ATLÂNTICA
UNIVERSITY HIGHER INSTITUTION

Automação Inteligente
Aplicação de Práticas de DevSecOps no contexto de RPA

Projeto Final de Licenciatura

Elaborado por:
Ângelo Neves - nº de aluno 20192365

Orientador:
Prof.ª Dra. Virgínia Araújo

Barcarena
Julho de 2022

Gestão de Sistemas e Computação

Automação Inteligente
Aplicação de Práticas de DevSecOps no contexto de RPA

Projeto Final de Licenciatura

Elaborado por
Ângelo Neves - nº de aluno 20192365

Orientador:
Prof.ª Dra. Virgínia Araújo

Barcarena
Julho de 2022

O autor é o único responsável pelas ideias expressas neste relatório

Agradecimentos

Gostaria de dirigir os meus sinceros agradecimentos a todos os elementos e meus colegas da Siemens GBS O DS pela participação no inquérito desenvolvido no âmbito deste projeto, e em especial ao manager e *Head of RPA Operations* Joaquim Cláudio Casimiro, que me incentivou e apoiou na busca de novas soluções de qualidade para o serviço RPA na Siemens GBS.

À minha orientadora, Prof.ª Dra. Virgínia Araújo, gostaria de agradecer pelo apoio e disponibilidade prestada, foi o azimute para o desenvolvimento deste projeto, orientando-me sempre o melhor caminho sem nunca perder o norte.

A todos os meus colegas finalistas do curso de licenciatura GSC, fomos uma equipa e a capacidade de entreajuda foi preponderante para o sucesso do meu percurso académico. Um agradecimento em especial ao Nuno Cunha, Leandro Cunha e José Caniço, “estamos juntos”.

Um agradecimento final à minha família, à minha esposa Sílvia Leão e ao meu filho Rafael Neves Leão, que foram o meu farol e o meu porto de abrigo, abrigando-me da tormenta e de tempos de tempestade. Sem eles este objetivo não era possível. Para ti minha estrelinha.

Resumo

Num ecossistema de automação inteligente, nomeadamente dentro do contexto de RPA (*Robotic Process Automation)*, existe a necessidade de rever os processos e práticas de desenvolvimento e operações, com o sentido de aliar as duas competências ao bem comum necessário em qualquer organização, a segurança da informação, de facto, é com a segurança que se viabiliza a qualidade, a eficiência e a rentabilidade.

A elaboração de orientações e melhores práticas da aplicação da cultura *DevSecOps*, é atualmente primordial em qualquer organização que desenvolva produtos/serviços de software em ambientes ágeis. Numa era de digitalização, cada vez mais as equipas necessitam de um método colaborativo e de envolver várias valências, desde a análise à implementação e evolução de um produto de software. A segurança da informação terá que ser parte integrante desde o início ao fim do ciclo de vida de um produto, pois sem isso, coloca-se em risco aspetos fundamentas de confidencialidade, integridade e disponibilidade do software e que pode ter sérias implicações nas atividades de negócio da organização.

Sem perder o foco nas necessidades do cliente, é necessário modelar as práticas de desenvolvimento de software, seguindo metodologias mais ágeis. Desta forma, as equipas conseguem modelar o software ao longo do seu ciclo de vida, na perspetiva de entregar mais valor ao cliente e numa maior certeza de que os requisitos, os planos e os resultados estão 100% alinhados com as necessidades do cliente.

O teor deste trabalho, análise e proposta de soluções de melhoria contínua, teve por base a implementação de uma plataforma de automação inteligente numa numa empresa multinacional de grande dimensão. Em paralelo, foram estudados os aspetos que geram resistências à implementação de uma metodologia *DevSecOps* no âmbito do desenvolvimento de código RPA.

Palavras-chave: DevOps, DevSecOps, RPA, Segurança, Robotics.

Abstract

In an intelligent automation ecosystem, namely in the context of Robotic Process Automation, there is a need to review the development and operation processes and practices, in order to combine the two skills to the common good necessary for any organization, security. Well, it is with safety that quality, efficiency, and profitability become possible.

The elaboration of guidelines and best practices for the application of the DevSecOps culture is currently essential in any organization that develops software within an agile approach. In an era of digitalization, teams increasingly need a collaborative method and to involve several valences, from analysis to the implementation and evolution of a software product. Information security will have to be an integral part from beginning to end of a product's lifecycle, as without it, fundamental aspects of confidentiality, integrity and availability of the software are put at risk and that can create serious implications for the organization business activities.

Without losing focus on customer needs, it is necessary to model software development practices, following more agile methodologies. In this way, teams are able to model the software throughout its lifecycle, with a view to delivering more value to the customer and with greater certainty that requirements, plans, and results are 100% aligned with customer needs.

The content of this project, analysis, and proposal of continuous improvement solutions, was based on the implementation of an intelligent automation platform in a large-scale multinational organization. In parallel, aspects that generate resistance to the implementation of a DevSecOps methodology within the scope of RPA code development were studied.

Palavras-chave: DevOps, DevSecOps, RPA, Security, Robotics.

Índice

Lista de Abreviaturas, Acrónimos e Siglas

AI	Artificial Intelligence
API	Application Programming Interface
AT&T	American Telephone and Telegraph
BP	Blue Prism
BP App	Blue Prism Application
BPA	Business Process Automation
BPM	Business Process Management
CA	Customer Acceptance
CD	Continuous Delivery ou Continuous Deployment
CI	Continuous Integration
CI/CD	Continuous Integration & Continuous Delivery/Deployment
CIA	Confidentiality, Integrity, and Availability
CRC	Code Review Checklist
CRM	Customer Relationship Management
DEV	Development
DevOps	Development & Operations
DevSecOps	Development, Security & Operations
DSR	Design Science Research
DV	Development
EM	Emergency
ERP	Enterprise Resource Planning
FC	Feasibility Check
GBS	Global Business Services
GUI	Graphical User Interface
H2R	Hire-to-Retire
HIPAA	Health Insurance Portability and Accountability Act
HRI	Human-Robot Interaction

IA	Inteligência Artificial
IBM	International Business Machines Corporation
ICFR	Internal Control over Financial Reporting
ID	Identification
IoT	Internet of Things
IT	Information Technology
KPI	Key Performance Indicator
MFA	Multi Factor Authentication
ML	Machine Learning
NLP	Natural Language Processing
NUPKG	NuGet Package
O2C	Opportunity-to-Cash
OCR	Optical Character Recognition
OI	Operational Instructions
OLA	Operational Level Agreement
P2P	Purchase-to-Pay
PA	Process Analyst
PCI-DSS	Payment Card Industry Data Security Standards
PDD	Process Definition Document
PoC	Proof of Concept
PR	Production
QA	Quality Assurance
R2R	Record-to-Report
RBAC	Role-based access control
RE Framework	Robotic Enterprise Framework
RPA	Robotic Process Automation
SaaS	Software as a Service
SAP	Systems, Applications and Products
SDD	Solution Design Document
Siemens AG	Siemens Aktiengesellschaft
SLA	Service Level Agreement
SLO	Service Level Objective

SME	Subject Matter Expert
SNOW	Service Now
SOA	Service-Oriented Architecture
SoD	Segregation of Duties
SOX	Sarbanes-Oxley Act
TA	Technical Architect
Target App	Target Application
TI	Tecnologia da Informação
TPD	Test Planning Document
TS	Test
UAT	User Acceptance Test
VBO	Visual Business Objects
WPF	Windows Presentation Foundation

Índice de Figuras

Índice de Tabelas

Índice de Gráficos

1 Introdução

Em qualquer empresa ou organização existem inúmeras tarefas administrativas, de baixo risco e que são obrigatórias para o seu correto funcionamento. Contudo, muitas destas tarefas, são repetitivas, e para além de consumirem tempo, estão obsoletas, desajustadas e consomem tempo que poderia ser usado de forma mais eficiente. Assim, cada vez mais as empresas procuram minimizar os seus impactos na produtividade/eficiência de cada colaborador.

As reuniões, as tarefas administrativas, os e-mails e os telefonemas, são algumas das ações que ocupam grande parte dos dias dos colaboradores de uma organização, sendo por vezes fonte de distração na execução de determinado trabalho manual repetitivo, para o qual seria necessária concentração por um longo período. Com isso, inevitavelmente o rendimento e a concentração ficam reduzidos substancialmente, refletindo-se na produtividade do colaborador e na sua contribuição para tarefas mais importantes das organizações.

Segundo a investigação publicada no artigo da Harvard Business School, realizada por Teresa Amabile e Andrew Brodsky (2018), algum tempo ocioso pode ser bem-vindo para alguém que tem trabalhado duro. Mas nós, humanos, ficamos entediados rapidamente. O estudo identificou que, ao se atribuir uma tarefa repetitiva a um funcionário por muito mais tempo do que o necessário, o indivíduo prefere alongar essa tarefa tediosa ao invés de terminar a tarefa o mais rapidamente possível (Brodsky & Amabile, 2018).

RPA, ou *Robotic Process Automation*, é uma tecnologia que utiliza robôs para desenvolver tarefas antes executadas por humanos. Não se trata de quaisquer tarefas, mas sim atividades repetitivas e que não exigem pensamento crítico.

1.1 Motivação / Enquadramento

Este projeto nasce como resultado da licenciatura em Gestão de Sistemas e Computação, onde se juntam sinergias da função profissional do autor na área da Gestão de Serviços de

Tecnologias de Informação, nomeadamente na gestão de plataformas de *Robotic Process Automation*, RPA, na Siemens Global Business Services.

A Siemens GBS, com mais de 20 anos de experiência, oferece serviços de digitalização e otimização de processos de negócios de forma transparente para as áreas internas, como por exemplo, serviços de Opportunity-to-Cash (O2C), *Purchase-to-Pay* (P2P), *Record-to-Report* (R2R), *Hire-to-Retire* (H2R), entre outros.

A segurança proativa e focada no cliente permite antecipar, em vez de reagir, a violações de dados ou ciberataques. O DevSecOps quando implementado corretamente, ou seja, desde o início do ciclo de vida do software, permite reduzir os custos associados à correção de falhas de segurança, incorporando a segurança em todas as etapas do processo de desenvolvimento do software. Esta abordagem também pode ser aplicada no contexto de *Robotic Process Automation* (RPA).

Os programadores em RPA interessados em produzir automatizações com qualidade, beneficiam com a introdução dos aspetos de segurança numa fase inicial do projeto de desenvolvimento de software. Podem facilmente modelar e testar vulnerabilidades do software, e consequentemente, seguindo uma abordagem de segurança no RPA, podem proceder à revisão do código da automatização cobrindo aspetos de segurança, para além de poderem automatizar auditoria de sistemas, monitorização e gestão de eventos de segurança numa fase preliminar à entrega do projeto para produção. Desta forma, previne-se que a automatização com falhas seja colocada em produção e exponha a empresa a incidentes de segurança da informação.

A segurança da informação deverá estar intrínseca em todas as plataformas *Robotic Process Automation*, como também, em todas as atividades de planeamento, desenho, construção, teste, implementação e evolução, focando a segurança e privacidade dos dados, autenticação e a capacidade de impor um controlo de acessos baseado na função, (RBAC – *Role-based access control*), de forma a restringir o acesso à aplicação com base nas funções de cada utilizador.

Um melhor controlo e gestão das atividades no âmbito de RPA na Siemens, através da aplicação das práticas *DevSecOps* e com automatização da revisão de código, seria uma mais-valia significativa para a qualidade do software e das entregas, reduzindo significativamente o número de incidentes em produção.

1.2 Objetivos

O objetivo principal deste trabalho que rege o desenvolvimento das propostas neste estudo é a elaboração de orientações e melhores práticas da aplicação da cultura de *DevSecOps* no contexto da implementação do serviço *Robotic Process Automation*. Considerando o objetivo principal anteriormente identificado, é fundamental neste projeto, melhorar a agilidade no processo de desenvolvimento de processos robóticos, tendo em conta os constrangimentos impostos pelo desenvolvimento deste tipo de automatização no ambiente à escala empresarial. As exigências crescentes no que diz respeito à heterogeneidade na modelação de novas automatizações, tende a criar desafios no que refere à acomodação de novos processos em produção e que por vezes descura no especto fundamental da segurança da informação.

Por considerar que as melhores soluções resultam da partilha e do envolvimento das equipas envolvidas, procura-se estabelecer uma estratégia de sensibilização para os seguintes temas:

- Sensibilização das equipas para a uniformização de ferramentas, tornando este tipo de informação transparente para todas as equipas envolvidas.
- Estudo da maturidade de *Continuous Integration & Continuous Delivery* (CI/CD), das equipas de desenvolvimento e sensibilização das mesmas quanto à relevância deste tema.
- Aplicar e monitorizar o processo das práticas de *DevSecOps* no contexto RPA.

Seguindo em linha com o âmbito acima descrito, neste projeto, propõem-se os seguintes objetivos específicos a atingir:

- **Objetivo 1** - Identificar as capacidades e benefícios do *DevOps* e suas áreas de abrangência.
- **Objetivo 2** - Elaborar a comparação entre *DevOps* e *DevSecOps*, demonstrando os seus benefícios.
- **Objetivo 3** - Fazer um levantamento do processo de desenvolvimento de código RPA na Siemens GBS.
- **Objetivo 4** - Analisar o processo de melhoria contínua aplicado na Siemens GBS.

- **Objetivo 5** - Formular uma proposta de melhoria para o processo de *code review* e CI/CD (*Continuous Integration/Continuous Deliver/Continuous Deployment*).
- **Objetivo 6** - Criação de um modelo RPA para inclusão de práticas *DevSecOps*.

1.3 Estrutura do documento

Este projeto é composto por nove capítulos. O primeiro capítulo refere-se à introdução, nomeadamente motivação/enquadramento e objetivos e estrutura do documento. O segundo capítulo apresenta a base teórica sobre *DevOps* e *DevSecOps*, *Robotics Process Automation* e a aplicação de automação inteligente Blue Prism. O terceiro capítulo apresenta a metodologia de pesquisa utilizada neste estudo, *Design Science Research* e *Design Thinking*. No quarto capítulo, é apresentada a fase do projeto no contexto "*as is*". No quinto capítulo, apresenta-se o desenvolvimento e demonstração da proposta de melhoria contínua e levantamento dos requisitos. No sexto capítulo, trata-se da especificação dos requisitos, desenho e implementação, detalhe dos requisitos funcionais e não funcionais referentes à solução proposta. O sétimo capítulo, refere-se à validação e da solução proposta. No oitavo capítulo, estabelece-se a conclusão ao estudo desenvolvido, são também apresentados os possíveis trabalhos futuros e as limitações sentidas ao longo deste projeto. E por último, o nono capítulo, são colocadas as referências bibliográficas. A título de complementação de dados, foram colocados no final do documento os apêndices e os anexos que constam referenciados em certos capítulos.

2 Fundamentos Teóricos

2.1 DevOps vs DevSecOps

As práticas de desenvolvimento modernas contam com a metodologia *Agile*, que priorizam a melhoria contínua versus etapas sequenciais do tipo *Waterfall*. Se as equipas de desenvolvimento trabalharem isoladamente sem considerar as operações e a segurança, o produto desenvolvido pode apresentar problemas operacionais ou vulnerabilidades de segurança que podem ser ineficientes ao nível financeiro ou operacional.

Na conferência Agile de 2008 em Toronto, Canadá, Patrick Debois e Andrew Schafer realizaram uma sessão ad hoc intitulada "Infraestrutura ágil" sobre a aplicação dos princípios Agile à infraestrutura em oposição a um código de aplicação. Embora fossem as únicas pessoas que compareceram, rapidamente ganharam seguidores com ideais semelhantes, incluindo o co-autor John Willis, (Kim et al., 2016). A discussão deu origem ao grupo *Agile System Administration* em Grupos do Google por Andrew Shafer e Patrick Debois. Embora o grupo não tenha sido muito popular, mas isso levou a uma discussão fascinante.

Mais tarde, na conferência Velocity de 2009, John Allspaw e Paul Hammond fizeram a seguinte apresentação seminal, “10 Deploys per Day: Dev and Ops Cooperation at Flickr”, onde descreveram como criaram metas partilhadas entre *Development* e *Operations* e usaram práticas de integração contínuas para tornar a implementação, parte do trabalho diário de todas as equipas envolvidas, (Kim et al., 2016).

Patrick Debois não esteve presente nessa apresentação, mas ficou tão empolgado com a ideia de Allspaw e Hammond que criou o primeiro “DevOpsDays” em Gante, Bélgica em 2009. E foi onde o termo *DevOps* foi cunhado. O sucesso do evento inspirou outros eventos do “DevOpsDays” em diferentes países. Esses eventos agiram como um catalisador para a conversa e um movimento de base.

O *DevOps* ganhou notoriedade nos últimos anos como forma de combinar os principais princípios operacionais com os ciclos de desenvolvimento, reconhecendo que esses dois processos devem coexistir durante o ciclo de vida do produto, (Figura 1). As operações do pós-desenvolvimento em silos podem facilitar a identificação e o tratamento de possíveis problemas, mas essa abordagem atrasa a entrega. A implementação das operações em paralelo com os processos de desenvolvimento de software permite que as organizações reduzam o tempo de implementação e aumentem a eficiência geral.

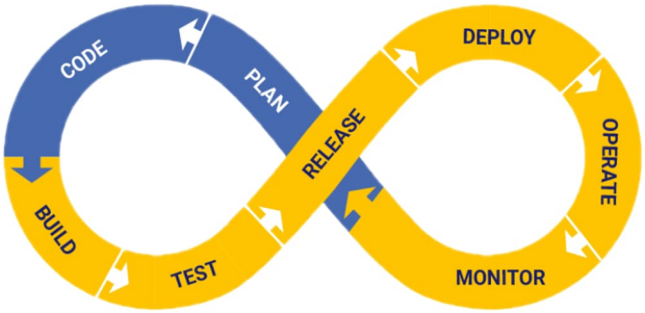


Figura 1 - Modelo de DevOps no contexto de integração e entrega contínuas

Na segurança da informação referem-se as ferramentas e técnicas necessárias para desenhar e construir um software que resista a ciberataques, com o objetivo de prevenir, detetar e responder a ameaças de segurança da informação o mais rápido possível. A segurança do software era comumente abordada após a conclusão do desenvolvimento, com intervenção de equipas específicas na área da cibersegurança não integradas com as equipas de desenvolvimento e de operações. Nesta abordagem em silos, o processo de desenvolvimento e o tempo de reação a qualquer correção de vulnerabilidades torna-se mais longo. Este processo cria dificuldades para qualquer equipa envolvida no ciclo de vida do produto, pois limita a visão dos problemas de segurança no contexto do ambiente de produção.

O DevSecOps é a integração eficiente dos testes de segurança e a proteção durante todo o ciclo de vida de desenvolvimento e implementação de software, ou seja, é necessário pensar na segurança da aplicação e na infraestrutura desde o início. Nesta abordagem de segurança integrada, o foco não é apenas estabelecer uma camada de proteção em torno de aplicações e

dados, mas em todo o contexto da implementação/integração, operação/manutenção e utilização por parte do utilizador final.

Assim como o *DevOps*, o *DevSecOps* é uma mentalidade que precisa de ser partilhada por todos os membros da equipa que participa no desenvolvimento e implementação do software. A adoção de uma mentalidade de segurança da informação e de cibersegurança na coexistência com outros requisitos, permite partilhar a responsabilidade sobre qualquer tecnologia ou técnica específica, elaborar metodologias de segurança que permitam um maior controlo e rapidez na gestão de vulnerabilidades e riscos de segurança.

Tal como o *DevOps*, os objetivos do *DevSecOps* são lançar software com maior qualidade, de forma rápida e segura. Se a segurança for implementada apenas no final do pipeline de desenvolvimento, as organizações que usam *DevOps* podem tornar-se menos eficientes, pois ao não adotarem a segurança integrada, a probabilidade de revisões duplicadas e recompilações desnecessárias aumenta, resultando num tempo de entrega mais longo, ou mesmo na criação de um código menos seguro.

De acordo com Myrbakken & Colomo-Palacios (2017), *DevSecOps* é o movimento que trabalha no desenvolvimento e integração de métodos de segurança modernizados que podem acompanhar o *DevOps*. O *DevSecOps* é um trio tático que interliga três áreas diferentes: desenvolvimento, segurança da informação e operações, (Figura 2). O objetivo é integrar perfeitamente a segurança no seu pipeline de integração contínua e entrega contínua, *Continuous Integration & Continuous Delivery/Continuous Deployment,* doravante denominado por CI/CD, em ambientes de desenvolvimento, pré-produção e produção.

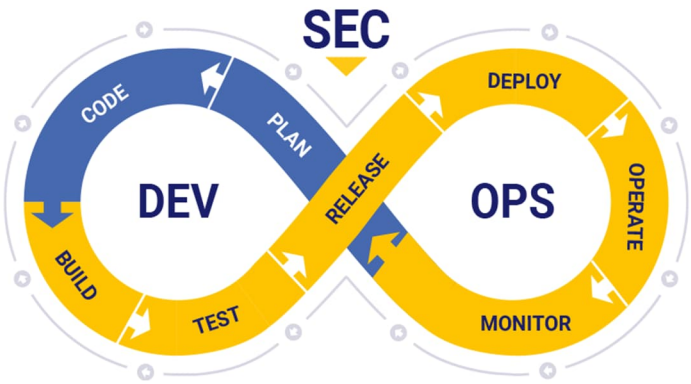


Figura 2 - Modelo de DevSecOps no contexto de integração e entrega contínuas (Apps Built Better DevSecOps, 2021)

O processo de integração continua – CI *Continuous Integration*, consiste na prática dos programadores de software enviarem constantemente as suas alterações para um repositório comum onde as mesmas são sistematicamente validadas através de diversos tipos de testes. O objetivo deste processo passa pela automatização de testes que são desencadeados a cada alteração submetida no repositório comum, garantindo desta forma que as alterações não violem a integridade do artefacto de software.

