MicroSTAMP: microserviços para os passos 1 e 2 da técnica STPA

João Hugo Marinho Maimone, Thiago Franco de Carvalho Dias,

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

joao.maimone@sou.unifal-mg.edu.br, thiago.dias@sou.unifal-mg.edu.br, pagliares@bcc.unifal-mg.edu.br

Resumo

STPA (*System-Theoretic Process Analysis*) é uma técnica que auxilia *analistas de safety* na confecção de análises de *hazards* (condições de perigo) em sistemas sociotécnicos. Embora *analistas de safety* estejam normalmente interessados em uma análise de *hazards* completa, a partir da execução dos 4 passos da STPA, artefatos produzidos em cada um dos passos podem ser valorizados de forma independente por outros *stakeholders*, tais como especialistas de domínio, engenheiros de sistemas, desenvolvedores de *software* e projetistas de sistemas. Embora existam algumas soluções em *software* para suporte à STPA, nenhuma delas é desenvolvida com foco em reuso das implementações de cada um dos 4 passos STPA, de forma isolada, entre ferramentas distintas. Este trabalho tem dois objetivos *(i) criar a MicroSTAMP, um conjunto de dois microserviços para suporte parcial à técnica STPA (passos 1 e 2) e (ii) demonstrar que é possível reutilizar os microserviços, integrando a MicroSTAMP com outras ferramentas para suporte integral à STPA (passos 1 a 4)*. Inicialmente, criamos a MicroSTAMP para suporte aos passos 1 e 2 da STPA. Na sequência, demonstramos que é possível integrar a MicroSTAMP a uma segunda ferramenta (WebSTAMP) de forma a fornecer suporte completo aos 4 passos da STPA. Demos o nome de PowerSTAMP o resultado da

integração da MicroSTAMP com a WebSTAMP. Podemos concluir que a MicroSTAMP beneficia diversos *stakeholders*, desde os que possuem interesse nos resultados de passos específicos da STPA (passos 1 e 2) até aqueles que necessitam de análises de *hazards* completas, a partir da aplicação dos 4 passos da técnica.

Palavras-chave: STAMP, STPA, Microserviços, MicroSTAMP, PowerSTAMP.

Abstract

STPA (System-Theoretic Process Analysis) is a technique that aids *safety analysts* in the development of hazard analyses for socio-technical systems. Although *safety analysts* are usually interested in a complete hazard analysis, by performing the 4 steps of the STPA, artifacts produced in each of the steps can be independently valued by other stakeholders, such as *domain experts*, *systems engineers*, *software developers* and *system designers*. Although there are some software solutions to support STPA, none of them are developed focusing on reuse of the implementations of each of the 4 STPA Steps, in isolation, between different tools. This work has two objectives *(i) to create MicroSTAMP, a set of two microservices that partially supports the STPA technique (steps 1 and 2) and (ii) to demonstrate that it is possible to reuse the microservices, by integrating MicroSTAMP with other tools to fully support STPA (steps 1 to 4)*. Initially, we created MicroSTAMP to support STPA steps 1 and 2. Next, we demonstrate that it is possible to integrate MicroSTAMP with a second tool (WebSTAMP) in order to provide full support for the four steps of STPA. We named PowerSTAMP the result of the integration of MicroSTAMP with WebSTAMP. We conclude that MicroSTAMP benefits several stakeholders, from those who are interested in the results of specific steps of the STPA (steps 1 and 2) to those who need a complete hazard analysis, by applying the 4 steps of the STPA technique.

Keywords: STAMP, STPA, Microservices, MicroSTAMP, PowerSTAMP.

1 Introdução

STPA (*System-Theoretic Process Analysis*) é uma técnica que auxilia *analistas de safety* na tarefa de confecção de análise de *hazards* (condições de perigo) e prevenção de perdas (acidentes) em sistemas sociotécnicos (LEVESON; THOMAS, 2018).

A STPA possui 4 passos: *(i) Define Purpose of the Analysis (Identificar o objetivo da análise); (ii) Model the Control Structure (Modelar a estrutura de controle); (iii) Identify Unsafe Control Actions (Identificar ações de controle inseguras); (iv) Identify Loss Scenarios (Identificar cenários de perdas)*. Embora normalmente *stakeholders* como *analistas de safety*, estejam interessados em uma análise completa com resultados de todos os passos, artefatos produzidos em cada um dos passos podem ser valorizados de forma independente por outros *stakeholders*, tais *como especialistas de domínio*, *engenheiros de sistemas*, *projetistas de sistema*, *desenvolvedores* e *testadores de software*, etc.

Especialistas de domínio, por exemplo, podem estar mais interessados nos objetivos de uma análise de *hazards* (definidos no *Passo 1* da STPA). *Engenheiros de sistemas* podem estar mais preocupados em definir a estrutura de controle responsável por impor as restrições de *safety* no sistema (definida no *Passo 2* da STPA). *Projetistas de sistemas, desenvolvedores e testadores de software* usam as restrições de *safety* produzidas pelo *Passo 3* da STPA para se protegerem de contextos pelos quais uma ação de controle pode ser insegura. *Desenvolvedores de software* e *engenheiros de sistemas*, por exemplo, podem estar mais interessados nos requisitos e recomendações de *design* produzidos pela STPA a partir de cenários gerados no *Passo 4*.

