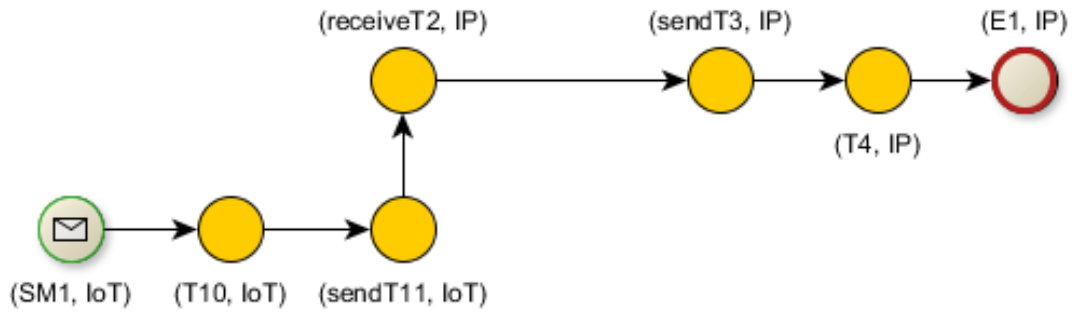

Figura 3.14 - Caminho do grafo que não respeita propriedades do algoritmo que encontra caminhos candidatos

O algoritmo começa a sua travessia a partir da lista de nós iniciais do grafo. O evento de início *SM1* é um desses nós. Caso a segunda propriedade não existisse, este caminho seria aceite, pois existe uma aresta de comunicação onde ambos os elementos fazem parte do caminho (*sendT1* e *receiveT2*). Contudo, este caminho é irrelevante na obtenção de uma solução, pois existirá um caminho de maior dimensão com início em *S1* que, por conter mais elementos (*S1* e *sendT1*), será um melhor candidato. Isto porque pretendemos comparar o maior número possível de elementos que façam sentido, com o conjunto de padrões que identificámos e, embora este exemplo contenha elementos relevantes, omite dois outros elementos que poderiam enquadrar-se com um dos casos. *SM1* é o elemento de destino de uma aresta de comunicação e, por *sendT1* (o elemento de origem) não fazer parte do caminho, não existe qualquer benefício em este ser aceite. Por consequência da segunda propriedade, a primeira surgiu, porque não haverá a possibilidade de existirem caminhos com começo em eventos de início que façam parte de uma comunicação. Por outro lado, também não faz sentido considerar o caminho apresentado na Figura 3.15, pois não existem modificações a efetuar em elementos que não comunicam com a rede IoT.

Figura 3.15 - Caminho do grafo que não contém comunicações com a rede IoT

Caminhos do grafo com nós que contêm elementos BPMN que podem ser executados na rede de dispositivos IoT:

[sendT1, SM1, T10, sendT11, receiveT2, sendT3, SM2, T12, E4];

[sendT1, SM1, T10, sendT11, receiveT2, sendT3];

[sendT1, SM1, T10, sendT11, E3];

[sendT1, receiveT2, sendT3, SM2, T12, E4];

[ST1, sendT5, SM3, T13, sendT14, receiveT6, IF1, IF2];

[ST1, sendT5, SM3, T13, sendT14, receiveT6, IF1, sendT7, SM4, T15, E6];

[ST1, sendT5, SM3, T13, sendT14, receiveT6, IF1, sendT7, ST2, sendT8, SM5, T16, E7];

[ST1, sendT5, SM3, T13, sendT14, receiveT6, IF1, sendT7, ST2, sendT8, IF2];

[ST1, sendT5, SM3, T13, sendT14, E5];

[ST1, sendT5, receiveT6, IF1, sendT7, SM4, T15, E6];

[ST1, sendT5, receiveT6, IF1, sendT7, ST2, sendT8, SM5, T16, E7].

Explicação: são removidos dos caminhos do grafo todos os nós que contenham elementos BPMN que não podem ser transferidos para a rede IoT (tarefas). Também são removidos nós que contenham eventos de início não condicionados (como o *S1*) que, apesar de puderem ser transferidos, não se enquadram em nenhum dos padrões. Permanecem os elementos que já fazem parte da rede ou que poderão ser executados pelos dispositivos IoT. Como cada nó é um par composto por uma referência para um elemento BPMN e o nome do participante a que este pertence, os elementos que fazem parte da rede IoT são facilmente identificáveis. Os dispositivos IoT conseguem executar eventos condicionados por um instante (alarmes), eventos condicionados por uma mensagem, tarefas de envio e receção de mensagens e *gateways*. Também conseguem utilizar objetos de dados, mas estes só serão verificados caso se confirme a aplicação de um dos padrões e, por isso, não são relevantes neste passo.

Resultado (passo 3 do algoritmo):

Lista de caminhos candidatos:

[[sendT1, SM1, T10, sendT11, receiveT2, sendT3, SM2, T12, E4],
[ST1, sendT5, SM3, T13, sendT14, receiveT6, IF1, sendT7, ST2, sendT8, SM5, T16, E7]].

Explicação: neste passo são removidos caminhos cujos nós estejam contidos em caminhos de maior dimensão. Isto inclui subcaminhos (até ao último nó) ou caminhos que não realizam desvios. A Figura 3.16 ilustra os caminhos candidatos obtidos para este caso de uso.

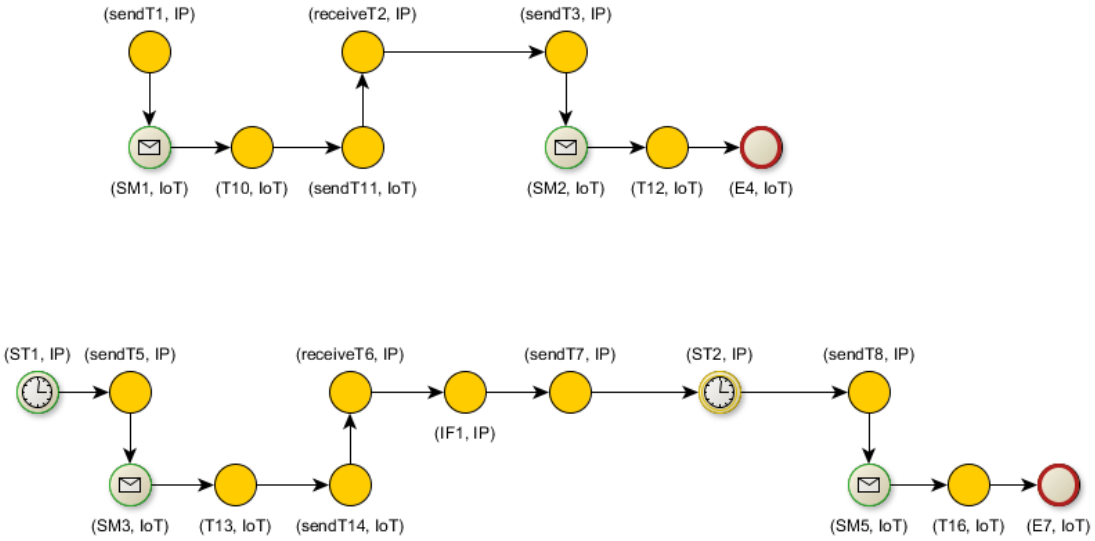

Figura 3.16 - Caminhos candidatos resultantes do caso de uso

Se tivermos em conta o caminho [sendT1, SM1, T10, sendT11, E3], é fácil verificarmos que a sua remoção da lista de caminhos candidatos deveu-se a este estar contido (à exceção de *E3*) em outro caminho de maiores dimensões. Por outro lado, o caminho [ST1, sendT5, receiveT6, IF1, sendT7, ST2, sendT8, SM5, T16, E7] é removido por não conter os nós que estão entre *sendT5* e *receiveT6*. Como existe um caminho que realiza este desvio e possui os mesmos nós que o caminho anterior, este é descartado. Os restantes caminhos removidos encontram-se entre estes dois casos.

Nome: Redefinir participantes

Descrição: os caminhos candidatos são comparados aos padrões de modificação. Caso o caminho coincida com um dos padrões, é aplicada a transformação correspondente, caso contrário, nada é alterado e o modelo BPMN de entrada será o mesmo do de saída. Mesmo que sejam detetados vários padrões ao longo dos caminhos candidatos, só será efetuada a modificação do primeiro padrão encontrado, de modo a facilitar a validação do modelo resultante.

Objetivo: redefinir os participantes com base nos caminhos candidatos obtidos. É realizada uma modificação à lista de processos se os caminhos candidatos coincidirem com um dos padrões de modificação.

Entrada:

- Lista de processos;
- Colaboração;
- Lista de caminhos candidatos;
- Lista de associações.

Saída:

- Nova lista de processos;
- Nova colaboração;
- Lista de associações atualizada.