A entrega contínua - CD *Continuous Delivery* é uma extensão do processo de CI. Nesta fase o objetivo é ter sempre um artefacto de software pronto a ser entregue para produção. Isto significa que, para além da automatização dos testes é necessário ter um processo de entrega automática, onde tão frequentemente quanto o necessário, são construídas e controladas as versões dos artefactos de software com as eventuais alterações sucessivamente efetuadas.

A implementação contínua - CD *Continuous Deployment* é o último passo de automatização de todo o processo. Nesta fase pretende-se que qualquer alteração efetuada que passe com sucesso os passos anteriores seja entregue para produção de forma automática, sem qualquer intervenção humana. Desta forma, o resultado dos testes anteriores determina a inconsistência do código e previne a entrega de um produto com menor qualidade.

À medida que novos tipos de ciberataques aumentam, é cada vez mais importante a proteção dos ambientes de desenvolvimento e do CI/CD. Foco na segurança logo na fase inicial do ciclo de desenvolvimento, mantendo-se depois pelo ciclo de vida do produto, de forma a ajudar os

programadores a escreverem código mais seguro, a adotar as melhores práticas de segurança e responder rapidamente a vulnerabilidades.

2.2 Robotics Process Automation

Robotic Process Automation, doravante RPA, refere-se à tecnologia de software que facilita a construção, implementação e gestão de robôs de software que emulam ações humanas interagindo com outros softwares e sistemas digitais. Estes robôs realizam um conjunto de tarefas seguindo um processo, sem qualquer intervenção humana. Todas essas tecnologias reduzem a força de trabalho manual, o que permite às organizações automatizar as operações de negócios de maneira ágil e económica.

Trata-se de tecnologia de automatização de processos de negócios, (BPA – *Business Process Automation*), baseada em robôs de software, (*software robotics*), onde também poderá ser integrada a inteligência artificial, (AI – *Artificial Intelligence*). Estes robôs são comumente identificados como *digital workforce*.

A RPA pode usar integrações API (*Application Programming Interface*), bem como outras tecnologias de automatização, incluindo Inteligência Artificial (IA), assim como modelos de *Machine Learning* (ML), serviços cognitivos como *Chatbots, Natural Language Processing* (NLP) e *Optical Character Recognition* (OCR).

Os robôs podem ser utilizados para simular atividades que os humanos executam através de ecrãs ou aplicações, para capturar, interpretar e processar transações. Isto pode desencadear respostas, criando e manipulando dados de uma forma consistente e previsível. São tipicamente de baixo custo e fáceis de implementar. Não requerem o desenvolvimento ou uma integração complexa dentro de sistemas existentes. Os benefícios potenciais são claros, uma vez que o RPA permite a implementação de processos consistentes, fiáveis e previsíveis de uma forma rentável.

Através desta tecnologia, tarefas repetitivas podem ser automatizadas, permitindo que os funcionários se concentrem no trabalho mais especializado e crítico, proporcionando às empresas um aumento da produtividade. Também, pode ser visto nas organizações como um potencial método para racionalizar as operações comerciais, reduzindo os custos de pessoal e também reduzir o erro humano. Esta consistência pode levar a menos erros em processos chave, consequentemente a aumentos de receitas e um melhor serviço ao cliente, o que leva a uma maior satisfação e retenção de clientes.

No sentido demonstrativo da aplicabilidade desta tecnologia, apresentam-se a seguir, cinco exemplos de implementação do RPA nas organizações:

Área de Recursos Humanos

Contratação e Integração - Contratar apenas uma pessoa pode levar semanas e pode ser dispendioso. O processo de contratação e integração contém várias tarefas repetitivas e baseadas em regras nas quais o RPA pode ajudar.

Por exemplo, um robô pode fornecer candidatos 24 horas por dia com mais precisão e sem preconceitos. Depois de procurar candidatos, o processo de automatização robótico também pode selecionar currículos e candidatos.

Num outro exemplo, quando a empresa contrata um funcionário, o RPA pode gerir grande parte da documentação que a empresa é responsável por preencher e atualizar em vários sistemas.

Gestão de pagamento de salários - O processamento dos salários a cada mês é uma tarefa repetitiva e demorada que o departamento de Recursos Humanos em todas as empresas tem que gerir. Devido ao volume envolvido, poderão ocorrer erros e imprecisões, o que causa a necessidade de correções que podem resultar em atrasos nos pagamentos.

A automatização RPA pode verificar os dados de funcionários em vários sistemas e validar o controlo de horas, ganhos e deduções fiscais, também pode processar os benefícios tributáveis e outros reembolsos.

Área da Contabilidade e Finanças

Processamento de Faturas - A contabilidade e gestão financeira são operações comerciais vitais, mas as tarefas envolvidas são tediosas, propensas a erros e não geram receita diretamente.

A RPA apresenta-se como uma solução eficiente e produtiva para muitas dessas tarefas. Por exemplo, o processamento de faturas, Figura 3, é uma das tarefas mais demoradas. As faturas chegam por vários canais, depois são combinadas com os pedidos de compra e geralmente precisam de ser aprovadas por pessoas diferentes para o pagamento. Desta forma, é possível criar regras para enviar automaticamente faturas à entidade certa para aprovação, criando assim uma melhoria no *workflow* de aprovação para pagamento. Também é possível automatizar o processo de revisão das ordens de compra, melhorando a *checklist* para revisão adicional antes de enviar para pagamento.

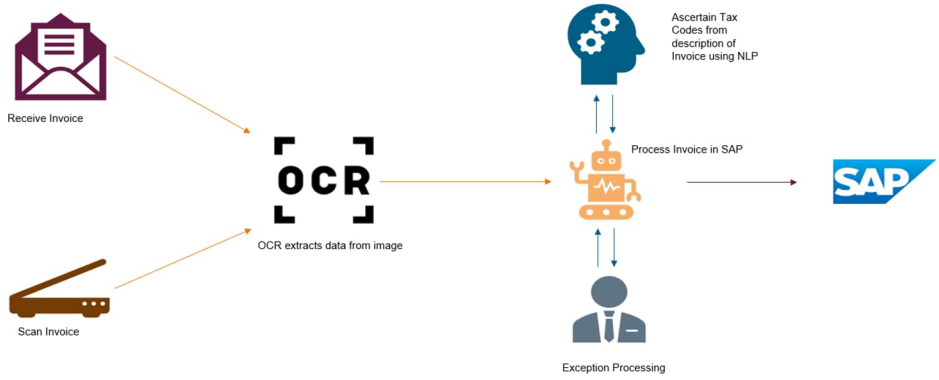


Figura 3 - RPA no processamento de faturas.

A área de contabilidade e financeira tem um processamento avultado de dados, o processamento manual envolvido em contabilidade e finanças incide numa taxa de esforço acrescida. É comum ser um dos pontos de partida para a implementação da RPA nas organizações.

Área do Retalho

Gestão de stocks – A expressão do valor agregado da automatização é notória à medida que as empresas se adaptam às tendências do comércio eletrónico. Em particular, os retalhistas poderão usar as soluções de RPA para gestão de stock. Especialmente para acompanhar vários produtos em várias regiões.

Com a evolução do mercado, e na da digitalização é necessário garantir que as empresas retalhistas têm stock suficiente para responder à procura, mas também precisam obter informações de gestão do stock sobre a sua procura e outras tendências do mercado.

Aliado ao BPM, (*Business Process Management*), a RPA pode ajudar a resolver esses problemas por meio de uma variedade de automatizações. Como por exemplo, reduzir erros de inventário, automatizar os alertas para tendências de inventário de stock baixo, otimizar os níveis de stock para maximizar o fundo de maneio, avaliar os números de vendas.

Área de suporte ao cliente

Um grande número de problemas e perguntas de um cliente podem ser resolvidos de maneira rotineira e padronizada, tornando o suporte ao cliente pronto para a RPA.

A RPA pode categorizar as consultas e enviá-las ao departamento correto, como suporte técnico, faturação ou área comercial, entre outros.

Com a RPA, é possível gerar uma resposta mais rápidas e aumentar a satisfação do cliente apenas incrementando uma melhoria no processo de *BackOffice* de um apoio ao cliente.

2.3 Blue Prism

A Blue Prism oferece controlo seguro, escalável e centralizado das tarefas automatizadas. Um grupo de especialistas em automatização de processos formou em 2001 a Blue Prism, uma das poucas empresas líderes de mercado de RPA. Esta tecnologia permite que pessoas que não são programadores de software, automatizem determinados processos de negócios de forma rápida e barata. Está direcionada para processos que são altamente orientados por regras e cuja exigência é muito tática ou de curta duração, vocacionada para justificar o desenvolvimento nas organizações de IT que favorecem a arquitetura orientada a serviços (SOA), como também as que englobam um conjunto de ferramentas de gestão de processos de negócios, (Slaby, 2012).

A Plataforma de Automatização Inteligente constitui uma ferramenta RPA que possui a capacidade da força de trabalho virtual alimentada por robôs de software. O software é desenvolvido em Microsoft .Net Framework e suporta várias plataformas como por exemplo IBM Mainframe, Windows, Windows Presentation Foundation, (WPF), com Java ou web. A ferramenta oferece um design visual numa abordagem *top-down*, visão do nível mais geral para o nível mais específico, e com funcionalidades *drag-and-drop*, permitindo que até mesmo os utilizadores não técnicos automatizem um processo arrastando componentes através de uma interface *user friendly*.

Garante a conformidade com as políticas de segurança instituídas (configurável) e fornece recursos robustos, na medida que protege os dados através de criptografia e ofuscação. Os algoritmos garantem conectividade, armazenamento e acesso a dados de forma segura.

No campo da segurança das credenciais necessárias para a automatização, possibilita a integração de um *password vault* para gestão das credenciais usadas na automatização, como por exemplo CyberArk ou IBM Security Secret Server.

Ao nível de controlo de acessos, permite a gestão de forma a restringir as funções por grupo de utilizadores, como por exemplo, autoriza o acesso específico dos utilizadores a grupos de robôs, processos e objetos. O software Blue Prism suporta Payment Card Industry Data Security Standards, (PCI-DSS), Health Insurance Portability and Accountability Act, (HIPAA), e Sarbanes-Oxley Act, (SOX), de forma a fornecer a segurança e a governança necessárias, (Blue Prism Security Guide, 2020).

Inclui uma interface de gestão centralizada de versões de *releases* para os processos automatizados, proporcionando visibilidade e controlo. A Plataforma de Automatização Inteligente da Blue Prism regista cada login do sistema, mudança na ação da gestão nas decisões e ações tomadas pelos robôs, como também regista os estados operacionais de cada execução da automatização.

Permite escalabilidade com gestão centralizada. Esta ferramenta foi projetada para funcionar de forma inteligente sem necessidade monitorização/ação "*in loco*" de todas as execuções que ocorrem no processo automatizado. Para tal, o software proporciona um módulo de gestão de agendamentos, (*schedule*), que permite executar de forma automática o processo automatizado

de acordo com um agendamento específico. Assim, todos os processos podem ser automatizados conforme a necessidade e podem ser monitorizados centralmente. Um *Control Room* aprimorado fornece feedback detalhado em tempo real sobre o status e a integridade do robô para uma visão *end-to-end* em toda a força de trabalho digital.

Fornece alguns *dashboards,* mas contempla a possibilidade de integração com Sistemas de Monitorização, como exemplo o Splunk. Implementando *add-ons* complementares, a Plataforma de Automatização Inteligente oferece a implementação de *Data Gateways*, fornecendo um método centralizado e fácil de usar para enviar dados do Blue Prism para utilização na monitorização com sistemas externos, armazenamento de dados para um longo período de retenção e para alimentar modelos de *machine learning.*

3 Metodologia

3.1 Design Science Research

Design Science Research (DSR) é uma nova visão de investigação à pesquisa, ou uma forma de perspetivar e idealizar sobre a pesquisa. No contexto da pesquisa de design, podemos considerar que se trata mais do que uma metodologia, embora inclua certas metodologias e defina um conjunto de técnicas de análise, como também, perspetivas positivistas e de interpretação para a realização de pesquisas nos sistemas de informação. O termo pode seguir em linha com a seguinte citação, “como um processo contínuo, com erros e acertos e ajustes ao longo de todo o processo” (Junior et al., 2013).

Neste tópico, é detalhada a metodologia científica na qual este projeto se baseou. O DSR orienta a elaboração de pesquisas científicas envolvendo o desenvolvimento de artefactos. A palavra Artefacto vem do latim "arte + factus", significa feito com arte, com técnica. Pode ser definido como um artifício, algo construído artificialmente, de maneira intencional, ou seja, tem uma utilidade específica, (Simon, 1996). Podemos considerar que seja o resultado de um projeto. Um artefacto é projetado para uma finalidade, numa abordagem de epistemologia-metodológica de pensar e fazer ciência, envolvendo a pesquisa interdisciplinar, como por exemplo, Sistemas e Tecnologias da Informação.

O método de pesquisa DSR ajuda no que se refere à solução de problemas reais nas organizações, muito devido ao seu pragmatismo, é adequada à investigação de problemas de natureza prática, em vez de, por outro lado, focar-se na verificação de leis naturais ou teorias comportamentais (Hevner et al., 2004).

Neste método, é possível identificarmos dois objetivos assinaláveis. Desenvolver um artefacto para resolver um problema prático num contexto específico e gerar novos conhecimentos técnicos e científicos. O conhecimento técnico necessário para a construção de um artefacto será diferente do conhecimento científico. Sem discriminação de relevâncias de valor, o conhecimento técnico e conhecimento científico são ambos relevantes, mas precisamos

reconhecer que são conhecimentos distintos, ainda que possam frequentemente ser confundidos:

> Os termos ciência e tecnologia quase sempre andam tão juntos que muitas pessoas têm dificuldade em distingui-los. Porém, a ciência é a busca do conhecimento e das explicações. A ciência constrói teorias para explicar os fatos observados. [...] Ao contrário da ciência, a tecnologia não tem por vocação explicar o mundo. Ela é prática e existe para transformar o mundo, não para teorizar sobre ele. (Wazlawick, 2014).

DSR é um método para desenvolvimento de soluções e, portanto, compatível com o procedimento científico largamente aceite em engenharia de software. O modelo DSR consiste num conjunto de elementos que precisam de estar coerentemente interrelacionados. Os principais elementos do Modelo-DSR estão representados na Figura 4. O Artefacto deve estar fundamentado em conjeturas e deve ser desenhado para resolver um problema num determinado contexto. Uma avaliação empírica, possibilita avaliar se o problema foi resolvido e se são válidas as conjeturas que fundamentam o desenvolvimento do artefacto. É necessário ter conhecimento sobre o problema e o contexto, fundamentar as teorias que constituem o quadro teórico, identificar soluções correlacionadas e realizar uma "pesquisa para o design", (Estado da Arte), (Pimentel et al., 2020). O pesquisador deve definir questões/hipóteses de pesquisa relacionadas à aceitação do artefacto, (baseadas nos critérios de aceitação), bem como questões/hipóteses de pesquisa relativas às conjeturas que fundamentam o design do artefacto.

3.2 Design Thinking

Outra abordagem metodológica que complementa o método DSR utilizada neste projeto também segue o método de o *Design Thinking*. Esta prática requer um pensamento criativo para a elaboração de possíveis soluções para um problema em busca de uma solução que seja satisfatória. Enquanto o método DSR foca a vertente técnica do artefacto a desenvolver, o *Design Thinking* foca a sua aplicabilidade na perspetiva do utilizador.

O método abdutivo, que consiste em estudar fatos e propor uma teoria para explicá-los, é característico do pensamento conceptual dos designers. A abdução é considerada um processo criativo, por isso é o mais indicado para compreender uma situação ou problema. Dada a forma como o processo criativo é intrínseco ao tipo de raciocínio para descoberta de soluções, o método abdutivo é o único método científico que permite a introdução de uma nova ideia. (Fischer & Gregor, 2011)

Dan Nessler, considera que cada fase deste processo de *Design Thinking*, alterna momentos de divergência e convergência de ideias, envolvendo as etapas de pesquisa, síntese, ideação e implementação. As fases divergentes exigem que se abra e leve em consideração tudo o que for possível ou que se desenvolva o maior número possível de ideias e potenciais soluções. As fases convergentes exigem uma redução, ou melhor que se refine o resultado apurado, colocando diretamente as ideias e abordagens de forma a ser possível criar decisões, (Nessler, 2018).

Enquanto no Modelo DSR especifica o quê deve ser feito numa pesquisa, (quais os elementos da pesquisa que devem ser pensados/realizados pelo pesquisador), o processo *Design Thinking* determina a ordem em que as coisas devem ser feitas. Essa é a diferença entre modelo e processo: o primeiro diz o que é, o segundo diz quando fazer.

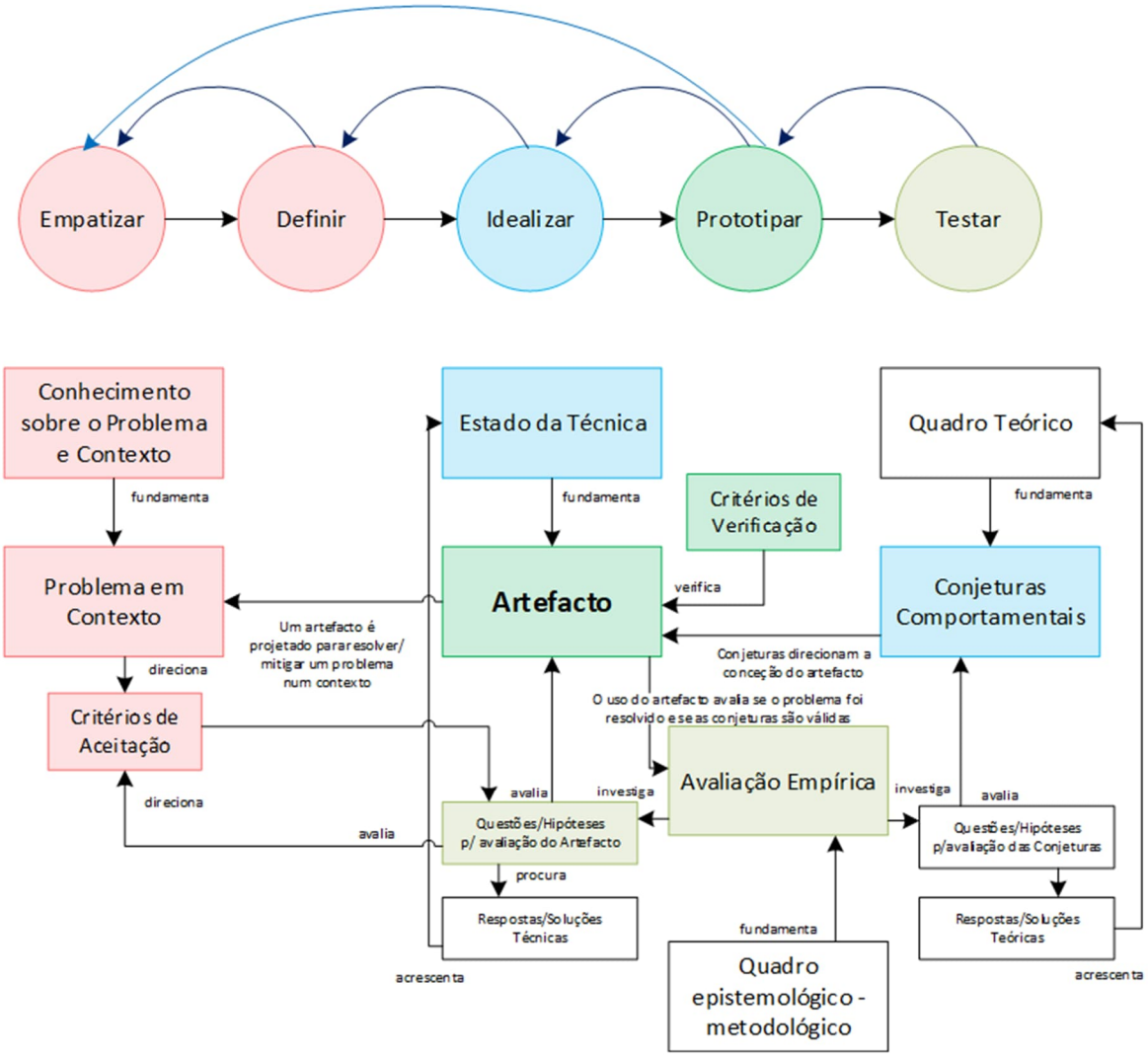


Figura 4 - Mapeamento entre o processo de Design Thinking e os elementos do DSR

O *Design Thinking* é caracterizado como a sequência de etapas, Empatizar, Definir, Idealizar, Prototipar e Testar. A Figura 4 ilustra a comparação com o método DSR. Empatizar e Definir no *Design Thinking*, equivale a identificar um Problema em Contexto no modelo DSR, incluindo a produção de Conhecimento sobre o Problema e Contexto. Já a Ideação equivale a projetar um Artefacto no modelo DSR, o que precisa ser feito com base em Conjeturas Comportamentais e em toda a revisão da literatura técnica e teórica subjacente. Prototipagem, em *Design Thinking*, equivale a implementar o Artefacto, enquanto Testar equivale a realizar uma Avaliação Empírica no modelo DSR.