A confecção de uma análise de *hazards* com STPA é uma tarefa complexa que gera muita documentação (normalmente necessária) por diversos motivos, tais como restrições legais, regulamentações governamentais, auditoria e certificação de sistemas. Dessa forma, os *analistas de safety* precisam de ferramentas de *software* para organizar a documentação, lidar com a complexidade e direcionar e sistematizar o processo de criação de uma análise de *hazards.*

Embora existam algumas soluções em *software* para suporte à STPA, nenhuma delas é desenvolvida com foco em reuso das implementações, de forma isolada, de cada um dos seus 4 passos entre ferramentas distintas. O não foco em reuso impede, por exemplo, que partes da análise de *hazards* sejam feitas em ferramentas distintas, não permitindo que analistas se beneficiem dos pontos fortes de cada ferramenta.

O reuso de partes individuais de uma análise de *hazards* com STPA é algo desejável já que, de maneira geral, com base na experiência dos autores deste artigo, não há ferramenta melhor ou pior para suporte completo à STPA. Algumas são melhores em alguns passos da STPA, enquanto outras oferecem melhor suporte para outros passos. Além disso, algumas ferramentas são aplicações *web*, já outras, aplicações *desktop*.

A XSTAMPP (ABDULKHALEQ; WAGNER, 2015) por exemplo, permite modelarmos estruturas de controle (*Passo 2* da STPA) de forma simplificada, "arrastando e soltando" ("*drag and drop*") componentes. Já na WebSTAMP (SOUZA *et al.*, 2019), o suporte ao *Passo 2* é fornecido via seleção de componentes em formulários *web* (o que normalmente é feito após *analistas de safet*y terem modelado a estrutura de controle em outra ferramenta, já que é mais fácil modelar estruturas de controle usando componentes visuais do que usando formulários *web*). Outro exemplo, é que a WebSTAMP suporta um conceito de regras diferente da XSTAMPP no *Passo 3* da STPA.

O fato de não existir uma ferramenta melhor ou pior para suporte completo à STPA é um dos motivos pelo qual *analistas de safety* acabam por utilizar mais de uma ferramenta para análises de *hazards*. Ao se utilizar mais de uma ferramenta, *analistas de safety* possuem o trabalho adicional de integrar e/ou mapear todas as partes das análises feitas em ferramentas distintas, se responsabilizando em manter a rastreabilidade entre os elementos presentes em passos distintos (por exemplo, de uma responsabilidade de um controlador em uma estrutura de controle identificada no *Passo 2*, deve ser possível navegar para uma restrição de *safety* do *Passo 1* a qual esta responsabilidade tenta impor).

Um dos motivos da dificuldade de reuso de funcionalidades individuais é o fato de que as ferramentas de *software* existentes para suporte à STPA se baseiam em arquiteturas monolíticas, construídas ao redor do conceito de camadas, com alto grau de acoplamento entre os componentes presentes em cada uma delas(MARTIN *et al.*,2017). Uma das vantagens reivindicadas no uso de arquiteturas de microserviços é minimizar problemas de alto acoplamento e baixo reuso presente em aplicações monolíticas (NEWMAN, 2021).

Dessa forma, o uso de uma arquitetura de microserviços para implementação de funcionalidades de cada um dos 4 passos de forma isolada tem o potencial de facilitar o reuso de implementações dos serviços, permitindo uso individual de cada um para suporte parcial à STPA, ou uso integrado de microserviços (possivelmente desenvolvidos por organizações/indivíduos independentes) para suporte integral à técnica.

Este trabalho tem dois objetivos *(i) criar a MicroSTAMP, um conjunto de dois microserviços para suporte parcial à técnica STPA (passos 1 e 2) e (ii) demonstrar que é possível reutilizar os microserviços, integrando a MicroSTAMP com outras ferramentas para suporte integral à STPA (passos 1 a 4)*.

Os microserviços da MicroSTAMP (*objetivo i*) podem ser usados individualmente ou em conjunto. Dessa forma, a MicroSTAMP fornece suporte parcial à técnica STPA. Dos 4 passos da técnica STPA, a MicroSTAMP suporta os dois primeiros a partir de um microserviço para o *Passo 1* (*Step 1 - Define the purpose of the analysis*) e outro microserviço para o *Passo 2* (*Step 2 - Model a control structure*). A realização do *objetivo ii* (detalhes na Seção 4) permite suporte completo à técnica STPA.

Para demonstração do potencial de reuso e minimização de acoplamento reivindicado por arquiteturas de microserviços, fizemos a integração da MicroSTAMP com uma ferramenta *web* para STPA (WebSTAMP). A WebSTAMP é uma aplicação monolítica, implementada com o conceito de camadas. Demos o nome de PowerSTAMP, o resultado da integração entre a MicroSTAMP e a WebSTAMP para análises de *hazards* mais completas do que as geradas pela MicroSTAMP de forma isolada.

Este trabalho está organizado da seguinte maneira: A Seção 2 apresenta os trabalhos relacionados. Uma breve introdução a microserviços e detalhes sobre a MicroSTAMP são descritos na Seção 3. A Seção 4 apresenta a PowerSTAMP. Os resultados estão presentes na Seção 5. As Seções 6 e 7 apresentam, respectivamente, discussões sobre o desenvolvimento deste trabalho e as conclusões e trabalhos futuros.

2 Trabalhos relacionados

Dentro dos trabalhos relacionados com o nosso, destacamos algumas ferramentas de *software* que fornecem suporte à STPA.

