Esquemas JSON e XML para a WebSTAMP

Gustavo Henrique Santiago da Silva, Rodrigo Martins Pagliares

Departamento de Ciência da Computação – Universidade Federal de Alfenas (UNIFAL-MG)

Av. Jovino Fernandes Sales, 2600 - Santa Clara – Alfenas – MG – Brasil.

Prédio C, 3º andar - CEP: 37.133.840

gustavo.santiago@sou.unifal-mg.edu.br, pagliares@bcc.unifal-mg.edu.br

RESUMO

A WebSTAMP é uma ferramenta em *software* para auxiliar analistas de *safety* na confecção de análises de *hazards* (condições de perigo) usando a técnica STPA (*Systems-Theoretic Process Analysis*). A WebSTAMP não possui um esquema formalizado que permita importar e exportar análises de *hazards* compatíveis com este esquema. Isto impede que analistas de *safety* utilizem análises feitas na WebSTAMP em outras ferramentas e usem análises exportadas de outras ferramentas na WebSTAMP. Este trabalho tem como objetivos (i) definir um esquema XML (*Extensible Markup Language*) e; (ii) definir um esquema JSON (*JavaScript Object Notation*) para a WebSTAMP, permitindo portabilidade de análises de *hazards* entre ferramentas de *software* que suportem os esquemas. Definimos os esquemas para XML e JSON usando XSD (*XML Schema Definition*) e JSON *Schema*, respectivamente. Os esquemas foram utilizados na WebSTAMP e em uma segunda ferramenta para validar as funcionalidades de importação e exportação de análises de *hazards* em XML e JSON a partir de exemplos presentes na literatura. Os esquemas criados para a WebSTAMP proporcionam portabilidade às análises de *hazards*, tornando possível usar as análises feitas na WebSTAMP em outras ferramentas que suportam os esquemas e vice-versa. Os resultados deste trabalho permitem concluir que os esquemas para XML e JSON auxiliam analistas de *safety*, tornando mais flexível a tarefa de confecção de análises de *hazards* tendo em vista que proporcionam liberdade na escolha de qual ferramenta usar, incluindo a possibilidade de usar mais de uma ferramenta, explorando os pontos fortes de cada uma.

Palavras-chave: XML; JSON; XSD; JSON *Schema*; STAMP; STPA; Análise de perigo.

ABSTRACT

WebSTAMP is a software tool to assist safety analysts in the application of the STPA (Systems-Theoretic Process Analysis) technique for the analysis of hazards. WebSTAMP does not have a formalized schema that allows importing and exporting hazard analysis that adhere to this schema. This prevents safety analysts from using analysis created in WebSTAMP in other software tools, and using analysis exported from other software tools in WebSTAMP. This work aims to (i) define an XML (Extensible Markup Language) schema and (ii) define a JSON (JavaScript Object Notation) schema for WebSTAMP allowing hazard analysis portability among software tools that support the schemas. We define the schemas for XML and JSON using XSD (XML Schema Definition) and JSON Schema, respectively. The schemas were used in WebSTAMP and in a second tool in order to validate the import and export features of hazard analysis in XML and JSON by the use of examples presented in the literature. The schemas created for WebSTAMP provide portability to hazard analysis, making it possible to use the analysis made in WebSTAMP in other applications that support the schemas, and vice versa. The results of this work allow us to conclude that the schemas for XML and JSON aid safety analysts, making their task of conducting hazard analysis more flexible, since they allow safety analysts to use the tools of their choice, including the possibility to use more than one tool, leveraging the strengths of each one.

Keywords: XML; JSON; XSD; JSON Schema; STAMP; STPA; Hazard Analysis.

1 INTRODUÇÃO

Os sistemas sócio-técnicos complexos desenvolvidos nos dias atuais impõem desafios para analistas de *safety* na definição de requisitos de segurança que, ao serem implementados, podem evitar perdas inaceitáveis, tais como perdas de vidas humanas, problemas econômicos, ou dano ao meio-ambiente (LEVESON; THOMAS, 2018). A identificação de requisitos de *safety* passa pela análise de condições de perigo (*hazards*) que devem ser controladas durante o desenvolvimento e operação do sistema.