No modelo DSR pretende-se dar ênfase à revisão da literatura, pois na avaliação empírica é exigido rigor científico. No *Design Thinking*, pretende-se testar a solução proposta para avaliar a sua utilidade, mas sem necessidade de contribuir com rigor para o conhecimento teórico-científico.

4 Caso de Estudo DevSecOps integrado na Plataforma RPA

Em adição às novas tecnologias funcionais existentes, tais como, IoT, *Big Data Analytics*, *Deep Learning*, *Inteligência Artificial*, *Machine Learning* e outras tecnologias relacionadas, a tecnologia RPA está a tornar-se uma das tecnologias disruptivas mais notáveis. Segundo a literatura, para suprir a falta de recursos humanos, a Interação Humano-Robô, (HRI - *Human-Robot Interaction*), (Slaby, 2012). O interesse sobre Robotic Process Automation é seriamente despoletado por empresas da Fortune 500 juntamente com novas start-ups.

Para os processos de negócios, o termo RPA mais comumente refere-se à configuração de software para fazer o trabalho anteriormente feito por pessoas, como transferir dados de várias fontes de entrada, como e-mail e folhas de calculo, para sistemas de registo como *Enterprise Resource Planning,* (ERP) e *Customer Relationship Management,* (CRM), (Lacity, et al., 2015).

David Schatsky, administrador da Deloitte, fundamenta num artigo publicado, que o desenho do processo é mais relevante para o Retorno do Investimento do que a tecnologia utilizada, o caso de uso publicado refere-se à experiência de um banco na implementação da tecnologia RPA, na qual o banco redesenhou seu processo de sinistros implementando 85 robôs para executar 13 processos, atendendo 1,5 milhões de solicitações por ano. O banco adicionou capacidade equivalente a 230 funcionários em tempo integral em aproximadamente 30% do custo de recrutamento de mais funcionários, (Schatsky et al., 2016).

É possível também verificar outros casos de uso, como por exemplo, AT&T, Deutsche Bank, Ernst & Young, Walgreens, estão entre as muitas empresas que adotam a tecnologia RPA, (Writer, 2019).

Relativamente ao presente caso de estudo, a Siemens GBS, em 2017 decidiu implementar a sua primeira plataforma RPA global de forma a servir os vários serviços internos. A tecnologia RPA eleita foi da Blue Prism, por ser uma das mais conceituadas e líderes de mercado. Um dos aspetos fundamentais, foi o facto de se tratar de uma das marcas pioneiras e com maior maturidade no mercado de tecnologias RPA. As soluções da BluePrism são projetadas para

empresas de grande dimensão. Fornecem um forte suporte para automatização de *back office* e, portanto, torna-se mais adequado para empresas de fabricação industrial e empresas de saúde, (Khan, 2020).

No caso de estudo em causa, a plataforma RPA é analisada na sua eficiência no que concerne à integração da metodologia *DevOps* com os requisitos de segurança da organização em causa. A Siemens AG, no contexto dos serviços digitais desenvolveu um serviço partilhado que contempla suporte ao *Business Process Management* interno. Uma plataforma Blue Prism RPA gerida centralmente que automatiza processos repetitivos, rotineiros e baseados em regras com base na entrada de dados estruturados. A plataforma RPA integra também com outras tecnologias para impulsionar a automatização de ponta a ponta.

Esta plataforma está desenhada considerando um ambiente de desenvolvimento, teste e produção. Todos os ambientes seguem uma segregação lógica e física dos ambientes, sendo que ao nível do ambiente de produção também existe a segregação física dos dados.

O Serviço RPA na Siemens é baseado na plataforma de automatização inteligente da Blue Prism. A arquitetura da plataforma de automatização inteligente é mostrada na Figura 5, a título de exemplo.

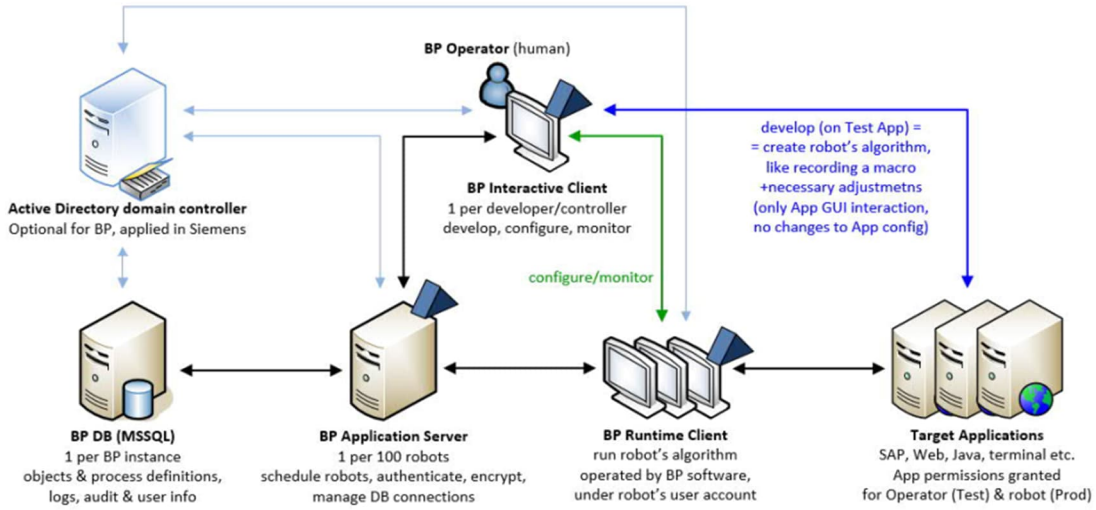


Figura 5 - Arquitetura da plataforma de automatização inteligente na Siemens GBS

A imagem e a descrição da Figura 5 demonstram os princípios e o conceito central da plataforma Blue Prism. Neste caso em específico, verifica-se um componente central denominado de servidor aplicacional, que gere de forma centralizada toda a conceção, configuração, parametrização e monitorização das automatizações RPA, um servidor de base de dados e um grupo de máquinas cliente para execução das tarefas automatizadas nos seus sistemas de destino/alvo. Existem várias possíveis abordagens e configurações da plataforma Blue Prism, dependendo das necessidades e restrições do negócio. Outro exemplo de implementação possível, é o *standalone deployment* que conta com o servidor SQL e o servidor aplicacional no mesmo servidor.

O software Blue Prism, doravante denominado de BP, executa um algoritmo predefinido no *Runtime Client*, ou seja, robô de software, isso permite que o software faça a autenticação nas aplicações de destino de forma criptografada e interagir com a GUI, (*Graphical User Interface*), das aplicações de destino, tal como, executar a leitura/gravação de dados nos campos da interface do utilizador, interação com elementos tais como botões ou seletores, etc., da mesma forma que um utilizador humano faria.

Um processo automatizado é capaz de operar várias aplicações de destino. Para operar uma aplicação de destino, o robô precisa de uma conta de utilizador e permissões adequadas dentro do sistema de destino, portanto, o robô está sujeito a segregação de funções, respeitando o princípio baseado em responsabilidades partilhadas de um processo-chave que dispersa as funções críticas desse processo por mais de uma pessoa ou departamento. Neste caso, a mesma autenticação num sistema, por exemplo SAP, não poderia registar uma ordem de compra e aprovar a mesma.

Os diagramas do processo automatizado são fluxos de trabalho de negócios, que atuam como programas de software. Esses diagramas utilizam conceitos básicos de programação e criam os fluxos do processo operacional como fluxogramas. São basicamente representações gráficas de fluxos de trabalho, para criar, analisar, modificar e dimensionar a capacidade do negócio.

Estes diagramas, definições dos processos automatizados, são armazenados na base de dados do BP e despachados para os *Runtime Clients* pelo servidor da aplicação do BP, de acordo com a programação pré-definida, com base no agendamento ou alocação da capacidade de trabalho

de forma inteligente, agendamento inteligente – a execução da automatização depende da existência de tarefas pendentes. Os *Runtime Clients* relatam os resultados da execução do processo automatizado para a base de dados aplicacional, via servidor da aplicação do BP; esses relatórios estão disponíveis para solução de problemas ou auditoria.

Toda esta automatização é desenvolvida por *developers* usando a aplicação Blue Prism *Interactive Client*, nomeadamente *Process Studio.* Este componente do Blue Prism oferece recursos como lógica de negócios, chamada de objetos, *loops* de controlo e variáveis. Cada processo criado tem a Página Principal que é executada primeiro. O *output* será a execução de um processo de negócios, de acordo com o documento de definição do processo de negócios.

O *Object Studio* é a área onde os *Visual Business Objects* (VBO) são criados. Esses objetos nada mais são do que programas diagramáticos que interagem com as aplicações externas de forma a estabelecer a comunicação para a automatização das tarefas. Um *Business Object* fornece uma interface com apenas uma aplicação externa. Sendo necessário criar novos objetos para cada nova aplicação.

Cada programador RPA tem acesso ao ambiente aplicacional Blue Prism de desenvolvimento. Para tal, foram criadas segregações de ambientes para que seja possível manter o SoD de acordo com o exigível para a segurança de toda a plataforma RPA. É neste ambiente que os processos e objetos são criados, sendo depois testados no ambiente de testes e somente após o *User Acceptance Test* ser efetuado com sucesso é que a automatização é distribuída para produção, toda esta integração é executada e gerida pelo *Release Manager*, por via do CI/CD ou de uma forma *ad-hoc*.

4.1 Funções do serviço RPA (aplicação Blue Prism e operação do serviço)

As Tabelas 1 e 2 apresentam as funções envolvidas no Serviço RPA no nível da plataforma Blue Prism e no nível de operação do Serviço RPA. Essas funções são geridas e controladas pelo Serviço RPA.

Tabela 1 - ***Funções e Responsabilidades do Serviço RPA***

Nome da Função	Área da Função	Tipo de Função	Tarefas
RPA Service Owner	Service	Management	Representa o serviço em toda a organização; Participa na negociação de Acordos de Nível de Serviço (SLAs) e Acordos de Nível Operacional (OLAs) para o serviço; Fornece informações para o desenvolvimento de indicadores-chave de desempenho (KPIs) e métricas para fornecer status sobre a integridade do serviço; Coordena a comunicação interna com a equipa operacional.
Process Analyst	Service	Management	Realiza esclarecimentos sobre os novos pedidos de automação, requisitos e custos; Gere as reuniões entre cliente, *Developers* , *Control Room Manage* r e *Release Manager*; Coordena e acompanha os novos pedidos de automação desde o início até a entrega para Produção.
Architect	Service	Management	Estima o esforço de entrega e prepara a oferta do projeto; Gere a capacidade e atribuição dos recursos dos *developers*; Cria o *Solution Design Document*, (SDD), para cada projeto; Revê os padrões para o desenvolvimento e executa a revisão do código; Gere a business object library e garante a reutilização dos objectos entre as equipes de desenvolvimento. A função de Architect é conjunta com a função de Developer.
Developer	BP App	Functional	Cria o diagrama da automação, com base no PDD, no ambiente desenvolvimento e instâncias de teste da aplicação de destino; Executa as alterações ao código do diagrana de automação e coopera na revisão do código juntamente com o arquiteto, (princípio dos 4 olhos); Coopera com o *Release Manager* na documentação ao suporte à qualidade e na gestão de releases para ambientes de Teste e Produção; Presta suporte à resolução de problemas na automação com base nos incidentes registados pelo *Control Room Manager* ou Cliente.

Informação recolhida do modelo de operações referido no RPA Operations Handbook da Siemens GBS

*Tabela 2 - **(Cont.) Funções e Responsabilidades do Serviço RPA***

Nome da Função	Área da Função	Tipo de Função	Tarefas
Tester	BP App	Functional	Executa o *User Acceptance Test* do diagrama do processo automatizado no ambiente de teste e na instância de teste do aplicativo de destino; Mantém a função de Developer no ambiente de desenvolvimento.
Release Manager	BP App	Functional	Cria os pacotes de novas automações ou pacotes de atualizações para distribuição entre ambientes Blue Prism (Desenvolvimento, Teste, Emergência, Produção); Gere a integração e entregas continuas, CI/CD; Gere as credenciais das aplicações necessárias para a automação.
Control Room Manager	BP App	Functional	Configura clientes virtuais de acordo com os requisitos específicos das tarefas do Robô; Agenda e monitoriza as tarefas do robô em Produção; Soluciona os problemas ocorridos nas tarefas do robô; Gere a capacidade da *virual workforce* (robôs RPA).
Runtime Resurce (Robot)	BP App	Functional	Executa a automação; De acordo com o SoD, acede aos sistemas necessários para executar a automação programada.
IT Service Manager	Service	Management	Responsavel geral por todo a plataforma RPA incl. projeto, operação, custo, segurança e continuidade de negócios; Gere a plataforma Blue Prism e infraestrutura de TI relacionada com RPA, incl. coordenação da entrega de serviços e suporte de fornecedores; É o gestor da aplicação Blue Prism no que concerne à segurança; Coordena o projeto e implementação de mudanças na plataforma Blue Prism e infraestrutura RPA relacionada; É o Administrador do sistema Blue Prism em todos os ambientes.
System Administrator	BP App	Functional	Gere a aplicação Blue Prism de ponta-a-ponta em todos os ambientes, Desenvolvimento, Teste, Emergência e Produção; Executa o controlo de acessos da aplicação Blue Prism; A função de *System Administrator* é conjunta com a função de *IT Service Manager*.
Target Application Owner	Target App	Management	Aprova ou garante a implementação do acesso refrente ao sistema a automatizar; Garante recursos para apoiar a organização e execução do UAT; Aprova e garante a implementação do acesso ao sistema produtivo da aplicação a automatizar após o UAT.

Informação recolhida do modelo de operações referido no RPA Operations Handbook da Siemens GBS

4.2 Modelo Operacional do Serviço RPA

No serviço RPA criado exclusivamente para a organização Siemens, visa a prestação de serviços para desenvolvimento e gestão da operação de automações RPA, para diferentes unidades de negócio da organização. Verifica-se que a procura por serviços internos automatizados é crescente e a integração tecnológica é heterogénea. Pode-se afirmar que cada tarefa de automatização executada pela RPA depreende um nível de desenvolvimento

específico, o que torna cada robô de software único, tanto no que concerne aos acessos aplicacionais, como pelo diagrama do processo desenvolvido.

O modelo de desenvolvimento em prática é baseado no *Software Factory approach*, tal como pode ser verificado na ilustração da Figura 6. O desenvolvimento de automatizações RPA baseado na abordagem fábrica de software, pode fornecer benefícios quando comparado às abordagens convencionais de desenvolvimento de software. Entre estes benefícios, destaca-se, a consistência na entrega, dado poder-se partilhar os mesmos recursos e lógicas semelhantes, embora seja necessário partilhar o conhecimento, como por exemplo, formações, documentação e *frameworks*. No entanto, usar esta abordagem para aplicar consistentemente o conhecimento adquirido anteriormente durante o desenvolvimento de várias automatizações RPA, pode ser um processo ineficiente e propenso a erros. Outro dos benefícios é a qualidade, devido à integração de código reutilizável é possível poupar tempo e recursos no desenvolvimento da automatização, permitindo gastar mais tempo trabalhando nos recursos exclusivos de cada automatização. O expectável é que se consiga reduzir a probabilidade de falhas de design e erros de código, mas sem que exista consistência na excelência de entrega, dificilmente se consegue reduzir o esforço para entregar com qualidade. Por último, aborda-se a produtividade, a eficiência na consistência e qualidade permite entregar cada projeto em menor tempo possível, possibilitando uma maior capacidade de entrega de novos projetos usando os mesmos recursos.

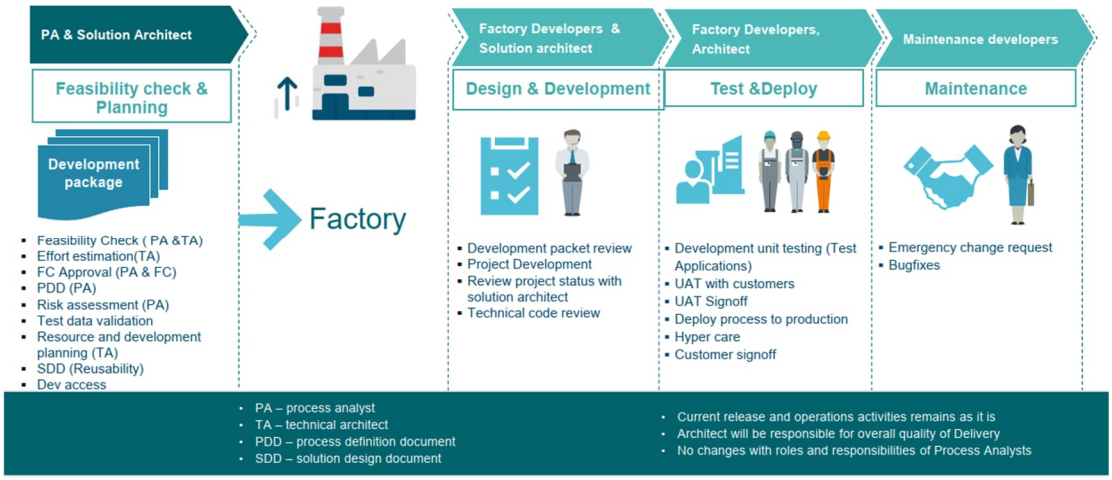


Figura 6 - Software factory approach (Abordagem de fábrica de software para o RPA na Siemens.)

Como já foi referido, cada robô virtual tem uma configuração própria, considerando a necessidades específicas de cada automatização, o que limita a execução do mesmo processo automatizado por várias máquinas. Pelo facto de serem cumpridas as segregações de acessos, também torna impossível escalar a capacidade da automatização ao nível da execução dos processos por uma lista de robôs. Muitas das aplicações validam os acessos pelo *Domain Account* e executam *Multi Factor Authentication,* (MFA), utilizado recursos de autenticação federada e/ou por via de certificados digitais. O que torna cada ID do robô único nos acessos aplicacionais, desta forma, é considerado o seguinte, uma tarefa automatizada específica corre somente num robô, mas um robô poderá correr mais do que uma tarefa automatizada, desde que não exista conflito nas funções executadas ao nível da segregação de deveres ou confidencialidade.

Os processos de automatização robótica são agendados, controlados e monitorizados pela equipa *Control Room*. Cabe a esta equipa verificar a operacionalidade do serviço RPA, sinalizar falhas e acomodar novas automatizações em produção. Nesta sala de controlo os eventos são registados em tempo real.

4.3 A plataforma RPA e a integração e entrega contínuas

O objetivo da *Release*, (lançamento de uma versão de código), é garantir o transporte seguro de RPA *packages*, (códigos BluePrism), do ambiente de origem para o ambiente de destino, realizando controlos de entrega dos *packages* por meio de ferramentas e *logs* das *releases*.

O controlo de qualidade está ilustrado na Figura 7, onde é possível identificar os passos desde a entrada de novos pedidos de automatização até à saída para produção. Todos as novas automatizações passam pelo processo de teste e de UAT, (*User Acceptance Test*), sendo que são entregues em produção num período de *hypercare* onde a monitorização e recolha de *logs* permite depurar erros não encontrados durantes as fases anteriores.

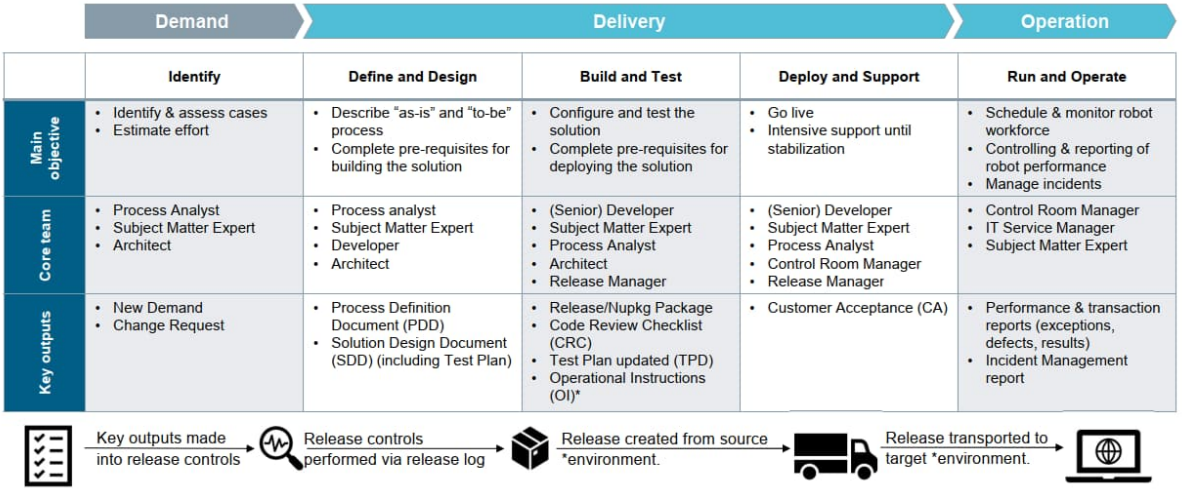


Figura 7 - Quality Control (Controlo de qualidade para o RPA na Siemens).

