UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
IM – INSTITUTO DE MATEMÁTICA / INSTITUTO TÉRCIO PACITTI
PPGI – PROGRAMA DE PÓS GRADUAÇÃO EM INFORMÁTICA
DOUTORADO EM INFORMÁTICA

IDENTIFICAÇÃO DE GARGALOS EM PROCESSO DE DESENVOLVIMENTO DE SOFTWARE: UMA PROPOSTA BASEADA NOS PRINCÍPIOS DA TEORIA DAS RESTRIÇÕES

SILDENIR ALVES RIBEIRO

Tese de Doutorado submetida ao Programa de Pós Graduação em Informática (PPGI), do Instituto de Matemática / Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro (UFRJ) como parte das exigências do curso de Pós-Graduação Stricto Senso em Informática, área de concentração Gestão de Sistemas Complexos, para obtenção do titulo de Doutor em Informática.

Orientador: Eber Assis Schmitz

Rio de Janeiro - RJ
Abril de 2018

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
IM – INSTITUTO DE MATEMÁTICA / INSTITUTO TÉRCIO PACITTI
PPGI – PROGRAMA DE PÓS GRADUAÇÃO EM INFORMÁTICA
DOUTORADO EM INFORMÁTICA

FOLHA DE APROVAÇÃO

IDENTIFICAÇÃO DE GARGALOS EM PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE: UMA PROPOSTA BASEADA NOS PRINCÍPIOS DA TEORIA DAS RESTRIÇÕES.

SILDENIR ALVES RIBEIRO

Tese de Doutorado submetida ao Programa de Pós Graduação em Informática (PPGI), do Instituto de Matemática / Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro (UFRJ) como parte das exigências do curso de Pós-Graduação Stricto Senso em Informática, área de concentração Gestão de Sistemas Complexos, para obtenção do titulo de Doutor em Informática.

Aprovada em 11 de Abril de 2018, por:

Eber Assis Schmitz, PhD. (Orientador)
Dep. de Ciência da Computação / IM – Inst. de Matemática / PPGI – Inst. Tércio Pacitti
Universidade Federal do Rio de Janeiro (UFRJ)

Antônio Juarez Sylvio Menezes de Alencar, D.Phil
Programa de Pós Graduação em Informática – PPGI / Instituto Tércio Pacitti
Universidade Federal do Rio de Janeiro (UFRJ)

Mônica Ferreira da Silva, D.Sc.
Programa de Pós Graduação em Informática – PPGI / Instituto Tércio Pacitti
Universidade Federal do Rio de Janeiro (UFRJ)

Marcos Kalinowski, D.Sc.
Departamento de Informática – DI
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)

Ricardo de Almeida Falbo, D.Sc.
Departamento de Informática – DI / Centro Tecnológico – CT
Universidade Federal do Espírito Santo (UFES)

Rio de Janeiro - RJ
Abril de 2018

CIP - Catalogação na Publicação

A484i

Alves Ribeiro, Sildenir
Identificação de Gargalos em Processo de Desenvolvimento de Software: Uma Proposta Baseada nos Princípios da Teoria das Restrições / Sildenir Alves Ribeiro. -- Rio de Janeiro, 2018.
221 f.

Orientador: Eber Assis Schmitz.
Tese (doutorado) - Universidade Federal do Rio de Janeiro, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática, 2018.

1. Processo de Software. 2. Teoria das Restrições. 3. Identificação de Gargalos. 4. Engenharia de Software. 5. Engenharia de Software Experimental. I. Assis Schmitz, Eber, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a).

AGRADECIMENTOS

"Num mundo que se faz deserto, temos sede de encontrar um amigo."

Antoine De Saint-Exupéry

Embora este trabalho seja o resultado de um projeto que implica em boa parte o esforço de natureza individual, avocado na disciplina, na dedicação e na perseverança, nada teria acontecido sem o apoio das pessoas que estão a minha volta. Deixo registrada minha profunda gratidão à tríade que me acompanhou nesta longa caminhada: família, amigos e professores.

A família é o porto seguro, os pilares de sustentação, a cura do *stress* e o alimento da alma. A família é a curva do rio que retém todos os detritos em momentos de inundação e suaviza o fluxo da vida. É a família que ampara e sofre com as nossas angústias e emana felicidade com as nossas conquistas. À minha família, minha eterna gratidão por fazer parte de mim.