Um *hazard* é um estado ou conjunto de condições de um sistema que, em combinação com um conjunto particular de piores condições ambientais, pode ocasionar perdas (acidentes) (LEVESON; THOMAS, 2018). Duas aeronaves não respeitando uma separação mínima entre elas durante voo é um exemplo de *hazard*, já que em combinação com outros fatores, tais como problemas de comunicação entre os pilotos, tripulações, controladores de tráfego aéreo, ou baixa visibilidade causada por mau tempo, podem ocasionar perdas inaceitáveis.

STPA (*System-Theoretic Process Analysis*) é uma técnica para análise de *hazards* (LEVESON; THOMAS, 2018) baseada no modelo de causalidade de acidentes conhecido como STAMP (*System-Theoretic Accident Model and Processes*) (LEVESON, 2012). A STPA pode ser usada tanto por analistas de *safety* quanto por analistas de *security*.

Devido à complexidade e tamanho dos sistemas analisados com STAMP/STPA, ferramentas em *software* se tornam imprescindíveis no auxílio a analistas de *safety* na confecção de análises de *hazards*. Existem ferramentas de *software* que fornecem suporte ao modelo STAMP e a técnica STPA: SafetyHAT (BECKER; HOMMES, 2014), XSTAMPP (ABDULKHALEQ; WAGNER. 2015), STAMP Workbench (IPA, 2018) e WebSTAMP (SOUZA et. al., 2019).

Dentre as ferramentas em *software* citadas, a *WebSTAMP* é uma ferramenta *web* que fornece aos analistas de *safety* requisitos essenciais para análises de *hazards*, bem como sistematização e automação de determinados passos da STPA. Tivemos acesso ao código fonte da WebSTAMP e suporte de seus mantenedores em dúvidas sobre seu uso e sobre o seu código-fonte. Dessa forma, optamos por usar a WebSTAMP como meio para atingir os objetivos deste trabalho.

A WebSTAMP não possui um esquema formalizado que permita importar e exportar análises de *hazards*. Isto impede que analistas de *safety* utilizem análises feitas na WebSTAMP em outras ferramentas e usem análises exportadas de outras ferramentas na WebSTAMP.

Este trabalho tem como objetivos: (i) definir um esquema XML (*Extensible Markup Language*) e; (ii) definir um esquema JSON (*JavaScript Object Notation*) para a WebSTAMP. Os esquemas permitem a portabilidade de análises de *hazards* entre ferramentas de *software*

compatíveis com os esquemas criados. Definimos o esquema para XML usando XSD, enquanto que para o esquema JSON utilizamos JSON *Schema*.

Todos os elementos do domínio STAMP/STPA suportados pela WebSTAMP são contemplados nos esquemas criados, permitindo representar qualquer análise de *hazards* criada com a WebSTAMP e capacitando implementar as funcionalidades de importação e exportação de análises de *hazards* entre ferramentas que suportem os esquemas criados.

Por fim, para validar a portabilidade de análises de *hazards* entre ferramentas, usamos exemplos de análises de hazards presentes na literatura, tais como: bomba de insulina (SOUZA et. al, 2019) e sistema de porta de trem (LEVESON, 2012). Detalhes sobre a validação dos esquemas criados podem ser encontrados na Seção 6.

O restante deste trabalho está organizado da seguinte maneira: A Seção 2 apresenta o modelo STAMP e a técnica STPA, cujos conceitos são implementados na WebSTAMP. Na Seção 3, apresentamos uma revisão da literatura sobre técnicas, linguagens e esquemas que podem ser usados para representação de análises de *hazards* com STPA. Apresentamos os trabalhos relacionados na Seção 4. Na Seção 5 são apresentados os esquemas JSON e XML, principais contribuições deste trabalho. A validação dos esquemas JSON e XML a partir de exemplos disponíveis na literatura é discutida na Seção 6. Por fim, resultados, discussões, conclusões e trabalhos futuros são apresentados na Seção 7.

2 STAMP/STPA

STAMP é um modelo de causalidade de acidentes baseado na teoria de sistemas (BERTALANFFY, 1969). O modelo STAMP tem como objetivo prevenir e explicar as causas de acidentes em sistemas grandes e complexos a partir do estabelecimento de controles e imposição de restrições de segurança (*safety*) que mantenham os sistemas livres de acidentes.

O modelo STAMP é fundamentado em três conceitos básicos: restrições, níveis hierárquicos de controle e modelo de processo. As restrições limitam a operação do sistema de forma a impedi-lo de migrar para um estado de alto risco. Os níveis hierárquicos constituem estruturas de controle na qual componentes de nível superior impõem restrições aos de nível inferior. Já o modelo de processo é um modelo do processo controlado.

