Concepção de Sistemas ETL Seguros e Confiáveis em Alloy

Designing Safe and Reliable ETL Systems Using Alloy

Mariana Capelo, Universidade do Minho, PORTUGAL, pg27777@alunos.uminho.pt
Orlando Belo, Universidade do Minho, PORTUGAL, obelo@di.uminho.pt

Resumo

Ao longo dos últimos anos foram apresentadas diversas propostas para suporte à modelação conceptual e lógica de processos de povoamento de *data warehouses* - processos de ETL. Todavia, estes processos apresentam usualmente um grau de especificidade elevado, acarretando requisitos de dados bastante complexos e rotinas de transformação muito elaboradas, cuja correção é frequentemente de difícil validação. Na modelação de processos de ETL, a utilização da linguagem de especificação Alloy introduz um formalismo inovador perante as abordagens tradicionalmente utilizadas, mantendo a flexibilidade necessária para lidar com comportamentos específicos dos processos ETL. Adicionalmente, as especificações criadas podem ser analisadas e validadas, oferecendo maior confiança quanto à sua correção, uma característica imprescindível no sucesso de produtos de software complexos. Neste artigo, inspirados pelos avanços registados nesta área de trabalho, apresentamos e discutimos formas de especificar e validar processos de ETL - blocos de operações e as suas dependências - utilizando a linguagem Alloy.

Palavras-chave: Sistemas de *Data Warehousing*; Povoamento de *Data Warehouse*, Especificação Formal de Software, Especificação e Validação de Processos de ETL. Alloy.

Abstract

Over the last few years, several proposals have been presented for supporting conceptual and logical modelling of data warehousing populating processes - ETL processes. However, these processes usually have a high degree of specificity, which entails very complex data requirements and elaborate processing routines – often difficult to validate. In ETL process modelling, the use of the Alloy specification language introduces an innovative formalism to the traditional approaches, maintaining the flexibility for handling the specific behaviours of an ETL process. Additionally, Alloy specifications can be analysed and validated, offering greater confidence in its correctness, which is essential for the success of complex software products. In this paper we present and discuss how to specify and validate ETL processes - blocks of operations and their dependencies - using Alloy, inspired by advances in this area of research, which show the potential of using a formal language in the ETL process modelling domain.

Keywords: *Data Warehousing Systems; Populating a Data Warehouse, Formal Specification of Software, ETL Processes Specification and Validation, Alloy.*

1. INTRODUÇÃO

É vulgar encontrarmos na base de um sistema de suporte à decisão um sistema de *data warehousing* (SDW) assegurando o fornecimento de informação pertinente aos vários processos de negócio que são tratados no seu ambiente. Os SDW (Inmon, 2005) (Kimball e Ross, 2002) são sistemas concebidos de forma a armazenar, num único repositório de dados – o *data warehouse* –, informação temporal, não volátil, cuja organização foi concebida de forma a sustentar as diversas

ISSN 2183-489X

dimensões de análise utilizadas pelos agentes de decisão. Além disso, são sistemas especializados na centralização de dados. Como tal, estão munidos com serviços e estruturas que lhes permitem angariar uma grande variedade de dados, a partir de diferentes fontes de informação, usualmente com características heterogéneas, e armazená-los de acordo com as estruturas multidimensionais que foram estabelecidas no *data warehouse*. Porém, a construção e desenvolvimento de um SDW é uma tarefa bastante complexa, envolvendo processos de trabalho e desenvolvimento de componentes de software, não raras vezes, complexos e sofisticados. Dentro do leque desses componentes evidencia-se o sistema de povoamento do *data warehouse*, vulgarmente designado por processo de ETL (Extração, Transformação e Carregamento). Este processo é merecedor de especial atenção pelas suas próprias características e pelas dificuldades de projeto e implementação que levanta na grande generalidade dos projetos de *data warehousing*. Os processos de ETL (Kimball e Caserta, 2004) têm como objetivo principal alimentar um SDW. Para que a qualidade dos dados que são processados seja garantida, os processos de ETL necessitam de mecanismos para tratamento de dados e para recuperação de erros. Adicionalmente, o fluxo de execução de um processo de ETL apresenta, frequentemente, algumas dependências entre componentes de serviço, processamento de dados, e execução de tarefas em paralelo. Uma implementação de menor qualidade de um processo de ETL terá consequências negativas não só a nível do desenvolvimento e da manutenção do sistema, bem como a nível da qualidade dos dados do próprio SDW (English, 1999).

Apesar do peso que o desenvolvimento de um processo ETL acarreta num projeto de um SDW, e da importância da sua correção, hoje ainda temos algumas dificuldades em encontrar uma abordagem definitiva, simples e rigorosa, que nos permita modelar e validar um processo de ETL. Com base no conhecimento adquirido em (Oliveira e Belo, 2017) e (Oliveira et al., 2016), alguns dos primeiros esforços que conhecemos na especificação formal de tarefas ETL utilizando Alloy, decidimos apresentar e discutir neste artigo algumas das formas de especificação e validação de processos de ETL - blocos de operações e as suas dependências - utilizando uma linguagem de especificação formal, em particular a linguagem Alloy.

A Alloy (Jackson, 2012) é uma linguagem textual de especificação desenvolvida no MIT em 1997, que permite descrever estruturas complexas de um sistema, bem como as restrições e comportamentos associados a este. Uma das suas principais características é a sua capacidade analítica, que permite validar as especificações declaradas, através de uma ferramenta: o Alloy *Analyzer*. A análise de especificações em Alloy é implementada através da descoberta de instâncias de um dado modelo ou de contraexemplos de uma asserção relacionada com esse modelo, alertando para a existência de erros no sistema modelado. No contexto da modelação de processos de ETL, a utilização da linguagem Alloy introduz um formalismo inovador, mantendo a flexibilidade necessária para lidar com comportamentos específicos de processos ETL.