Mesmo após a *release* para produção a monitorização contínua é efetuada. Sempre que existe uma falha não recuperável na operação do robô, o *Control Room Manager* deverá alertar o *Developer* e o *Process Owner* para o facto do robô não estar a conseguir executar a tarefa programada. São recolhidas evidencias da operação e verificado o erro em detalhe. O incidente poderá ser originário de diferentes causas raiz. Pode ser algo relacionado com o software que corre na máquina virtual, problemas de comunicação/rede ou problemas ao nível da automatização. Este último, poderá ser identificável caso algo na aplicação a automatizar tenha sido alterado, ou mesmo, o próprio processo de negócio alterou-se, mas essas alterações não foram refletidas no processo RPA. No caso de se tratar de uma falha ao nível do *workflow* do processo RPA, (diagrama do processo), será necessária a intervenção do *developer* para a resolução do problema.

Com está descrito na Tabela 3, existem 3 tipos relevantes de processos de entrega. As entregas do tipo 1, novas automatizações e grandes alterações a automatizações RPA que se encontram em produção, tipo2, pequenas alterações às automatizações existentes em produção e por fim as alterações de tipo 3, que tem por base a correção ou adaptação urgente de uma automatização que se encontra em produção.

*Tabela 3 - **Tipos de processos de entrega***

	Type I New Automation and Major Change Request	Type II Minor Change Request	Type III Emergency Change Request
Description	Request for automating a: • process • process variation • process step This type Includes significant changes to a current automation requiring new Process and Business Objects. OR Flowchart/Sequence/RE Framework.	Request for atomic changes to an existing automation due to continuous improvement. (Ex.: Variable values, Configuration files, Credentials, Schedules). In general, this type does not require the creation of new Business Objects nor Process Objects.	Request for changing an existing automation due to errors in production. This type must not require the creation of new Workflows/Objects.
Request form	New Demand or Change Request in the Service Now (ticketing tool) initiated by Subject Matter Expert/Process Analyst.	Minor Change Request entry in Service Now (ticketing tool) started by Subject Matter Expert.	Service Now (ticketing tool) entry requested by Control Room Manager or Subject Matter Expert.
Effort	Variable depending on the complexity.	Included in Maintenance service (less than a day effort).	Included in Maintenance service.
Comments	The change requires approval of Process Owner before execution.	The change does not require approval of Process Owner before execution.	The change is executed and the code is approved by Senior Developer/Architect.

Tipos de processos de entrega para o serviço RPA na Siemens.

Nos casos em que é necessário corrigir uma automatização que já se encontra em produção, na maioria das vezes, o *developer* irá necessitar de acesso à aplicação a automatizar em produção para perceber quais as diferenças em relação ao que foi desenvolvido no ambiente de qualidade. Para colmatar essa falha foi criado um ambiente Blue Prism chamado de Emergência.

No ambiente de Blue Prism de Emergência, é possível ao *developer* utilizar os sistemas produtivos de forma a minimizar quaisquer diferenças verificadas entre os ambientes de qualidade e os ambientes de produção. Ou seja, numa automatização onde é programada a manipulação de dados existentes somente em ambientes aplicacionais de produção, tais como, faturas ou ordens de compra em SAP, por vezes, os ambientes de QA, não têm a mesma qualidade em volume de dados nem na sua heterogeneidade, o que torna mais difícil, treinar a automatização com dados "*dummy*".

O desenvolvimento neste ambiente de Emergência está dentro do planeamento da Siemens para o seu *Business Continuity Management*, possibilitando assim a recuperação a uma falha de uma forma mais célere, reiniciando o serviço o mais rápido possível para que o negócio não tenha um impacto elevado devido ao *downtime* ocorrido pelo incidente ou desastre.

Normalmente o processo automatizado é colocado em modo de *debugging*, (depuração), de forma a auditar todo o *workflow* do diagrama do código RPA, o *developer* terá a possibilidade

de acompanhar "*in loco*" toda atividade automatizada a fim de perceber qual a raiz do problema. Esta monitorização é executada utilizando o princípio dos quatro olhos, (*four-eyes principle*), é sempre importante que ao menos duas pessoas participem neste processo, de forma a mitigar erros ou práticas ilícitas. O acesso às credenciais do robô em análise será sempre reduzido ao mínimo necessário, dado que a automatização poderá executar as autenticações sem que as passwords sejam partilhadas ou visualizadas. Todas as credenciais partilhadas serão alteradas após o *troubleshooting* em Emergência. Este processo só será possível de forma temporária e após a aprovação de acordo com o controlo de acessos implementado.

4.3.1 Processo automatizado para a integração e entrega contínuas

No processo de integração e entrega contínuas é adotada uma abordagem de automatização integrando o conceito da RPA na gestão do pipeline. A própria automatização RPA gere o conceito CI/CD *pipeline*, permitindo que toda gestão de entrega de novas automatizações passe a ser efetuada de forma automatizada. A Tabela 4 fornece uma visão geral dos controlos de versão do código RPA em relação ao tipo de *release* e ao ambiente Blue Prism a que se destina.

Tabela 4 - ***Controlo dos processos de entrega***

Release types	Release Environment						
	DV-TS	DV-TS,PR	DV-TS,EM,PR	TS-PR	PR-DV	PR-DV,EM,TS	PR-EM
New Automation / Major Change request	SDD			UAT			
	PDD			OI			
	TPD						
	TCR						
Release Upgrade (Continuous testing)	TCR						
Minor Change request	TCR	TCR		*UAT			
	Incident #	Incident #		Incident #			
	TPD*	*OI		*OI			
Emergency change request	TCR	TCR		Incident #	Incident #	Incident #	Incident #
	Incident #	Incident #					
Rollback				+Incident #			
Shared Components (Shared object & Process)	TCR (TA)		TCR (TA)			+Incident #	
Remarks	New Automation / Major Change request, the quality gates will be validated by release manager *Conditional control, applicable as required +Rollback ticket can only be requested by SME / Process analyst +Shared components to be released from Production to other environment to be requested by Technical Architects Incident should be mapped to the configuration item of the use case						

Abbreviation	
DV	Development
TS	Test
PR	Production
EM	Emergency
SDD	Solution Design Document
PDD	Process Definition Document
TPD	Test Plan Document
TCR	Technical Code Review
Incident #	SNOW Ticket Number
TA	Technical Architect
UAT	User Acceptance Test
OI	Operational Instruction

Matriz de controlo para cada ambiente RPA Blue Prism na Siemens.

Nesta automatização são considerados os vários ambientes e as limitações de acordo com a Tabela 5. O desenho do processo baseia-se na automatização da gestão de filas de trabalho em RPA, interligando com uma API de uma ferramenta externa – *low code process automation* - nomeadamente Mendix, onde é possível automatizar o processo de *release management*, coordenando as várias etapas deste processo, desde a entrada de um pedido de alteração, como também as validações das versões do código do Blue Prism e os respetivos *release logs*. Desta forma, é possível gerir o *release* log e controlar o estado do pedido de cada tipo de alteração. Com esta abordagem de integração, é potenciada a gestão do *backlog* de acordo com os novos pedidos de serviço ou alterações a automatizações RPA já implementadas.

Tabela 5 - ***Tipos de processos de entrega***

	Release Automation Process per BP environment			
Release activities	**Development**	**Test**	**Emergency**	**Production**
New Automation / Major Change request	Not Applicable	**Partially automated	Fully automated	Fully automated
Release Upgrade (Continuous Testing)	Not Applicable	**Partially automated	Fully automated	Fully automated
Minor Change request	Not Automated	**Partially automated	Fully automated	Fully automated
Emergency Change request	Not Applicable	Not Applicable	Fully automated	Fully automated
Rollback	**Partially automated	**Partially automated	Fully automated	Fully automated
Shared Components (Shared Object & Process)	**Partially automated	**Partially automated	Fully automated	Fully automated
Updating Environmental Variables	Not Automated	**Partially automated	*Partially automated	*Partially automated
Updating Credentials	Not Applicable	**Partially automated	***Partially automated	***Partially automated
Remarks	*Due to the release control process, manual release may apply ** Developers have release privileges to perform their own tasks. *** Some credentials are not implemented through automation, for example Windows accounts.			

Matriz dos processos de entrega por ambiente RPA na Siemens

O fluxograma apresentado na Figura 8, ilustra o desenho do processo na integração desta *release tool*. Neste processo é usado uma aplicação *low code*, em Mendix, que permite gerir a documentação e o controlo de versões de cada *release*. O CI/CD é implementado na medida do controlo de qualidade, dependendo sempre que todos os inputs, como por exemplo, preparação do package, (código) ou documentação, sejam processados manualmente previamente a serem executados pela automatização em Mendix. Após a verificação da existência de uma nova *release* ou alteração a um código existente, é despoletada a própria automatização em RPA para

executar a entrega de acordo com o ambiente/instância de destino. É como se tratasse de um robô RPA a executar tarefas para ser próprio serviço.

É importante não existir incoerências aos testes executados em UAT para os *New Automation /Major Change Requests*. A documentação deve evidenciar que tipo ou nível de testes foram executados para uma fácil avaliação à integridade e resiliência do código. Este procedimento ainda não se encontra totalmente automatizado e é nesta fase, que se determina o controlo de qualidade e aceitação dos termos para a autorização à passagem do código a produção. Quanto maior for o número de incidentes/*bugs*, maior será recorrência ao *debugging* em Emergência, o que leva a um maior número de *Emergency Change Requests*, que por sua vez incrementam o pipeline no *delivery* do CI/CD. Em certos casos, como por exemplo alterações pequenas, *Minor Change Requests*, as validações ou testes são ignorados e a passagem a produção é direta.

Os shared componentes, ou Blue Prism *business objects*, são os objetos de negócios que fornecem uma série de ações que podem ser chamadas por um processo. Por exemplo, um objeto de negócios pode ser projetado para interagir com uma aplicação de mainframe contendo detalhes do cliente. O processo de alto nível pode exigir que o endereço de um cliente seja atualizado. Um objeto de negócios útil conteria a ação "Definir endereço do cliente" e receberia um parâmetro "ID do cliente" para identificar no sistema subjacente qual endereço do cliente deve ser atualizado. O objeto de negócios então interage com sistema de mainframe de destino para realizar sua tarefa.

Ao compartilhar estas funcionalidades úteis, permite que sejam reutilizadas em vários processos automatizados. Um único objeto de negócios pode ser usado por vários processos ao mesmo tempo. No fluxograma da Figura 8, estes objetos estão evidenciados por se tratar de um processo de desenvolvimento e *release* não relacionado apenas com cada automatização, mas sim, com todas as automatizações que utilizem a mesma aplicação/sistema de destino e partilhem o mesmo objeto de negócios.

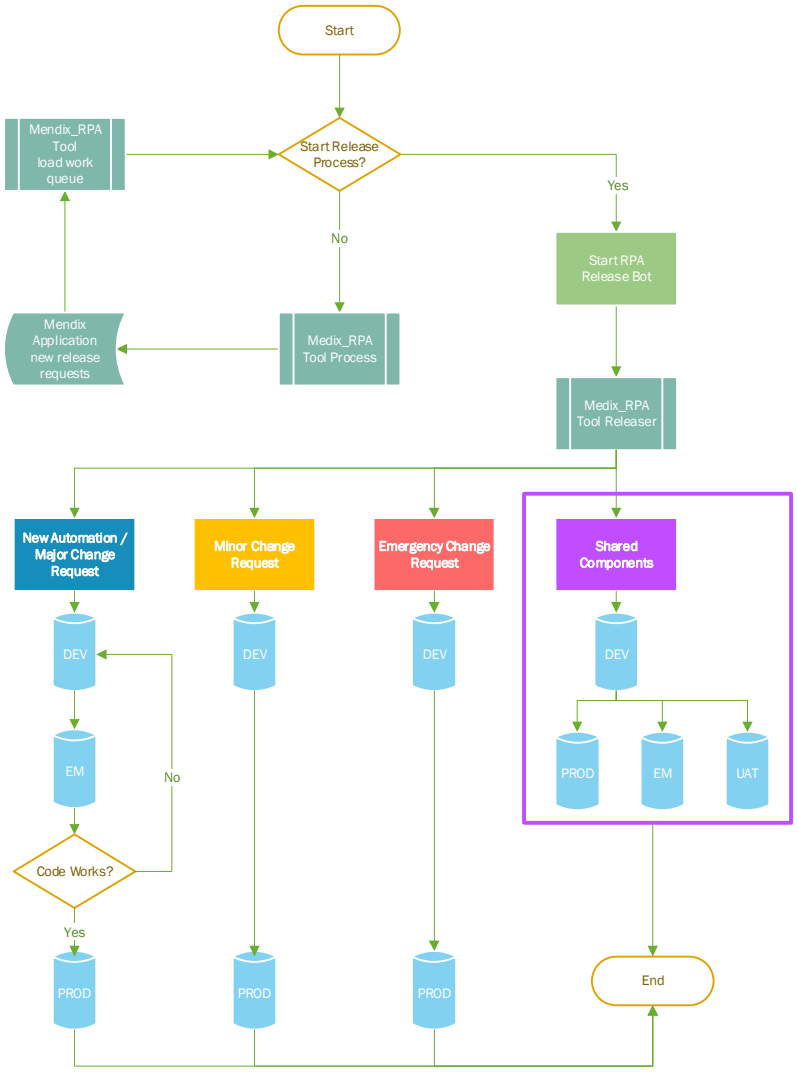


Figura 8 - Diagrama da ferramenta de automatização para o CI/CD da plataforma RPA na Siemens.

5 Análise ao processo de melhoria contínua – Levantamento de Requisitos

A Siemens GBS prima pela excelência nos seus serviços digitais e procura processos de melhoria contínua que permitam ajustar os seus serviços aos mais exigentes controlos de qualidade.

A garantia da qualidade é a potencialização da redução do erro ou falha face à entrega do serviço providenciado. No caso da RPA, os métodos utilizados visam acomodar qualquer desenvolvimento ao *Quality Assurance*, (QA), das aplicações alvo – sistemas que serão manipulados pela automatização RPA – e que por norma não traduzem o melhor ambiente possível para se desenvolver processos com a maior resiliência ao erro.

Desta forma é necessário criar outros mecanismos que possam ajudar a criar processos com a melhor qualidade. O aumento da automatização da segurança no ciclo de desenvolvimento reduz o risco de erros e o perigo de má administração, o que poderia levar inadvertidamente a ataques ou paragens no serviço RPA. Listando a seguir, os requisitos a considerar na solução a propor, quer ao nível funcional como não funcional.

5.1 Requisito de Negócio 1: Melhoria na qualidade de entrega – code review

Uma revisão de código ou *code review* visa melhorar a qualidade do produto final, neste caso, abordaremos o código RPA. Trata-se de uma abordagem sistemática para rever o código de outros programadores quanto a erros e muitas outras métricas de qualidade. Além disso, uma revisão de código verifica se todos os requisitos foram implementados corretamente. Este processo deve ser planeado e executado numa fase inicial do desenvolvimento, o *timing* é preponderante, a revisão deve ser antecipada o quanto antes, pela razão de que um *code review* tardio e não planeado com antecedência, tem uma maior probabilidade de ser forçada a entrada em produção de novo código e os problemas tenham que ser tratados quando os robôs já estiverem em ser executados em produção, o que obviamente traz as usas consequências.

É crucial definir o processo de revisão corretamente. Na pior das hipóteses, as revisões de código podem parecer um obstáculo. Na melhor das hipóteses, as revisões de código ajudam a manter o desempenho de equipa de desenvolvimento nos níveis de excelência de qualidade.

E o que dizer em relação à segurança, deve também ser integrada nesta fase?

A automatização de processos robóticos na Siemens lida com muitos dados confidenciais. Os robôs de software de RPA processam informações de vários sistemas da empresa e fazem login em diferentes contas usando credenciais fornecidas para automatizar tarefas diárias de negócios, como transferência de ficheiros, processamento de pedidos e realização de pagamentos de salários. Nesta abordagem, a plataforma de automatização tem acesso às informações dos funcionários, clientes e fornecedores da empresa.

A segurança deve ser um foco em todo o ciclo de vida do desenvolvimento. É essencial regular os problemas de segurança da RPA com um conjunto de controlos especializados. Criar modelos de ameaças durante a fase de design, educar os *developers* sobre práticas de programação seguras e realizar revisões frequentes de código com o pessoal de segurança envolvido ajudará a aumentar a qualidade geral do código e reduzir o número de problemas relatados pela revisão de código seguro.

5.2 Requisito de Negócio 2: Melhoria contínua – testes contínuos

O desenvolvimento de sistemas robóticos é um processo contínuo. Não pode ser um evento único e deve evoluir para lidar com as fraquezas e ameaças. Normalmente, os códigos do *Robotic Process Automation* são concluídos na prioridade para concluir a implementação num ritmo maior, devido ao qual a segurança é, por vezes, descurada.

Verifica-se que a abordagem no desenvolvimento dos *use cases* – processos RPA para suporte ao negócio - não seguem uma linha de melhoria contínua. O desenvolvimento dos objetos RPA fica estanque nas versões das aplicações a automatizar à data em que foi desenvolvido pela primeira vez. Mesmo que outro objeto seja desenvolvido para uma versão da mesma aplicação

mais recente, não são efetuados esforços para atualizar o mesmo objeto nos *use cases* já existentes.

Uma abordagem sem planeamento para melhorias contínuas potencia os riscos para a continuidade do negócio, mais especificamente, em caso de um volume grande de objetos estarem baseados numa versão antiga de uma dada aplicação.

É possível descrever este risco num simples exemplo, a falta de testes inseridos na melhoria contínua permitirá que existam discrepâncias nos objetos utilizados em relação às aplicações alvo, (*target applications*). Ou seja, um objeto RPA criado para um específico *browser* de forma a suportar a automatização a uma dada aplicação web. Se esta aplicação passar a ser suportada somente noutro tipo de browser, a automatização não estará preparada para funcionar com outro *browser*, o que quer indicar que para toda e qualquer aplicação que é sujeita a atualizações de segurança e consequentemente upgrades de versões, é necessário ir acomodando a automatização à nova realidade, de forma a não permitir automatizações ineficazes ou pior ainda, forçar que a plataforma RPA mantenha a suas aplicações desatualizadas para poderem funcionar com objetos que não foram alvo de um processo de melhoria contínua. Desta forma, cria-se uma abertura a vulnerabilidades no que concerne à segurança. Em suma, se não existir uma melhoria nos processos RPA que se encontram dependentes deste browser, um grande volume da operação poderá parar, o que é notoriamente uma das situações que deveria estar incluída numa metodologia de testes contínuos a fim de procurar soluções atempadamente.

Não existe nenhuma automatização RPA que não seja afetada pelo tempo, a cada semana surgem novas tecnologias no mercado e a Siemens acompanha as atualizações necessária para que a sua infraestrutura permaneça segura, seguindo a mesma lógica, para evitar a existência de software já não suportado e em fim do seu ciclo de vida. Por conta destas mudanças e novidades, é preciso perceber o desenvolvimento na RPA como algo mutável.

5.3 Requisito de Negócio 3: Melhoria na qualidade de entrega – Operações

Como parte de uma estrutura de governança de automatização de processos robóticos, são necessárias análises regulares de risco e auditorias das atividades de processamento de RPA. Os funcionários sob responsabilidade do serviço RPA devem ser claros sobre suas responsabilidades de segurança, que incluem gerir o acesso ao ambiente de automatização de processos robóticos, registar e monitorizar as operações e assim por diante. Deve haver deveres definidos para a realização de avaliações regulares da conformidade de segurança da informação da RPA e uma *checklist* de requisitos de segurança para as tecnologias de automatização de processos robóticos em vigor. A respetiva catalogação dos níveis de Confidencialidade, Integridade e Disponibilidade, (CIA), de cada processo RPA, deve ser tida em conta de forma a agilizar a identificação de riscos em consequentes auditorias de Controle Interno sobre Relatórios Financeiros (ICFR).

No paradigma dos acessos concedidos aos robôs RPA, a abordagem de *"least privilege"* determina que o acesso do robô a outras aplicações e sistemas seja limitado ao necessário para executar tarefas. Os danos no caso de um ataque, são minimizados ao restringir o número de aplicações ou sistemas aos quais os robôs de RPA têm acesso. Este cenário é especialmente crítico no caso de um ciberataque para impedir que os *hackers* executem várias aplicações numa máquina cliente e usurpar direitos de administrador local para instalar *spyware* e outros *malwares*. Todas as permissões de acesso codificadas diretamente nos scripts do código RPA, (*hard-coded*), devem ser substituídas por pedidos de API, com cada solicitação vinculada diretamente aos direitos de acesso necessários armazenados num repositório central. Isso adiciona outra camada de proteção, tornando um ataque menos provável.

Se a segurança na plataforma RPA falhar, os *logs* das operações necessitarão de ser examinados e revistos pelas equipas de TI e segurança. Os *logs* da automatização de processos robóticos devem ser guardados num sistema separado de forma a proteger a sua segurança e integridade forense.

6 Proposta aos processos de melhoria contínua

6.1 Especificação de Requisitos da Solução

Após uma análise das questões envolvidas neste projeto, obteve-se assim uma lista de requisitos da solução a apresentar, e que foram considerados importantes para procurar a abordagem mais correta para o problema identificado. Estes requisitos são apresentados de seguida em forma tabular, contendo uma breve descrição, a sua prioridade e dificuldade estimadas de implementação.