STPA é uma técnica baseada em STAMP para análise de *hazards*. STPA possui quatro passos: (i) definir o objetivo da análise; (ii) modelar a estrutura de controle; (iii) identificar as ações de controle inseguras e; (iv) identificar os cenários de perdas.

No primeiro passo, definem-se os objetivos, limites do sistema, as perdas (acidentes) que o sistema deve prevenir, *hazards* em nível de sistema associados às perdas e, por fim, restrições de segurança (*safety constraints*) em nível de sistema associadas aos *hazards*.

Uma estrutura hierárquica de controle do sistema é criada no segundo passo de forma a impor as restrições de segurança identificadas no primeiro passo e a controlar o sistema impedindo o surgimento de *hazards*. A estrutura de controle detém relacionamentos e interações funcionais entre os componentes do sistema.

No terceiro passo são analisadas as ações de controle para cada controlador presente na estrutura de controle criada no passo 2. Um controlador é todo componente da estrutura de controle que impõe ações de controle a outro componente do sistema. Esta análise tem como objetivo identificar os contextos nos quais tais ações de controle se tornam inseguras (*unsafe control actions*) para posteriormente criar as restrições de segurança em nível de componente.

O quarto passo permite identificar os cenários e fatores causais que justificam a execução ou omissão de ações de controle inseguras por cada componente da estrutura de controle, além de permitir a geração de casos de testes, recomendações de *design*, requisitos e procedimentos que devem ser atendidos no *design*, desenvolvimento e operação do sistema analisado ou construído.

3 REVISÃO DA LITERATURA

Dentre as técnicas que podem ser usadas no desenvolvimento de um esquema formalizado destacam-se: ontologias (VERDONCK et al., 2019), UML *Profiles* (OMG, 2021b), meta-modelos (OMG, 2021a), DSL's (*Domain-Specific Languages*) (FOWLER, 2010), XSD (W3C, 2012b) e JSON Schema (PEZOA et. al., 2016).

Ontologia é uma técnica de organização de informação baseada na criação de modelos que definem formalmente categorias, propriedades, relações e entidades de um determinado domínio. O objetivo principal de uma ontologia é a representação formal e compartilhamento de conhecimento (VERDONCK et al., 2019).

UML *Profiles* são utilizados em casos nos quais os elementos padrões UML não são apropriados para modelar uma aplicação, devido ao propósito geral da linguagem UML. Dessa forma, UML *Profiles* são extensões que permitem especializar elementos UML, permitindo a personalização da UML para domínios específicos (OMG, 2021b).

Um metamodelo é um modelo que descreve outro modelo como instância, isto é, um metamodelo define por meio de metadados as regras, estrutura e sintaxe de dados de como criar classes de um domínio específico (OMG, 2021a).

DSL's são linguagens confeccionadas para um domínio específico. Ou seja, uma DSL é uma gramática formal bem definida, cujo objetivo é resolver problemas de um domínio específico. Alguns exemplos de DSL's: CSS, *makefile*, SQL, entre outras (FOWLER, 2010).

XSD e JSON *Schema* tornam possível definir conteúdo, semântica, e a estrutura que instâncias XML e JSON devem seguir. XSD é usado para documentos XML (W3C, 2012b) e JSON Schema para documentos JSON (PEZOA et. al., 2016).

Os esquemas propostos neste trabalho usam XSD e JSON *Schema* para a formalização de esquemas para a WebSTAMP. A escolha destas tecnologias se deu pelo fato de que elas são amplamente utilizadas nos dias atuais para desenvolvimento de aplicações distribuídas corporativas, tanto em aplicações monolíticas quanto em aplicações com arquitetura de microsserviços baseadas em computação nas nuvens (MEGARGEL; SHANKARARAMAN; WALKER, 2020).

4 TRABALHOS RELACIONADOS

A necessidade de abordagens e modelos para a representação do conhecimento e da informação é amplamente reconhecida (FRANK; VLADIMIR; BRUCE, 2008). Diversos modelos de representação de conhecimento são usados como suporte na aplicação de análises de *hazards*.

Rejzek, Krauss e Hilbes (2015) propõem uma abordagem baseada em STPA com UML com a finalidade de promover modelagem de sistemas guiada por segurança (*safety*). Parte dessa abordagem é o desenvolvimento de um perfil UML (*UML profile*) especificado para STPA, visando representar a informação gerada na aplicação da técnica.