Passos:

1. Inicializar uma nova lista de processos com todas as informações (nome, id, artefactos, tipo de processo, se é executável, entre outros) da lista de processos original à exceção dos elementos de fluxo que contém.
2. Inicializar uma nova colaboração com todas as informações (nome, id, participantes, entre outros) da colaboração original exceto a lista de fluxos de mensagens que contém.
3. Iterar sobre lista de processos e adicionar à nova lista de processos todos os elementos de fluxo que não façam parte de nenhum dos nós dos caminhos candidatos. Só serão adicionados fluxos de sequência cujos elementos de origem e destino tenham sido adicionados à nova lista de processos.
4. Iterar sobre cada caminho candidato em busca de um dos possíveis padrões. Caso um padrão seja detetado, é realizada a transformação correspondente (descritas na secção 3.2) e deixam-se de verificar mais ocorrências. Após a transformação aplicada (ou na ausência desta), são iterados os nós do caminho candidato e são tomadas as seguintes decisões:
    - Caso o nó atual contenha uma tarefa de envio de mensagens, verificam-se os dois nós seguintes. Se o nó seguinte fizer parte de outro participante, cria-se um fluxo de mensagem entre a tarefa de envio e o elemento de fluxo do nó seguinte. Caso contrário, cria-se um fluxo de controlo e unem-se os dois.
    - Caso o nó atual contenha uma tarefa de receção de mensagens, verificam-se os dois nós anteriores. Cria-se um fluxo de controlo para unir o elemento de fluxo do nó anterior à tarefa de receção caso o nó anterior pertença ao mesmo participante.
    - Caso o nó atual contenha um *gateway*, criam-se fluxos de controlo para unir o *gateway* a cada um dos elementos de fluxo sucessores.
    - Para todos os casos, o elemento de fluxo do nó atual é adicionado à nova lista de processos e, caso não tenha sido antes, é adicionado o fluxo de controlo que liga o elemento de fluxo do nó anterior ao elemento de fluxo atual.
5. Eliminar todos os elementos BPMN que não possuem dependências e fluxos de controlo ou fluxos de mensagens que apontem para elementos que não foram adicionados à nova lista de processos. Estes cenários podem ocorrer devido ao passo 1.

Caso existam *gateways* diretamente ligados a um evento de fim e o *gateway* só possua um antecessor, este é ligado ao evento de fim e o *gateway* é removido. Caso se trate de um *gateway* paralelo, então tanto o *gateway* como o evento de fim são removidos e o antecessor do *gateway* é ligado ao sucessor.

6. Caso existam elementos BPMN que não possuam sucessor, são criados eventos de fim como seus sucessores.
7. Caso algum dos participantes possua *lanes*, itera-se a nova lista de processos e verifica-se se os seus elementos BPMN já estavam referenciados nas *lanes* da lista de processos original. É criada uma referência na *lane* respetiva da nova lista de processos, para os elementos de fluxo que estavam referenciados na lista de processos original.
   7.1. Para o caso em que o elemento BPMN não pertença à lista de processos original, a referência é criada de duas formas: se o elemento BPMN for um evento de início condicionado por um instante, então este será atribuído à mesma *lane* que o seu sucessor, caso contrário, percorrem-se os antecessores do elemento BPMN atual até se encontrar um cuja *lane* é conhecida.
8. Atualizar as referências de ligações de entrada e de saída de todos os elementos BPMN da nova lista de processos.

Aplicação ao caso de uso:

Padrão 1:

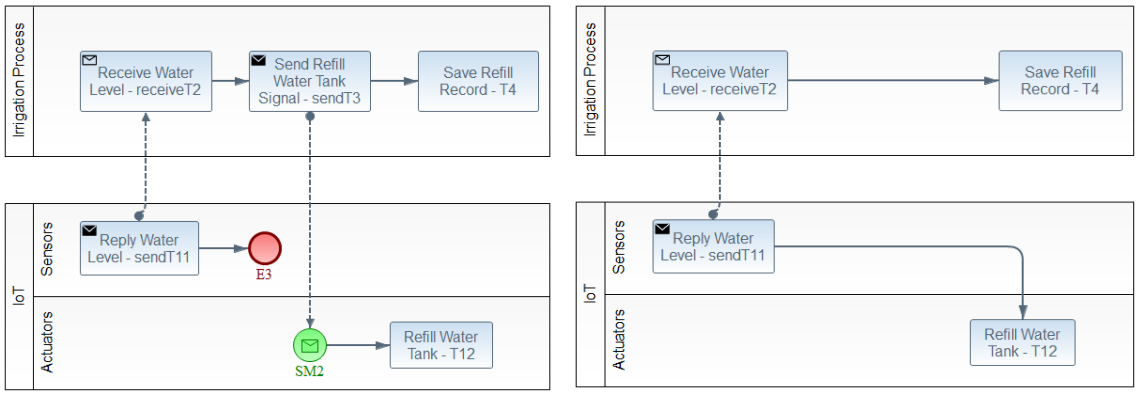

Figura 3.17 - Excerto do caso de uso que ilustra o primeiro padrão e respetiva transformação

A Figura 3.17 mostra um excerto do caso de uso com os elementos BPMN relevantes na aplicação do primeiro padrão. No caso de uso, *receiveT2* antecede *sendT3* e, como *sendT11* e *SM2* fazem parte da rede de dispositivos IoT é possível aplicarmos este padrão. *SendT3* e *SM2* são removidos e, de modo a manter o fluxo de controlo, *receiveT2* é ligado a *T4* e *sendT11* é ligado a *T12*. *E3* é removido, visto que foi adicionado à nova lista de processos no passo 3 por não haver um nó que o referencie na lista de caminhos candidatos e, por não possuir ligações que o referenciem na nova lista de processos, é removido no passo 5.

Padrão 2:

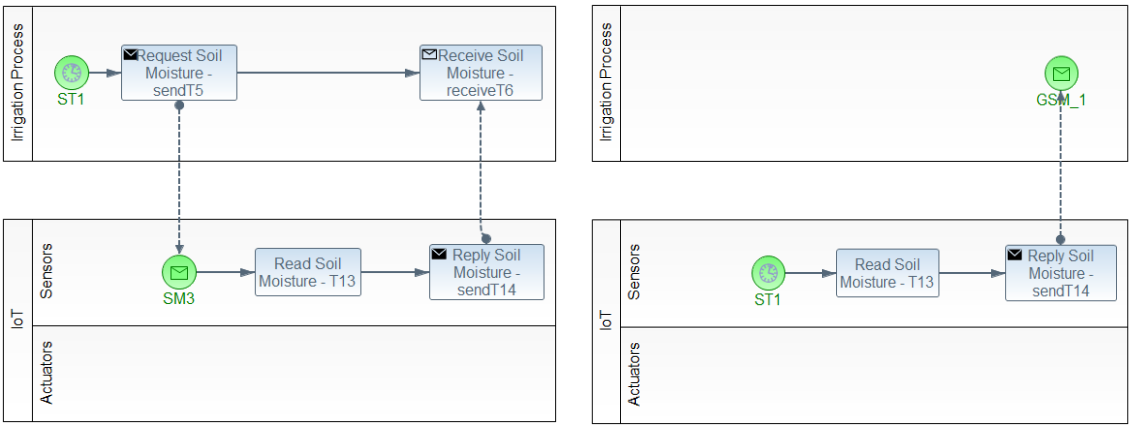

Figura 3.18 - Excerto do caso de uso que ilustra o segundo padrão e respetiva transformação

É possível observar na Figura 3.18 que *ST1* antecede *sendT5* que, por sua vez, antecede *receiveT6*. A transformação a aplicar é eliminar *sendT5*, substituir *SM3* por *ST1* e substituir receiveT6 por um evento de início condicionado por uma mensagem (*GSM_1*).

Padrão 3:

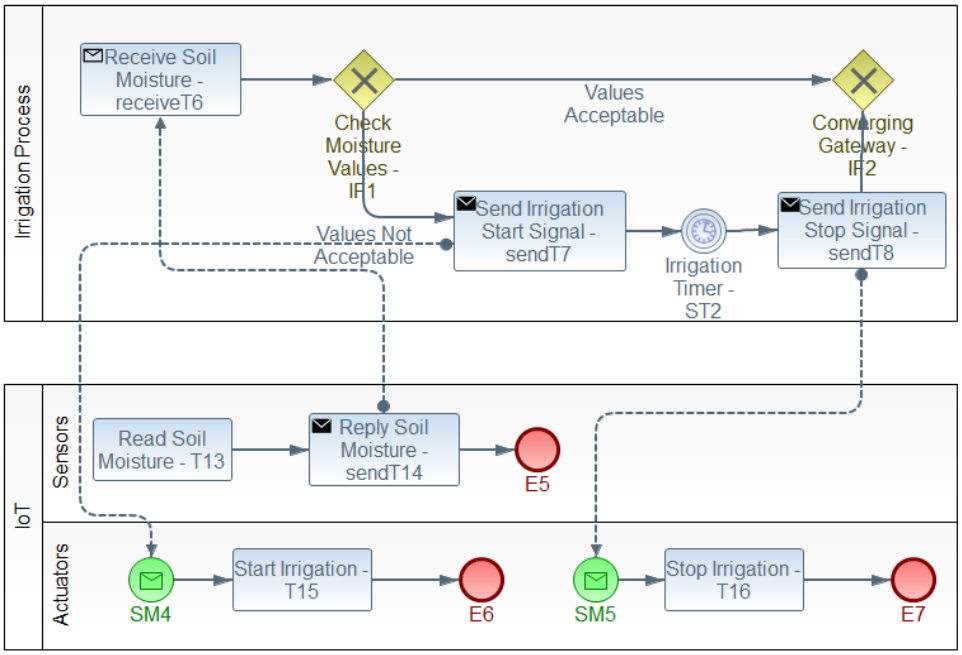

Figura 3.19 - Excerto do caso de uso que ilustra o terceiro padrão

A Figura 3.19 ilustra um excerto do caso de uso com os elementos BPMN relevantes na aplicação do terceiro padrão. O padrão é detetado por *sendT14*, *receiveT6* e *IF1*. A transformação a efetuar para este excerto encontra-se na Figura 3.20