*Tabela 6 - **Requisitos Funcionais***

RF		Prioridade	Dificuldade
1	Entrega de novos RPA *use cases* para Produção - Pretende-se incrementar maior qualidade ao artefacto entregue, criando um novo processo complementar para *code review*. Com a ajuda de novas soluções automatizadas, é possível fazer rastreio complementar ao nível da qualidade do código RPA entregue por cada *developer*, estabelecendo mais dados de análise que possibilitam identificar pontos de melhoria e criar uma maior transparência no trabalho desenvolvido.	Alta	Média
2	Change Management para RPA use cases em Produção – Considera-se fulcral a alteração ao *workflow* para o *change management*. Sendo necessário redefinir as prioridades no que concerne a alterações em emergência. Facilitar a metodologia para o envolvimento de testes conjuntos com as operações ainda em pré-produção (*DevOps*), de forma a existir um levantamento correto dos requisitos.	Alta	Média

*Tabela 7 - **Requisitos não Funcionais***

RNF	Descrição	Prioridade	Dificuldade
1	Integração do *DevSecOps* na entrega continua - Pretende-se conceber uma alteração ao modelo de operação de forma que seja possível alinhar os processos CI/CD com práticas *DevSecOps*. Aliar a segurança como aspeto importante durante e após desenvolvimento do artefacto.	Média	Alta
2	Integração do *DevSecOps* no Change Management – Após alteração no requisito funcional 2, será importante criar práticas de *DevSecOps* para o *change management*, envolvendo melhorias no que concerne aos processos de testagem do código RPA e mitigação de obstáculos, como por exemplo, cultura de práticas de desenvolvimento já obsoletas ou resistência à mudança do paradigma operacional.	Média	Alta

6.2 Desenho e Implementação

A Figura 9 demonstra uma visão global da nova abordagem RPA com *DevSecOps* e das partes com as quais interage, aplicada ao caso de estudo neste projeto. Neste diagrama, é tida em conta os aspetos de *Quality Assurance* logo numa primeira abordagem. Ao contrário do fluxograma da Figura 8, é identificado numa primeira etapa os *quality gates* necessários para que se tenham em conta aspetos na formalização dos testes e melhoria continua, como também reduzir o risco de segurança da informação devido ao persistente e recorrente *bug fixing* após entrega da automatização para produção. É necessário precaver todo e qualquer tipo de duplicação de código existente, nomeadamente código referente a objetos RPA. A eficiência neste processo, garante uma padronização no contexto de funcionalidades do ecossistema RPA e permite aplicar uma maior eficiência na entrega. Permitindo assim, melhorias ao código existente sem redobrar esforço e complexidade na manutenção de todo o repositório de código RPA.

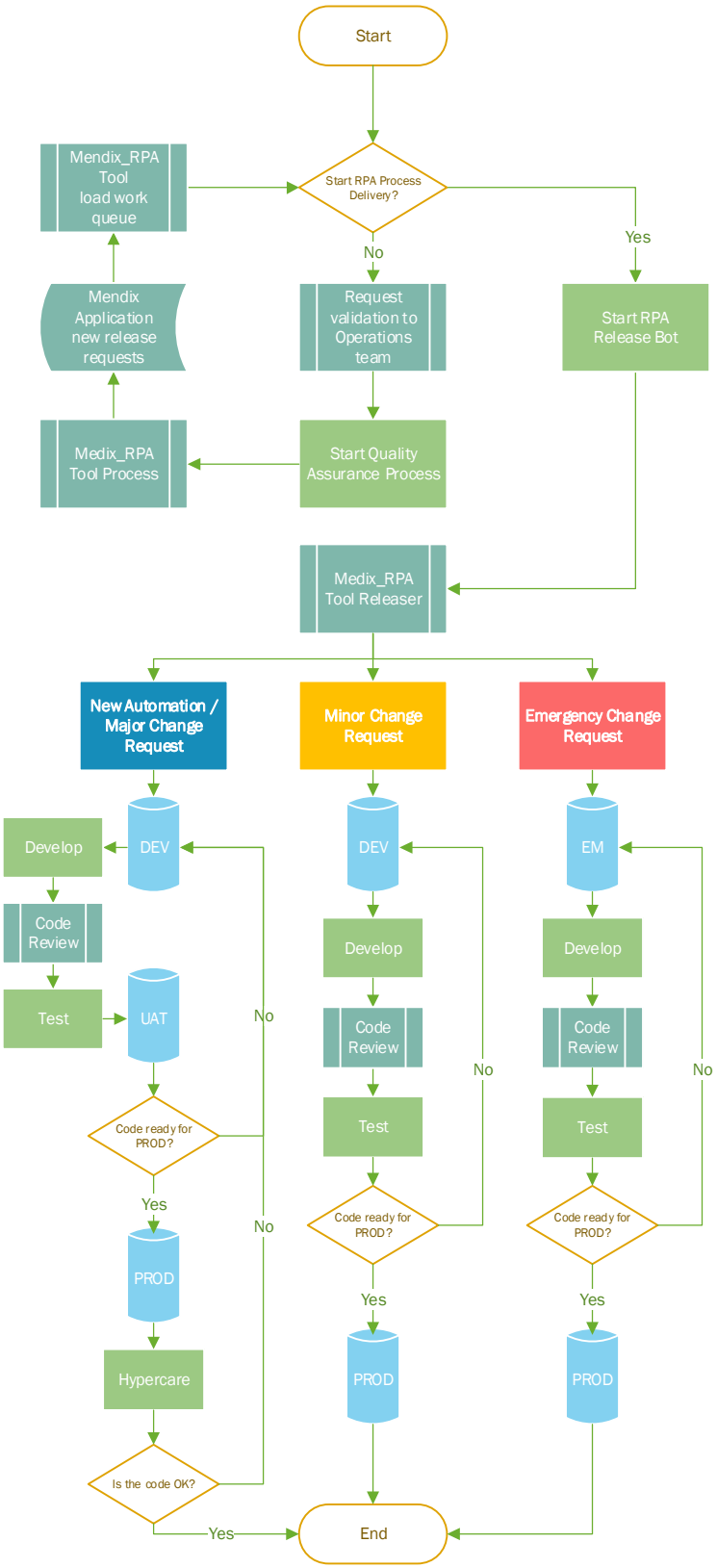


Figura 9 - Visão global do novo modelo CI/CD para processos RPA

Neste tópico são descritos os mecanismos e procedimentos relevantes para a obtenção de uma solução com as características pretendidas.

6.2.1 Requisito Funcional 1: Entrega de novos RPA use cases para Produção

O processo de desenvolvimento de novas automatizações RPA, necessitam numa primeira instância de envolver um maior critério no que concerne à segurança e qualidade do código. Um código mais seguro é um código com maior qualidade. As ferramentas automatizadas de revisão de código são essenciais para padronizar e dimensionar os esforços de desenvolvimento do código RPA. É necessário rever o *script*/código RPA o mais cedo possível, para que o tempo despendido no desenvolvimento não seja em vão e de forma a minimizar as chances de se refazer o código novamente.

A empresa Blue Prism dispõe de uma *checklist,* (Anexo I e II), para verificação do código RPA de forma manual. Este processo manual é moroso e tende a ser tendencioso, dado que certos aspetos poderão ser descurados numa tentativa de serem somente corrigidos após entrada do código em Produção. O *technical architect* é o responsável pela formulação do *Technical Code Review document*, mas não existe um processo isento e sem intervenção manual que permita recolher o mais fidedigno relatório. Não obstante, as revisões manuais continuam a ser importantes em aspetos funcionais que poderão ser identificáveis numa primeira instância. Mas para a formulação do TCR, será importante a validação dos requisitos funcionais e de segurança de forma autónoma. Esse processo autónomo permite reduzir falhar e incrementar eficiência a todo o processo de revisão de código.

Existem no mercado soluções para revisão do código RPA de forma rápida e automática, como por exemplo, RoboReview do grupo Reveal, CodeIQ da Binaryway, entre outras. Estas ferramentas potenciam a resposta à qualidade e podem ser usadas juntamente com *checklists* internas. Com esta abordagem, podemos obter um maior controlo durante o desenvolvimento e receber inputs valiosos para a melhoria contínua, como também, receber alertas de riscos de segurança que nos permitirão agir de forma mais rápida. Conciliando com novas ferramentas,

é possível reduzir o esforço de entrega até 30% e garantir um aumento de qualidade significativo.

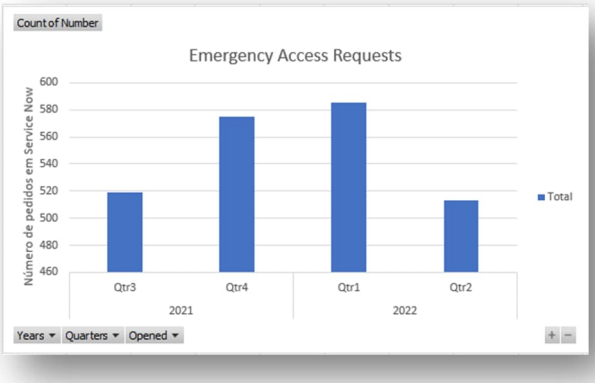
Na Siemens GBS, o processo de entrega de novos RPA uses cases para produção, por vezes, não seguem uma metodologia Lean, ou seja, uma forma de otimizar as pessoas, recursos, esforço e energia de uma organização para criar valor para o cliente. É importante perceber se existe eficiência e produtividade em todo este processo. No que foi possível analisar, existe um desperdício de tempo e recursos no que concerne ao desenvolvimento de novos use cases. É constante o recurso a adaptações após entrega em produção, numa medida de mitigação de problemas, onde o processo de desenvolvimento foi descurado no respeita a um *workflow* correto e preponderado para entrega de código RPA com maior qualidade. Os dados alvo de análise foram relacionados com os pedidos de acesso para desenvolvimento/*debugging* em Emergência, pelo que foi possível apurar, ver Tabela 8 e Gráfico 1, quase todos os novos códigos RPA entregues para produção, recorre a alterações após "*go live*", visto que o número de alterações é muito acima do use cases em produção. O que reflete uma inconsistência na qualidade de entrega. As razões que elevam a recorrer a alterações de emergência são muito variadas, mas serão analisadas nos subcapítulos seguintes.

Tabela 8 - ***Contagem dos pedidos abertos para acessos em Emergência***

Data	Número de pedido de Acesso a Emergência
⊟2021	1094
⊞Qtr3	519
⊞Qtr4	575
⊟2022	1098
⊞Qtr1	585
⊞Qtr2	513
Total	2192

Estes números foram recolhidos do sistema ticketing Service Now da Siemens GBS, Julho 2021 a Junho 2022.

Gráfico 1 - ***Pedidos de Emergência por Trimestre (Julho 2021 a Junho 2022)***



Desta forma e como ilustra a Figura 10, o processo de entrega para produção, mesmo seguindo uma automatização dentro do conceito CI/CD, deve em certa medida, respeitar princípios de eficiência, tendo sempre em conta a superprodução e manter a redução no que respeita o *Cost Of Poor Quality*, que é o custo da baixa qualidade, denominado como gasto extra que a empresa tem quando não aplica o Lean nos processos, sistemas, fluxos e serviços atuais.

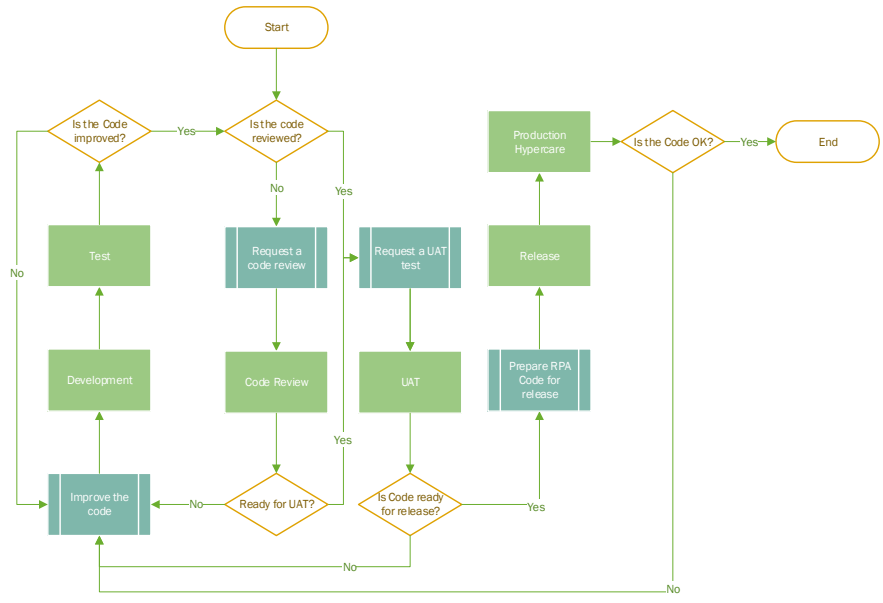


Figura 10 - Fluxograma representativo de entrega de código RPA para Produção

Uma estratégia a adotar será a acomodação de melhores práticas de teste, sendo que um dos maiores desafios será as diferenças existentes entre os ambientes de qualidade e de produção. Será importante tentar replicar ao máximo estes ambientes, as diferenças não deverão ser substanciais, mas é necessário ter em conta que as aplicações alvo continuaram a ter diferenças no que respeita aos ambientes de qualidade e produção.

Uma solução que poderá ser adotada, é a adaptação de testes RPA com acessos controlados a aplicações alvo em produção. Neste âmbito, o que se pretende é reduzir as diferenças entre ambientes sem conceder a possibilidade de desenvolvimento diretamente em pós-produção RPA.

O que se pretende com esta abordagem é rastrear ao máximo os acessos e minimizar a partilha de credenciais ou uso de contas de automatização por terceiros. É possível analisar o seguinte, numa análise do risco de segurança, no caso de não ser possível criar um ambiente de qualidade fidedigno e eficiente, é preferível atribuir um acesso de um sistema alvo em produção diretamente ao *developer*, do que permitir que o mesmo utilize uma conta da automatização para executar alterações necessárias ao processo RPA.

Sempre que um membro de uma equipa utilize acessos que não os seus para execução das suas tarefas, cria um problema de rastreabilidade e necessidade de aplicar o "principio de 4 olhos" – o acompanhamento de um terceiro durante a execução do acesso – de forma a controlar toda e qualquer alteração no código após produção. Este processo é limitativo e ineficaz, sendo preferível rastrear e controlar o acesso direto do *developer* a sistemas de produção, até para aspetos de auditoria, pois todo o *logging* será registado identificando o utilizador do *developer* em vez de apenas referir um utilizador genérico, como por exemplo VirtualWorkForce1.

6.2.2 Requisito Não Funcional 1: Integração do DevSecOps na entrega contínua

A segurança deve acompanhar o processo de desenvolvimento do início ao fim, bem como a identificação das funções de operações devem estar também definidas, de forma que seja possível adaptar ou acomodar no código RPA apresentado, soluções aplicadas no conceito da

operação. Devem ser executados testes mais próximos do ambiente de produção, não apenas com contexto aplicacional, mas também em todo o ecossistema envolvente. Este ponto é primordial na acomodação de noções de segurança, respeitar o *framework* de segurança existente na Siemens, serve acima de tudo como balizador no que toca ao escrutínio de regras de segurança seguidas e permite seguir um guia complacente nas melhores práticas de segurança.

No contexto de *DevSecOps*, a equipa de desenvolvimento deverá estar a par deste *framework* de segurança e cooperar proactivamente com a equipa de segurança. No caso de algumas dúvidas existentes no próprio serviço RPA, é sugerido que envolvam as operações durante o processo de desenvolvimento para que o primeiro *assessment* ao nível de infraestrutura e segurança seja executado a fim de levantar potenciais riscos.

6.2.3 Requisito Funcional 2: Change Management para RPA use cases em Produção

O serviço RPA na Siemens necessita de várias acomodações e constantes alterações após entrada em produção, por isso, é crucial entregar novas automatizações com uma maior segurança e qualidade. Para tal, seguindo o ponto acima abordado, é importante criar-se transparência na entrega para produção. Os testes deverão seguir os requisitos necessários para entrada em produção. A execução de testes em contextos diferentes que não a simulação do mesmo ambiente onde a automatização irá correr em produção, poderá por si só, criar discrepância na efetividade da automatização no ambiente produtivo, ou até, descurar certas vulnerabilidades de segurança. De facto, o código não "sobrevive" apenas por ser funcional, deverá considerar os aspetos inerentes na qual a automatização vai correr, como por exemplo, arquitetura do sistema, versões de sistemas operativos, configurações de infraestrutura e rede, aspetos relacionados com autorizações e autenticações em ambiente de produção, nomeadamente autenticações seguras e uso de "*vaults*" para guardar a credenciais sem recursos a autenticações *hard-coded*. Quanto mais próximo for o teste da realidade em produção, maior será a qualidade de entrega.

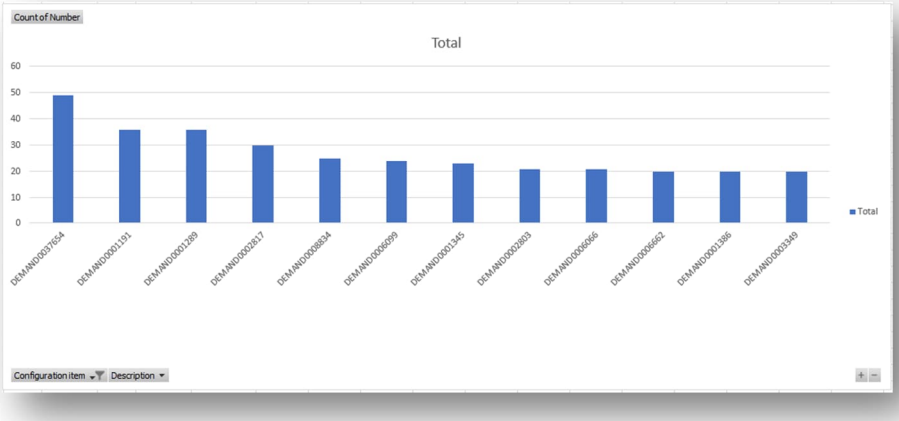
Todo este processo de verificação e teste deve ser sempre considerado de forma dinâmica e continua. No que toca aos códigos RPA já desenvolvidos e entregues, por vezes, ficam estanques no tempo e não acompanham as alterações constantes dentro do ecossistema da RPA. A qualidade e segurança não podem nem devem ser garantidas somente na primeira fase de entrega, é necessário recorrer a processos de melhoria contínua para que processos que já tenham sido entregues possam também ser revistos e aprimorados nos aspetos chave. Os objetos RPA que comportam manipulações em sistemas antigos, (*target applications*), devem ser alvo de verificação e acomodação de novas alterações a fim de poderem adaptar-se a atualizações desses sistemas, sem correrem o risco de causarem impacto por falta de adaptação novas funcionalidades ou meras alterações de versão que poderão causar a paragem da automatização.

Este, tem sido um ponto bastante debatido, o esforço de rever o código RPA já entregue poderá ser grande, mas permitir que a operação seja afetada por alterações nos sistemas alvo causa, por norma, um maior esforço, tanto financeiro como operacional, para acomodar em tempo recorde as alterações necessárias a fim de levantar o serviço novamente e mitigar o impacto gerado.

6.2.4 Requisito Não Funcional 2: Integração do DevSecOps no Change Management

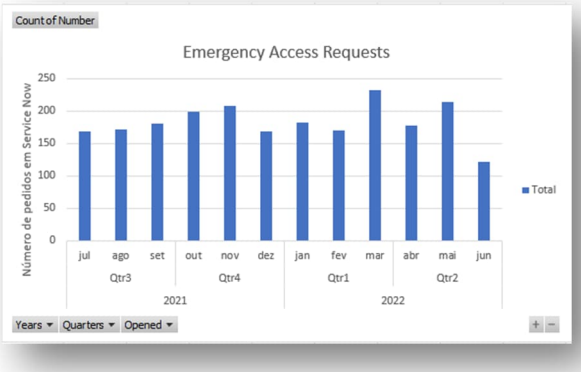
As alterações constantes do código após entrada em produção, criam em certa parte, um potencial risco no que concerne à segurança. Sempre que é necessário alterar o código após entrada em produção, cria um maior desafio no controlo de acessos, nomeadamente, por se tratar de um ambiente produtivo com acesso às aplicações alvo de sistemas em produção. No Gráfico 2, podemos verificar os 12 RPA *use cases* – num total de 321 – com mais alterações desde Julho de 2021 a Junho de 2022.

Gráfico 2 - ***Top 12 dos use cases com mais alterações entre Julho 2021 e Junho 2022***



A maioria dos meses tem um volume significativo de pedidos de acesso a Emergência, tal como ilustra o Gráfico 3. Estes pedidos poderão também englobar o *Hypercare* – o use case entra em produção, mas requer uma monitorização efetiva por parte do *developer* no que respeita às primeiras execuções – normalmente são executadas várias alterações durante este período, o que representa vários pedidos de *debugging* em produção até estabilizar o RPA use case. Pode levar um mês ou mais, dependerá muito da recorrência do agendamento desta automatização, certas automatizações correm uma vez por semana, quinze em quinze dias ou apenas no final do mês.

Gráfico 3 - ***Contagem dos pedidos de alterações em EM por mês***



Será importante rever os controlos de testagem por parte dos arquitetos ou *senior developers*, pois o facilitismo de *debugging* em produção cria, de forma crescente, um risco ao nível de segurança. Durante uma alteração ao código deve-se ter em conta os aspetos fundamentais da segurança, não só olhando ao código desenvolvido, mas também a todo o processo inerente ao desenvolvimento de uma alteração a um RPA *use case*, como por exemplo, necessidade de entregar o código para produção sem o processo de testagem correto, ou o não envolvimento do cliente para mitigar obstáculos que poderão ocorrer durante o desenvolvimento, leva em certa parte, a um potencial risco de *debugging* em pós-produção. Os ambientes de teste deverão simular ao máximo todas as potenciais funcionalidades e caso não seja possível aplicar todos os requisitos funcionais, deve a equipa de desenvolvedores, envolver logo em primeira instância a equipa de operações.