A SafetyHAT, *Safety Hazard Analysis Tool* (BECKER; VAN EIKEMA HOMMES, 2014) direciona os *analistas de safety* na identificação de ações de controle inseguras (*Passo 3*) e de fatores causais (*Passo 4*). O direcionamento se dá a partir do fornecimento de frases-guia (*guide phrases*) associadas com sistemas da área de transporte que facilitam a confecção da análise de *hazards*. Dentre as características da SafetyHAT, destacam-se a rastreabilidade entre artefatos e geração de documentação passível de ser auditável. Uma limitação da ferramenta reside no fato da SafetyHat possuir as *guide phrases* e as *guide words* apenas para sistemas da área de transporte, ou seja, não podem ser aproveitadas em outras áreas de domínio/aplicação.

A XSTAMPP (ABDULKHALEQ; WAGNER, 2015) é uma plataforma de código aberto, baseada no *Eclipse Plug-in Development Environment* (PDE) (BLEWITT, 2016). O objetivo geral da XSTAMPP é desenvolver uma plataforma extensível para suporte ao modelo STAMP (*System Theoretic Accident Model and Processes*) (LEVESON, 2011) e suas técnicas. A XSTAMPP pode ser usada por *analistas de safety* em diferentes áreas de domínio/aplicação e atualmente suporta as técnicas STPA, CAST (*Causal Analysis based on System Theory*) (LEVESON, 2019) e STPA-Sec (*STPA for security*) (YOUNG; LEVESON, 2014).

A RM Studio (RISK MANAGEMENT STUDIO, 2022) é uma ferramenta que possui um módulo com suporte a STPA, que pode ser usado de forma isolada ou

integrado com outros módulos da RM Studio, fornecendo soluções para a área de gestão de riscos.

SAHRA, *STPA based Hazard and Risk Analysis* (REJZEK; KRAUSS; HILBES, 2015), é uma extensão da ferramenta *Enterprise Architect* (SPARX SYSTEMS PTY LTD, 2022) que integra STPA em um ambiente de modelagem UML/SysML. A SAHRA fornece ferramentas, perfis UML, padrões e *templates*, além de suporte a editores específicos para modelagem com STPA.

A CAIRIS (*Computer Aided Integration of Requirements and Information Security*) (ALTAF *et al.*, 2020) é uma plataforma de código aberto para análise de *safety*, *security* e usabilidade em sistemas. A plataforma oferece suporte a *analistas de safety* permitindo rastreabilidade automática entre elementos STPA, geração automática de modelos visuais, documentação e suporte para identificar e validar cenários de perdas. CAIRIS possui interoperabilidade com o *software* Microsoft Excel permitindo importar planilhas *Excel* em modelos STPA e vice-versa.

A STAMP Workbench (IPA, 2018) é uma ferramenta de código aberto, desenvolvida com a linguagem de programação Java, para suporte à STAMP/STPA. Uma das funcionalidades previstas na STPA Workbench é a possibilidade de exportar diagramas criados pela ferramenta no formato de imagens vetoriais, como por exemplo, o formato SVG (*Scalable Vector Graphics*).

A WebSTAMP (SOUZA *et al.*, 2019) é uma uma aplicação *web*, em constante evolução, compatível com o modelo STAMP e as técnicas STPA e STPA-Sec. A WebSTAMP tem como objetivo auxiliar os *analistas de safety e security* nos 4 passos da técnica STPA, direcionando e automatizando a confecção de análises de *hazards*.

Conforme discutido na Seção 1, nenhuma das ferramentas apresentadas nesta seção é desenvolvida com foco em reuso das implementações, de forma isolada, de cada um dos passos da STPA entre ferramentas distintas.

3 MicroSTAMP: microserviços para STPA

A MicroSTAMP é uma aplicação *web* constituída de dois microserviços para suporte aos passos 1 e 2, respectivamente. Os microserviços foram implementados com o *framework* MVC (*Model*, *View*, *Controller*) conhecido como Spring Boot (MUSIB, 2022).

Cada microserviço da MicroSTAMP trabalha individualmente, de forma independente, respondendo a requisições HTTP (*Hypertext Transfer Protocol*) baseadas no estilo arquitetural REST (*Representational State Transfer*) e gerenciando a persistência de dados em banco de dados relacionais.

Com RESTful, requisições e respostas transportam dados de maneira padronizada no formato JSON (*JavaScript Object Notation*). O uso de um formato padrão para comunicação (JSON) permite por exemplo, que microserviços se comuniquem mesmo que implementados em linguagens de programação distintas (Isto é demonstrado na Seção 4 em que os microserviços da MicroSTAMP, escritos com Spring Boot na linguagem Java, são integrados com a ferramenta WebSTAMP implementada em Laravel (GRIFFIN, 2020), um framework PHP de código aberto para a criação de aplicações web).

3.1 Um microserviço para o *Passo 1* da STPA: Definir o objetivo do sistema

A MicroSTAMP suporta na íntegra o *Passo 1*, com *endpoints* RESTful para cada um dos artefatos necessários para este passo (objetivos do sistema - *system goals*, perdas - *losses*, condições de perigo em nível de sistema - *system level hazards* e restrições de *safety* em nível de sistema - *system safety constraints*). Também implementamos o conceito de suposições (*assumptions*), uma particularidade da WebSTAMP (SOUZA *et al.*, 2019) que pode ajudar a enriquecer a análise.