A amizade se traduz ou se define por: sentimento de grande afeição, simpatia, apreço, companheirismo e outros tantos adjetivos. O amigo é o ser que detém a amizade em sua essência. A amizade da sentindo a vida e preenche lacunas existentes em cada um de nós. O amigo é o irmão que Deus nos permitiu escolher. E por falar em amizade, não poderia deixar de agradecer aos meus amigos do CEFET/RJ, especialmente ao colegiado de Automação Industrial do Campus Maria da Graça, pela compreensão e por suprirem as minhas ausências em determinados momentos em função da construção deste projeto. A todos os meus amigos, o crédito pelo incentivo, pela descontração e pelo companheirismo.

O professor é um ser fabuloso que carrega consigo um mar de conhecimento formado por gotas de sabedoria, de dedicação e de extraordinária paixão pelo ofício. O professor é o bem maior de uma nação! É o responsável direto pela construção e pela prosperidade de um país! Ao professor deve ser

dado o reconhecimento por todo o sucesso conquistado em nossa jornada. Pois o fracasso é fruto da não obediência aos seus ensinamentos ou em razão maior, por não ter dado a devida importância ao conteúdo por ele ministrado. Sem a participação dos meus professores durante toda a minha vida acadêmica, esse trabalho sequer teria sido cogitado. Portanto, agradeço todo o esforço e dedicação daqueles que compartilharam seus conhecimentos e experiências e contribuíram diretamente para a minha formação. Aos professores do PPGI/ITP/IM/UFRJ meus sinceros agradecimentos. Em especial, agradeço ao professor ***Eber Assis Schmitz*** pela orientação, pela dedicação, pela ajuda, pelo apoio incondicional e por ter acreditado desde o início na viabilidade deste projeto. Agradeço ainda à minha querida irmã e primeira professora ***Maria D'Ajuda Ribeiro***, por ter principiado todo este processo.

Por fim, agradeço a Deus pela luz, pela esperança, pela proteção e pelos milagres diários que tem me proporcionado. Deus, sem tua paz nada seria possível, sem tua presença nada faria sentindo!

DEDICATÓRIA

"Que a família celebre a partilha do abraço e do pão!"

Padre Zezinho

Aos meus pais, ***Joaquim Ribeiro de Carvalho*** *e* ***Oracy Alves Ribeiro****, pela vida.*

À minha esposa, ***Viviani de Moraes Freitas Ribeiro****, pelo carinho, pelo afeto e pelos presentes que se traduzem como "o amor maior".*

À minha filha ***Manuela Freitas Ribeiro*** *e ao meu filho* ***Rafael Freitas Ribeiro****, prova real de que o amor não é abstrato.*

"Cada sonho que você deixa para trás é um pedaço do seu futuro que deixa de existir".

Steve Jobs.

30 frases de Steve Jobs para você se inspirar. Por: Mariana Desidério, Revista EXAME, Editora Abril - 05 de Outubro de 2015.

RESUMO

RIBEIRO, Sildenir Alves: (2018) – Identificação de Gargalos em Processos de Desenvolvimento de Software: Uma Proposta Baseada nos Princípios da Teoria das Restrições. Orientador: Prof. Eber Assis Schmitz, PhD. Rio de Janeiro-RJ: UFRJ/IM/PPGI/Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais (NCE). Tese de Doutorado, Doutorado em Informática.