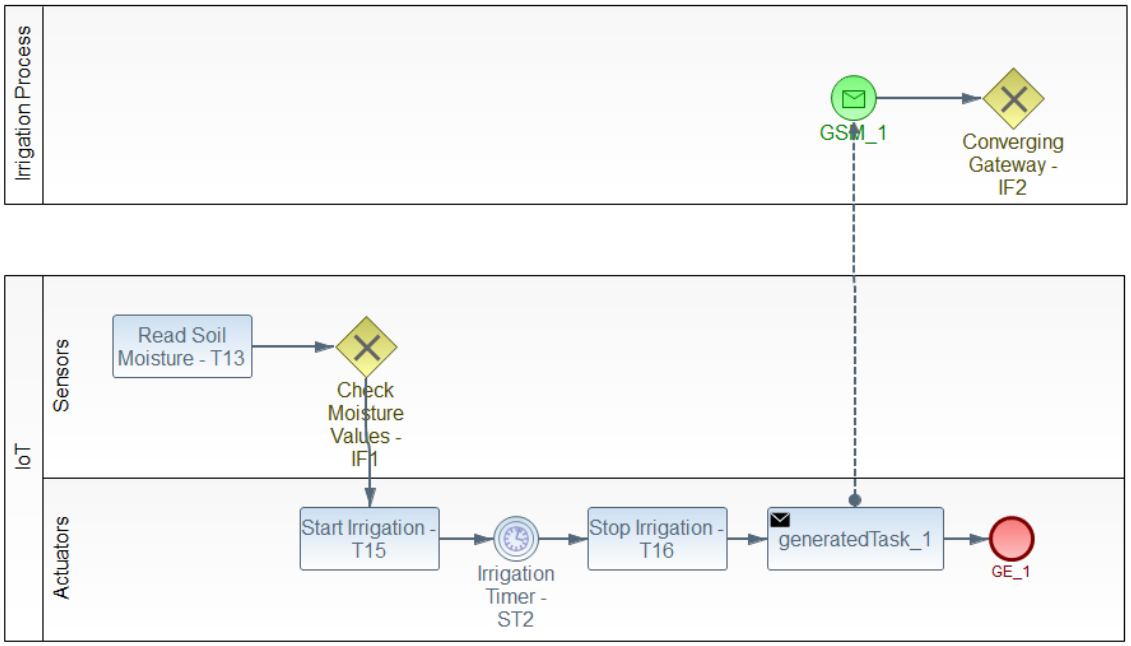

Figura 3.20 - Excerto do caso de uso que ilustra a transformação a aplicar ao terceiro padrão

As tarefas *sendT14* e *receiveT6* são removidos (assim como *E5*) e *IF1* é transferido para a rede de dispositivos IoT e ligado à tarefa antecessora de *sendT14* (*T13*). Para manter o fluxo de controlo original, é necessário decidir o que acontece aos elementos que permaneceram no processo de irrigação e que dependiam de *IF1*. No primeiro ramo, temos *SendT7* que é uma tarefa de envio e, por isso, pode ser movida para a rede de dispositivos IoT. Como tal, procuramos a primeira tarefa que sucede a *sendT7* na rede de dispositivos IoT. A operação passa por eliminar *sendT7*, *SM4* e ligar *IF1* a *T15*. *ST2* pode também ser transferido para a rede de dispositivos IoT (por se tratar de um evento intermédio condicionado por um instante) e é ligado a *T15*. É aplicado o mesmo procedimento que foi feito a *sendT7* a *sendT8*. Neste caso, a primeira tarefa encontrada na rede de dispositivos IoT a partir de *sendT8* foi *T16* e, portanto, esta é ligada a *ST2*. Contudo, *sendT8* antecedia outro elemento (*IF2*). Para manter o fluxo de controlo prévio, o procedimento utilizado é criar uma tarefa de envio de mensagem na rede de dispositivos IoT (neste caso, *generatedTask*_1 que vai suceder *T16*) e o respetivo evento de início condicionado por uma mensagem no participante de destino (neste caso, *GSM*_1 que vai anteceder *IF2*). É também criado um evento de fim (*GE*_2) que é ligado à *generatedTask_1* para terminar o processo na rede de dispositivos IoT. Quanto ao segundo ramo, como este começa por um *gateway* (*IF2*), nenhuma ação é tomada.

Nome: Desenhar colaboração

Objetivo: criar o novo modelo BPMN com as alterações efetuadas.

Entrada:

- Modelo de processos de negócio BPMN;

- Lista de processos;
- Nova lista de processos;
- Nova colaboração;
- Lista de *data stores*;
- Lista de associações.

Saída:

- Novo modelo de processos de negócio BPMN.

Passos:

1. Criar um novo modelo de processos de negócio BPMN, inicializá-lo com as informações do modelo original e com a nova lista de processos, a nova colaboração e a lista de *data stores*.
2. Utilizar o *Graphviz* para gerar as coordenadas de cada elemento que fará parte do novo modelo. Este passo é realizado da seguinte forma:
   2.1. Eliminação de ficheiros DOT previamente criados.
   2.2. Criação de ficheiros DOT com base nas dependências dos elementos de fluxo. É gerado um ficheiro por cada participante.
   2.3. Execução de um comando no *Graphviz* que, a partir dos ficheiros DOT, gera ficheiros de texto com coordenadas para cada nó do grafo.
3. Iterar sobre cada participante da colaboração e atribuir-lhe a dimensão correspondente (altura e largura) proveniente do ficheiro de texto gerado.
   3.1. Caso existam *lanes*, são realizadas operações que determinam a dimensão de cada *lane*, visto que o *Graphviz* apenas gera a dimensão total dos participantes. Estas operações têm por base a distância de elementos entre *lanes* e o valor mais baixo da ordenada Y de entre todos os elementos como forma de determinar os limites inferiores e superiores de cada *lane*.
   3.2. É verificada a ordenada Y de um elemento de cada uma das *lanes*. As *lanes* são ordenadas com base nestes valores, podendo a sua ordem não corresponder à apresentada no modelo original. Isto acontece porque o *Graphviz*, ao organizar os elementos, pode trocar as posições das *lanes*.
4. Iterar sobre cada ficheiro de texto e atribuir a cada elemento de fluxo as coordenadas geradas que lhes foram atribuídas. São aplicadas translações, com base nas dimensões

conhecidas dos participantes e das *lanes*, de modo a enquadrar cada elemento no participante ou *lane* certos.

5. Para cada fluxo de mensagem da nova colaboração, identificar as coordenadas dos elementos de origem e destino e usá-las para posicionar os fluxos de mensagens corretamente. Este passo é necessário, porque o *Graphviz* não lida com elementos de participantes diferentes.

6. Caso o modelo original possua objetos de dados ou *data stores*, itera-se sobre a lista de associações e, através das coordenadas de origem e destino dos elementos de cada associação, posicionam-se as associações. O mesmo é feito para cada atividade que possua associações de dados.

Aplicação ao caso de uso:

```
digraph G {
rankdir=LR;
      subgraph cluster_1{
         StartEvent_2 -> Task_3;
         Task_3 -> SendTask_2;
         StartEvent_5 -> Task_7;
         Task_7 -> SendTask_6;
         SendTask_6 -> GeneratedEndEvent_1;
      }
      subgraph cluster_2{
         StartEvent_6 -> Task_8;
         Task_8 -> EndEvent_6;
         Task_5 -> EndEvent_3;
         StartEvent_7 -> Task_2;
         Task_2 -> EndEvent7;
      }
{ rank=same; StartEvent_2; StartEvent_5; StartEvent_6; StartEvent_7; }
         SendTask_2 -> Task_5;
}
```

Figura 3.21 - Exemplo de um ficheiro DOT da rede IoT gerado a partir do caso de uso

A Figura 3.21 ilustra a criação de um ficheiro DOT com base na dependência dos elementos BPMN. Neste caso, o ficheiro DOT corresponde aos elementos da rede de dispositivos IoT. Cada elemento BPMN é representado pelo seu identificador e não pelo seu nome, porque o *Graphviz* interpreta cada espaço em branco como um elemento diferente.