Adicionalmente, uma vez que as especificações criadas em Alloy podem ser analisadas e validadas, é oferecida uma maior confiança quanto à correção do sistema – uma característica imprescindível para o sucesso de produtos de software complexos, como os processos de ETL. Assim, ao longo das próximas secções deste artigo, faremos uma breve exposição de alguns trabalhos relacionados com a modelação de processos de ETL (Secção 2), veremos como poderemos utilizar a linguagem Alloy na especificação de processos de ETL (Secção 3), ilustrando, sempre que for pertinente, a exposição realizada com alguns exemplos que consideramos importantes na aplicação da linguagem, e apresentaremos a especificação formal de alguns componentes de um processo de ETL, discutindo os seus aspetos mais pertinentes e as vantagens que retiraremos da utilização da linguagem Alloy no domínio da modelação de processos de ETL e na garantia da sua segurança e confiabilidade. Por fim, terminaremos este artigo com algumas conclusões sobre o trabalho realizado e apontaremos algumas linhas de trabalho para futuros processos de investigação (Secção 4).

2. TRABALHO RELACIONADO

A modelação de processos ETL tem sido alvo de estudo nas últimas décadas. A definição de uma metodologia ou notação padrão para a modelação de processos de ETL aceite pela comunidade é um passo importante para a padronização do próprio desenvolvimento de SDW. A utilização de padrões é uma forma de tornar a implementação mais metódica e regularizada, evitando erros ou falhas conhecidas através da utilização de elementos padrão. Ademais, uma padronização de processos de ETL pode permitir a sua automação. Atualmente, grande parte das ferramentas disponíveis no mercado já permitem algum tipo de automação de tarefas ETL, o que simplifica bastante o desenvolvimento dos processos de ETL. Porém, as notações ou os modos de funcionamento interno dessas ferramentas são bastante específicos (ou mesmo desconhecidos) para os seus utilizadores, devido ao seu estatuto proprietário. Apesar disso, temos vindo a assistir a algumas propostas muito interessantes no domínio das iniciativas de código aberto – *open source*.

Desde as abordagens para o desenvolvimento de processos de ETL inicialmente propostas por Inmon (2005) e Kimball e Ross (2002) que, algo similares, defendem um processo essencialmente *ad-hoc e* demonstram alguma falta de rigor e formalismo, têm surgido outras abordagens, que tentam esquematizar, de uma forma simples, os mecanismos inerentes aos processos de ETL. A modelação com base em meta modelos e meta dados, e a sua consequente instanciação, foi o primeiro marco no processo da esquematização dos processos de ETL, despoletando diferentes abordagens que tentaram utilizar linguagens de modelação como a UML ou a notação BPMN – na qual um processo de ETL é identificado como um processo de negócio –, enquanto outras abordagens, mais formais, apelaram à álgebra relacional ou linguagens de especificação formal para fazer esse mesmo trabalho. Em Vassiliadis e Simitsis (2002) foi defendida uma arquitetura

baseada em três níveis para suportar a modelação conceptual de processos de ETL, nomeadamente: uma camada de meta modelos com construtores genéricos (e.g. atributo, relação ou transformação), uma camada de *templates*, com construtores específicos de atividades de ETL (e.g. tabela de factos, dimensão ou atribuição de chaves de substituição) e, por fim, uma camada de esquema com instâncias das classes da camada de meta modelos e das subclasses contidas na camada de *templates*. Apesar de conceptualmente bem dividida e de fácil compreensão, a arquitetura sugerida conduz ao desenvolvimento de diagramas bastante complexos, pouco legíveis, mesmo quando aplicada a casos de processos de ETL que consideramos simples. Além desta proposta, também no âmbito da modelação conceptual destes processos e de forma a padronizá-los, foi proposta uma extensão à notação UML por Trujillo e Luján-Mora (2003). Nesta abordagem o processo de desenvolvimento de um *data warehouse* é estruturado com base num modelo integrado, constituído por esquemas para representar os dados operacionais, o *data warehouse* (esquema concetual e esquema de armazenamento físico) e o modelo de negócio, que são complementados com esquemas de mapeamento (entre o esquema de dados operacionais e o esquema concetual do *data warehouse* e entre o esquema concetual do *data warehouse* e o seu esquema de armazenamento). Posteriormente, com Akkaoui e Zimanyi (2009), os processos de ETL foram modelados como processos de negócio. Para isso, propuseram uma extensão à notação BPMN de forma a que fosse possível lidar com as especificidades de certas tarefas de ETL. A utilização da BPMN permite, através da linguagem de programação BPEL, uma tradução dos modelos criados para programas. Esta abordagem foi também explorada por outros autores, que encontravam no mapeamento de modelos conceptuais para um conjunto de primitivas de execução grande valor. A partir dessa base, tais autores criaram modelos conceptuais específicos para diferentes processos de ETL (Oliveira e Belo, 2013). Contudo, estes modelos continuavam a não apresentar uma componente lógica, que fosse capaz de reduzir a liberdade oferecida no momento de implementação das tarefas ETL.