O *Passo 1* da MicroSTAMP também trabalha com uma entidade `Project` (não presente na STPA) que armazena todos os artefatos citados acima, e possibilita ao *analista de safety* gerenciar múltiplos projetos diferentes.

Uma vantagem do *Passo 1* da MicroSTAMP em relação às ferramentas similares apresentadas na Seção 2 é o refinamento de *hazards* em *sub-hazards* (LEVESON; THOMAS, 2018), uma etapa opcional do *Passo 1* da STPA que pode ser muito útil em análises mais complexas, ajudando a guiar o desenvolvedor nos passos subsequentes.

3.2 Um microserviço para o *Passo 2* da STPA: Modelar a estrutura de controle

O microserviço da MicroSTAMP para o *Passo 2* permite a modelagem de estruturas de controle. O microserviço define uma entidade nomeada `Component`. A partir da entidade `Component`, derivamos os cinco tipos de componentes utilizados em análises STPA: `Actuator`, `Controller`, `ControlledProcess`, `Sensor` e `Environment`.

A entidade `Component` possui um campo `father` que representa o "pai" do componente em questão. Este campo possibilita a criação de estruturas aninhadas de múltiplos *layers*, com um componente pai possuindo `N` componentes filhos em um nível inferior a ele, como ilustrado na Figura 3.1, na qual o componente sistema de freio (*Braking System*) é filho de veículo físico (*Physical Vehicle*). Outra forma de visualizar a relação entre componentes pai e filho é observar na parte superior da Figura 3.1 (Seção *Components*) que filhos aparecem aninhados em relação a seus pais.

Para todo `Component` do tipo `Controller`, também é possível estabelecer múltiplas `Responsibilities`, que são responsabilidades que os componentes devem ter para satisfazer as *System Safety Constraints* definidas no *Passo 1* (LEVESON; THOMAS, 2018).

MicroSTAMP Step 2

Components

Collapse All

Name	Border	is visible?	Type
Driver	SOLID	true	Controller
Auto-Hold Module	SOLID	true	Controller
▼ Physical Vehicle	DASHED	true	ControlledProcess
Braking System	SOLID	true	ControlledProcess
Propulsion System	SOLID	true	ControlledProcess

Connections

Collapse All

Source	Target	Connection Type	Style
Environment	Physical Vehicle	PROCESS_INPUT	SOLID
Physical Vehicle	Environment	PROCESS_OUTPUT	SOLID
▼ Environment	Driver	PROCESS_INPUT	SOLID
Visual cues			LABEL
Physical feedback			LABEL
▶ Auto-Hold Module	Driver	FEEDBACK	SOLID
▼ Driver	Propulsion System	CONTROL_ACTION	SOLID
Accelerate			LABEL

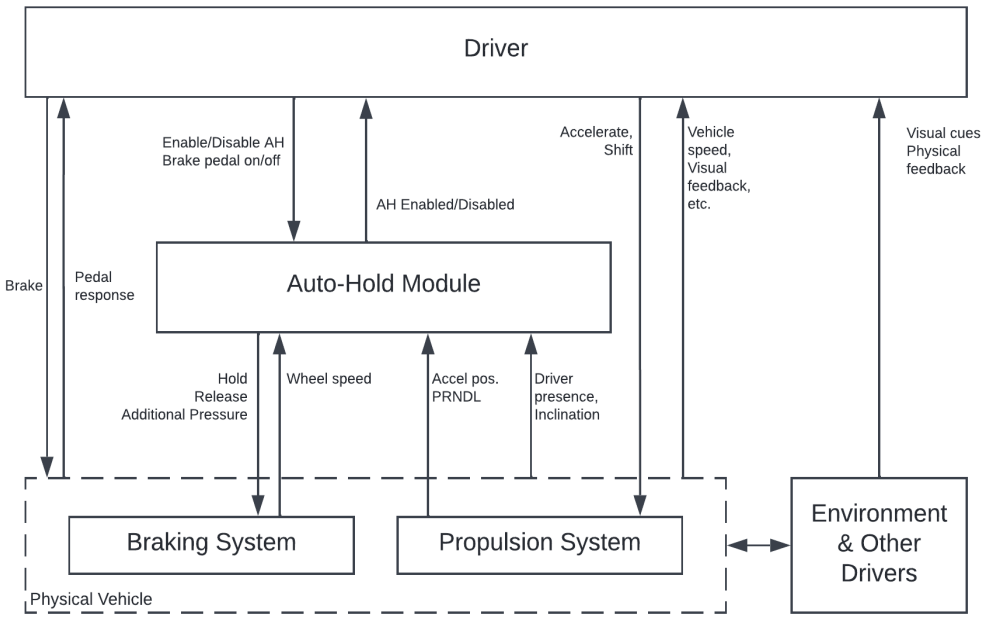

Figura 3.1 - Tela do *Passo 2* da MicroSTAMP (LEVESON; THOMAS, 2018)

Para as conexões entre componentes (`Component`) utilizamos uma entidade chamada `Connection`, que permite criar "*feedback loops"* de controle entre componentes distintos, além de fornecer o transporte de informações e rastreabilidade entre os múltiplos componentes criados.