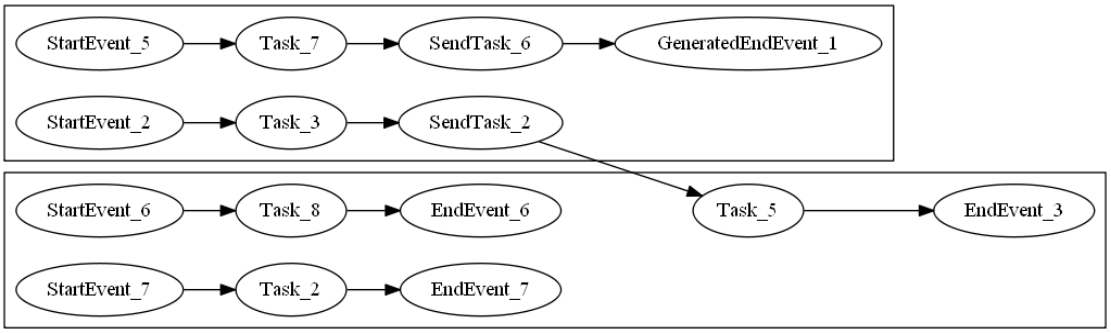

Figura 3.22 - Imagem gerada pelo Graphviz que representa o posicionamento dos elementos do exemplo

A Figura 3.22 ilustra a distribuição de coordenadas que o *Graphviz* atribuiu aos elementos provenientes do ficheiro DOT do exemplo anterior. Como referido no passo 3.2, a distribuição dos elementos com base nas suas dependências pode gerar inversão na ordem das *lanes* para a conveniência do *Graphviz*. Este privilegia colocar elementos com dependências próximos entre si, o que possibilita este problema caso os elementos pertençam a *lanes* diferentes. A coordenada (0,0) encontra-se no canto inferior esquerdo para o *Graphviz*, enquanto que, para o visualizador de BPMN que optámos utilizar no Eclipse, a origem encontra-se no canto superior esquerdo. Por isso, devido às tarefas *SendTask_2* e *Task_5*, irá ocorrer inversão de *lanes*, porque podemos observar que o conjunto de elementos com início em *StartEvent_2* foi posicionado na *lane* superior (segundo o *Graphviz*) de forma a ficar próximo com a *Task_5*.

3.5 Iterações da solução sobre o caso de uso

Esta secção ilustra a execução consecutiva da solução desenvolvida sobre o caso de uso até que não haja mais padrões detetáveis que nos permitam decompor o processo. Cada iteração tem por base o modelo da iteração anterior, pelo que a primeira iteração tem por base o modelo apresentado na Figura 3.1.

Primeira iteração:

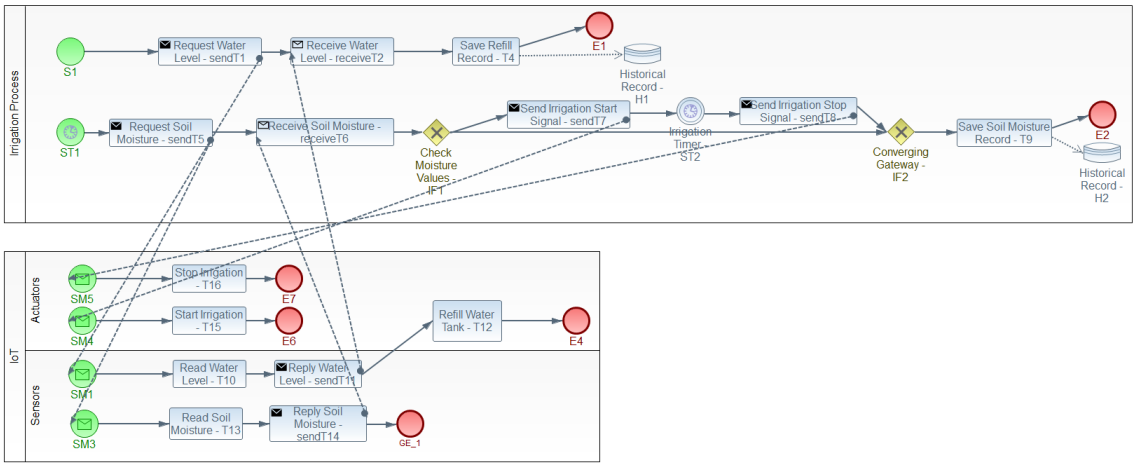

Figura 3.23 - Modelo BPMN resultante da aplicação da solução sobre o caso de uso

Caminhos candidatos: [[sendT1, SM1, T10, sendT11, receiveT2, sendT3, SM2, T12, E4], [ST1, sendT5, SM3, T13, sendT14, receiveT6, IF1, sendT7, ST2, sendT8, SM5, T16, E7]].

Padrão detetado: 1 (devido a *receiveT2* e *sendT3*).

Operações realizadas: *sendT3*, *SM2* e *E3* foram eliminados; *sendT11* foi ligado a *T12*; *E5* foi substituído por *GE_1*.

Inversão de *lanes*: Sim.

Número de elementos eliminados: 3.

Número de comunicações reduzidas: 1.

Segunda iteração:

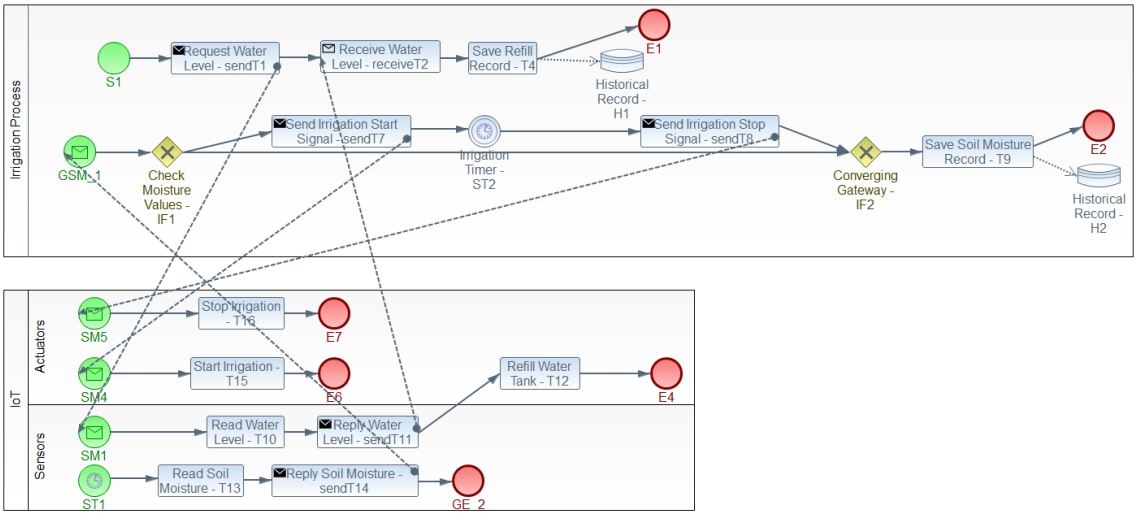

Figura 3.24 - Modelo BPMN resultante da aplicação da solução sobre a primeira iteração

Caminhos candidatos: [[sendT1, SM1, T10, sendT11, T12, E4], [ST1, sendT5, SM3, T13, sendT14, receiveT6, IF1, sendT7, ST2, sendT8, SM5, T16, E7]].

Padrão detetado: 2 (devido a *ST1*, *sendT5 e receiveT6*).

Operações realizadas: *sendT5*, *SM3*, *receiveT6* e *GE_1* foram eliminados; *ST1* foi transferido para a rede de dispositivos IoT e ligado a *T13*; *receiveT6* substituído por *GSM_1* e *GE_1* foi substituído por *GE_2*.

Inversão de *lanes*: Não.

Número de elementos eliminados: 4.

Número de comunicações reduzidas: 1.

Terceira iteração:

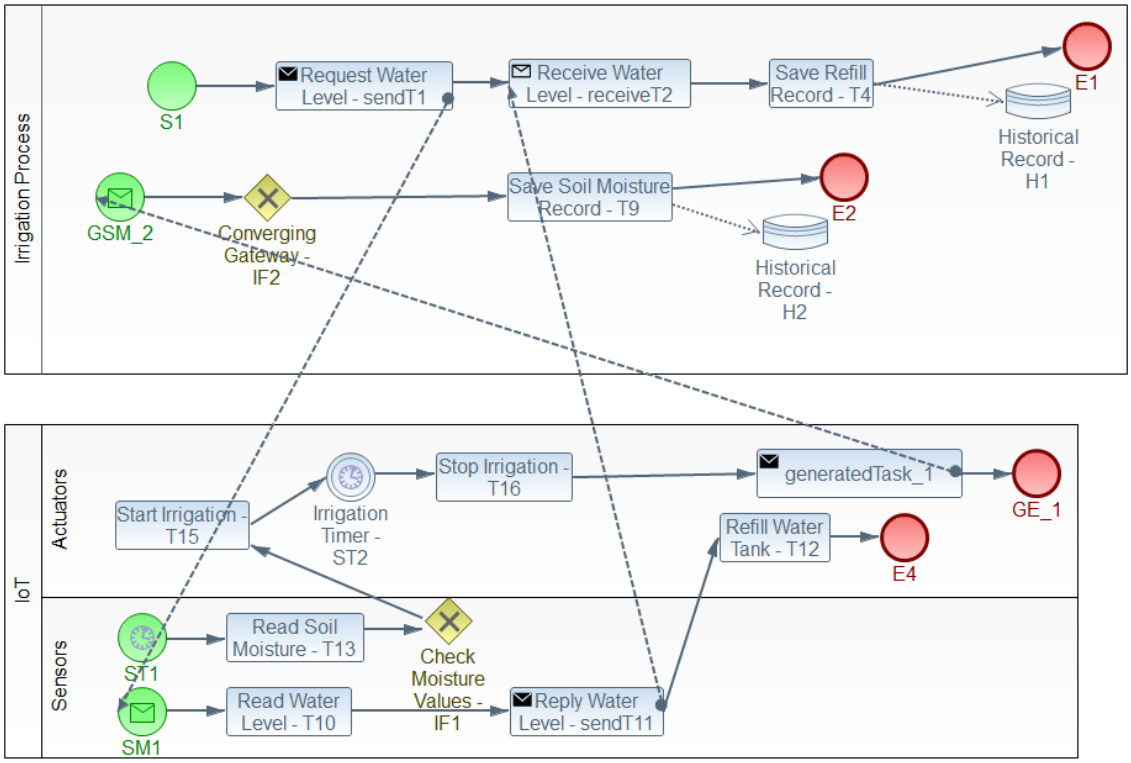

Figura 3.25 - Modelo BPMN resultante da aplicação da solução sobre a segunda iteração

Caminhos candidatos: [[sendT1, SM1, T10, sendT11, T12, E4], [ST1, T13, sendT14, GSM_1, IF1, sendT7, ST2, sendT8, SM5, T16, E7]].