Mais tarde, em Santos (2015) a especificação de processos de ETL foi abordada utilizando operadores e árvores de álgebra relacional, o que adicionou uma componente formal à sua definição e construção. Nesse estudo foram especificados alguns processos padrão ETL, como foi o caso das tarefas de captura de alteração de dados – *Changing Data Capture* (CDC) – ou da garantia da qualidade dos dados – *Data Quality Enforcement* (DQE) –, particularmente em termos da decomposição, conformidade e conciliação de dados. Além disso, foram também estudadas casos de padrões ETL para suporte a tarefas relacionadas com dimensões com variação de dados de diferentes tipos e a atribuição de chaves de substituição – *Surrogate Key Pipelining* (SKP). Contudo, esta abordagem falha na captação de aspetos intrínsecos dos processos em questão. Os operadores utilizados na especificação não permitem a modelação de comportamentos, como o caso da deteção e tratamento de erros, uma vez que a álgebra relacional é orientada à manipulação

de dados e não ao controlo de fluxos de operações. Assim, apesar do formalismo que acarreta, esta abordagem não satisfaz como um todo.

Mais recentemente, encontrámos em Oliveira e Belo (2017) e Oliveira et al. (2016) duas iniciativas na utilização da linguagem de especificação formal Alloy para modelar processos de ETL. Em (Oliveira et al., 2016) é apresentada a definição de um padrão ETL – *Dimensão com Variação e Manutenção de Histórico* (SCD-H) –, enquanto padrão de transformação, e a sua especificação em Alloy, incluindo a sua modelação estrutural e comportamental. O trabalho realizado evidencia o objetivo principal desta abordagem: a implementação de processos de ETL com um elevado grau de abstração e com um formalismo bem definido na sua modelação, mas sem acarretar uma grande carga matemática. Adicionalmente, a especificação realizada pode ser validada, tirando partido das capacidades de análise da linguagem Alloy. A utilização da linguagem Alloy para a modelação e validação de processos de ETL introduz um formalismo inovador perante as abordagens atualmente existentes, mantendo a flexibilidade necessária para lidar com comportamentos específicos dos processos de ETL e permitindo a validação e consequente confiança na qualidade do sistema. Nas próximas secções veremos como resultou a utilização da linguagem de especificação formal Alloy, na especificação e validação de processos de ETL, seus blocos de operações e dependências.

3. ESPECIFICAÇÃO DE PROCESSOS ETL EM ALLOY

3.1. Especificações Formais

As especificações formais são técnicas matemáticas utilizadas para descrever um sistema e analisar o seu comportamento. O seu propósito é auxiliar na implementação de sistemas e produtos de software, ajudando à sua modelação através da verificação de propriedades chave, com vista a garantia da sua correção e robustez. As especificações formais utilizam notações matemáticas para descrever de forma precisa as propriedades de um dado sistema. Contudo, fazem-no sem restringir a forma como essas propriedades são mantidas ou alcançadas – uma especificação formal descreve o que um sistema deve fazer sem especificar como é feito (Spivey, 1989).

A filosofia por detrás da utilização de especificações formais no desenvolvimento de software baseia-se na aplicação de abstrações no seu projeto. As abstrações em software implicam uma definição precisa, consistente e completa das propriedades do sistema a desenvolver. Isto conduz a uma maior compreensão do próprio sistema e da sua modelação, ajudando na descoberta de problemas ainda na fase inicial de desenvolvimento. Adicionalmente, ao se definirem as propriedades do sistema a modelar, a sua complexidade conceptual é reduzida e torna-se mais fácil a sua decomposição modular. As linguagens de especificação formal são utilizadas para expressar especificações formais numa dada linguagem, cujo vocabulário, sintaxe e semântica são

formalmente definidas, o que significa que são baseadas em conceitos matemáticos bem compreendidos.

3.2. A Linguagem Alloy

A linguagem de especificação formal Alloy (Jackson, 2012) é uma linguagem textual desenvolvida no MIT em 1997, que permite descrever estruturas complexas, bem como as suas restrições e comportamentos. A linguagem Alloy está profundamente enraizada na notação Z (Spivey, 1989). A Alloy permite descrever estruturas com um conjunto mínimo de notações matemáticas, utilizando, também, notações de modelação de objetos, o que permite simplificar a classificação de objetos e a associação de propriedades. Adicionalmente, os modelos Alloy podem ser analisados e validados. Os processos de análise em Alloy são implementados através da descoberta de instâncias de um modelo ou de contraexemplos de uma asserção relacionada com o modelo. A análise é executada de forma exaustiva através da utilização de um SAT *solver*, dentro de um domínio com cardinalidade definida pelo utilizador. Ao contrário da maioria das notações conhecidas, a Alloy não se baseia na prova completa de teoremas - a análise é feita sobre um espaço limitado de casos, sendo, no entanto, completa dentro do espaço estabelecido. Assim, a abordagem seguida pela Alloy consegue, em geral, apenas revelar a presença de erros num modelo dentro do *scope* definido, e não provar a sua completa ausência. Contudo, a hipótese *Small Scope* postula que a maior parte dos erros pode ser encontrada em modelos com um *scope* pequeno. Desta forma, a Alloy tem vindo a ser usada com sucesso para a modelação de problemas em diferentes áreas e de diferentes naturezas.

A Alloy é uma linguagem orientada a modelos, que defende o desenho de abstrações, enquanto ideias reduzidas à sua forma essencial. Modelar um sistema em Alloy é um exercício que exige um nível alto de compreensão do problema e uma forma genérica de analisar a solução a desenvolver. Esta simplificação leva a uma maior clareza no processo de planeamento, permitindo detetar vários problemas de desenho nesta fase inicial. Além disso, a modelação em Alloy não é um processo complexo, tendo uma curva de aprendizagem suave e sendo o seu resultado final facilmente legível.