Uma `Connection` é uma associação entre dois objetos do tipo `Component`: um `Component` como origem (*source*) e um segundo `Component` como destino (*target*). Ela possui um tipo, podendo ser uma `ControlAction`, um `Feedback`, ou um `CommunicationChannel` e pode possuir vários rótulos (*labels*), conforme pode ser visto na Figura 3.1, onde os rótulos "*Visual cues*" e "*Physical feedback*" estão vinculados à `Connection` entre o ambiente (*Environment*) e o motorista (*Driver*).

Além dos componentes citados acima, também temos um componente especial criado por *default* para qualquer estrutura criada no *Passo 2* da MicroSTAMP, designado `Environment`. Este componente tem o intuito de representar interações (entradas e saídas) entre o ambiente e o sistema sendo analisado. Ao utilizar o componente `Environment` como entrada para os sistema via uma `Connection`, usamos conexões do tipo `ProcessInput` ou `Disturbance`, e caso seja utilizado como saída, usamos uma conexão do tipo `ProcessOutput`, (LEVESON; THOMAS, 2018)

O *Passo 2* da MicroSTAMP também possibilita a criação de variáveis para todo `Component` do tipo `Controller` ou `ControlledProcess`. Estas variáveis também podem ter `N` estados vinculados a elas. Este conjunto de variáveis e estados representam o conteúdo de informação que os componentes da estrutura de controle possuem. A informação representada por estas variáveis é transmitida entre os componentes por meio das conexões entre eles, uma `Connection` do tipo `ControlAction` por exemplo pode mandar a informação de uma variável contida em um `Controller` para um `ControlledProcess`.

Todas as entidades descritas nesta subseção são gerenciadas pela entidade `ControlStructure`, responsável por armazenar todas as informações de uma estrutura de controle criada no *Passo 2*.

4 PowerSTAMP: Exemplo da integração da MicroSTAMP com a WebSTAMP

Embora os microserviços da MicroSTAMP produzam valor de forma independente para diferentes *stakeholders*, eles, de forma isolada ou em conjunto, fornecem apenas suporte parcial à STPA (mais especificamente, suporte aos dois primeiros passos dos quatro presentes na técnica).

Normalmente, *analistas de safety* precisam de uma análise de *hazards* completa com STPA (com os artefatos produzidos nos quatro passos da técnica). Esta seção apresenta como podemos integrar a MicroSTAMP com uma ferramenta independente (WebSTAMP) para suporte completo à STPA, englobando os quatro passos da técnica.

A WebSTAMP, apresentada brevemente na Seção 2, é uma ferramenta *web* organizada em uma arquitetura monolítica constituída de camadas. Ela oferece suporte a todos os passos da técnica STPA.

Uma vantagem notável desta ferramenta é a abordagem utilizada para o passos 3 e 4 da STPA, que auxilia o analista a identificar ações de controle inseguras (*unsafe control actions)* e cenários de perdas (*loss scenarios)* de forma automatizada, cobrindo os aspectos necessários para uma análise STPA ou STPA-Sec.

Entretanto, para o *Passo 1* da STPA, ela não possibilita a criação de *sub-hazards*. Com relação ao *Passo 2* da STPA, a ferramenta possui a desvantagem de não conseguir modelar estruturas de controle aninhadas, nem atribuir responsabilidades para seus componentes, além de não possuir suporte para interações entre o sistema sendo modelado e o ambiente. As limitações (desvantagens) da WebSTAMP para os passos 1 e 2 não estão presentes na MicroSTAMP.

Sendo assim, apresentamos a PowerSTAMP, resultado da integração das ferramentas MicroSTAMP e WebSTAMP, objetivando suporte a análises de *hazards* mais abrangentes e automatizadas, cobrindo todos os aspectos da STPA. A PowerSTAMP apresenta o melhor das duas ferramentas: o suporte para o *Passo 1* e para o *Passo 2* da STPA via MicroSTAMP com o suporte do *Passo 3* e *Passo 4* da

WebSTAMP.

Como pode ser visto na Figura 4.1, a PowerSTAMP é uma ferramenta que utiliza o fluxo de controle da WebSTAMP, porém aplicando o *Passo 1* e *Passo 2* da MicroSTAMP. Optamos por não mostrar os artefatos de entrada e saída de cada passo na Figura 4.1 para não poluir a imagem.