Padrão detetado: 3 (devido a *sendT14*, *GSM_1 e IF1*).

Operações realizadas: *sendT14*, *GSM_1*, *sendT7*, *SM4*, *sendT8*, *SM5*, *E6*, *E7* e *GE_2* foram eliminados; *IF1 e ST2* foram transferidos para a rede de dispositivos IoT; foi criado um *gateway* paralelo que se ligou a *T13*, *GE_2* e *IF1*, contudo, como este estava diretamente ligado a um evento de fim foi eliminado (bem como *GE_2*) e *IF1* tomou a sua ligação a *T13*, *IF1* ligou-se a *T15* e ST2 ligou-se a *T15* e *T16*; foi criada a tarefa de envio de mensagens *generatedTask_1* e ligada a *T16*, foi criado o evento de fim *GE_1* e ligado a *generatedTask_1* e foi criado *GSM_2* para substituir *GSM_1*.

Inversão de *lanes*: Não.

Número de elementos eliminados: 9.

Número de comunicações reduzidas: 2.

Quarta iteração:

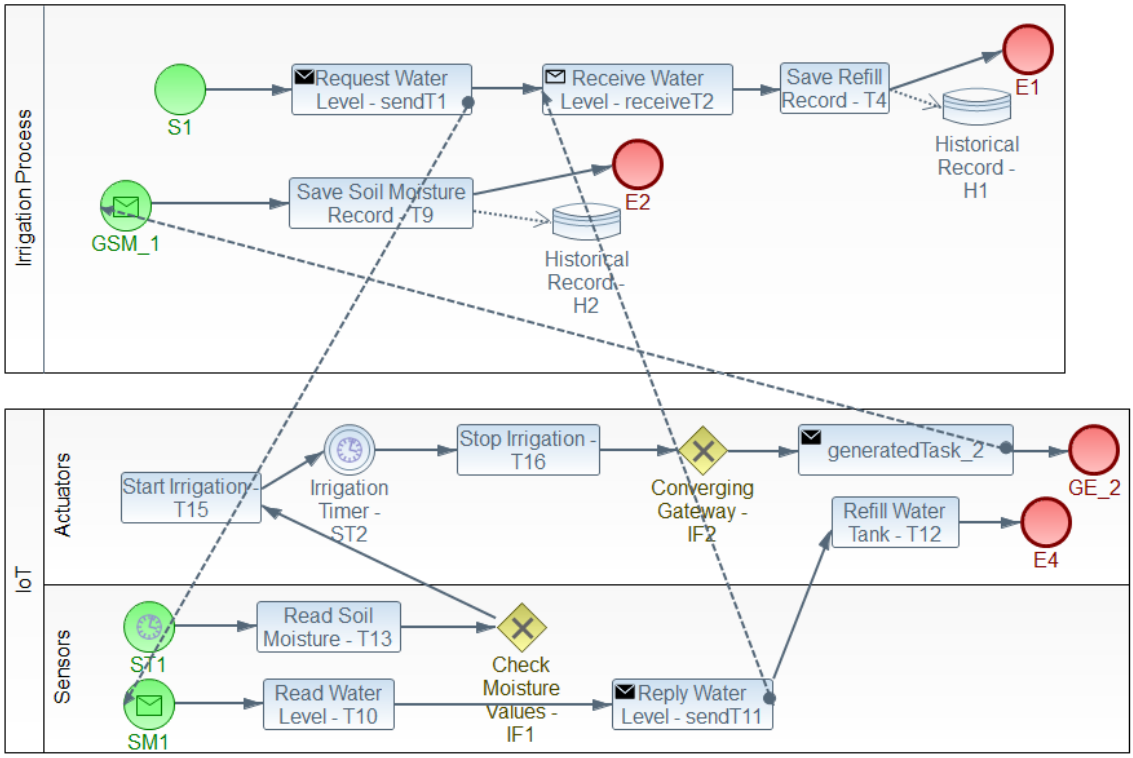

Figura 3.26 - Modelo BPMN resultante da aplicação da solução sobre a terceira iteração

Caminhos candidatos: [[ST1, T13, IF1, T15, ST2, T16, generatedTask_1, GSM_2, IF2], [sendT1, SM1, T10, sendT11, T12, E4]].

Padrão detetado: 3 (devido a *generatedTask_1*, *GSM_2 e IF2*).

Operações realizadas: *generatedTask_1*, *GSM_2* e *GE_1* foram eliminados; *IF1* foi transferido para a rede de dispositivos IoT; foi criado um *gateway* paralelo que se ligou a T16, IF2 e GE_1, contudo, como este estava diretamente ligado a um evento de fim foi eliminado (bem como *GE_1*) e *IF2* tomou a sua ligação a *T16;* foi criada a tarefa de envio de mensagens *generatedTask_2* e ligada a *IF2*, foi criado o evento de fim *GE_2* e ligado a *generatedTask_2* e foi criado *GSM_1* para substituir *GSM_2*.

Inversão de *lanes*: Não.

Número de elementos eliminados: 3.

Número de comunicações reduzidas: 0.

As Figuras 3.23, 3.24, 3.25 e 3.26 ilustram a decomposição do modelo BPMN original do caso de uso até à solução obtida com uso do nosso protótipo. É possível observar que, das 7 comunicações realizadas entre o processo de irrigação e a rede de dispositivos IoT no modelo original, a solução obtida mostra que apenas 3 são necessárias se tirarmos proveito das capacidades que a rede IoT oferece. É também possível observar que nem sempre obtemos redução no número de comunicações (como ilustrado na quarta iteração), mas a operação realizada é igualmente importante, pois a transferência de parte do processo de negócio para os dispositivos IoT reduz a quantidade de processamento centralizado. Neste caso em concreto, o *gateway IF2* poderia até mesmo ser removido, mas de forma manter a coerência com o controlo de fluxo original, optámos manter o maior número de elementos que nos é permitido. A solução contém um total de 24 elementos BPMN, enquanto que o caso de uso original possuía 35. Este é também um aspeto importante a salientar, porque a legibilidade do modelo foi drasticamente melhorada e o modelo agora contém os elementos que são de facto necessários para a realização do trabalho que o processo de negócio pretendia executar.

Capítulo 4 Protótipo

Neste capítulo descrevemos as ferramentas utilizadas pelo protótipo que concebemos para a realização dos procedimentos de decomposição descritos no capítulo anterior. O nosso ambiente de desenvolvimento é o *Eclipse Luna* (versão 4.4.2) [22] e o protótipo foi concebido na linguagem de programação Java. Tirámos partido de dois *plugins Eclipse* (*BPMN2 Modeler* e *SonarLint*) que nos auxiliaram na visualização gráfica dos modelos BPMN e na produção de código fonte de qualidade.

4.1 jBPM

O *jBPM* [24] é uma aplicação JavaEE [23] que implementa a norma BPMN 2.0. Permite modelar, executar e monitorizar o ciclo de vida de processos de negócio. O *jBPM* foca-se em processos de negócio executáveis, ou seja, processos de negócio que contêm um nível de detalhe suficiente para um motor de BPM os executar.

Para apoiar os processos de negócio durante o seu ciclo de vida, o *jBPM6* contém as seguintes ferramentas e características:

- Editor gráfico baseado em *Eclipse* e em *web* para a criação gráfica de processos de negócio (ao estilo *drag and drop*);
- Transações e persistência baseadas em JPA/JTA [25, 26];
- Serviços de tarefas realizáveis por pessoas baseados no *WS-HumanTask*, de modo a incluir tarefas que têm de ser executadas por atores humanos;
- Consola de gestão de instâncias de processos, de formulários de tarefas, de tarefas e de relatórios;
- Repositório de processos (opcional);
- Histórico para monitorização, análise e pesquisas;
- Integração com *Seam* [27], *Spring* [28], *OSGi* [29], entre outros;
- API remota para o motor de processo apresentado, como um serviço (REST, JMS, Remote Java API).

O *jBPM* pode ser associado ao projeto *Drools* [30] para o desenvolvimento de um ambiente unificado para a modelação de lógica de negócio como combinação de processos,

regras e eventos. Em particular, utilizamos o servidor aplicacional *JBoss* com a aplicação *jBPM* e o *BPMN2 Modeler* numa versão pré-instalada do Eclipse, a Luna 4.2.2 que utilizamos para o protótipo.