Souag et al. (2015) apresentam uma ontologia voltada para o processo de análise de requisitos de segurança (*security*) em conjunto com um ambiente para facilitar o seu uso. É demonstrado que utilizar a ontologia como suporte na análise de requisitos de segurança auxilia positivamente o processo.

Gurgel, Hirata e Bezerra (2015) desenvolveram uma ferramenta *desktop* cujo objetivo é auxiliar analistas de *safety* na execução do terceiro passo da STPA. A ideia central da ferramenta é automatizar a detecção de contextos perigosos por meio de regras definidas pelo analista. Uma das funcionalidades da ferramenta é utilizar XML para salvar o progresso da análise. Para tanto, foi desenvolvido um XSD para formalizar o conhecimento gerado pela ferramenta.

Krauss et. al. (2016) integram a metodologia STPA com um ambiente UML, promovendo uma modelagem de sistema guiada por segurança (*safety*). Dessa forma, é desenvolvida pelos autores a extensão SAHRA (*STPA based Hazard and Risk Analysis*) utilizando a ferramenta de modelagem *Sparx Systems Enterprise Architect* (EA) (SPARX SYSTEMS, 2021). A extensão SAHRA inclui uma DSL (*Domain Specific Language*) para STPA que fornece diagramas, caixa de ferramentas e perfis UML para representação do conhecimento de análises de segurança.

Rosa, Jino e Bonacin (2018) apresentam uma ontologia denominada SecAOnto (*Security Assessment Ontology*) que visa formalizar os conceitos e informações da análise de segurança (*security*). A SecAOnto é baseada em OWL (*Ontology Web Language*) (W3C, 2012) e seu conteúdo vem de glossários, vocabulários, taxonomias, outras ontologias e diretrizes de mercado. A SecAOnto é desenvolvida de forma a ser extensível, com o intuito de que seja útil para pesquisadores da área de análise de segurança que desejam formalizar conhecimento em seus sistemas, métodos ou técnicas.

Pereira, Hirata e Nadjm-Tehrani (2019) apresentam uma abordagem para auxiliar analistas de *safety* e *security* na realização de análises de segurança. Essa abordagem baseia-se na utilização de uma ontologia como suporte para a técnica de análise de segurança STPA-Sec (YOUNG, 2019). A ontologia desenvolvida pelos autores formaliza o conhecimento que advém da análise de segurança gerada na aplicação da técnica STPA-Sec.

Ahmad et. al. (2018), com a finalidade de auxiliar analistas de *safety* em análises de *hazards*, propõem uma ontologia que formaliza os fundamentos conceituais do modelo

STAMP denominada SHRO (*STAMP Hazard and Risk Ontology*). A ontologia facilita a aplicação de métodos de análise de *hazards* baseados no modelo STAMP. Para avaliação da ontologia, os autores usaram como caso de uso, exemplos reais no domínio de segurança na aviação.

Abdulkhaleq e Wagner (2015) a fim de fornecer suporte a analistas de *safety* e *security* propuseram a ferramenta XSTAMPP. A ferramenta implementada pelos autores atende a execução da técnica STPA em diferentes áreas e pode ser estendida por meio de adição de recursos. Parte da abordagem dos pesquisadores foi a implementação do formato ".hazx" que tem como sua base um XSD compatível com STAMP/STPA, isto é, que representa o conhecimento gerado na aplicação da técnica.

É nesse cenário que esse trabalho se insere, propondo esquemas JSON e XML como via para representar o conhecimento gerado na aplicação da técnica STPA. Nossa abordagem baseia-se em utilizar os esquemas (XML e JSON) implementados para proporcionar portabilidade às análises de *hazards* confeccionadas dentro da WebSTAMP. Para tanto, implementamos as funcionalidades de importação e exportação de análises de *hazards* usando nossos esquemas como base.

5 ESQUEMAS JSON E XML

Esquema é um formato independente de tecnologia que define a estrutura de um domínio de informação. Isto é, um esquema é capaz de representar a semântica de um domínio específico de forma portável em qualquer sistema.

Tendo em vista que nosso intuito é prover portabilidade às análises de *hazards* entre diferentes ferramentas de *software*, optamos por definir esquemas formalizados que representam os dados das análises. Vale ressaltar que apenas um esquema (JSON ou XML) é necessário para atingir o objetivo de portabilidade entre análises, todavia optamos por implementar ambos fornecendo mais flexibilidade para analistas e desenvolvedores de ferramentas, optando pelo esquema que possuem maior familiaridade.