3.3. Especificação de Modelos em Alloy

Os modelos escritos em Alloy são micromodelos, declarativos, estruturais e analisáveis. O desenvolvimento de modelos em Alloy é realizado no seu próprio ambiente e tira partido da ferramenta Alloy Analyzer, sendo um desenvolvimento interativo no qual os cenários obtidos através da análise do modelo são considerados para a sua construção e refinação incremental. Uma das principais características da linguagem é o conceito de relação. Na verdade, a Alloy baseia-se em duas definições: átomos e relações. Os átomos são entidades indivisíveis, imutáveis e não-

interpretadas (sem qualquer tipo de propriedades nativas). A forma como tudo evolui a partir dos átomos deve-se à entidade de relação, que é responsável por relacionar os mesmos. De referir que, um conjunto de átomos é uma relação unária, um escalar é uma relação unária com apenas uma entrada, e os relacionamentos mais complexos são traduzidos em relações entre átomos. Na prática, um modelo Alloy é constituído por uma declaração de módulo, um conjunto de importações de módulos e um conjunto de parágrafos. Estes podem ser uma declaração de assinatura, uma restrição ou um comando. Uma declaração de assinatura representa um conjunto de átomos e pode introduzir campos que representam relações entre assinaturas. As restrições são divididas em factos (invariantes do modelo), predicados (restrições a serem usadas e cumpridas em diferentes contextos), funções (expressões reutilizáveis com capacidade de retornar um valor) e asserções (implicações a serem confirmadas). Os comandos são instruções para analisar o modelo construído – nomeadamente, o comando *run* procura soluções ou instâncias de um predicado específico e o comando *check* procura contraexemplos que contrariem uma determinada asserção. Para a escrita de um modelo em Alloy, é possível tirar partido de uma lógica relacional que combina quantificadores de lógica de primeira ordem e operadores de cálculo relacional.

De seguida, veremos os exemplos que trabalhámos para aplicar e analisar a viabilidade e utilidade da linguagem de especificação formal Alloy na construção e validação de modelos de processos de ETL, abordando a modelação estrutural das entidades das tarefas ETL escolhidas bem como a sua modelação dinâmica. Os exemplos escolhidos são, na nossa opinião, componentes chave que podem ser encontrados na generalidade dos processos de ETL e, como tal, bastante convenientes para a satisfação dos objetivos que traçámos para este trabalho.

3.4. Especificação de Modelos ETL em Alloy

A modelação dos exemplos que escolhemos seguiu um dos princípios da modelação de sistemas e da própria linguagem Alloy, princípio este que dita que não se devem procurar modelos completos, mas sim modelos adequados ao propósito da modelação. Assim, excluímos do processo de modelação todos os objetos de dados e propriedades dos subsistemas sem interesse para os componentes ETL a modelar. Antes de apresentarmos a modelação realizada dos componentes ETL escolhidos, façamos, de forma breve, a apresentação do esquema conceptual (Figura 1) do sistema de povoamento de um *data mart*, que utilizámos como base de trabalho para este artigo. Em termos gerais, no esquema apresentado vemos os principais (sub)processos ETL, organizados de forma bastante convencional, abordando as principais etapas de um processo de ETL. Assim, podemos identificar, num primeiro grupo (E), os processos envolvidos com a extração de dados nas fontes de informação do SDW, que são, basicamente, processos *Change Data Capture* (CDC), que determinam os dados a importar para o *data mart*. Depois, num segundo grupo (T), essencialmente constituído por processos de transformação, encontramos os processos de ETL

para garantir a correção e integridade dos dados coletados – *Data Quality Enhancement* (DQE) –, para fazer a conciliação de dados provenientes das diversas fontes – *Data Conciliation* (DCI) – e para fazer a substituição de chaves operacionais por chaves de substituição – *Surrogate Key Pipelining* (SKP). Por fim, num terceiro grupo (L), vemos as operações de carregamento de dados no *data mart*, que integram processos de atualização de dados em dimensões – *Slowly Changing Dimensions* (SCD) – e carregamento de dados em tabelas de factos – *Intensive Data Loading* (IDL).

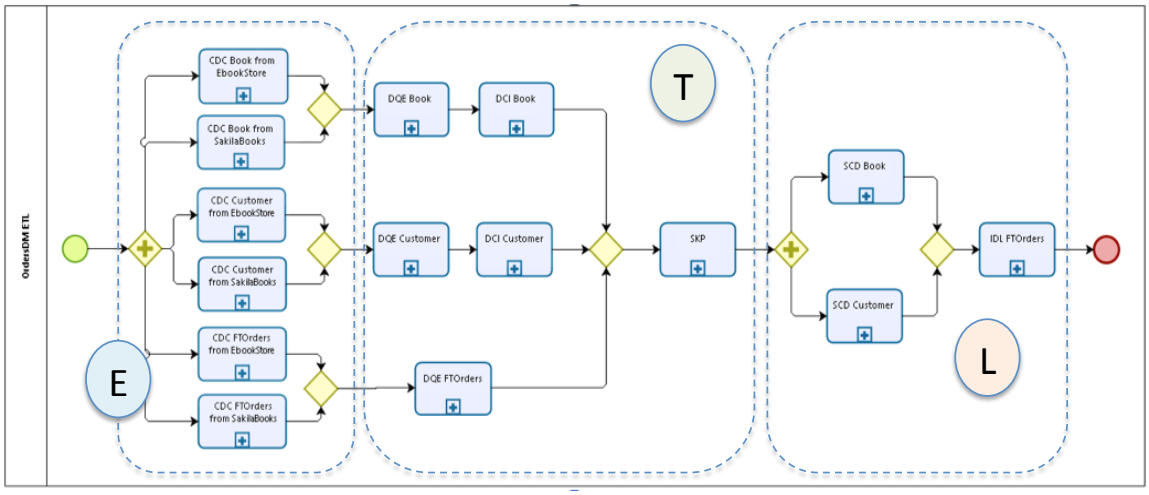


Figura 1 – Modelo conceptual do sistema de povoamento em BPMN.