4.2 BPMN2 Modeler

O *BPMN2 Modeler* [31] é um *plugin* de visualização gráfica de modelos BPMN, compatível com a norma BPMN 2.0. É patrocinado pela *Red Hat* com o propósito de substituir o editor de visualização baseado em Eclipse do *jBPM*. O *BPMN2 Modeler* também pode ser utilizado para documentar processos de negócio internos e a forma como uma organização interage com outras unidades ou parceiros de negócio.

Diferentes tipos de diagramas permitem visualizar processos de diferentes perspetivas e modelar exatamente a forma como uma organização funciona. O *BPMN2 Modeler* permite criar três tipos de diagramas: diagramas de processo, diagramas de colaboração e diagramas de coreografia, como indicado na Figura 4.1.

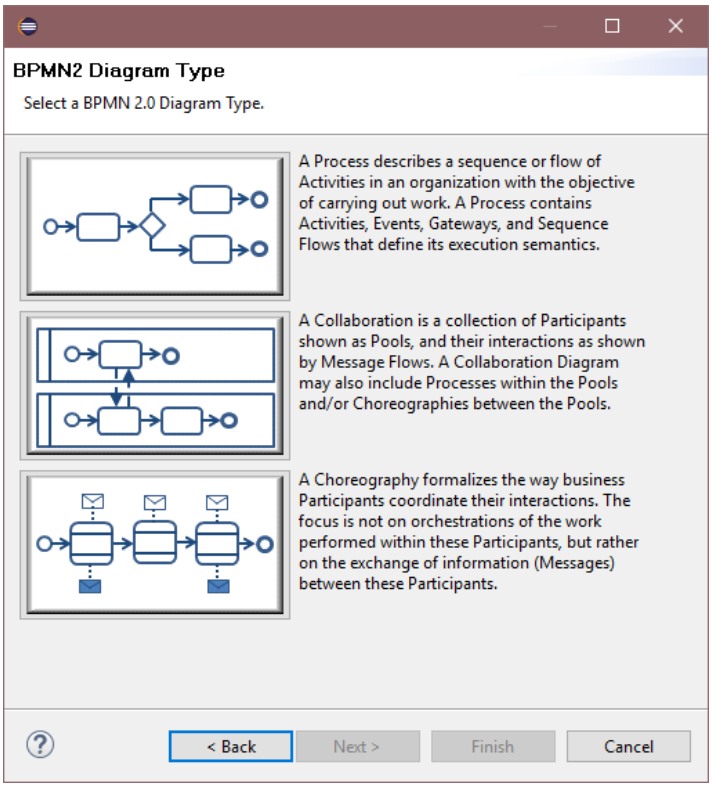

Figura 4.1 - Diferentes tipos de diagramas suportados pelo BPMN2 Modeler

O *BPMN2 Modeler* possui o seu próprio mecanismo de encaminhamento de ligações. Este força a que todas as ligações (fluxos de sequência, fluxos de mensagens, etc) sejam predispostas segundo a aplicação de um algoritmo de *layout*. Existem três algoritmos de *layout* que determinam como as linhas das ligações são encaminhadas da origem até ao destino:

- *Manual Bendpoint:* a linha é desenhada diretamente da origem até ao destino e, caso uma forma seja movida ao ponto de colidir com uma linha das ligações, o editor não fará o reencaminhamento dessa ligação.
- *Automatic Bendpoint:* a linha é desenhada diretamente da origem até ao destino e o editor procura reencaminhar as ligações de forma a não colidir com formas.
- *Manhattan:* as ligações são desenhadas como uma série de segmentos de linha horizontais e verticais desde a origem até ao destino. Os *bendpoints* são automaticamente recolocados quando necessário.

É possível aplicarmos um algoritmo de *layout* diferente para cada tipo de ligação. Na prática, poderíamos aplicar um destes algoritmos de *layout* à solução desenvolvida, mas como o *Graphviz* gera coordenadas para todos os elementos, optámos por aproveitar as coordenadas geradas como pontos de origem e destino nas nossas ligações.

O *BPMN2 Modeler* possui também um mecanismo de validação que verifica, em todo o ficheiro BPMN, elementos que estejam mal configurados ou que sejam desconhecidos e reporta-os com um sinal de aviso ou erro para o utilizador corrigir.

4.3 SonarLint

O *SonarLint* [32] é um *plugin* do Eclipse que utilizamos para garantir a qualidade do código fonte. O seu propósito é o de fornecer *feedback* sobre o código que está a ser construído acerca de faltas ou problemas de qualidade no código fontes. É, portanto, um analisador de código estático. O *SonarLint* também oferece informações sobre código duplicado, padrões de código, testes unitários, cobertura do código, complexidade do código, comentários e vulnerabilidades de segurança.

4.4 Graphviz

O *Graphviz* [20] é uma ferramenta *open-source* que permite criar grafos a partir de *scripts* produzidos de linguagens descritivas. Para grafos direcionados, a linguagem a utilizar é o DOT. Tiramos proveito do *Graphviz* para criar um grafo que contenha os elementos da solução produzida pelo nosso protótipo, devidamente esquematizados.

Existem quatro fases principais quando se utiliza o DOT para a criação de grafos. O procedimento de *layout* (geração de coordenadas) utilizado pelo DOT requer que os grafos sejam acíclicos, por isso, a primeira fase é a de quebrar quaisquer ciclos que existam no grafo. A próxima fase é a de atribuição de *ranks* ou níveis aos nós. Por defeito, o algoritmo de disposição de nós de um grafo segue o esquema *top-to-bottom*, ou seja, os nós são dispostos verticalmente

de forma a que os nós antecessores estejam sempre num nível superior aos seus sucessores. Na terceira fase, os nós são ordenados pelo seu *rank* de forma a evitar sobreposições. Na quarta fase, são atribuídas as coordenadas aos nós de modo a que tenham arestas curtas e, caso existam, são criadas as arestas que possuam curvatura.

Podemos descrever três tipos de objetos com a linguagem DOT: grafos, nós e arestas. Na primeira linha do *script* DOT é definido o nome e tipo de grafo que pretendemos. Nas linhas que se seguem, podemos definir atributos para o grafo em causa, criar nós, arestas, subgrafos e definir atributos para cada um dos objetos. Um nó é criado quando o seu nome aparece pela primeira vez no ficheiro. Uma aresta é criada quando dois nós são unidos pelo operador de arestas '→'. Um subgrafo é definido pela palavra-chave *cluster*. Na nossa solução, para cada *lane*, pertencente a um participante do modelo BPMN, é utilizado um *cluster*.

```
digraph G {
rankdir=LR;
        subgraph cluster_1{
            node1 -> node2;
            node3 -> node4;
        }
        subgraph cluster_2{
            node5 -> node6;
        }
            node3 -> node6;
}
```

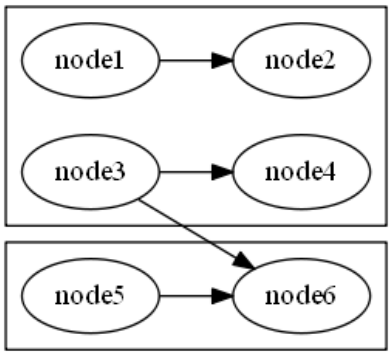

Figura 4.2 - Ilustração de um script na linguagem DOT e o seu resultado

A Figura 4.2 mostra um exemplo de um *script* escrito na linguagem DOT. A palavra "*digraph"* indica que pretendemos um grafo direcionado e o atributo "*rankdir=LR"* define a orientação do grafo da esquerda para a direita (por defeito a ordenação de elementos é de cima para baixo). Segue-se a definição de dois subgrafos e dos seus elementos e, por fim, a indicação da união de dois nós pertencente a subgrafos diferentes. O *Graphviz* processa esta especificação e retorna o resultado no formato de saída especificado pelo utilizador. O formato escolhido para este exemplo foi a figura (formato PNG) visível à direita do ficheiro DOT.