A extensa literatura sobre gargalos em processos de produção, baseada principalmente nos ensinamentos de *Eliyahu Moshe Goldratt*, afirma que todo sistema produtivo possui pelo menos um elemento restritivo que impede que as organizações atinjam a sua meta. Como o desenvolvimento de software é um processo produtivo, procurou-se com esta pesquisa primeiramente responder as seguintes perguntas: (1) existem gargalos no processo de desenvolvimento de software? (2) Se existem, é possível identificá-los ou tipificá-los? (3) Uma vez identificado o(s) gargalo(s) é possível dizer que o gargalo é o mesmo para cada equipe de desenvolvimento e para cada fase do processo? Para responder estas questões e consequentemente aplicar o segundo objetivo desta tese, que é o tratamento dos gargalos encontrados, foi desenvolvido durante as investigações uma metodologia experimental baseada em observações empíricas das atividades voltadas para o processo de produção de software. Os ensaios experimentais foram realizados em ambiente de aprendizagem de Engenharia de Software, onde foi simulado um processo produtivo real, porém executado com aprendizes do curso de Ciência da Computação da Universidade Federal do Rio de Janeiro. Os estudantes foram divididos em grupos de trabalho e envolvidos no projeto laboratorial da disciplina de Engenharia de Software. Cada grupo (equipe de desenvolvimento) representa uma Linha de Produção (LP) no processo produtivo e cada componente (aluno) representa uma máquina ativa em uma linha de produção. O conjunto de LPs determina o ambiente produtivo (*shop)* de cada *quase-experimento*. Ao todo foram elaborados três *shops*, que determinaram o ambiente produtivo de três rodadas experimentais. Todas as LPs de cada *shop* executaram as mesmas tarefas dentro de um mesmo domínio e com o mesmo apoio ferramental. O Processo Unificado (PU) foi adotado como o Processo de Desenvolvimento de Software (PDS). O PU sofreu algumas modificações e adaptações alinhadas com os objetivos e o propósito desta pesquisa, e foi denominado de *PU-Like*. Os princípios da Teoria das Restrições (TOC) foram empregados para identificar e tratar as restrições qualitativas. Posteriormente, após a análise quantitativa dos dados, os gargalos encontrados no PDS foram tipificados e categorizados. Para simular o ambiente produtivo e escalonar as tarefas no *shop*, foram empregados os conceitos de *Job Shop Scheduling (JSSP)* em sua variação dinâmica, denominada de *Dynamic Job Shop Problem (DJSP).* Como resultado, a pesquisa apresentou um método que permitiu evidenciar uma série de restrições qualitativas que foram categorizados em dois grupos: (1) Restrições de Ordem Comportamental (RC): que são restrições locais associadas às máquinas (pessoas) e que afetam diretamente a produção de uma LP; e (2) Restrições de Ordem Técnica (RT): que são restrições qualitativas associadas a capacidade técnica dos indivíduos em resolver problemas de construto no PDS. Os gargalos qualitativos (restrições) foram identificados e tratados em tempo de execução com a aplicação da TOC. O método desenvolvido permitiu ainda analisar os dados quantitativamente e assim identificar os gargalos do PDS. Os gargalos produtivos encontrados estão associados a uma atividade e/ou artefato, e são causados por uma

restrição qualitativa (RC ou RT), que impedem a maximização da produção de uma LP. Os gargalos quantitativos foram identificados após a análise dos dados e ao fim do processo produtivo (entrega do software construído. Os resultados experimentais colhidos nos três *"quase-experimentos"* foram analisados estatisticamente para que as hipóteses levantadas pudessem ser aceitas ou refutadas, e desta forma responder as questões centrais sobre a existência de gargalos no PDS. A análise conclusiva sobre a temática abordada nesta tese, apontou a atividade de Codificação como o gargalo do Processo de desenvolvimento de Software estudado, seguido da atividade de Documentação.

Palavras chave: Identificação de Gargalos, Teoria das Restrições, Processo Unificado, Processo de Desenvolvimento de Software, Job Shop Scheduling Dinâmico, Análise Quali-quantitativa.

ABSTRACT

RIBEIRO, Sildenir Alves: (2018) – Bottlenecks Identification in Software Development Process: A Proposal Based on the Principles of the Theory of Constraints. Advisor: Prof. Eber Assis Schmitz, PhD. Rio de Janeiro-RJ: UFRJ/IM/PPGI/Tércio Pacitti Institute of Applications and Computational Research (NCE). Doctoral Thesis, Doctorate in Informatics.