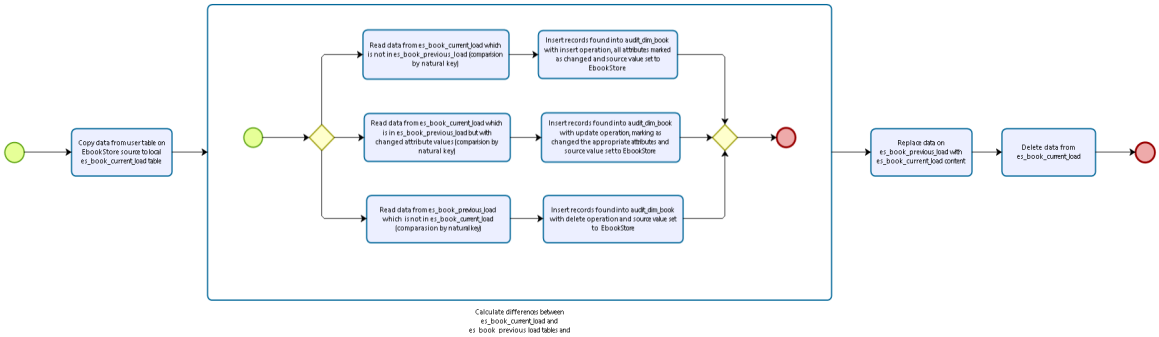


Figura 2 – Modelo conceptual de um componente CDC.

No âmbito geral deste artigo não incorporámos uma descrição detalhada da especificação completa do sistema de povoamento da Figura 1, uma vez que apenas pretendemos revelar a utilidade da utilização da linguagem Alloy na especificação formal e validação de componentes ETL. Como tal, de todas as tarefas ETL referidas, abordaremos em detalhe apenas os aspetos relacionados com a captura das modificações de dados ocorridas nas fontes de informação do caso considerado (Figura 2). Este processo, conhecido como CDC, é, como sabemos, um dos casos mais emblemáticos de um sistema de povoamento e um dos seus processos de trabalho mais críticos – no nosso caso, este tipo de processo envolveu mais de um terço do esforço de especificação e desenvolvimento.

```
/*
 * Predicate: CDCEbookStoreBooks
 * Receives two states (previous and next). The next state has the AuditBooks table with the new audit
 * books from EbookStore and an updated EbookStoreBookPreviousLoad table.
*/
pred CDCEbookStoreBooks[s, s': State]
{
   // Pre-conditions
   (EbookStoreBookCurrentLoad.rows).s = (StagingArea/EbookStore/Books.rows).s

   // Post-conditions
   // Calculate differences between current and previous load and feed AuditBooks accordingly.
   // D - All books from PreviousLoad which are not present in CurrentLoad in the previous state s
   // generate an AuditBook entry with Delete operation and EbookStore source in the next state s'
   all pl : (EbookStoreBookPreviousLoad.rows).s |
      ( no cl : (EbookStoreBookCurrentLoad.rows).s | cl.id = pl.id )
      implies
      ( one a' : (AuditBooks.rows).s' | a'.id = pl.id and a'.title = pl.title and a'.ISBN = pl.ISBN and
         a'.description = pl.summary and a'.timestamp = pl.last_update and a'.source = EbookStore and
         a'.op = D and a'.title_changed = False and a'.ISBN_changed = False and a'.description_changed = False
      )

   // I - All books from CurrentLoad which are not present in PreviousLoad in the previous state s
   // generate an AuditBook entry with Insert operation and EbookStore source in the next state s'
   all cl : (EbookStoreBookCurrentLoad.rows).s |
      ( no pl : (EbookStoreBookPreviousLoad.rows).s | cl.id = pl.id )
      implies
      ( one a' : (AuditBooks.rows).s' | a'.id = cl.id and a'.title = cl.title and (...) and
         a'.source = EbookStore and a'.op = I and
         a'.title_changed = True and a'.ISBN_changed = True and a'.description_changed = True
      )
 (...)
   // AuditBooks: s' -> s (entries present in s' are either present in s or were generated by a calculated difference from s)
   all a' : (AuditBooks.rows).s' | ( a' in (AuditBooks.rows).s ) iff not
      (
         ( one pl: (EbookStoreBookPreviousLoad.rows).s | a'.id = pl.id and (...) and
            a'.source = EbookStore and a'.op = D and a'.title_changed = False and (...) and
            ( no cl : (EbookStoreBookCurrentLoad.rows).s | cl.id = pl.id )
         ) or
         ( one cl: (EbookStoreBookCurrentLoad.rows).s | a'.id = cl.id and (...) and
            a'.source = EbookStore and a'.op = I and a'.title_changed = True and (...) and
            ( no pl : (EbookStoreBookPreviousLoad.rows).s | cl.id = pl.id )
         ) or
         ( one cl: (EbookStoreBookCurrentLoad.rows).s |
            some pl : (EbookStoreBookPreviousLoad.rows).s |
               cl.id = pl.id and (cl.title != pl.title or cl.summary != pl.summary or cl.ISBN != pl.ISBN) and
               a'.id = cl.id and (...) and a'.source = EbookStore and a'.op = U and
               (
                  ( cl.ISBN != pl.ISBN )
                  implies
                  ( a'.ISBN_changed = True and a'.title_changed = True and a'.description_changed = True)
                  else
                  (   a'.ISBN_changed = False and
                     (cl.title != pl.title implies a'.title_changed = True else a'.title_changed = False) and
                     (cl.summary != pl.summary implies a'.description_changed = True else a'.description_changed = False)
                  ) ) )
      )

   (EbookStoreBookPreviousLoad.rows).s' = (EbookStoreBookCurrentLoad.rows).s
   (EbookStoreBookCurrentLoad.rows).s = none
```

Figura 3 – Fragmento da Especificação do Modelo em Alloy do predicado CDCEbookStoreBooks.