O nosso protótipo gera automaticamente ficheiros DOT e processa-os, obtendo um ficheiro de texto, por cada especificação, com as coordenadas de todos os elementos que compõem um modelo BPMN. A estrutura das especificações DOT geradas é a seguinte:

- A primeira linha é sempre a indicação de que pretendemos gerar um grafo direcionado de nome G;

- A segunda e terceira linhas contêm os atributos "*rankdir=LR"* e "*splines=false*", respetivamente. Este último atributo evita que haja arestas com curvatura, poupando assim um passo no procedimento de *layout* do DOT;
- Caso o participante possua *lanes*, a quarta linha será o atributo "*newrank=true*". Este atributo faz com que o algoritmo de atribuição de *ranks* faça um conjunto de *ranks* global, ignorando a existência de *clusters*. Isto permite que os nós sejam sujeitos a múltiplas restrições. O algoritmo de atribuição de *ranks* é recursivo para *clusters*, o que impossibilita a utilização de restrições em nós de *clusters* diferentes. Por isso, utilizamos este atributo para depois selecionarmos alguns nós e forçá-los a um mesmo *rank*, de modo a alinhá-los verticalmente;
- As próximas linhas são para a criação de nós e arestas. São iterados todos os fluxos de sequência do participante em causa, de modo a aferir as dependências que cada nó possui. De igual forma, são iteradas todas as associações e as dependências entre nós são conhecidas através dos elementos BPMN de origem e de destino. São também iteradas as associações de dados de todas as atividades para incluir as dependências das atividades com os objetos de dados ou *data stores* correspondentes. Cada nó é representado pelo identificador do elemento BPMN correspondente. Se existirem *lanes*, é utilizada a nomenclatura "*subgraph cluster_*" seguida de um número que corresponde ao número de *lanes* que foram iteradas. Neste caso, os nós são englobados nos *clusters* que correspondem à *lane* que os respetivos elementos BPMN pertencem;
- Se existirem *lanes*, a próxima linha irá corresponder à utilização do termo "*rank=same"* seguido do id de todos os eventos de início que o participante possui. Este atributo força a que seja atribuído a todos os eventos de início o mesmo *rank* e, como os eventos de início não possuem antecessores, serão alinhados o mais à esquerda possível e, deste modo, estaremos ao mesmo tempo a alinhar todos os *clusters* verticalmente;
- Por fim, caso existam *lanes*, as últimas linhas corresponderão a dependências de nós que pertençam a *clusters* diferentes. Estas dependências são detetadas caso o elemento de origem, de um dado fluxo de sequência, pertença a uma *lane* diferente do elemento de destino.

4.5 O protótipo

O protótipo teve por base a proposta preliminar ao problema de decomposição de processos de negócio BPMN apresentada em [1]. Esta abordagem utiliza técnicas baseadas em partições para agrupamento de atividades em subprocessos que serão atribuídos por participantes separados. São quatro os passos de decomposição desta abordagem:

1. Criação de um grafo que captura as dependências de fluxo de controlo do processo de negócio;
2. Elaboração de uma tabela de dependências (a partir do grafo gerado) onde estão explícitas as dependências de todos os elementos do processo de negócio;
3. Geração de uma tabela de dependências transitivas, a partir do fecho transitivo da tabela de dependências do passo anterior. Este passo tem também em conta os pares de comunicações da tabela de dependência e os elementos BPMN que podem ser executados nos dispositivos da rede IoT;
4. Redefinição de participantes ao transferir atividades para a rede IoT. Partindo das dependências transitivas, as tarefas de comunicação são eliminadas e os restantes elementos são transferidos para a rede IoT e são criadas novas tarefas de comunicação quando necessário de modo a manter o fluxo de controlo do processo de negócio original.

A primeira tentativa de implementação do protótipo a partir deste procedimento de decomposição começou por ignorar a criação de um grafo intermédio para as tabelas de dependência. Em vez disso, cada dependência era um triplo constituído pelo elemento antecessor, a ligação e o elemento sucessor. A tabela de dependências é criada iterando sobre estes triplos para extrair as dependências de cada elemento. O próximo desafio foi o de definir os elementos BPMN que poderiam ser executados pelos dispositivos da IoT. A ideia era excluir, dos caminhos transitivos, os elementos que não fizessem parte da definição anterior, bem como todos os elementos de comunicação. Deste modo, teríamos apenas de realizar operações sobre os elementos que seriam alvo de algum tipo de transformação. Contudo, esta abordagem foi abandonada durante a implementação das dependências transitivas. A partir dos triplos, é complicado validar que todos os caminhos de cada fluxo de execução do processo de negócio tinham sido visitados. Também surgiram questões sobre que ações aplicar aos elementos dos caminhos transitivos, pois a informação que estes caminhos oferecem é bastante limitada.

A abordagem atual identifica os padrões em que é de facto necessário modificar o processo de negócio e transferir elementos para a rede de dispositivos IoT. Em seguida foi definida a

transformação a aplicar no grafo e o algoritmo para identificar os padrões. A implementação do protótipo segue a versão descrita no capítulo anterior.

O protótipo tirou partido da implementação em Java fornecida pela norma BPMN (que segue a estrutura descrita no guião formal [33]) para tratar dos processos de negócio e seus elementos. Perceber a representação do esquema BPMN em Java também foi um desafio. Cada elemento BPMN é uma classe cuja letra inicial é um "T" seguida do elemento em causa. Por exemplo, os processos são representados pela classe `TProcess` e os eventos de início pela classe `TStartEvent`. Contudo, esta implementação impõe algumas restrições desnecessárias que fizeram com que tivéssemos que reescrever parte de uma classe para adaptarmos à nossa solução. Por exemplo, existem várias classes que possuem atributos representados por uma lista que no máximo terá um elemento, ou a utilização de *wildcards* em listas que dificultam bastante a sua utilização.

O desenho da solução foi a fase que consumiu maior tempo na construção do protótipo. Isto porque é necessário corresponder as coordenadas geradas pelo *Graphviz* com o sistema de coordenas do *BPMN2 Modeler*. O código foi reescrito várias vezes devido aos seguintes problemas:

- Encontrar um fator de escala a aplicar aos elementos BPMN que seja adequada, diferente do fator de escala utilizada em objetos de dados e *data stores*, pois este tipo de objetos ficam bastante deformados e necessitam de um fator de escala menor;
- Aplicar translações aos elementos para os centrar, afastar das margens dos participantes ou para os colocar no participante/*lane* certos. Isto porque, caso as coordenadas geradas pelo *Graphviz* para os elementos fossem diretamente aplicadas, os elementos mais à esquerda estariam diretamente em cima do texto que contém o nome do participante e os elementos mais acima estariam a tocar na margem do participante;
- Encontrar um fator de escala a aplicar aos participantes suficiente para enquadrar todos os elementos;
- Realizar uma translação para que os participantes não fiquem sobrepostos;
- O *Graphviz* não gera coordenadas para cada *cluster* que produz, apenas para o participante, por isso, foi necessário perceber que operações teríamos de realizar de modo a enquadrar as *lanes* dentro dos participantes e sem folgas entre estas;
- Perceber porque motivo, em certos modelos, é necessário inverter a posição das *lanes* para que os elementos contidos nestas fiquem visíveis. Inicialmente, julgou-se que a inversão de *lanes* ocorria sempre que existiam ligações entre elementos de *lanes* diferentes. Contudo, em certos modelos tal não ocorria e, por isso estamos sujeitos à

disposição de elementos que o *Graphviz* nos concede. É mais fácil inverter a posição das *lanes* do que mover individualmente cada elemento para uma posição que se enquadre na posição original da sua *lane*.