Ambos esquemas (JSON e XML) possuem o mesmo poder representacional. Esquemas XML possuem a vantagem de serem mais difundidos na comunidade acadêmica e pela disponibilidade de trabalhar com metadados que o faz capaz de melhorar a capacidade de

recuperação de informações. Esquemas JSON são mais recentes e documentos aderentes a JSON provocam menos excessos de processamento e/ou armazenamento (*overhead*), pois são menores que documentos XML. Dessa forma, atualmente o formato JSON vem sendo cada vez mais usado em aplicações *web* para troca de informações.

O Quadro 5.1 apresenta um trecho de um esquema JSON (parte esquerda do quadro) e uma possível instanciação deste esquema (parte direita do quadro) compatíveis com um dos componentes presentes na estrutura de controle (Passo 2 da STPA) denominado controlador. Um sistema de porta de trem foi usado como exemplo de domínio (LEVESON, 2021).

Quadro 5.1 - Trecho do esquema JSON criado para um componente da análise de *hazards* e um exemplo (instanciação) de documento JSON compatível ao esquema.

```
{
   "type": "object",
   "required":[
      "id", "name",
      "type", "id_analysis"
   ],
   "properties": {
      "id": {
         "type": "integer"
      },
      "name": {
         "type": "string"
      },
      "type": {
         "type": "string",
         "enum": ["human","auto"]
      }
      "id_analysis": {
         "type": "integer"
      }
   }
}
```

```
{
  "id": 1,
  "name": "Train Door
          Controller",
  "type": "auto",
  "id_analysis": 1
}
```

Fonte: autor da pesquisa, 2022.

O Quadro 5.2 apresenta um trecho de um esquema XML criado (parte esquerda do quadro) e uma possível instanciação deste esquema (parte direita do quadro) para um exemplo de um controlador de uma porta de trem.

Quadro 5.2 - Trecho de um esquema XML criado para um componente da análise de *hazards* e um exemplo (instância) de documento XML compatível ao esquema.

```
01 <xs:schema>
02 <xs:simpleType name="Type_control">
03   <xs:restriction base="xs:string">
04     <xs:enumeration value="human"/>
05     <xs:enumeration value="auto"/>
06   </xs;restriction>
07 </xs:simpleType>
08
09 <xs:element name="controller">
10   <xs:complexType>
11     <xs:sequence>
12       <xs:element name="id"
13         type="xs:integer"/>
14       <xs:element name="name"
15         type="xs:string"/>
16       <xs:element name="type"
17         type="Type_control"/>
18       <xs:element name="id_analysis"
19           type="xs:integer"/>
20     </xs:sequence>
21  </xs:complexType>
22 </xs:element>
23 </xs:schema>
```

```
01 <controller>
02    <id>
03       1
04    </id>
05    <name>
06       Train Door Controller
07    </name>
08    <type>
09       auto
10    </type>
11    <id_analysis>
12       1
13    </id_analysis>
14 </controller>
15
16
17
18
19
20
21
22
23
```

Fonte: autor da pesquisa, 2022.

6 EXEMPLOS DE USO DOS ESQUEMAS JSON E XML

No intuito de validar os esquemas JSON e XML criados, usamos dois exemplos de análise de *hazards* presentes na literatura: controlador de uma porta de trem (LEVESON, 2012) e bomba de insulina (SOUZA et al., 2019).