7 Validação da Proposta

Para validação dos requisitos funcionais e não funcionais propostos, foram criadas duas abordagens:

- Avaliação do método *DevSecOps* abordado por via de um questionário.
- Criação de um PoC para testar a aplicabilidade da solução proposta de *code review*.

Numa primeira abordagem, foi importante efetuar um questionário à equipa de desenvolvimento e operações, na medida de procurar perceber qual a perspetiva de cada membro das equipas, quanto à qualidade entregue dos artefactos de software como também analisar o entendimento e relação entre *DevOps* e *DevSecOps* no contexto do serviço do RPA na Siemens GBS.

Estas equipas encontram-se distribuídas por diferentes países, India, Turquia, Brasil, Estados Unidos da América, Portugal e Chéquia. O inquérito foi enviado por email a 45 pessoas tendo sido obtidas 30 respostas. A taxa de sucesso foi de 66,66%, ou seja, a amostra representa a maioria da população. Os resultados pormenorizados ao inquérito efetuado podem ser consultados no Apêndice I.

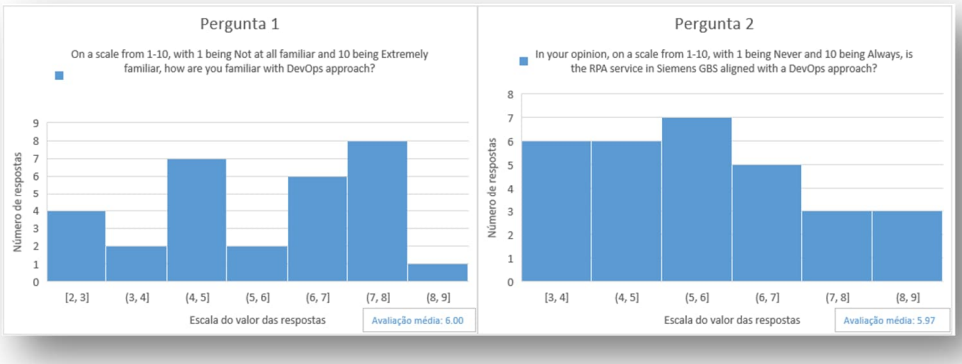
Na segunda abordagem, foi verificada a aplicabilidade da solução proposta para *code review*. O software escolhido foi o RoboReview do grupo Reveal. Criado para utilizadores RPA, o RoboReview fornece uma plataforma *cloud*, SaaS, que permite efetuar um *code review* do código RPA de forma proativa. O utilizador ao fazer upload do seu código, recebe em poucos segundos um relatório pormenorizado de acordo com as práticas recomendadas de automação inteligente. Pode também, consultar os *dashboards* disponíveis para avaliar a melhoria continua do seu processo de desenvolvimento.

Os dados apresentados nos *dashboards* do RoboReview, apresentam um score relativo à qualidade do produto desenvolvido, baseando-se nos seguintes critérios, capacidade de gestão, reutilização, escalabilidade, segurança, estabilidade e compreensão do código.

7.1 Análise em detalhe aos Resultados do Questionário

O questionário teve por base avaliar o contexto atual do modelo de operações, visando áreas como desenvolvimento, operações e segurança dentro do serviço RPA da Siemens GBS. Nas primeiras duas perguntas, efetuou-se uma sondagem sobre qual o envolvimento das equipas com práticas de *DevOps*. Tal como demonstra o Gráfico 4, a grande maioria dos inquiridos sente-se familiarizada com as práticas de *DevOps*, cerca de 80%, avaliando de moderadamente a extremamente familiarizados, as respostas estiveram por base uma escala de 1 a 10, onde 1 corresponde a "não familiarizado" e 10 a "extremamente familiarizado". Mas acaba por não ter grande aplicabilidade no dia a dia, tanto no desenvolvimento como nas operações, cerca de 40% do total dos inquiridos avaliaram a sua aplicabilidade como raramente a moderado, as respostas estiveram por base uma escala de 1 a 10, onde 1 corresponde a "nunca" e 10 a "sempre".

*Gráfico 4 - **Análise aos dados recolhidos das questões 1 e 2***



Em relação à segurança, Gráfico 5 e 6, na pergunta 3 os inquiridos demonstram que são preocupados com o tema da segurança no RPA, apenas 10% avaliaram como não sendo um tema com o qual se preocupam. Na pergunta 4, em relação à segurança no código RPA, 20% avaliam como pouco seguro a moderado. Relativamente à pergunta 6, os inquiridos avaliaram-se como tendo um conhecimento geral sobre o *framework* de segurança de TI estabelecido na

Siemens ao nível global. E na pergunta 6, também consideram que a plataforma RPA na Siemens é segura.

*Gráfico 5 - **Análise aos dados recolhidos das questões 3 e 4***

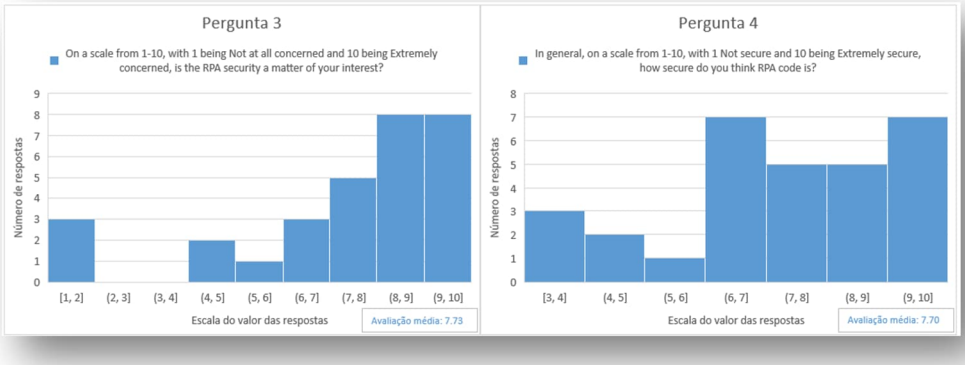
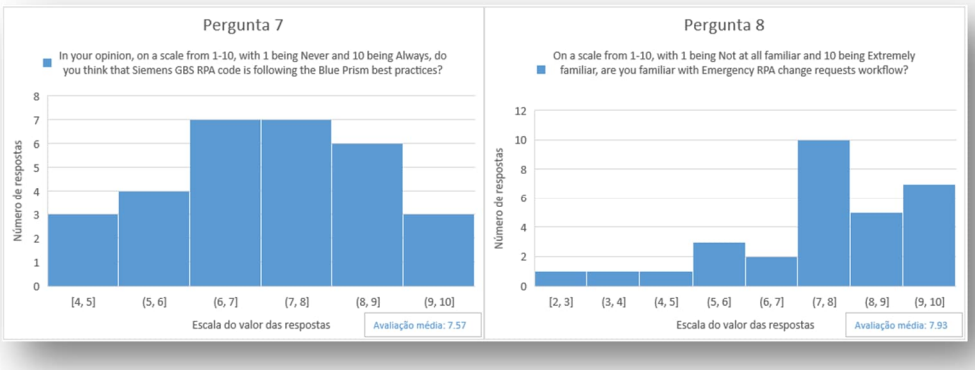


*Gráfico 6 - **Análise dos dados recolhidos das questões 5 e 6***

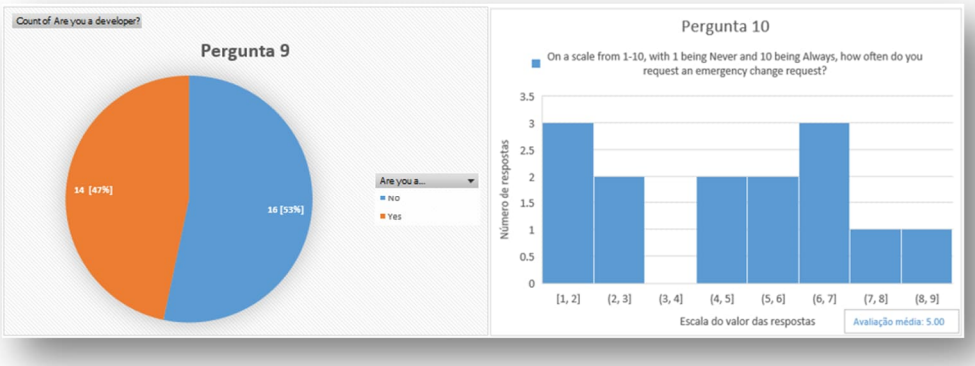
No conceito geral, baseada numa opinião, as respostas à pergunta 7 e 8, Gráfico 7, demonstram que existe uma ideia estabelecida de que o código RPA na Siemens segue as melhores práticas de desenvolvimento sugeridas pela Blue Prism. E que os inquiridos têm um entendimento geral sobre o processo para pedidos de alterações ao código RPA em Emergência.

Gráfico 7 - ***Análise aos dados recolhidos das questões 7 e 8***



Na pergunta 9 e 10 elaborou-se um levantamento de quantos *developers* estariam a responder a este inquérito e quantos destes *developers* utilizam o ambiente de emergência para *debugging* após entrada do código em Produção. Sendo visível no Gráfico 8, de que 47% respondeu que é *developer*, e na pergunta 10, 64.28% dos *developers* inquiridos afirma que pedem acesso ao ambiente de emergência de forma moderada a regular, para desenvolvimento dos seus RPA *use cases*.

Gráfico 8 - ***Análise aos dados recolhidos das questões 9 e 10***



Na pergunta 11, destinada a todos os inquiridos, elaborou-se uma sondagem acerca de quais as razões que, na opinião dos inquiridos, levaria a mais pedidos de emergência para resolução de falhas na execução do código RPA em produção. Dentro do ecossistema da plataforma RPA, foram então levantados vários aspetos ou razões que podem eventualmente determinar que o código RPA seja alterado num ambiente de emergência, após entrada em produção. Após inquérito foi criada a seguinte lista:

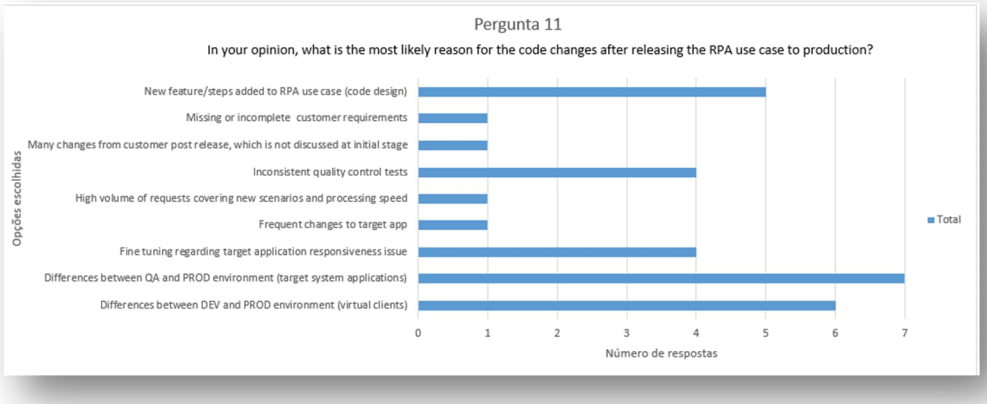
- Testes de controle de qualidade inconsistentes.
- Diferenças entre os ambientes de desenvolvimento e produção no contexto dos clientes virtuais.
- Diferenças entre os ambientes de qualidade e produção no contexto das aplicações alvo.
- Ajuste do código em relação ao problema de resposta da aplicação de destino ou alvo.
- Novas funcionalidades adicionadas ao design do código RPA.
- Muita complexidade adicionada ao código RPA.
- Outras opções colocadas pelos inquiridos:
    - Requisitos do cliente ausentes ou incompletos.
    - Muitas mudanças do cliente após entrega em produção e que não foram discutidas numa fase inicial.
    - Alto volume de novos *change requests* que envolvem novos cenários e velocidade de processamento.
    - Alterações frequentes na aplicação alvo.

No Gráfico 9, pode-se analisar o valor dos resultados e verificar quais as razões com maior relevo. Sendo que as três principais razões atribuídas pelos inquiridos, por ordem decrescente, são: diferenças entre os ambientes de qualidade e produção no contexto das aplicações alvo, diferenças entre os ambientes de desenvolvimento e produção no contexto dos clientes virtuais, e por último, novas funcionalidades adicionadas ao design do código RPA.

Outras das evidências relevantes são, razões inerentes aos testes de controle de qualidade inconsistentes e ajuste do código em relação ao problema de resposta da aplicação de destino ou alvo, ambas com o mesmo número de respostas.

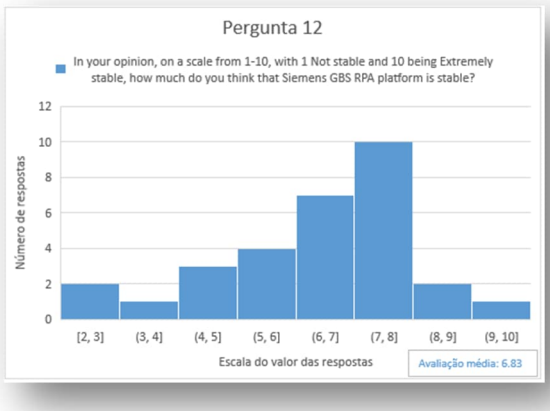
Existem outras razões também escolhidas, dado a pergunta permitir ao inquirido colocar as suas razões, opção outra. O âmbito das mesmas, espelha o que já foi referido quanto ao levantamento de requisitos necessário para desenvolver o código, a fase de UAT não abordada de forma correta ou a necessidade constante de alterar o código RPA pelo motivo da aplicação alvo ser alterada constantemente.

Gráfico 9 - ***Análise aos dados recolhidos da questão 11***



Por último, foi abordada a estabilidade da plataforma RPA na Siemens GBS, os inquiridos estabelecerem uma avaliação tendencialmente positiva, somente 20% dos inquiridos avalia que a plataforma RPA é pouco a moderadamente estável.

Gráfico 10 - ***Análise aos dados recolhidos da questão 12***



7.1.1 Análise global ao Questionário

Apesar das equipas envolvidas no serviço RPA da Siemens GBS, possam ter conhecimentos de práticas de *DevOps*, não se verifica que têm criado uma metodologia que permita a colaboração de várias equipas, de forma que o paradigma, no que se refere ao desenvolvimento de uma automação inteligente, possa beneficiar de um novo contexto e de um melhor aproveitamento de recursos, a fim de mitigar problemas já debatidos nos capítulos anteriores deste trabalho.

A segurança é um tema sério para as equipas envolvidas no serviço RPA da Siemens GBS, mas verifica-se que quanto à sua aplicabilidade, as coisas poderão não ser tão lineares, dado que o cumprimento de tais práticas seguras, num envolvimento *DevSecOps*, necessitaria de um levantamento de requisitos colaborativo, de menor práticas de desenvolvimento/*debugging* em pós produção ou mesmo prevenir o desencadear de um fluxo elevado de pedidos para adaptações do código a novas funcionalidades, quando nesse cenário, o que é recomendável é o retrocesso do código para a fase de desenvolvimento, seguindo o fluxograma de acordo com o seguinte ciclo, desenvolvimento, revisão de código, teste, UAT, produção, tal como ilustrado na Figura 10, (pág. 43).

Um dos aspetos importantes de referir está relacionado com as diferenças entre o ambiente de qualidade e de produção tanto ao nível de infraestrutura como aplicacional. No caso da infraestrutura, verifica-se que os testes podem ser efetuados num maior compromisso com o que existe em produção, ou seja, os resultados dão a transparência de que a fase de testes não é tão aprimorada, pois o próprio developer acaba por preferir executar *debugging* em Emergência do que explorar todas as alternativas existentes para testar o *use case* com maior efetividade. Em relação às diferenças das aplicações em qualidade e produção, é sugerido um método mais *Agile*, num acompanhamento mais próximo do cliente a fim de mitigar estas diferenças e reduzir as chances de *debugging* no pós produção.

7.2 Análise ao PoC com o RoboReview

Na análise executada ao processo de *code review*, e após avaliação às soluções disponíveis no mercado, verificou-se que o RoboReview apresenta ser uma opção viável para validação da solução proposta. Trata-se de uma plataforma *cloud*, onde é minimizado o tempo para implementação e recorre-se a um serviço estável de acordo com os requisitos propostos para desempenho de um *code review* aos *release packages* do Blue Prism. O RoboReview fornece uma plataforma SaaS para revisão do código RPA, para que se obtenha feedback de acordo com as práticas recomendadas de Automação Inteligente. Criado para utilizadores de RPA, especificamente para código de automações Blue Prism. O RoboReview foi desenvolvido pela Reveal Group usando a experiência, as melhores práticas e os padrões de qualidade desenvolvidos ao fornecer milhares de soluções automatizadas para empresas em todo o mundo.

Numa primeira análise, efetuou-se uma demonstração da solução juntamente com o fabricante do software, Reveal RoboReview. Na sessão exploratória, foi possível identificar as funcionalidades principais da solução, como também, a agregação de valor ao incluir um serviço que, além de executar a revisão do código de forma automática e célere, disponibiliza relatórios minuciosos que possibilitam a identificação de pontos de melhoria. Também avalia o desempenho da equipa ou de forma individual, relativamente à qualidade do código desenvolvido, permitindo assim publicar *dashboards* com todas as métricas necessárias, para que seja possível ter uma visão 360º sobre todos os aspetos importantes que determinam o valor da qualidade do código desenvolvido.

No âmbito da validação da solução proposta, foram utilizados os 12 RPA *use cases* que têm um maior número de acessos ao ambiente de Emergência para correção de erros urgentes. Os *use cases* em análise, são os casos com mais incidências de *bug fixing* em produção, tal como revisto no Subcapítulo 6.2.4, Gráfico 2. A recolha desta amostra, permite avaliar de forma geral qual a qualidade do código desenvolvido, identificar pontos de melhoria, como também, avaliar os riscos de segurança identificados.

Após upload e análise aos dados da revisão do código dos RPA *use cases* na plataforma RoboReview, foi possível verificar o seguinte. Logo numa primeira instância, Figura 11, o score global de qualidade era baixo, cerca de 62%.

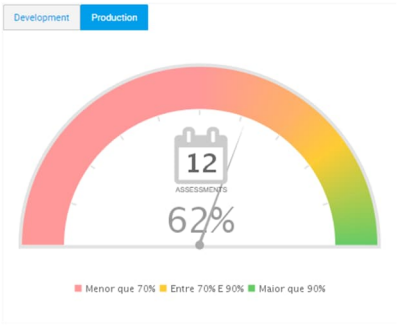


Figura 11 – Score global de avaliação do RoboReview

Analisando em detalhe os dados da avaliação global, Tabela 9, verifica-se que a maioria dos RPA *use cases* revistos estão abaixo dos 70%, sendo que apenas um atinge o score de 70%. Os resultados foram demonstrativos de que, em termos relativos, existe um problema de qualidade, o que é sugerível a uma análise mais aprimorada.

*Tabela 9 - **Score global por RPA use case***

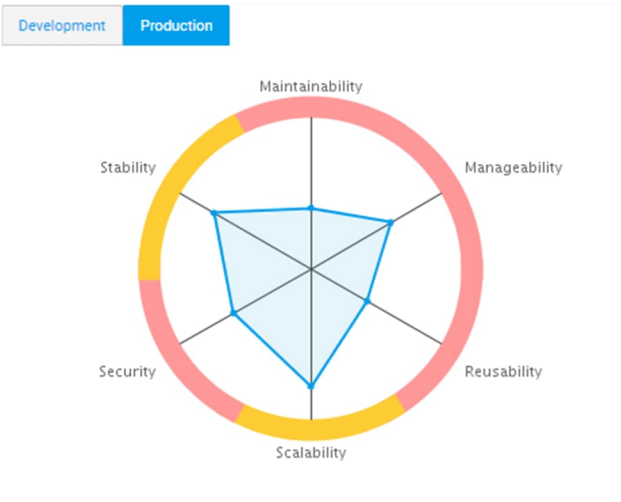
assessment_id	Date	Delivery Stage	Score
495114	13/jul/22	Production	60%
495384	13/jul/22	Production	59%
495657	13/jul/22	Production	66%
496199	13/jul/22	Production	70%
497010	13/jul/22	Production	63%
497565	13/jul/22	Production	58%
498225	13/jul/22	Production	59%
498899	13/jul/22	Production	64%
499480	13/jul/22	Production	58%
500317	13/jul/22	Production	62%
502160	13/jul/22	Production	61%
503997	13/jul/22	Production	58%

É necessário perceber quais os pontos de maior incidência negativa, de forma a ser possível delinear um plano de melhoria para mitigação das falhas encontradas. Os critérios a ter em conta são os seguintes, Capacidade de manutenção, Capacidade de gestão, Capacidade de reutilização, Escalabilidade, Segurança e Estabilidade. Analisando a Tabela 10, verifica-se que, os pontos que têm um impacto mais negativo assentam sobre um problema que é a segurança, são eles, capacidade de manutenção e reutilização, estes dois pontos impactam a segurança no que concerne ao facto de necessitarem de mais acessos ao ambiente de emergência para ser possível corrigir o *use case*. Cada acesso concedido para *bug fixing,* após entrada do código RPA em produção, aumenta o risco da segurança da informação. Numa visão mais periférica, obtemos o score radar do critério da segurança, que também não deixa de ser preocupante, Gráfico 11.