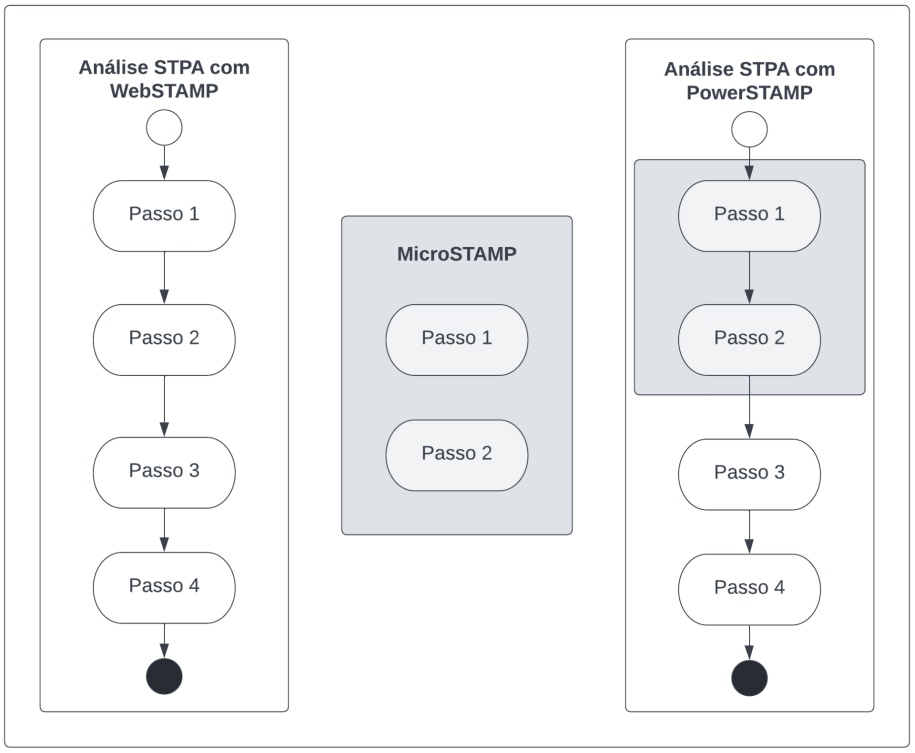

Figura 4.1 - PowerSTAMP, ferramenta que apresenta o melhor da WebSTAMP e da MicroSTAMP

Para o *Passo 1*, fornecemos os dados necessários para gerenciar todos os artefatos presentes no *Passo 1* da STPA utilizando os *endpoints* da MicroSTAMP via chamadas RESTful, porém mantendo as *views* (telas) da WebSTAMP, mapeando as ações de criação, edição, busca e remoção fornecidas pela WebSTAMP em tela para os serviços da MicroSTAMP.

Como a WebSTAMP não contempla todo o suporte fornecido pelo *Passo 2* da MicroSTAMP, optamos por aproveitar nossas *views* (telas via formulário *web*),

possibilitando que o *analista de safety* modele a estrutura de controle da forma necessária. Realizamos uma adaptação no formulário para redirecionar o *analista de safety* de volta para a plataforma WebSTAMP (*Passo 3*), entretanto, também seria possível capturar as informações sobre a estrutura modelada com chamadas RESTful aos *endpoints* do *Passo 2* da MicroSTAMP, não sendo necessário a alteração na nossa *view*.

Vale ressaltar que uma outra abordagem possível para integrar o *Passo 2* seria gerenciar por completo os artefatos da MicroSTAMP por chamadas RESTful, adaptando assim a *view* da WebSTAMP para contemplar o suporte a estruturas mais completas.

Já para o *Passo 3* e para o *Passo 4*, recuperamos os artefatos necessários criados nos passos anteriores, novamente fazendo uso de chamadas RESTful nos *endpoints* da MicroSTAMP. Dessa forma, utilizando as entidades criadas na MicroSTAMP, conseguimos seguir utilizando a abordagem da WebSTAMP em ambos os passos.

Após a integração das duas, foi possível realizar a análise de um exemplo: um sistema crítico de bomba de insulina presente na literatura (SOUZA *et al.*, 2019). Ele contempla todos os passos da técnica STPA.

5 Resultados

A MicroSTAMP pode ser integrada com a WebSTAMP para suporte completo à técnica STPA com o benefício adicional de melhorar as análises da WebSTAMP tendo em vista que os microserviços para o *Passo 1* e *Passo 2* da MicroSTAMP possuem melhor suporte aos passos 1 e 2 da STPA do que as funcionalidades atualmente implementadas no monolito da WebSTAMP.

Na PowerSTAMP, conseguimos realizar uma análise de *hazards* presente na literatura, com os dois passos iniciais da análise sendo executados pelos dois microserviços presentes neste trabalho, e com os dois passos finais da STPA (3 e 4) sendo executados pela WebSTAMP.

Os microserviços da MicroSTAMP para os passos 1 e 2 da técnica STPA

fornecem valor de forma independente pois cobrem todos os aspectos da técnica, possibilitando a definição de objetivos de análises de *hazards* e modelagem de estruturas de controle multi-níveis.

Conforme demonstrado com a PowerSTAMP, os microserviços da MicroSTAMP podem ser reutilizados e integrados em uma ferramenta como a WebSTAMP, para atendimento às necessidades de *analistas de safety* que demandam análises mais completas com suporte aos 4 passos da técnica STPA.

Por seguir o padrão RESTful e realizar a comunicação dos microserviços com JSON, a MicroSTAMP não se limita à tecnologias específicas para ser integrada. O uso da arquitetura de microserviços possibilita a integração e reuso da MicroSTAMP com qualquer ferramenta que necessite de um suporte completo ou parcial aos passos 1 e 2.

6 Discussões

Nós reivindicamos neste trabalho que resultados parciais obtidos a partir de cada um dos passos individuais da STPA também fornecem valor para diversos *stakeholders*, além de habilitar *analistas de* safety a usarem a ferramenta que acharem mais adequada para realização de cada um dos passos. Esta é a motivação para criação da MicroSTAMP, fornecendo microserviços para os passos 1 e 2 da STPA.

Os microserviços da ferramenta foram testados e validados de forma isolada e em conjunto. Além dos testes, apresentamos o microserviço do *Passo 1* para um especialista em STPA para validação (SARGENT, 2020). Nós fizemos a validação do *Passo 2* da MicroSTAMP a partir da modelagem de todas estruturas de controle presentes nos apêndices do livro referência em STPA (LEVESON; THOMAS, 2018). Ao todo, modelamos 10 estruturas de controle presentes nos Apêndices B e G, para diversos setores da indústria (operações de veículos espaciais autônomos, máquina de terapia de prótons, sistema fictício de interceptação de mísseis, sistema de frenagem automática, dentre outros).