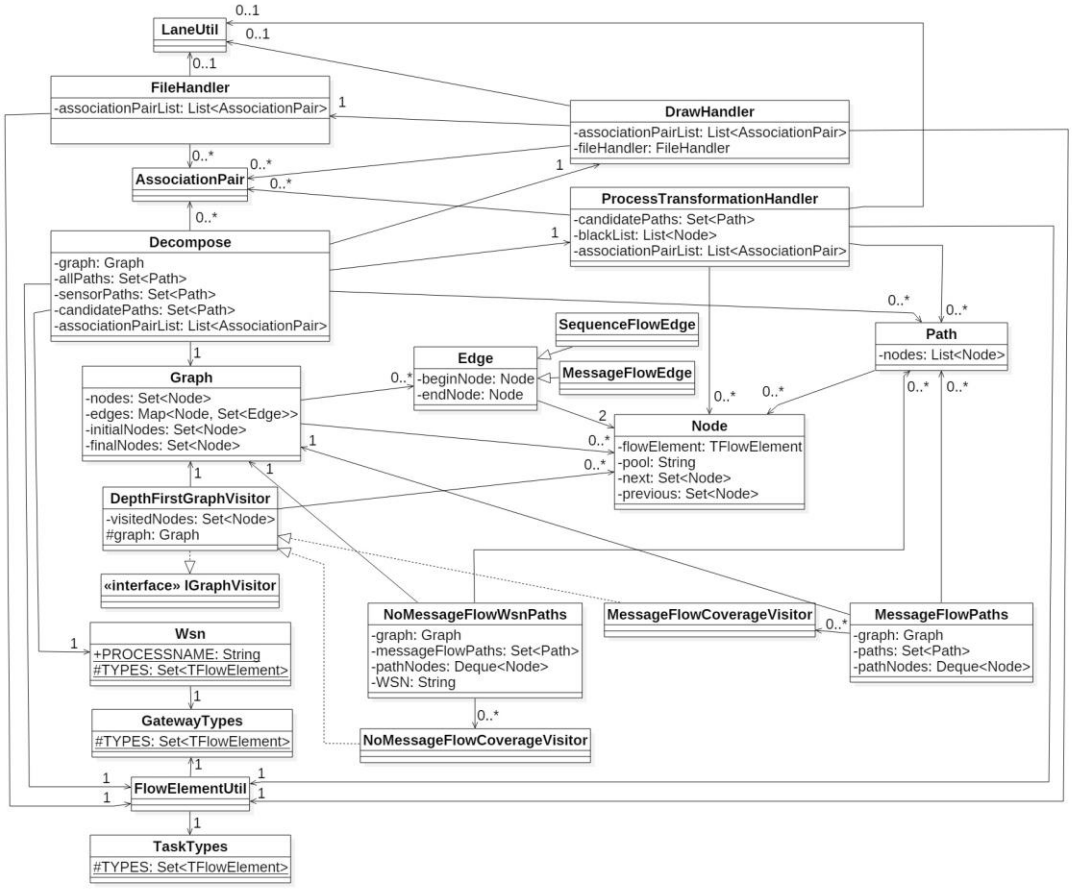

Figura 4.3 - Diagrama de classes do protótipo

A Figura 4.3 ilustra o diagrama de classes do protótipo concebido. A classe `Decompose` é responsável por executar as outras classes que intervêm diretamente no processo de decomposição. A classe `ProcessTransformationHandler` é responsável pelas transformações a efetuar no processo de negócio, descritas no algoritmo de redefinição de participantes. As classes `FileHandler` e `DrawHandler` contêm os métodos necessários para a criação dos ficheiros DOT e desenho da solução final, respetivamente. A classe `DepthFirstGraphVisitor` é a que implementa a pesquisa pelo grafo e as classes `NoMessageFlowWsnPaths` e `MessageFlowPaths` contêm os métodos necessários para realizar uma pesquisa no grafo que retorne todos os caminhos que não contenham arestas de comunicação ou que, obrigatoriamente, contenham arestas de comunicação, respetivamente. As classes `Wsn`, `GatewayTypes` e `TaskTypes` são utilizadas de forma a facilitar a comparação de tipos. Por exemplo, `GatewayTypes` é composta por um conjunto de elementos que contém todos os objetos que representam um *gateway* (`TGateway`, `TInclusiveGateway`,

`TParallelGateway`, entre outros). Por último, `FlowElementUtil` e `LaneUtil` contêm métodos comuns a várias classes.

Capítulo 5 Conclusão e trabalho futuro

Os processos de negócio usam cada vez mais informação disponibilizada por dispositivos IoT, de modo a darem respostas atempadas de acordo com o contexto. No entanto, os dispositivos IoT possuem capacidade computacional suficiente para executar partes de processos de negócio, reduzindo assim o processamento central e o número de mensagens trocadas e, consequentemente, aumentando a autonomia energética destes dispositivos.

Tendo em conta que, normalmente, a modelação de processos de negócio continua a seguir uma abordagem centralizada, o trabalho aqui proposto permite a decomposição automática de processos de negócio dependentes da IoT. A decomposição tem em conta as restrições impostas pelo fluxo de controlo e pelo fluxo de dados e reduz o número de comunicações necessárias com a rede de dispositivos IoT.

Neste momento, este trabalho está a ser integrado com um tradutor de BPMN para código que pode ser executado em dispositivos IoT, com o objetivo de auxiliar as várias fases de desenvolvimento de processos de negócio: definição (com eventual decomposição), instalação (tradução e instalação do código traduzido nos dispositivos) e execução (assegurando a comunicação entre o motor central de execução de processos e os dispositivos). A abordagem também poderá ser generalizada de modo a suportar outras variantes de processos de negócio além da rede IoT.

Outro trabalho que nos propomos é o de realizar uma avaliação quantitativa ao procedimento de decomposição que apresentamos. Esta avaliação permitirá confirmar que o protótipo faz o que é suposto e correto e se a diminuição do número de comunicações com uma entidade central é de facto benéfica. Contudo, poder-se-á afirmar que a remoção de elementos no processo de negócio torna os modelos BPMN correspondentes difíceis de compreensão, dependendo do número de elementos removidos. Será necessário ter este aspeto em conta na realização da avaliação e verificar se o ganho de eficiência do processo de negócio compensa a possível perda de legibilidade do modelo BPMN.

Bibliografia

1. Domingos, D., Martins, F., & Caiola, L.: *Decentralising Internet of Things Aware BPMN Business Processes*. *In: International Conference on Sensor Systems and Software*, pp. 110-119. Springer (2014).
2. Duipmans, E. F., Pires, L. F., & da Silva Santos, L. O. B.: *Towards a BPM cloud architecture with data and activity distribution. In:* IEEE 16th *International Conference on Enterprise Distributed Object Computing Conference Workshops* (EDOCW), pp. 165-171. IEEE (2012).
3. Fdhila, W., Dumas, M., Godart, C., & García-Bañuelos, L.: *Heuristics for composite web service decentralization*. *Software & Systems Modeling*, vol. 13(2), pp. 599-619 (2014).
4. Fdhila, W., Yildiz, U., & Godart, C.: *A flexible approach for automatic process decentralization using dependency tables*. *In*: IEEE *International Conference on Web Services* (ICWS), pp. 847-855. IEEE (2009).
5. Hoenisch, P., Schuller, D., Schulte, S., Hochreiner, C., & Dustdar, S.: *Optimization of complex elastic processes*. IEEE *Transactions on Services Computing*, vol. 9(5), pp. 700-713 (2016).
6. Nanda, M. G., Chandra, S., & Sarkar, V.: *Decentralizing execution of composite web services*. *In*: ACM *Sigplan Notices*, vol. 39(10), pp. 170-187. ACM (2004).
7. OASIS.: *Web services business process execution language version* 2.0. *Technical report, Organization for the Advancement of Structured Information Standards* (2007).
8. 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*: IEEE/ACS 11th *International Conference on Computer Systems and Applications* (AICCSA), pp. 470-477. IEEE (2014).
9. Sadiq, W., Sadiq, S., & Schulz, K.: *Model driven distribution of collaborative business processes*. *In*: IEEE *International Conference on Services Computing*, SCC'06, pp. 281-284. IEEE (2006).
10. Wodtke, D., Weißenfels, J., Weikum, G., & Dittrich, A. K.: *The Mentor project: Steps towards enterprise-wide workflow management*. *In*: *Twelfth International Conference on Data Engineering*, pp. 556-565. IEEE (1996).

11. Yu, Y., Ma, H., & Zhang, M.: A genetic programming approach to distributed execution of data-intensive web service compositions. In: Australasian Computer Science Week Multiconference. ACM (2016).
12. Caetano, A., Silva, A., Tribolet, J.: Business Process Decomposition: An Approach Based on the Principle of Separation of Concerns. Enterprise Modelling and Information Systems Architectures, vol. 5(1) (2010).
13. Fan, S., Hua, Z., Storey, V., Zhao, J.: A process ontology based approach to easing semantic ambiguity in business process modeling. Data & Knowledge Engineering, vol. 102, pp. 57–77 (2016).
14. Yousf, A., Freitas, A., Dey K., A., Saidi, R.: The Use of Ubiquitous Computing for Business Process Improvement. In: Transactions on Services Computing, vol. 9(4), pp. 621-632. IEEE (2016).
15. Atzori, L., Iera, A, Morabito, G.: *The Internet of Things: A survey. Computer Networks*, vol. 54, pp. 2787-2805 (2010).
16. Moreno, M., Úbeda B., Skarmeta A., Zamora M.: *How can We Tackle Energy Efficiency in IoT Based Smart Buildings. Sensors*, vol. 14(6), pp. 9582-9614 (2014).
17. Rault T., Bouabdallah A., Challal Y.: *Energy efficiency in wireless sensor networks: A top-down survey. Computer Networks*, vol. 67, pp. 104-122 (2014).
18. Lee G., Kim J.: *The Internet of Things – A Problem Statement. In: International Conference on Information and Communication Technology Convergence*. ICTC (2010).
19. Zorzi M., Gluhak A., Lange S., Bassi A.: *From today's INTRAnet of things to a future IN-TERnet of things: a wireless- and mobility-related view. In: IEEE Wireless Communications*, vol. 17(6). IEEE (2010).

Webgrafia

20. Página inicial do *Graphviz*, http://www.graphviz.org/, acedido em 2017/07/13

21. Página da especificação *Business Process Model and Notation* do *Object Management Group*, http://www.omg.org/spec/BPMN/2.0, acedida em 2017/07/13.

22. Página inicial do Eclipse, https://eclipse.org/luna/, acedido em 2017/07/16.

23. Página da Oracle que faz referência ao JavaEE http://www.oracle.com/technetwork/java/javaee/overview/index.html, acedido em 2017/07/16.

24. Página inicial do *jBPM JBoss*, http://jbpm.jboss.org/, acedido em 2017/07/16.

25. Página da Oracle que faz referência ao JPA, http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html, acedido em 2017/07/16.

26. Página da Oracle que faz referência ao JTA, http://www.oracle.com/technetwork/java/javaee/jta/index.html, acedido em 2017/07/16.

27. Página inicial da infraestrutura Seam, http://seamframework.org/, acedida em 2017/07/16.

28. Página inicial da infraestrutura Spring, https://spring.io/, acedida em 2017/07/16.

29. Página inicial da OSGi, https://www.osgi.org/, acedida em 2017/07/16.

30. Página inicial do Drools, https://www.drools.org/, acedido em 2017/07/16.

31. Página do guião do utilizador do *BPMN2 Modeler*, https://www.eclipse.org/bpmn2-modeler/documentation/BPMN2ModelerUserGuide-1.0.1.pdf, acedido em 2017/07/18.

32. Página inicial do *SonarLint*, http://www.sonarlint.org/index.html, acedido em 2017/07/18.

33. Guião formal da norma BPMN, http://www.omg.org/spec/BPMN/2.0/PDF, acedido em 2017/08/05.