Vejamos, então, como funciona o processo CDC apresentado na Figura 2. Essencialmente, este processo captura as modificações que ocorrem numa fonte de informação **(“**EbookStore”) relativas ao povoamento de uma tabela de dimensão **(“**Book”), utilizando uma abordagem de análise diferencial, dada a natureza da fonte em questão**.** Para isso, o processo começa por copiar a tabela da fonte indicada para uma tabela local (“es_book_current_load”) que é posteriormente comparada com uma segunda tabela (“es_book_previous_load”) copiada anteriormente na última extração realizada. O resultado desta comparação é depois utilizado para atualizar a tabela de auditoria (“audit_dim_book**”**) da respetiva dimensão – através da inserção de registos que refletem a inserção de novas entradas, a deleção ou a atualização de valores de entradas existentes. Para o caso particular desta fonte (“EbookStore”), os seus dados são mapeados de forma bastante direta para a tabela de auditoria da dimensão “Book”. Veja-se, a título de exemplo, os casos**:** books.id →

id , books.title → title, books.summary → description, books.ISBN → ISBN, book.last_update → timestamp**.**

Na Figura 3 podemos ver um fragmento da modelação do predicado "CDCEbookStoreBooks", que modela a operação de captura de dados alterados relativos à dimensão "Books" a partir da fonte "EbookStore", tal como foi descrito anteriormente na exposição do processo apresentado na Figura 2. Este predicado recebe como argumentos um estado inicial e um estado final (nomeadamente, "s" e "s'", elementos da assinatura "State"), e descreve a alteração que se pretende obter com a operação. Neste caso, as tabelas de carregamento atual e de carregamento anterior são modeladas pelas assinaturas "EbookStoreBookCurrentLoad" e "EbookStoreBookPreviousLoad", correspondentemente. A tabela da fonte de dados da qual é realizada a extração é representada pela assinatura "Books" do módulo "EbookStore", e a tabela de auditoria utilizada é "AuditBooks". Estas assinaturas foram modeladas de forma específica. Cada assinatura define uma relação "rows", que relaciona os elementos da assinatura com um estado e um conjunto de elementos de linha específicos – declarados de acordo com a modelação lógica de cada tabela representada. Em Alloy, os predicados podem ser analisados quanto às suas pré-condições, pós-condições e condições de quadro. A pré-condição envolvida no predicado (do processo) CDC em análise refere-se à cópia dos dados na tabela da fonte para a tabela local da área de retenção do SDW. Para tal, na pré-condição define-se que o conjunto de linhas da tabela de carregamento atual no estado inicial é exatamente igual ao conjunto de linhas da tabela da fonte no mesmo estado. Por exemplo, na Figura 3, podemos ver que tal é alcançado com "(EbookStoreBookCurrentLoad.rows).s = (StagingArea/EbookStore/Books.rows).s". Por sua vez, as pós-condições dos predicados em estudo envolvem a alteração do conteúdo da tabela de auditoria, da tabela de carregamento anterior e da tabela de carregamento atual. A tabela de auditoria, no estado final, contém as entradas do estado anterior e novas entradas baseadas nas diferenças entre as tabelas de carregamento. Se "newEntries" simbolizar o conjunto de novas linhas de auditoria calculadas a partir das tabelas de carregamento, "prevStateEntries" simbolizar as entradas na tabela de auditoria do estado anterior e "nStateEntries" representar o conjunto de todas as linhas na tabela de auditoria no estado final, queremos especificar que "nStateEntries=prevStateEntries ∪ newEntries". O conjunto "prevStateEntries" é obtido de forma direta com o acesso às linhas da tabela de auditoria da dimensão no estado inicial - "(AuditBooks.rows).s". O conjunto "newEntries" é o resultado da comparação dos elementos do conjunto "row" de cada tabela de carregamento no estado inicial. Para cada diferença calculada, é declarada a presença de um elemento na relação "rows" da assinatura "AuditBooks" no estado final, que a reflete. A estes elementos são adicionados dados de controlo a nível de cada atributo (por exemplo: "title_changed" com valor booleano). Como parte das pós-condições do predicado, é especificada a cópia do conteúdo da tabela de carregamento atual no estado inicial para a tabela de carregamento anterior no estado final do predicado –