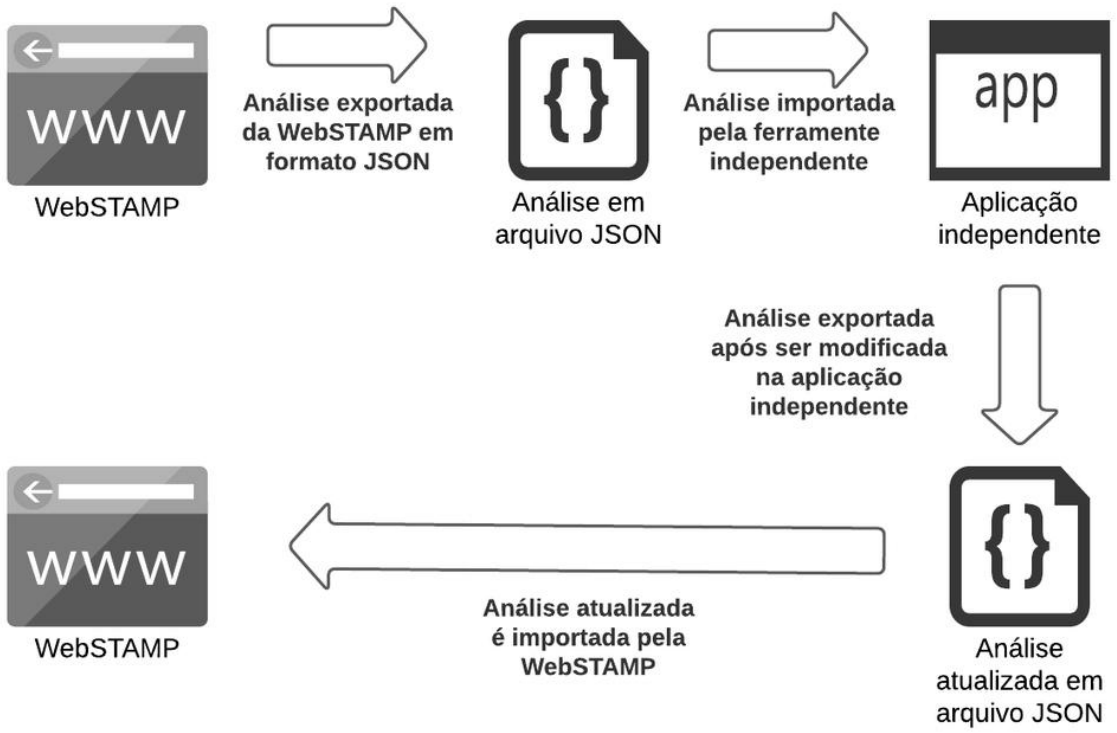
Para tornar possível a validação, modificamos o código fonte da WebSTAMP para incluir as funcionalidades de importação/exportação de análises de *hazards*.

Além disso, criamos uma pequena aplicação *desktop*, independente da WebSTAMP, também com as funcionalidades de importação e exportação de análises de *hazards* para simular a portabilidade das análises entre as duas ferramentas.

A Figura 6.1 descreve os experimentos realizados para validação dos esquemas JSON e XML implementados. Na figura exibimos o esquema JSON como referência, mas raciocínio análogo pode ser usado para o esquema XML. Da mesma forma, a figura apresenta o exemplo de controlador de porta de trem, mas o raciocínio é o mesmo para qualquer análise de *hazards*, incluindo o exemplo de bomba de insulina que usamos para validação dos esquemas JSON e XML.

O cenário descrito na figura 6.1 contempla (iniciando na parte superior à esquerda) uma análise de *hazards* criada na WebSTAMP, exportada no formato JSON, e importada em uma aplicação independente. Uma vez importada pela aplicação independente, a análise de *hazard* é editada e exportada, para ser importada novamente pela WebSTAMP. A análise importada na WebSTAMP contém a edição feita na aplicação independente, demonstrando a portabilidade da análise entre as ferramentas em *software* bem como a flexibilidade concedida a analistas de *safety* no que tange a escolha da ferramenta a utilizar para as análises. Durante o processo de importação em ambas ferramentas, a análise de *hazard* exportada é validada por meio do esquema JSON criado.

Figura 6.1 - Experimento de portabilidade de análise de *hazards* entre as ferramentas de *software* (WebSTAMP e ferramenta criada pelos autores para validação dos esquemas JSON e XML).



Fonte: autor da pesquisa, 2021

No cenário descrito para o experimento, se o arquivo JSON da análise a ser importada estiver em formato válido, os dados são importados e armazenados no banco de dados da WebSTAMP (parte inferior à esquerda na figura) como uma nova análise (projeto).

7 RESULTADOS, DISCUSSÕES, CONCLUSÕES E TRABALHOS FUTUROS

Por meio dos esquemas (XML e JSON) criados, conseguimos proporcionar portabilidade às análises de *hazards* entre a WebSTAMP e uma aplicação *desktop* independente, via funcionalidades de importação e exportação. Salientamos que em ambas as aplicações, as análises (projetos) estavam prontas para serem utilizadas assim que importadas: devidamente persistidas no banco de dados da WebSTAMP, e devidamente armazenadas em memória na aplicação independente.