A flexibilidade do uso de microserviços individuais para suporte aos passos da STPA possui um custo intrínseco: suposição que os *analistas de safety* ou outros *stakeholders* possuem as entradas necessárias para um passo em particular, dificultando a verificação de rastreabilidade entre artefatos produzidos por passos distintos.

Assim sendo, uma desvantagem do uso dos microserviços da MicroSTAMP de forma isolada é a suposição que *analistas de safety* ou outros *stakeholders* tenham em mãos as entradas necessárias para execução do microserviço. Por exemplo, uma das entradas do *Passo 2* é o conjunto de restrições de *safety* (*safety constraints*), criadas durante o *Passo 1*. Caso o *analista de safety* esteja usando o microserviço apenas do *Passo 2*, as entradas podem ter sido produzidas por outra ferramenta, anotadas em um papel, ou até mesmo a partir de um processo mental do analista.

Conforme mencionado, outra desvantagem sobre o uso isolado dos passos da STPA se dá com relação a verificação de rastreabilidade dos artefatos criados pelos 4 passos da STPA. Se cada um dos passos tiver sido criado por ferramentas distintas, manter a rastreabilidade torna-se uma atividade manual e propensa a erros.

A PowerSTAMP suporta verificação de rastreabilidade entre os artefatos presentes nos 4 passos. Isso é possível pois ela mantém referências a um `Project` (*Passo 1*) e a uma `ControlStructure` (*Passo 2*) associados à WebSTAMP. Com isso, é possível recuperar os artefatos criados na MicroSTAMP via requisições REST, pois os mesmos também estão vinculados ao projeto criado na WebSTAMP.

Conseguimos, por exemplo, recuperar os *hazards* do *Passo 1* (criados na MicroSTAMP), pois eles estão mapeados em um `Project` específico para o projeto criado na WebSTAMP. De forma similar, conseguimos recuperar objetos do tipo `Component` e `Connection` do *Passo 2*, pois eles foram criados em uma `ControlStructure` dedicada para o projeto em questão e criada previamente na MicroSTAMP.

Dessa forma, é possível recuperar estes artefatos (criados nos passos 1 e 2) no *Passo 3* da WebSTAMP, para então definir as ações de controle inseguras (h*azardous control actions*) e as restrições de *safety* e *security* associadas *(safety & security constraints)*, e por fim utilizá-las no *Passo 4* para definir os cenários de perdas (*loss scenarios*).

Portanto, a PowerSTAMP atende *analistas de safety* que necessitam de uma visão completa de uma análise de *hazards* com STPA, permite verificação automatizada de rastreabilidade e as entradas podem ser reaproveitadas de passos anteriores pela própria ferramenta.

Durante a integração, um dos problemas que nos deparamos foi a necessidade de adaptar as *views* da WebSTAMP para que elas recuperassem os artefatos construídos na MicroSTAMP. Isso foi necessário pois a exibição dos resultados na WebSTAMP acontece nas *views*, a partir de acessos diretos à base de dados existente.

Este problema não ocorre para uso com outras *views* (*interfaces* gráficas) que sejam desacopladas de um *back-end*. Para esses casos, nenhuma alteração na *interface* é necessária, pois a MicroSTAMP produz respostas no formato JSON, independente de tecnologia para construção de *views*.

Outra dificuldade encontrada no desenvolvimento da PowerSTAMP foi a rastreabilidade do projeto criado na WebSTAMP com os artefatos construídos na MicroSTAMP.

Para o *Passo 1*, mapeamos o fluxo de controle do projeto para a MicroSTAMP. Dessa forma, ao criar um novo projeto na PowerSTAMP, na realidade, o mesmo está sendo criado e persistido na MicroSTAMP, e não na WebSTAMP. Nos passos subsequentes, recuperamos e trabalhamos com esse projeto criado e a partir daí, todos os artefatos que o *analista de safety* criar são mapeados para este mesmo projeto na MicroSTAMP, mantendo a rastreabilidade entre eles.

Já para o *Passo 2*, precisamos criar previamente uma estrutura de controle na MicroSTAMP e então mapeá-la na WebSTAMP, para que seja possível gerenciar as entidades da estrutura de controle modelada. Com a informação do `Project` e

`ControlStructure`, conseguimos recuperar as entidades desenvolvidas comunicando via JSON e requisições REST, mantendo a rastreabilidade entre todos os artefatos do projeto.

7 Conclusões e trabalhos futuros

A partir dos resultados obtidos, podemos concluir que a MicroSTAMP suporta parcialmente análises de *hazards* com STPA utilizando de microserviços criados para os passos 1 e 2 da técnica. Os microserviços da MicroSTAMP fornecem valor para diversos *stakeholders*, mas podem ser ainda mais poderosos quando integrados com outras ferramentas para suporte integral à STPA.

O microserviço para o *Passo 1* da MicroSTAMP suporta por completo o *Passo 1* da técnica STPA (LEVESON; THOMAS, 2018), desde a identificação de objetivos de sistema (*system goals*), perdas (*losses*), condições de perigo - *hazards (incluindo sub-hazards)* até restrições de *safety* em nível de sistema (*system safety constraints*).