Tabela 10 - ***Score global por categoria***

Team	Category	Score
Siemens	Maintainability	40%
	Manageability	61%
	Reusability	43%
	Scalability	78%
	Security	59%
	Stability	74%

Gráfico 11 - ***Score Radar global por categoria***



Relativamente aos pontos de consideração levados a cabo pelo *code review* da solução RoboReview, encontram-se no apêndice II, os resultados completos com as pontuações mais baixas até às mais altas.

No que concerne à segurança, Tabela 11, verificou-se que é importante que o registo das ações e etapas do processo esteja de acordo com a Política de Segurança de TI, "*Process Logging Plocy Adherence*" e "*Object Logging Policy Adherence*". As filas de trabalho do processo automatizado, "*Process Work Queue*", necessitam de estar configuradas de forma a fornecerem a funcionalidade para armazenar, gerir, partilhar e relatar o trabalho do processo RPA. É imperativo, como parte de seu processo, que, quando as credenciais são usadas, necessitem de estar armazenadas no Blue Prism Credential Management ou num repositório igualmente seguro, "*Secure Password Storage*".

Tabela 11 - ***Score global sobre os critérios da Segurança***

Rank	Consideration Type	Category	Consideration	Score
1	Blue Prism Process Considerations	Security	BPP403 Process Logging Policy Adherence	0%
2	Blue Prism Object Considerations	Security	BPO404 Object Logging Policy Adherence	1%
10	Blue Prism Process Considerations	Security	BPP404 Process Work Queue	43%
18	Blue Prism Process Considerations	Security	BPP401 Secure Password Storage	60%
38	Blue Prism Process Considerations	Security	BPP402 No Hard Coded Passwords	100%
38	Blue Prism Object Considerations	Security	BPO403 Technology Attributes	100%
38	Blue Prism Object Considerations	Security	BPO402 No Environmental Data	100%
38	Blue Prism Object Considerations	Security	BPO401 No Customer Data	100%

No caso dos processos/objetos alvos de revisão, foram identificados alguns pontos críticos relacionados com a capacidade de manutenção, e que devem ser melhorados como parte do processo de melhoria contínua, nomeadamente no aspeto de capacidade de manutenção, Tabela 12. Nesse processo, é importante a conceção de um critério de melhoria contínua no que concerne aos códigos de automatização já enviados para produção, pois a plataforma RPA não pode ser considerada como algo estático que não necessita de alterações para acomodação de novas versões de *software*, como também novas funcionalidades.

Seguindo os padrões de melhores práticas, os processos devem ter numeradas as páginas que compões a sua lógica. Um processo é composto por uma ou mais páginas. Por padrão, quando se cria um processo, inicia-se apenas com uma página chamada *main page*, que é o ponto de partida do processo. Partindo da página principal, é possível criar várias páginas e fazer com que sejam relacionadas à página principal. A ideia de criar múltiplas páginas é abstrair alguma lógica/funcionalidade relacionada dentro do código RPA. Desta forma, uma boa estruturação permite, reduzir a complexidade de análise do mesmo código por outro *developer*, facilitando assim a sua manutenção.

Os blocos devem ser usados para agrupar *data items* (*container* que armazena os dados) e *Collections* que se encontram relacionados. Foi verificado que nem sempre esta prática foi seguida adequadamente. Normalmente, uma coleção (*collection*) será usada para recuperar um grande número de informações diferentes de um objeto de negócios, numa única transação.

Nem todos os objetos e ações seguem as melhores práticas relativamente ao *convention naming*. Pois, é necessário permitir que informações úteis sejam deduzidas dos nomes dos artefactos, de forma a demonstrar relacionamentos entre o nome e o âmbito a que se destina.

Tabela 12 - ***Score global sobre os critérios da Capacidade de Manutenção***

Rank	Consideration Type	Category	Consideration	Score
1	Blue Prism Process Considerations	Maintainability	BPP605 Block Grouping Used	0%
1	Blue Prism Process Considerations	Maintainability	BPP604 Page Numbering	0%
4	Blue Prism Object Considerations	Maintainability	BPO605 Block Grouping Used	5%
8	Blue Prism Object Considerations	Maintainability	BPO603 Object and Action Names	31%
23	Blue Prism Object Considerations	Maintainability	BPO602 Meaningful Documentation Precursors	71%
27	Blue Prism Object Considerations	Maintainability	BPO601 Proper Element Names	81%
29	Blue Prism Process Considerations	Maintainability	BPP601 High Level Main Page	84%

No que concerne à capacidade de gestão existe um ponto fundamental que deve ser melhorado tanto ao nível dos processos como dos objetos, Tabela 13. Deve-se ter especial atenção com as exceções à lógica do código, como por exemplo, usando a captura do log do erro, criação de uma *Tag*, monitorização às exceções simultâneas e tentar recuperar da exceção. A falta de mecanismos de controlo destas exceções, pode criar sérios problemas com o desempenho do

processo e a robustez geral. As exceções não tratadas provavelmente irão causar que o processo RPA termine de forma incompleta. Devem ser seguidas as melhores práticas da Blue Prism, de forma que os detalhes de exceção existam e permaneçam consistentes. Para suporte mais detalhado os *developers* devem tentar seguir o tutorial Blue Prism *Exception Handling Guide*.

Tabela 13 - ***Score global sobre os critérios da Capacidade de Gestão***

Rank	Consideration Type	Category	Consideration	Score
6	Blue Prism Process Considerations	Manageability	BPP103 Process Exception Coverage	20%
9	Blue Prism Process Considerations	Manageability	BPP101 Process Exception Handling	33%
17	Blue Prism Process Considerations	Manageability	BPP105 Control Room Configuration	58%
22	Blue Prism Object Considerations	Manageability	BPO103 Exception Detail	69%
25	Blue Prism Process Considerations	Manageability	BPP104 Consecutive Exceptions	75%
26	Blue Prism Object Considerations	Manageability	BPO102 Exception Recovery	80%
32	Blue Prism Process Considerations	Manageability	BPP102 Process Exception Detail	89%
33	Blue Prism Object Considerations	Manageability	BPO105 Exception Types	92%

Relativamente à capacidade de reutilização, Tabela 14, verifica-se que nem todos os processos em análise, utilizaram o modelo de processo Blue Prism. Usar um modelo como base para construir um novo processo traz os seguintes benefícios:

- Diminuir o tempo de desenvolvimento do processo.
- Fornecer consistência em todos os Processos, permitindo uma compreensão e suporte mais fáceis.
- Auxiliar os desenvolvedores a manter as melhores práticas e trabalhar dentro da Metodologia de Desenvolvimento de uma organização.

Ainda relativo aos processos, foi possível verificar que nem todos os itens de dados de todo o ambiente estão configurados como variáveis de ambiente. A capacidade de alterar itens de dados sem abrir a configuração do processo fornece uma abordagem segura e flexível para gerir dados de todo o ambiente.

Em relação aos objetos, nem todas as ações dentro do objeto estão livres da lógica de negócios que deveria estar no nível do processo. Incluir a lógica de negócios ou regras num objeto faz

com que o objeto seja reutilizável apenas por processos que tenham as mesmas regras. A possibilidade de chamar outras ações publicadas acopla as ações para que elas só possam ser reutilizadas em conjunto e não individualmente. No que respeita os objetos base, construir ações reutilizáveis é extremamente importante para construir um repositório de objetos para processos futuros.

Tabela 14 - ***Score global sobre os critérios da Capacidade de Reutilização***

Rank	Consideration Type	Category	Consideration	Score
2	Blue Prism Process Considerations	Reusability	BPP201 Process Template Adherence	1%
5	Blue Prism Process Considerations	Reusability	BPP202 Environment Variables	15%
7	Blue Prism Object Considerations	Reusability	BPO205 Action Chaining	27%
11	Blue Prism Object Considerations	Reusability	BPO203 Action Reusability	49%
13	Blue Prism Object Considerations	Reusability	BPO204 Business Logic Location	53%
14	Blue Prism Object Considerations	Reusability	BPO206 Page Complexity	54%
37	Blue Prism Object Considerations	Reusability	BPO202 Logical Modeller Tree	99%

7.2.1 Análise global à solução de revisão de código RoboReview

Após o teste piloto para validação da plataforma RoboReview da Reveal, para solução de revisão do código RPA de forma automatizada, é possível identificar os benefícios adquiridos ao juntar mais uma ferramenta a todo o processo de desenvolvimento e entrega de código RPA. Além, de ser possível incrementar um maior controlo nas verificações de padrões de segurança e de qualidade, permite também, ajudar aos *developers* a terem um ponto de comparação mediante um processo de melhoria contínua, graças à disponibilização dos *dashboards* que analisam os dados individualmente, por *developer,* ou por equipa.

Uma análise ao código RPA seguindo uma *checklist* de forma manual, que é o que está determinado como processo interno de revisão de código para a Siemens GBS, proporciona um maior esforço de recursos e consumo de tempo, não permitido criar uma maior transparência e análise de agregação de dados para possível estudo de fatores críticos para melhoria no produto final.

8 Conclusão

Neste estudo foram abordados os critérios essenciais para os fundamentos teóricos e compreensão prática do modelo *DevOps* nas áreas de automação inteligente, permitindo assim a abertura à implementação do *DevSecOps* no serviço RPA da Siemens GBS. Para este conceito funcionar, é fundamental implementar uma metodologia *Agile* por todas as equipas inerentes ao serviço, mantendo por base uma cultura de cooperação e de envolvimento dos aspetos de melhoria contínua e segurança durante todo o ciclo de vida do produto/código RPA.

De acordo com o âmbito acima descrito, neste projeto, foram atingidos os seguintes objetivos, tal como descrito no capítulo 1.2:

Objetivo 1 e 2

Identificar as capacidades e benefícios do *DevOps* e suas áreas de abrangência e Elaborar a comparação do *DevOps* com o *DevSecOps* e demonstrar os seus benefícios – No Capítulo 2, fundamentos teóricos, pretende-se abordar os objetivos em análise. São identificados os valores chave desta abordagem num contexto de metodologias ágeis, e também, é efetuada a demonstração da evolução do *DevOps* para o *DevSecOps*.

Objetivo 3 e 4

Fazer um levantamento do processo de desenvolvimento de código RPA na Siemens GBS e Análise ao processo de melhoria contínua aplicado na Siemens GBS – No Capítulo 4 são abordados os pontos chave, desde a descrição do caso de estudo ao detalhe do modelo operacional implementado na Siemens GBS. No Subcapítulo 4.3 e Capítulo 5, analisa-se o procedimento implementado para o desenvolvimento do código RPA no serviço, como também, é elaborado o levantamento dos requisitos do negócio da Siemens GBS.

Objetivo 5 e 6

Formulação de uma proposta de melhoria para o processo de *code review* e CI/CD e Criação de um modelo RPA para inclusão de práticas *DevSecOps* – No Capítulo 6 desenvolve-se a

proposta de melhoria para o processo de revisão de código e de CI/CD, descrevendo as especificações dos requisitos da solução, funcionais e não funcionais. É também a proposta de uma abordagem RPA com *DevSecOps*, com a criação de um diagrama para o novo modelo de CI/CD para processos RPA.

Após levantamento técnico do modelo operacional do serviço de automação inteligente implementado na Siemens GBS, foi possível adquirir uma visão crítica sobre os pontos os pontos de melhoria e identificar quais os melhores métodos de resolução para questões de controlo de segurança e qualidade, durante o processo de desenvolvimento de um RPA *use case*.

O RPA na Siemens GBS, sendo um serviço onde ainda prevalece um modelo mais fechado e pouco colaborativo entre desenvolvimento e operações, é necessário primeiramente, ter em conta as resistências à mudança. A simples implementação de um modelo operacional que se torne disruptivo em relação ao "*as is*", pode correr o risco de nunca ser realmente colocado em prática. É necessário, facultar ferramentas que permitam aos *developers* e operações beneficiarem do especto da eficiência e qualidade na produção de um código RPA, e assim reduzir o *bug fixing* após entrega em produção, de forma a diminuir consideravelmente paragens no serviço dos robôs RPA. Este objetivo só será possível caso ambas as equipas trabalhem num modelo colaborativo.

Após levantamento do modelo operacional e levantamento de requisitos, foi elaborada a proposta de melhoria e entrega contínua, num conceito que envolvesse os critérios de segurança num modelo colaborativo, *DevSecOps*. O modelo operacional foi revisto e adaptado às melhores práticas de desenvolvimento, dando maior destaque para um processo de melhoria contínua em detrimento de uma abordagem *factory*. Pois, o objetivo será a introdução da eficiência com menor custo de manutenção. Ao prescindir de um modelo de entrega totalmente separado do modelo de operações, ganha-se benefícios no que respeita à manutenção dos casos RPA após entrega em produção. As vantagens poderão ser significativas, quando se trata de uma implementação em grande escala, que requer constantes adaptações ou alterações ao código já desenvolvido.

Para validação da proposta de melhoria, foram então inquiridos os membros das equipas de operações e desenvolvimento, de forma a recolher dados que demonstrassem a perspetiva de cada inquirido aos critérios de qualidade e segurança do código e plataforma RPA na Siemens GBS, mas também, perceber qual o nível de entendimento em relação aos fluxos de desenvolvimento e *change management* de acordo com as melhores práticas de desenvolvimento do código RPA.

A proposta desenvolvida, teve por base a melhoria de um processo de revisão de código que fosse possível adaptar ao já existente CI/CD. A solução RoboReview, demonstrou ser uma boa aposta na validação de certos padrões às melhores práticas existente para o desenvolvimento do código RPA da Blue Prism.

Após análise ao 12 *use cases* com mais pedidos de *debugging* no ambiente de Emergência, pós produção, foi possível perceber que a qualidade geral está perto dos 60%, o que indica que, existem critérios que necessitam de um maior foco ao nível de melhoria de qualidade. São eles, capacidade de manutenção, capacidade de reutilização e segurança, pois apresentam uma avaliação muito baixa, em termos relativos, no que se refere à qualidade, encontram-se abaixo dos 60%. Todos os outros critérios também necessitam de ser alvo de um plano de melhoria, mas estão acima dos 60% do score de qualidade, o que não quer indicar que seja aceitável, pois um código aceitável deverá situar-se não abaixo dos 75%, mas permite escalonar um processo de melhoria com menor criticidade para cada ponto identificado.

Esta solução automática de revisão de código espelha a existência de falhas no código RPA que necessitam de ser corrigidas. A utilização e integração desta solução no serviço RPA da Siemens GBS, poderá fomentar a qualidade e acima de tudo a melhor gestão de recursos. Será uma grande aposta para a estabilização da plataforma RPA, permitindo desenvolver processos automatizados mais seguros, estáveis, resilientes e com menor esforço.

8.1 Limitações

No âmbito deste projeto de investigação, foram identificados vários desafios, que na maioria dos casos, foram ultrapassados, mas certa parte, criou uma limitação no que se refere ao tamanho da amostra estudada.

O estudo teve por base uma implementação em larga escala de um serviço RPA, na Siemens GBS, que no seu portfolio de use cases RPA, conta com cerca de centenas de processos e objetos para a automação inteligente. Perante o tempo possível e de forma a ser exequível a conclusão do projeto académico, a amostra recolhida poderá ser pequena dado a dimensão da empresa. Mas pressupõe bons dados para uma investigação a fundo diretamente na Siemens, contando assim com a colaboração de várias equipas, de forma a ser possível analisar um maior número de amostras.

Outro aspeto de relevo a considera é a complexidade de um serviço RPA na Siemens GBS. Este serviço conta com interdependências estruturais, no que respeita a *frameworks* de segurança ou *quality checks* internos, que dependem de estudo e validação interna para ser possível acomodar uma alteração estrutural ao modelo de operações desencadeado por este projeto. Mas, pela sondagem feita por inquérito, passando pela validação técnica e análise aos resultados providos, este estudo apresenta-se com bons indicativos para poder ser seguido internamente pela Siemens GBS, obviamente sem nunca deixar de ser alvo de escrutínio interno prévio.

8.2 Trabalhos Futuros

Para implementação futura, seria a implementação da plataforma de *code review*, RoboReview, no âmbito do desenvolvimento RPA, especialmente na equipa de manutenção. Este processo traria uma maior coerência ao estudo, pois beneficiaria de múltiplos inputs elaborados pelas equipas de desenvolvimento e operações.

Outro dos aspetos fundamentais seria a revisão do processo de *change management*, de forma a prevenir que certas alteações sejam executadas diretamente em produção, (Emergência) e que, em certa parte, não seguem as melhores práticas no que concerne ao desenvolvimento de

software. A estabilidade uma plataforma RPA em produção, somente é conseguida caso existam requisitos fundamentais que não devem ser descurados, pois a permeabilidade de tais exceções, potenciam riscos e torna proeminente os ataques à segurança da plataforma RPA. Em certa forma, o processo proposto para melhoria na revisão do código RPA, irá ajudar a reduzir os pedidos de alteração do código após entrega para produção, o que satisfaz também uma das exigências na melhoria do processo de *change management.*

Outros dos aspetos a considerar, e estando relacionado com o processo de pedidos de *debugging* em emergência, será a melhoria no processo de *access management*, visto que, existe uma permeabilidade no que se refere a pedidos de *debugging* onde a grande maioria dos casos é legível para acesso a produção, não existindo um acompanhamento correto por parte dos *senior developers* e arquitetos durante a fase de testes e *code review*.

9 Bibliografia

American Idle: Workers Spend Too Much Time Waiting for Something to Do—HBS Working Knowledge. (2018, janeiro 31). https://hbswk.hbs.edu/item/american-idle-employees-are-wasting-way-too-much-time

Apps Built Better: Why DevSecOps is Your Security Team's Silver Bullet. (2021, julho 14). https://threatpost.com/apps-built-better-devsecops-security-silver-bullet/167793/

Banks, M. R., Willoughby, L. M., & Banks, W. A. (2008). Animal-Assisted Therapy and Loneliness in Nursing Homes: Use of Robotic versus Living Dogs. *Journal of the American Medical Directors Association*, *9*(3), 173–177. https://doi.org/10.1016/j.jamda.2007.11.007

Baskerville, R. (2008). What design science is not. *European Journal of Information Systems*, *17*(5), 441–443. https://doi.org/10.1057/ejis.2008.45

Boulton, C. (2018, setembro 3). What is RPA? A revolution in business process automation. *CIO*. https://www.cio.com/article/227908/what-is-rpa-robotic-process-automation-explained.html

Brodsky, A., & Amabile, T. M. (2018). The downside of downtime: The prevalence and work pacing consequences of idle time at work. *Journal of Applied Psychology*, *103*(5), 496–512. https://doi.org/10.1037/apl0000294

Clair, C. L. (2018). *The Forrester Wave™: Robotic Process Automation, Q2 2018*. 24.

Daptardar, S. (2021). *A REVIEW- THE GOLDEN TRIANGLE OF RPA, AI AND DIGITAL TRANSFORMATION*. *03*(01), 5.

Fischer, C., & Gregor, S. (2011). Forms of Reasoning in the Design Science Research Process. Em H. Jain, A. P. Sinha, & P. Vitharana (Eds.), *Service-Oriented Perspectives in*

Design Science Research (pp. 17–31). Springer. https://doi.org/10.1007/978-3-642-20633-7_2

Gomes, K., & Rogness, D. (2018). *HONORS THESIS ABSTRACT THESIS SUBMISSION FORM*. 19.

Guilherme, G., & Fernandes, C. (2020). *Departamento de Eletrónica, Universidade de Aveiro Telecomunicações e Informática 2020*. 96.

Hevner, A., & Chatterjee, S. (2010). Design Science Research in Information Systems. Em *Management Information Systems Quarterly—MISQ* (Vol. 28, pp. 9–22). https://doi.org/10.1007/978-1-4419-5653-8_2

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). *Design Science in Information Systems Research*. 32.

Intelligent Automation RPA Platform. (2022). Blue Prism. https://www.blueprism.com/products/intelligent-rpa-automation/

Junior, J. C. da S. F., Machado, L., Klein, A. Z., & Freitas, A. S. de. (2013). DESIGN RESEARCH: APLICAÇÕES PRÁTICAS E LIÇÕES APRENDIDAS. *Revista de Administração FACES Journal*. https://doi.org/10.21714/1984-6975FACES2015V14N1ART1999

Khan, S. (2020). COMPARATIVE ANALYSIS OF RPA TOOLS-UIPATH, AUTOMATION ANYWHERE AND BLUEPRISM. *International Journal of Computer Science and Mobile Applications*, *8*, 1–6. https://doi.org/10.47760/ijcsma.2020.v08i11.001

Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. (2016). *The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations* (First edition). IT Revolution Press, LLC.

Koskinen, A. (2019). *DevSecOps: Building security into the core of DevOps*. https://jyx.jyu.fi/handle/123456789/67345

Kukkuru, M. G., & Sucharita, S. (2018). *Robotic Process Automation and Quality Assurance – A Perspective*. 4.

Lacity, M., Willcocks, L., & Craig, A. (2015). *Paper 15/02 Robotic Process Automation at Telefónica O2*. 19.

Madakam, S., Holmukhe, R. M., & Jaiswal, D. K. (2019). The Future Digital Work Force: Robotic Process Automation (RPA). *Journal of Information Systems and Technology Management*, *16*(1), Article 1. https://doi.org/10.4301/S1807-1775201916001

Mayoral-Vilches, V., García-Maestro, N., Towers, M., & Gil-Uriarte, E. (2021). *DevSecOps in Robotics* (arXiv:2003.10402). arXiv. http://arxiv.org/abs/2003.10402

Myrbakken, H., & Colomo-Palacios, R. (2017). DevSecOps: A Multivocal Literature Review. Em A. Mas, A. Mesquida, R. V. O'Connor, T. Rout, & A. Dorling (Eds.), *Software Process Improvement and Capability Determination* (pp. 17–29). Springer International Publishing.