Trabalhos científicos que utilizam esquemas para representar formalmente o conhecimento gerado pela técnica STPA são escassos. Destacamos a abordagem dos pesquisadores Gurgel, Hirata e Bezerra (2015), já citada na Seção 2. Parte da abordagem dos pesquisadores utiliza XML e XSD para a funcionalidade de salvar o progresso das análises. Entretanto, o trabalho é focado no terceiro passo da STPA, e consequentemente o XSD criado representa formalmente apenas o conhecimento gerado na aplicação do terceiro passo da técnica.

Os esquemas implementados neste trabalho, cobrem todos os elementos presentes nos quatro passos da técnica STPA. Isso permite que nossos esquemas possam ser utilizados em qualquer ferramenta em *software*, técnica ou método voltado para o STPA, desde que suportem os esquemas propostos.

Um dos pontos fracos de nosso trabalho é o alinhamento de nossos esquemas (JSON e XML) com o modelo de domínio STAMP/STPA implementado na WebSTAMP. Não que a WebSTAMP não forneça suporte adequado ao modelo STAMP e a técnica STPA, mas pelo fato de que é possível definições diferentes das implementadas na WebSTAMP acerca de relacionamentos entre os elementos de domínio STAMP/STPA.

Por exemplo, na WebSTAMP, análise de *hazards* são representadas pela abstração `Projeto` (`Project`) e todas demais abstrações estão diretamente relacionadas a `Project`. Planejamos como trabalho futuro minimizar este acoplamento, distribuindo os relacionamentos acoplados diretamente com `Project` entre outros elementos do domínio (por exemplo, `Losses`, `Hazards` e `System Safety Constraints`).

Outra limitação do trabalho é a suposição de que a WebSTAMP é uma implementação de referência STAMP/STPA. Embora a WebSTAMP esteja em contínua evolução, alguns elementos STAMP/STPA ainda não são suportados, como por exemplo o conceito de *sub-hazards* (Passo 1 da STPA) e definição de responsabilidades de controladores (Passo 2 da STPA).

Dessa forma, não temos a pretensão de que os esquemas propostos neste trabalho se tornem um padrão de referência para uso de todas as ferramentas que suportem STAMP/STPA, embora a criação de referido padrão possa ser algo desejável pela comunidade de *safety*.

Apesar disso, podemos concluir que os esquemas propostos auxiliam analistas de *safety*, tornando mais flexível a tarefa de confeccionar análises de *hazards* tendo em vista que proporcionam liberdade na escolha de qual ferramenta usar, incluindo a possibilidade de utilizar mais de uma ferramenta, permitindo explorar os pontos fortes de cada uma (desde que suportem os esquemas propostos). Igualmente os esquemas podem auxiliar desenvolvedores e pesquisadores envolvidos com STAMP/STPA na representação do conhecimento gerado na aplicação de técnica.

8 REFERÊNCIAS

ABDULKHALEQ, A.; WAGNER, S. XSTAMPP: an eXtensible STAMP platform as tool support for safety engineering. [s. l.], mar. 2015.

AHMAD, J.; KOSTOV, B.; LALIS, A.; KREMEN, P. Ontological Foundation of hazards and risks in STAMP. **Semantic Web**, 2018.

BECKER, C.; HOMMES, V. E. Q. Transportation systems safety hazard analysis tool (SafetyHAT) user guide. [s. l.], 24 mar. 2014.

BERTALANFFY, L. V. **General Systems Theory: Foundations, Development, Applications**. New York: George Braziller Inc, 1969.

FOWLER, M. **Domain Specific Languages**. 4. ed. [*S. l.*]: Addison Wesley, 2010.

FRANK, V. H.; VLADIMIR, L.; BRUCE, P. **Handbook of knowledge representation**. [S. l.]: Elsevier, 2008.

GURGEL, D. L.; HIRATA, C. M.; BEZERRA, J. M. A rule-based approach for safety analysis using STAMP/STPA. **2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)**, 2015. DOI 10.1109/DASC.2015.7311464. Disponível em: https://bit.ly/3u5HFqF. Acesso em: 28 mar. 2022.

IPA. **STAMP Workbench**. [*S. l.*], 30 mar. 2018. Disponível em: https://bit.ly/37dj8GW. Acesso em: 30 mar. 2022.

KRAUSS, S. S.; REJZEK, M.; REIF, M.; HILBES, C. Towards a modeling language for Systems-Theoretic Process Analysis (STPA): Proposal for a domain specific language (DSL) for modeldriven Systems-Theoretic Process Analysis (STPA) based on UML. **IAMP Safety Critical Systems Working Papers**, [s. l.], 3 nov. 2016.