Já o microserviço para o *Passo 2* da MicroSTAMP possibilita a modelagem de estruturas de controle de vários níveis, com atribuição de responsabilidades para seus componentes, rastreabilidade para restrições de *safety* em nível de sistema, definição de variáveis e estados, definição do ambiente (`Environment`) na estrutura, e atribuição de conexões entre seus componentes.

Dessa forma, podemos concluir que a integração dos dois microserviços da MicroSTAMP com outras soluções permite a criação de análises de *hazards* mais completas quando comparadas com outras ferramentas com suporte integral à STPA, como por exemplo, a WebSTAMP.

É possível concluir também que a arquitetura de microserviços se mostrou uma boa candidata para ferramentas de análise STPA em relação à arquitetura monolítica. Como a STPA é constituída de passos individuais, ao se desenvolver um serviço para um passo em particular, é possível integrar e reutilizar esse serviço em mais de uma ferramenta, uma vantagem que a arquitetura monolítica não possui.

Dado que cada passo da técnica STPA pode produzir valor para diferentes

stakeholders e que, como dito anteriormente, os microserviços tem potencial em serem reutilizados e integrados com outras ferramentas STPA, recomendamos o desenvolvimento de microserviços para os passos 3 e 4 da técnica.

Uma limitação notável do *Passo 2* da MicroSTAMP é a falta de um suporte para modelagem de estruturas de controle de forma visual, "arrastando e soltando" (*drag and drop*) componentes, como presente em outras ferramentas, tal como a XSTAMPP (ABDULKHALEQ; WAGNER, 2015). Este suporte torna mais simples e intuitiva a tarefa de modelagem. Desta forma, recomendamos que o microserviço para o *Passo 2* da MicroSTAMP seja aperfeiçoado com suporte para criação de estruturas de controle via *drag and drop* de componentes.

Referências

ABDULKHALEQ, A.; WAGNER, S. **XSTAMPP**: An eXtensible STAMP platform as tool support for safety engineering, Boston: STAMP Workshop MIT, 2015.

ABDULKHALEQ, A.; WAGNER, S. **XSTAMPP 2.0:** new improvements to XSTAMPP Including CAST accident analysis and an extended approach to STPA. 5 ed.. Cambridge: STAMP Workshop, 2016.

ALTAF, A.; FAILY, S.; DOGAN, H.; MYLONAS, A.; THRON, E.. **Identifying safety and human factors issues in rail using IRIS and CAIRIS**. vol. 11980. Computer Security: Springer, Cham, 2020.

BECKER, C.; HOMMES, Q.Van Eikema. **Transportation systems safety hazard analysis tool (SafetyHAT)**. 1 ed.. United States: John A. Volpe National Transportation Systems Center, 2014.

BLEWITT, A..**Eclipse Plug-in Development**: Beginner's Guide. 2 ed..Birmingham: Packt, 2016.

EVANS, E.. **Domain-Driven Design**: Tackling Complexity in the Heart of Software. 1 ed.. Boston: Addison Wesley Professional, 2004.

GRIFFIN, J..**Domain-Driven Laravel:** Learn to Implement Domain-Driven Design Using Laravel. Illustrated ed..Nova York: Apress, 2020.

INFORMATION-TECHNOLOGY PROMOTION AGENCY(IPA).**STAMP Workbench**. Disponível em: https://www.ipa.go.jp/english/sec/reports/20180330.html, 2018. Data de acesso: 20/06/2022.

LEVENSON, Nancy G. **CAST Handbook**: How to learn more from incidents and accidents. 1 ed. 2019. Disponível em: https://bit.ly/3nlKHTv. Data de acesso: 26/06/2022.

LEVENSON, Nancy G. **Engineering a safer world**: Systems thinking applied to safety. 1 ed. Cambridge: The MIT Press, 2011.

LEVESON, Nancy G.; THOMAS, J. P. **STPA handbook**. Cambridge, MA, USA: 2018.

MARTIN, Robert C.; GREENING,James; BROWN, Simon; HENNEY, Kevlin; GORMAN, Jason. **Clean architecture:** a craftsman's guide to software structure and design. 1 ed.. Londres: Pearson, 2017.

MUSIB, S..**Spring Boot in Pratice.** 1 ed. Nova York: Manning, 2022.

NEWMAN, S.. **Building Microservices**: Designing Fine-Grained Systems. 2 ed.. California:O'Reilly Media, 2021.

REJZEK, M.; KRAUSS, Sven S.; HILBES, C.. **Tool qualification considerations for tools supporting STPA**. ed.128. Procedia Engineering, 2015.

Risk Management Studio (RM Studio). Disponível em: https://www.riskmanagementstudio.com/stpa-software-solution/. Data de acesso: 26/06/2022.

SARGENT, Robert G..**Verification and validation of simulation models:** an advanced tutorial. Winter Simulation Conference (WSC): IEEE, 2020.

SOUZA, Fellipe GR.; PEREIRA, Daniel P.; PAGLIARES, Rodrigo M.; NADJM-TEHRANI, Simin; HIRATA, Celso M. . **WebSTAMP**: a Web Application for STPA & STPA-Sec. Vol. 273. *MATEC Web of Conferences*: EDP Sciences, 2019.

Sparx Systems Pty Ltd. **UML Modeling and Lifecycle Tool Suite**. Disponível em: http://www.sparxsystems.com. Data de acesso: 26/06/2022.

YOUNG, William; LEVESON, Nancy G. **An integrated approach to safety and security based on systems theory**. v. 57. Communications of the ACM, 2014.