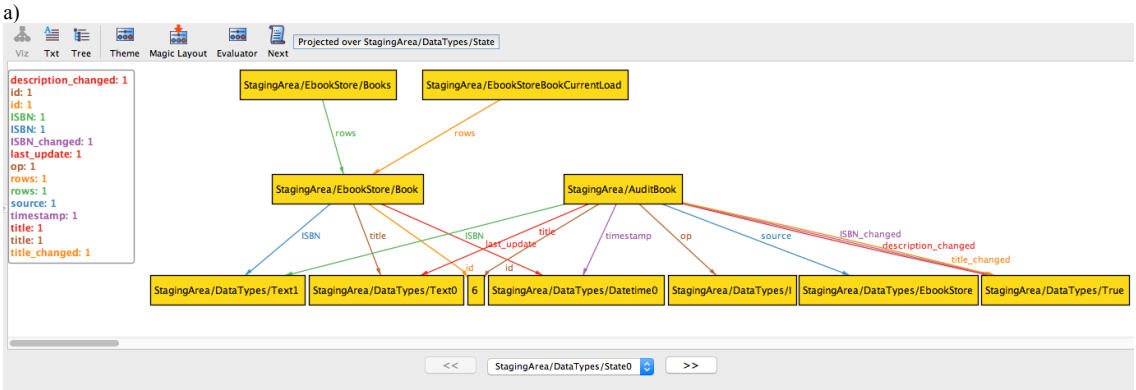
(EbookStoreBookPreviousLoad.rows).s' = (EbookStoreBookCurrentLoad.rows).s". Adicionalmente, é declarado que o conteúdo da tabela de carregamento atual no estado final do predicado é um conjunto vazio.

```
1.  run extractEbookStoreBookExample {
2.      some s: State, s': s.next | CDCEbookStoreBooks[s,s'] and ((EbookStoreBookPreviousLoad.rows).s) != ((EbookStoreBookCurrentLoad.rows).s)
3.  } for 2 but exactly 2 State
```

Figura 4 – Comando de execução para o predicado CDCEbookStoreBooks.

Os predicados de CDC são inicialmente analisados com a execução de um comando 'run' que inclui o predicado. Ao predicado podem ser adicionadas mais restrições no sentido de obter soluções mais significativas. Na Figura 4 podemos ver um exemplo de um comando de execução simples para análise do predicado "CDCEbookStoreBooks". Uma das soluções obtidas com a execução deste predicado apresentada na Figura 5.

a)



b)

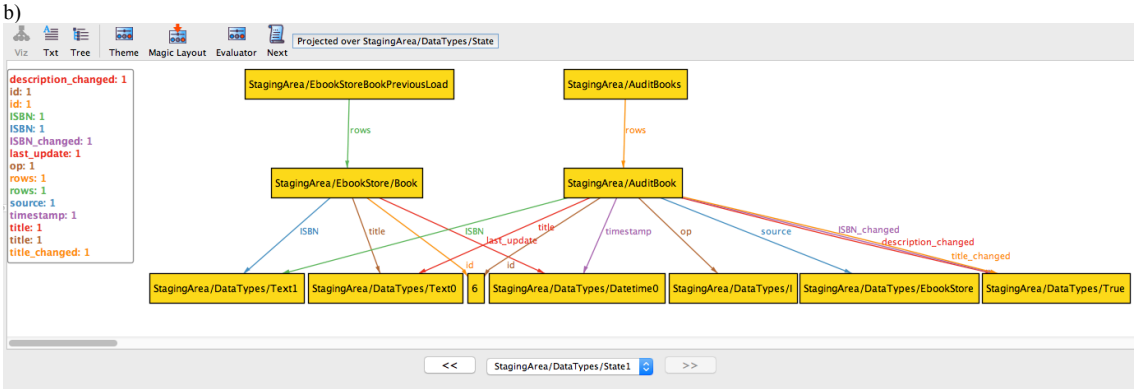


Figura 5 – Visualização de solução de execução do predicado "CDCEbookStoreBooks", projetada no estado anterior – bloco a) – e projetada no estado seguinte – bloco b).

Uma análise mais atenta desta solução permite concluir que, no estado inicial (State0), "EbookStoreBookCurrentLoad" contém o mesmo conjunto de linhas que "Books" da fonte "EbookStore" (um único registo de livro, com id 6). Também aqui, podemos ver que "EbookStoreBookPreviousLoad" não se relaciona com qualquer registo no estado inicial. No estado seguinte, contudo, "EbookStoreBookPreviousLoad" relaciona-se com o registo de um livro (com id 6), e "EbookStoreBookCurrentLoad" tem um conjunto vazio de linhas. A relação "AuditBooks" da área de retenção, que tem um conjunto de linhas vazio no estado inicial, encontra-se relacionada com um registo de auditoria no estado final ("AuditBook"), que é um

registo com uma operação de inserção, com valor de source (fonte) "EbookStore" e ISBN_changed, description_changed e title_changed com valor de True. Os seus atributos têm valores obtidos através de um mapeamento do novo registo de livro (id com valor de 6, title com valor Text0, etc.). Adicionalmente, os predicados são sujeitos a uma validação quanto à sua completude. Com isso, pretende-se verificar que nenhuma alteração de dados é descartada durante o processo: todas as alterações de dados na fonte causam um registo na tabela de auditoria. Para tal, é verificada a implicação de que, para quaisquer dois estados consecutivos, o predicado CDC com estes como argumento implica a existência de um registo na tabela de auditoria para todos os dados que foram inseridos, removidos ou atualizados na extração atual (em relação ao carregamento anterior).

Os exemplos utilizados são apenas uma pequena amostra da especificação geral realizada para o processo de ETL apresentado anteriormente na Figura 1. Foram selecionados de forma a revelar como é possível realizar uma especificação formal de um processo de ETL que, através dos modelos criados, nos permita garantir a correção do sistema e ao mesmo tempo aproximar o nível de conceptualização do nível de implementação, com os altos níveis de rigor e de detalhe que são utilizados na especificação. Porém, de facto, a amostra "peca" por ser escassa. Mas o seu alargamento tornaria este artigo bastante grande.