Nessler, D. (2018, junho 2). *How to solve problems applying a UXdesign Designthinking HCD or any Design Process from scratch v2*. Medium. https://uxdesign.cc/how-to-solve-problems-applying-a-uxdesign-designthinking-hcd-or-any-design-process-from-scratch-v2-aa16e2dd550b

Pimentel, M., Filippo, D., & Santos, T. M. dos. (2020). Design Science Research: Pesquisa científica atrelada ao design de artefatos. *RE@D - Revista de Educação a Distância e Elearning*, *3*(1), 37–61. https://doi.org/10.34627/vol3iss1pp37-61

Pries-Heje & Baskerville. (2008). The Design Theory Nexus. *MIS Quarterly*, *32*(4), 731. https://doi.org/10.2307/25148870

Rossi, M., Henfridsson, O., Lyytinen, K., & Siau, K. (2013). Design Science Research: The Road Traveled and the Road That Lies Ahead. *Journal of Database Management*, *24*(3), 1–8. https://doi.org/10.4018/jdm.2013070101

Schatsky, D., Muraskin, C., & Iyengar, K. (2016). Robotic process automation. 10.

Sen, A. (2021). DevOps, DevSecOps, AIOPS- Paradigms to IT Operations. Em P. K. Singh, A. Noor, M. H. Kolekar, S. Tanwar, R. K. Bhatnagar, & S. Khanna (Eds.), *Evolving Technologies for Computing, Communication and Smart World* (pp. 211–221). Springer. https://doi.org/10.1007/978-981-15-7804-5_16

Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). MIT Press.

Slaby, J. R. (2012). *Cheap, easy-to-develop software robots will eventually supplant many offshore FTEs*. 18.

Turkle, S. (2006). A Nascent Robotics Culture: New Complicities for Companionship. Em W. Wallach & P. Asaro (Eds.), *Machine Ethics and Robot Ethics* (1.ª ed., pp. 107–116). Routledge. https://doi.org/10.4324/9781003074991-12

Wazlawick, R. S. (2014). *Metodologia de Pesquisa Para Ciência da Computação*. https://www.sciencedirect.com/science/book/9788535277821

Writer, S. (2019, junho 12). RPA is poised for a big business break-out. CIO. https://www.cio.com/article/228818/rpa-is-poised-for-a-big-business-break-out.html

Anexos e Apêndices

Apêndice I – Resultados ao Questionário

Development and Operations on Siemens GBS RPA Service

30	21:43	Closed
Responses	Average time to complete	Status

1. On a scale from 1-10, with 1 being **Not at all familiar** and 10 being **Extremely familiar**, how are you familiar with DevOps approach?

30	*6.00*
Responses	Average Number

2. In your opinion, on a scale from 1-10, with 1 being **Never** and 10 being **Always**, is the RPA service in Siemens GBS aligned with a DevOps approach?

30	*5.97*
Responses	Average Number

3. On a scale from 1-10, with 1 being **Not at all concerned** and 10 being **Extremely concerned**, is the RPA security a matter of your interest?

30	*7.73*
Responses	Average Number

4. In general, on a scale from 1-10, with 1 **Not secure** and 10 being **Extremely secure**, how secure do you think RPA code is?

30	*7.70*
Responses	Average Number

5. On a scale from 1-10, with 1 being **Not a priority** and 10 being **Essential**, do you try to follow Siemens IT security recommendations in your daily work?

30	*8.67*
Responses	Average Number

6. Using a high-level overview, on a scale from 1-10, with 1 being **Not secure** and 10 being **Extremely secure**, do you think Siemens GBS has a secure RPA platform?

30	*8.73*
Responses	Average Number

7. In your opinion, on a scale from 1-10, with 1 being **Never** and 10 being **Always**, do you think that Siemens GBS RPA code is following the Blue Prism best practices?

30	*7.57*
Responses	Average Number

8. On a scale from 1-10, with 1 being **Not at all familiar** and 10 being **Extremely familiar**, are you familiar with Emergency RPA change requests workflow?

30
Responses

7.93
Average Number

9. Are you a developer?

Yes	14
No	16

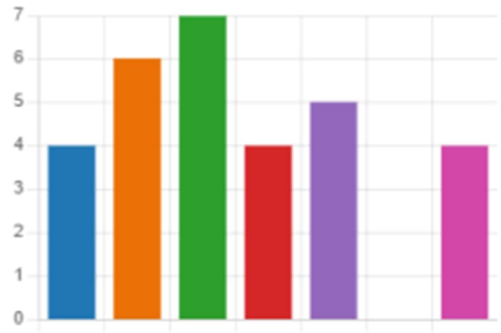
10. On a scale from 1-10, with 1 being **Never** and 10 being **Always**, how often do you request an emergency change request?

14
Responses

5.00
Average Number

11. In your opinion, what is the **most likely reason** for the code changes after releasing the RPA use case to production?

Inconsistent quality control tests	4
Differences between DEV and P...	6
Differences between QA and PR...	7
Fine tuning regarding target ap...	4
New feature/steps added to RP...	5
Too much complexity added to ...	0
Other	4



Apêndice II – Pontuação mais baixa a mais alta por consideração

Rank	Consideration Type	Category	Consideration	Score
1	Blue Prism Process Considerations	Security	BPP403 Process Logging Policy Adherence	0%
1	Blue Prism Process Considerations	Maintainability	BPP605 Block Grouping Used	0%
1	Blue Prism Process Considerations	Maintainability	BPP604 Page Numbering	0%
2	Blue Prism Object Considerations	Security	BPO404 Object Logging Policy Adherence	1%
2	Blue Prism Process Considerations	Reusability	BPP201 Process Template Adherence	1%
3	Blue Prism Object Considerations	Stability	BPO510 Image Handling	2%
4	Blue Prism Object Considerations	Maintainability	BPO605 Block Grouping Used	5%
5	Blue Prism Process Considerations	Reusability	BPP202 Environment Variables	15%
6	Blue Prism Process Considerations	Manageability	BPP103 Process Exception Coverage	20%
7	Blue Prism Object Considerations	Reusability	BPO205 Action Chaining	27%
8	Blue Prism Object Considerations	Maintainability	BPO603 Object and Action Names	31%
9	Blue Prism Process Considerations	Manageability	BPP101 Process Exception Handling	33%
10	Blue Prism Process Considerations	Security	BPP404 Process Work Queue	43%
11	Blue Prism Object Considerations	Reusability	BPO203 Action Reusability	49%
12	Blue Prism Process Considerations	Stability	BPP501 No Infinite Retry Loops	51%
13	Blue Prism Object Considerations	Reusability	BPO204 Business Logic Location	53%
14	Blue Prism Object Considerations	Reusability	BPO206 Page Complexity	54%
15	Blue Prism Object Considerations	Stability	BPO504 Attach Page Reference	56%
16	Blue Prism Object Considerations	Stability	BPO503 Valid Attach Status	57%
17	Blue Prism Process Considerations	Manageability	BPP105 Control Room Configuration	58%

18	Blue Prism Process Considerations	Security	BPP401 Secure Password Storage	60%
19	Blue Prism Process Considerations	Scalability	BPP301 Subpage Complexity	61%
20	Blue Prism Object Considerations	Stability	BPO511 Application Focus	66%
21	Blue Prism Object Considerations	Stability	BPO506 Wait Navigation	67%
22	Blue Prism Object Considerations	Manageability	BPO103 Exception Detail	69%
23	Blue Prism Object Considerations	Maintainability	BPO602 Meaningful Documentation Precursors	71%
24	Blue Prism Object Considerations	Stability	BPO505 Wait Start	74%
25	Blue Prism Process Considerations	Manageability	BPP104 Consecutive Exceptions	75%
26	Blue Prism Object Considerations	Manageability	BPO102 Exception Recovery	80%
27	Blue Prism Object Considerations	Maintainability	BPO601 Proper Element Names	81%
28	Blue Prism Object Considerations	Stability	BPO508 Wait Variables	83%
29	Blue Prism Object Considerations	Stability	BPO509 Wait Timeout Exception	84%
29	Blue Prism Process Considerations	Maintainability	BPP601 High Level Main Page	84%
30	Blue Prism Process Considerations	Scalability	BPP302 Subpage Size	86%
31	Blue Prism Process Considerations	Scalability	BPP303 Exception Retrying	88%
32	Blue Prism Process Considerations	Manageability	BPP102 Process Exception Detail	89%
33	Blue Prism Process Considerations	Stability	BPP503 Stop Decision Between Cases	92%
33	Blue Prism Object Considerations	Manageability	BPO105 Exception Types	92%
34	Blue Prism Process Considerations	Stability	BPP502 Appropriate Process Preparation	94%
34	Blue Prism Object Considerations	Stability	BPO507 Wait Conditions	94%
35	Blue Prism Process Considerations	Stability	BPP504 No Nested Retry Loops	95%
36	Blue Prism Object Considerations	Stability	BPO502 Valid Object Exposure	98%
37	Blue Prism Object Considerations	Reusability	BPO202 Logical Modeller Tree	99%
38	Blue Prism Process Considerations	Security	BPP402 No Hard Coded Passwords	100%

	38 Blue Prism Object Considerations	Security	BPO403 Technology Attributes	100%
	38 Blue Prism Object Considerations	Security	BPO402 No Environmental Data	100%
	38 Blue Prism Object Considerations	Security	BPO401 No Customer Data	100%

Anexo I – Blue Prism Checklist (Process Review Template)

PROCESS REVIEW

blueprism
Robotic Process Automation Software

Process Name	"Process (Name)"
Overall Status	

Topic	Considerations	Recommendations	References
Adherence to Process Templates	Does the Process adhere to Blue Prism (or a local standard) Process Templates?	*Using a template as the basis of building a new process has the following benefits:* *- Decrease Process development time* *- Provide consistency across all Processes, allowing easier understanding and support* *- Assist developers in keeping to best practice and to work within the Development Methodology of an organisation*	- Process Templates - Development Best Practice
Use of Work Queues	Does the Process use a Work Queue?	*Work Queues provide the functionality to store, manage, share and report on process work.*	- Work Queues Guide - Solution Design Overview
Use of Sub-pages	Does the Main Page contain only high-level process steps?		- Process Templates - Process Creation Tutorial
	Of the sub pages that are built, are they maintainable? (Not too complicated)		
	Of the sub pages used are they manageable? (Not too large?)		
	Are there duplicated pieces of functionality that could be created in a Sub-Page?	*Look to keep repeatable pieces of functionality in a Sub-Pages for ease of maintenance.*	
Exception Handling - Main Page	Are Exception cases handled gracefully?	*Not handling exceptions gracefully by capturing them and trying to recover from can create serious issues with the performance of the process and robustness.*	- Process Templates - Exception Handling Guide - Process Creation Tutorial - Solution Design Overview
	Are Exception details captured against the case in the Queue?	*Capturing the detail of the exception when it has occurred and updating the case with this information is vital for both operations and the RPA Developer team as they investigate the cause of the exception and the corrective action required.*	
	Are applications reset to the start point for the next case?	*Applications must be set to a known starting point before functionality can be re-tried or the next case can be processed.*	
	Are all relevant Exceptions handled?	*Unhandled exceptions are likely to cause a Process termination*	
	Does the Process check for multiple consecutive Exceptions and handle accordingly?	*Within the Blue Prism Best Practice Templates on the Portal on the sub page for "Mark Item as Exception", there is an example that can utilized on how to monitor for consecutive exceptions. Not performing this check may result in multiple exceptions.*	
Exception Handling - Sub Page	Do Sub-Pages allow retries of logic to allow for additional resilience?	*Retry logic for relevant exception types (e.g. System Exceptions) provide an increased chance of a case completing successfully.* *Sub Page retry handling can be found in the Blue Prism Best Practice Templates found on the Portal*	- Process Templates - Development Best Practice - Process Creation Tutorial
	Is the application correctly reset between retries?	*Applications must be set to a known starting point before functionality can be re-tried or the next case can be processed.*	
	Are only the relevant Exception Types retried?	*Retrying of Business or Validation Exceptions are unlikely to yield a different outcome.* *Retrying Login Exceptions may lock a user account*	
	Are there any undesired nested Retry Loops?	*Nested retry loops may result in too many retries*	
	Are retry loops infinite?	*Infinite retry loops may cause the process to become stuck in an undetected loop*	
Is the required Logic outside the Case Working phase in place?	Are any preparations carried out in the process? (Initialisation sub page for example; checking file locations, files exist, getting application credentials etc.)	*It is always good practice to have a sub page that conducts preliminary steps to check if files are in the location they should be, the robot can retrieve passwords etc.*	- Process Templates - Process Creation Tutorial - Solution Design Overview
	Are any Finalization steps carried out in the final parts of the process? (Moving files, sending emails etc.)	*It is always good to conduct a finalization check at the end of the process to ensure the final tasks have been carried out, especially ensuring applications have been closed down.*	
	Are the applications reset or closed down when a case is not being handled?		
	Are applications recovered following cases failing in readiness for the next case to be worked?		
	Is there a Stop decision point between cases?	*The ability to stop a process safely between cases or at a specified time add flexibility to the Process.* *An example of how this should be conducted can be found in the Blue Prism Best Practice Templates on the Blue Prism Portal*	

Use of Environment Variables	Are any environment wide data items configured as Environment Variables?	*The ability to change data items without opening the Process configuration provides a secure and flexible approach to managing environment wide data*	- Process Creation Tutorial
Use of Session Variables	Are the required data items configurable in Control Room?	*Session Variables provide more control of the process when running it from the control room e.g. amending a scheduled stop time, providing a flexible mechanism to control the number of cases worked in a session or to request the process to stop processing safely between cases.*	
Validation Errors	Does the Process contain any validation errors that should be corrected?	*Invalid stages will cause an internal exception if attempted to be executed*	- Development Best Practice
Use of Sub-Process	Does the Process repeatedly call a sub-process?	*Each call of a sub-process requires a call to the database to read the xml into memory. BP is reliant on .net garbage collection to tidy this memory up once the sub-process ends.* *Business Objects are read into memory only once and persist for the duration of the sessions.*	- Solution Design Overview
	Would the Sub-Process logic be more efficient in an Object?	*Each call of a sub-process requires a call to the database to read the xml into memory. BP is reliant on .net garbage collection to tidy this memory up once the sub-process ends.* *Business Objects are read into memory only once and persist for the duration of the sessions.*	
Should the Process be broken into smaller processes	Is there an opportunity to split out some functionality into a separate Process e.g. (Get Work, Send E-Mail, Create Results Report)	*A solid design will always benefit the digital workforce by utilizing the Work Queues to allow the workforce to retrieve work and work a simple task rather then from Start to End*	- Solution Design Overview - Development Best Practice - Process Creation Tutorial
Scalability	Can the process run on multiple machines in parallel, without having any effect? (Duplicate cases being loaded / worked, locking issues, same files being picked up and worked etc.)	*An appropriate design will enable the solution to be scaled safely as and when required*	
	Are there any steps, that must not be executed concurrently by multiple machines, that could possibly be so?	*An appropriate design will enable the solution to be scaled safely as and when required*	
	Are there any steps that must only be executed once, regardless of how many machines are running, that could be possibly executed multiple times? Consider whether the steps can be executed multiple times within the same session or across multiple sessions.	*An appropriate design will enable the solution to be scaled safely as and when required*	
Credentials	Are Passwords kept in the Credential Store or a similar secure repository?	*It is imperative as part of your process, credentials when used are stored within the Blue Prism Credential Management or in a similarly secure repository.*	- Secure Windows Authentication - Credential Manager - Active Directory Integration
	Are there any hard coded Passwords in the diagram?	*Hard coding of passwords in the process does not adhere to best practices nor will it confine to the IT Security Protocol.*	
	Do Credential names allow for unique Passwords per session?		
Logging	Does logging adhere to local Security Policy? (Is logging enabled for stages)	*It is important the logging of actions and stages in the process adheres to the IT Security Policy. If unsure what this is, please refer to your Lead RPA Technical Architect*	- Development Best Practice
Tagging	Using tags to record unique information?	*Tagging is not meant as a way of recording unique item information. Unique information should only be stored in the Item Data*	- Development Best Practice
	Presence of complex tag filtering, e.g. using wildcard inside searches?	*Wildcards inside searches should be avoided. These render the table index almost useless, and force the query to perform a table scan of the entire queue.*	- Development Best Practice
Other Comments	e.g. Are there any obvious logic Flaws?		
Other Comments	Are there any obvious logic Flaws?		

Anexo II – Blue Prism Checklist (Object Review Template)

OBJECT REVIEW

blueprism
Robotic Process Automation Software

Object Name	"Object (Name)"
Overall Status	

Topic	Considerations	Recommendations	References
Business Objects broken down into acceptable sizes	Are Business Objects broken up at 'an object per screen' or similar as per Best Practice?	*Creating multiple objects per application will provide greater efficiency in the process, minimise regression overheads and will enable multiple developers to work against an application.*	- Object Layer Design Tutorial
Application Modeller Tree broken down	Are the elements logically broken down by screen or each part of the screen?	*Creating an easy to comprehend tree will enable easy identification of elements within the tree.*	- Development Best Practice
Element - Names	Do the element names follow BP best practice or local naming convention?	*Creating an easy to comprehend tree will enable easy identification of elements within the tree.*	- Development Best Practice
Element attribute selection	Will the elements attribute match list result in a consistent / efficient match?	*An efficient attribute selection will ensure elements are located consistently and quickly at run time and are less likely to be impacted by a change to the application*	- Development Best Practice - Technology specific guides (e.g. Java / Browser)
	Do attribute values contain Customer data?	*Elements containing customer data or any data that can be perceived as a breach by the IT Security Policy should not be in the values for the attributes*	
	Do values contain any environment specific data?	*Environment specific attributes are more likely to cause an error as the solution moves between environments.*	
	Are there any technology specific attributes that are recommended to be checked / unchecked?	*Blue Prism Portal has a number of technology specific guides containing best practices for element attribute selection.*	
Documentation	Are Action descriptions, Pre-Conditions, Post Conditions, Input params & Output params documented to create a meaningful BOD?	*By creating comprehensive documentation within the object itself, Blue Prism's self-documenting functionality can be exploited to provide Object documentation to be referenced for future use.*	- Development Best Practice
Exposure	Are the Business Object exposures valid?		- Development Best Practice
Use of Attach	Does the Business Object have an 'Attach' Action that reads the connected status before Attaching?	*By determining is an object is already connected to the application before attempting to attach enables the 'Attach' action to be called multiple times safely and efficiently.*	
	Do all Actions use the Attach action?	*Attaching at the start of each action will ensure the Object is attached before attempting to automate the application.*	
Validation errors	Does the object contain any validation errors that should be corrected?	*Invalid stages will cause an internal exception if attempted to be executed*	- Development Best Practice
Correct use of Wait Stages	Does each action start with a Wait Stage to verify the application is in the correct state?	*Wait stages at the beginning of an action ensure the application is in the correct state prior to the action being executed.*	- Development Best Practice - Business Object Templates
	Are navigation stages between application screens followed by a Wait Stage to verify success?	*Wait stages following a navigate stage ensure the application is in the correct state prior to continuing*	
	Do Wait Stages have conditions (i.e. not arbitrary)? Do not include Arbitrary Waits if used for Surface Automation purposes only?	*Arbitrary Wait Stages are invariably either unreliable or inefficient. Look to use conditions wherever possible.*	
	Global variable enable a quick change to timeout values when application behaviour dictates.	*Blue Prism Best Practice templates utilize global timeout stages*	
	Do Wait Stages timeout to an exception?	*Wait Stages that do not timeout to another stage (usually an exception) will throw an 'internal' error resulting in more difficult error investigation.*	
Re-useable Actions	Are the actions re-useable?	*Building reusable action is extremely key to building up an Object Repository for future processes*	- Development Best Practice - Solution Design Overview
	Is there any Business Logic that should be in a Process?	*Including business logic or rules in an Object results in the object being only re-useable by Processes that have the same rules*	
Action Names	Do the names of objects and pages give the process developer a good idea of their purpose?		- Development Best Practice
	Are the action names generic and excluding any 'process specific' wordings?		
Action Size	Do Actions call other published Actions?	*Calling other published actions couples the actions together so they can only be re-used together and not individually.*	- Development Best Practice - Solution Design Overview
	Are there any overly complex pages that could be broken up? Think about; - For ease of use - For ease of reuse - For more effective testing - For increased efficiency - For better security		

Exception Handling	Do the objects try to recover exceptions (should be Process logic)?	*Recovering of exceptions should only occur in instances to make the action work, i.e. if finding an image fails but it can work after a retry.* *Typically, the steps to take following an exception should be decided in the Process layer as they may vary between different processes.*	- Exception Handling Guide
	Do Exceptions provide appropriate Type and Detail?	*As part of the Best Practice standards, those exception types and details need to remain consistent*	
	Do Exception Types follow Best Practice?	*Based on the Blue Prism Exception Handling Guide*	
Logging	Does logging adhere to local Security Policy?	*This is driven by the IT Security Policy*	- Development Best Practice
Images	Are Target Images definitions efficient?	*Creating images that are sometime too large and not efficient will impact the performance of the action*	
Application Focus	Is Focus ensured when required? For using Globals and Image Recognition?	*Activating the application will ensure the correct application is in focus to receive Global commands*	