LEVESON, N. G. **Engineering a Safer World:** Systems Thinking Applied to Safety. [S. l.: s. n.], 2012. ISBN 9780262298247.

LEVESON, N. G.; THOMAS, J. P. **STPA Handbook**. [S. l.: s. n.], 2018. Disponível em: https://bit.ly/3Kczz5i . Acesso em: 17 nov. 2021.

MEGARGEL, A.; SHANKARARAMAN, V.; WALKER, D. K. Migrating from monoliths to cloud-based microservices: A banking industry example. **Software Engineering in the Era of Cloud Computing**, [s. l.], 2 jan. 2020. DOI 10.1007/978-3-030-33624-0_4. Disponível em: https://ink.library.smu.edu.sg/sis_research/4725/. Acesso em: 31 mar. 2022.

OMG. **MetaObject Facility Specification**. [*S. l.*], 2021a. Disponível em: https://www.omg.org/mof/. Acesso em: 21 mar. 2022.

OMG. **UML Profile Category:** Specifications Associated. [*S. l.*], 2021b. Disponível em: https://www.omg.org/spec/category/uml-profile/About-uml-profile/. Acesso em: 18 set. 2021.

PEREIRA, D. P.; HIRATA, C.; NADJM-TEHRANI, S. A STAMP-based ontology approach to support safety and security analyses. **Journal of Information Security and Applications**, [*s. l.*], ago. 2019. DOI https://doi.org/10.1016/j.jisa.2019.05.014. Disponível em: https://www.sciencedirect.com/science/article/pii/S2214212619302042. Acesso em: 24 jun. 2021.

PEZOA, F.; REUTTER, J. L.; SUÁREZ, F.; UGARTE, M.; VRGOC, D. Foundations of JSON schema. **In Proceedings of the 25th International Conference on World Wide Web** , [s. l.], 11 abr. 2016. DOI https://doi.org/10.1145/2872427.2883029. Disponível em: https://dl.acm.org/doi/10.1145/2872427.2883029. Acesso em: 30 mar. 2022.

REJZEK, M.; KRAUSS, S. S.; HILBES, C. Safety driven design with UML and STPA. In: MIT STAMP WORKSHOP, 4, 2015, Boston MA. **Seminário[...].** Boston MA, 2015. Disponível em: https://bit.ly/3uQV4C0. Acesso em: 26 set. 2021.

ROSA, F. F.; JINO, M.; BONACIN, R. Towards an Ontology of Security Assessment: A Core Model Proposal. **Advances in Intelligent Systems and Computing**, [*s. l.*], 13 abr. 2018. DOI 10.1007/978-3-319-77028-4_12.

SOUAG, A.; SALINESI, C.; MAZO, R.; COMYN-WATTIAU, I. A SECURITY Ontology for Security Requirements Elicitation. **Engineering Secure Software and Systems**, [*s. l.*], 23 jun. 2015. DOI 10.1007/978-3-319-15618-7_13.

SOUZA, F. G. R.; PEREIRA, D. P.; PAGLIARES, R. M.; NADJM-TEHRANI S. WebSTAMP: a Web Application for STPA & STPA-Sec**. MATEC Web of Conferences,** [*s. l.*], 2019. DOI 10.1051/MATECCONF/201927302010.

SPARX SYSTEMS. **Enterprise Architect**. [S. l.], 5 nov. 2021. Disponível em: https://sparxsystems.com/. Acesso em: 22 mar. 2022.

VERDONCK, M.; GAILLY, F.; PERGL, R.; GUIZZARDI, G.; MARTINS, B.; PASTOR, O. Comparing traditional conceptual modeling with ontology-driven conceptual modeling: An empirical study. **Information Systems**, [*s. l.*], 20 mar. 2019. DOI https://doi.org/10.1016/j.is.2018.11.009

W3C. **OWL 2 Web Ontology Language document overview**. 2012a. Disponível em: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/. Acesso em: 24, set. 2021.

W3C. **XML Schema Definition (XSD)**. 2012b. Disponível em: https://bit.ly/37fQLb9. Acesso em: 25, set. 2021.

YOUNG, W. System-Theoretic Process Analysis for Security (STPA-SEC): Cyber security and STPA. In: 2019 STAMP Conference, 2019, Boston MA. **Seminário[...]**. Boston MA, 2019. Disponível em: https://bit.ly/3x2X4JZ. Acesso em: 23 set. 2021.