4. CONCLUSÕES E TRABALHO FUTURO

A modelação de componentes de processos ETL não é uma tarefa fácil. A aplicação de abstrações para simplificar um processo ETL num dado contexto de modelação tem claras vantagens. Porém, algumas as tarefas envolvidas num processo ETL frequentemente exigem grande atenção no seu detalhe e envolvem frequentemente especificidades próprias. Neste trabalho revelámos a utilização da linguagem de especificação formal Alloy para descrever partes – grupos de tarefas - de um processo ETL. A Alloy foi analisada quanto à sua adequação ao desafio de modelação de um processo de ETL, provando ser, na nossa perspetiva, uma boa candidata para a sua especificação e garantia da sua correção e confiabilidade. Os exemplos que escolhemos para analisar a aplicação da linguagem Alloy foram realizados segundo uma abordagem específica, adotando-se metodologias para a descrição dos sistemas (fontes utilizadas, *data warehouse* e área de retenção) de forma simples e realista. Como consequência, foi eliminado algum ruído associado a uma modelação mais genérica, conduzindo a especificações simples e a uma análise fácil de soluções com cardinalidade baixa. Porém, enfrentámos alguns problemas na modelação dos exemplos que escolhemos, em particular com a quantidade total de assinaturas declaradas. As tabelas envolvidas no processo foram definidas como assinaturas *singleton*, o que significa que existe um elemento da assinatura sempre presente no sistema. Este tipo de modelação conduziu a uma representação constante de todas as tabelas utilizadas no modelo, o que aumentou a sua complexidade.

Usualmente, este aumento de complexidade conduz a uma degradação da performance das análises executadas, especialmente visível na ausência de soluções de uma execução completa para o processo ETL. Além disso, com o grau de abstração utilizado nas especificações, perdeu-se a capacidade de modelar algumas operações atómicas, nomeadamente algumas transformações textuais como a concatenação de atributos utilizada como uma forma de conformação inicial e a capitalização de valores utilizada em alguns dos exemplos. Como consequência, não conseguimos realizar a sua especificação. O desenvolvimento dos vários exemplos segundo uma abordagem específica originou um modelo não reutilizável. No entanto, a modelação obtida desmascara semelhanças entre tarefas aplicadas em diferentes contextos, como é o caso de tarefas associadas a duas dimensões similares. Estas semelhanças revelam que uma modelação genérica das tarefas seria um avanço com grande valor e algo a explorar numa linha de trabalho a desenvolver futuramente. A especificação genérica de tarefas ETL pode tomar como base os predicados modelados de forma específica. No entanto, esforços realizados na generalização dos mesmos revelam que a análise resultante é bastante mais complexa e ruidosa. Além disso, a generalização da modelação não resolve o problema de complexidade encontrado. Uma das maiores dificuldades com a utilização da linguagem Alloy relaciona-se com a descrição do estado final de uma operação a nível das tabelas que se utilizam no processo. As operações desenhadas a nível mais atómico (por exemplo, sendo aplicada a uma linha) aproximam-se de uma abordagem processual, usualmente associada às tarefas realizadas. A descrição do resultado de uma operação a nível de tabelas é um exercício mais complexo, uma vez que é necessário ter em consideração o resultado final de diversas manipulações. Todos estes aspetos estão devidamente anotados para se estudar e avaliar a sua aplicação futuramente na especificação formal de modelos de processos ETL genéricos com Alloy.

AGRADECIMENTOS

Este trabalho foi suportado pelo COMPETE: POCI-01-0145-FEDER-007043, by FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2013.

REFERÊNCIAS

Akkaoui, Z., Zimányi, E., (2009). Defining ETL worfklows using BPMN and BPEL. Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP, (pp. 41-48).

Andoni, A., Daniliuc, D., Khurshid, S., & Marinov, D. (2003). Evaluating the "small scope hypothesis." *Unpublished*. Available at http://citeseerx.ist.psu.edu/viewdoc/download?doi =10.1.1.72.4000&rep=rep1&type=pdf [Accessed May 25, 2018]

English, L. P., 1999. Improving Data Warehouse and Business Information Quality: Methods for reducing costs and increasing profits. New York, USA: John Wiley & Sons, Inc.

Inmon, W. H., 2005. Building the Data Warehouse, 4th Edition, Wiley.

Jackson, D., (2012). Software Abstractions: Logic, Language and Analysis. The MIT Press.

Kimball, R., Ross, M., 2002. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, Wiley.

Kimball, R., Caserta, J., 2004. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data, Wiley.

Oliveira, B., Belo, O., (2013). Approaching ETL conceptual modelling and validation using BPMN and BPEL. 2nd International Conference on Data Management Technologies and Applications. July. DATA'2013.

Oliveira, B., Belo, O., Macedo, N., 2016. Towards a Formal Validation of ETL Patterns Behaviour. International Conference on Model and Data Engineering (pp. 156-165). Springer International Publishing.

Oliveira, B., Belo, O., 2017. Validating the Feasability of ETL Patterns Using Alloy, In Proceedings of 6th International Conference on Data Management Technologies and Applications (DATA'2017), Madrid, Spain, July, 24-26.

Sannella, D., Wirsing, M., (1999). Specification Languages. Em E. Astesiano, H. Kreowski, & B. Krieg-Bruckner (Edits.), Algebraic Foundations of Systems Specification (pp. 243-272). Springer Berlin Heidelberg.

Santos, V., (2015). A Relational Algebra Approach to ETL Modeling. PhD Thesis, University of Minho.

Spivey, J. M., (1989). An Introduction to Z and Formal Specifications. Software Engineering Journal, 4(1), 40-50.

Trujillo, J., Luján-Mora, S., (2003). A UML Based Approach for Modeling ETL Processes in Data Warehouses. Conceptual Modeling . International Conference on Conceptual Modeling (pp. 307-320). Springer Berling Heidelberg.

Vassiliadis, P., Simitsis, A., (2002). Conceptual modeling for ETL processes. Proceedings of the Fifth ACM International Workshop on Data Warehousing and OLAP (pp. 14-21). ACM.