The wide literature on bottlenecks in production processes, based mainly on the teachings of Eliyahu Moshe Goldratt, which states that every productive system has at least one restrictive element that preclude organizations of reaching their goal. Since software development is a productive process, this research sought first to answer some questions: (1) Do bottlenecks exist in the software development process? (2) If they exist, can they be identified and classified? (3) Once identified, is it possible to say that they are the same for each development team and for each stage of the process? A secondary objective of this thesis is to study methods of treatment of the restrictions arising during the software development process. In order to answer these questions, a methodology based on empirical observations of the activities executed during the software production process was developed. The experimental assays were carried out in a software engineering teaching environment with students of Computer Science Department of the Federal University of Rio de Janeiro. Each group (development team) represented a Production Line (PL) in the production process and each component (student) represented an active machine on a production line. The set of PLs determines the productive environment (Shop) in each of the *"quasi-experiments"*. Three Shops were elaborated, setting up the productive environment of the three experimental rounds. All the PLs of each shop perform the same tasks, within the same domain and with the same supporting tools The Unified Process (UP) was adopted as the Software Development Process (SDP). The UP has undergone some modifications and adaptations aligned with the objectives and the purpose of this research and were denominated of UP-Like. The principles of the Theory of Constraint (TOC) were used to identify and treat the qualitative bottlenecks found in the PDS. Posteriorly, after the quantitative data analysis, the bottleneck(s) found in PDS were classified. The concepts of Job Shop Scheduling Problem (JSSP) were used in its dynamic variation, called the Dynamic Job Shop Problem (DJSP) to simulate the productive environment and to scheduling the tasks in the shop. As result, the research presented a method that showed a series of qualitative constraints categorized into two groups: (1) Behavioral Constraints (BC): which are local constraints associated with machines (people) and that affect directly the production of an LP; and (2) Technical Constraints (TC): these are qualitative constraints associated with individuals' technical ability to solve construct problems in the PDS. In this work, the qualitative bottlenecks (constraints) are identified and treated, with the application of TOC, at execution time. The method developed in this work, allowed the identification of the PDS bottlenecks using the statistical analysis of the data collected during the experiment. The productive bottlenecks found are associated with an activity, and are caused by a qualitative restriction (BC or TC), which prevents the maximization of LP production. Quantitative bottlenecks were identified after data analysis and at the end of the production process (software delivery built). The experimental results obtained in the three quasi-experiments were analyzed statistically so that the hypotheses could be accepted or refuted, and thus answer the

questions about the existence of bottlenecks in the PDS., and The final results pointed to the activity “Coding” as the bottleneck of the Software Development Process studied, followed by the activity “Documenting”.

Keywords: Bottleneck Identification, Theory of Constraints, Unified Process, Software Development Process, Dynamic Job Shop Scheduling, Quali-quantitative Analysis.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

ABNT	Associação Brasileira de Normas Técnicas
ACM	Association for Computing Machinery
ANOVA	Analysis of Variance
ARA	Árvore da Realidade Atual
ARF	Árvore da Realidade Futura
BD	Bando de Dados
BPMN	Business Process Model Notation
C_{Max}	Capacidade Máxima
CMMI	Capability Maturity Model Integration
DBR	Drum, Buffer and Rope
DBUnit	Data Base Unit
DCC	Departamento de Ciência da Computação
Df	Diferença
DJS	Dynamic Job Shop
DJSP	Dynamic Job Shop Problem
DJSS	Dynamic Job Shop Scheduling
ES	Engenharia de Software
ESE	Engenharia de Software Experimental
Exp1	Experimento 1
Exp2	Experimento 2
Exp3	Experimento 3
FES	Fundamentos de Engenharia de Software
FDD	Feature Driven Development
FSSP	Floor Shop Scheduling Problem
G	Grupos (G1, G2,…, Gn)
GIT	Global Information Tracker
GIT-Hub	GIT-Head Up Butt (Hub = Command-line Wrapper for GIT)
GL	Graus de Liberdade (*gl*)
GS	Google Scholar
H_0	Hipótese Nula

H_1	Hipótese Alternativa
HSD	Honest Significant Difference
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IC	Intervalo de Confiança
IDE	Integrated Development Environment
ISO	International Organization for Standardization
IX	IEEE Xplorer
JSD	Job Shop Dinâmico
JSP	Job Shop Problem
JSS	Job Shop Scheduling
JSSP	Job Shop Scheduling Problem
JUnit	Java Unit
LP	Linha de Produção
LPOO	Linguagem de Programação Orientada a Objetos
MDS	Metodologia de Desenvolvimento de Software
MVC	Model View Controller
MPS.Br	Melhoria do Processo de Software . Brasileiro
Mq	Média dos quadrados
NBR	Normas Br / Normas Brasileiras
NP-Hard	**Non**-Deterministic **Polynomial**-Hard
OSSP	Open Shop Scheduling Problem
PDS	Processo de Desenvolvimento de Software
PO	Process Optimization
PU	Processo Unificado
PU-Like	Processo Unificado-*Like*
Q	Questão (Q1, Q2, ..., Qn)
Qp	Questão principal
Qs	Questão secundária
RAD	Rapid Application Development
RC	Restrição Comportamental
RL	Revisão da Literatura
RRC	Recurso com Restrição de Capacidade
RsRC	Recurso sem Restrição de Capacidade

RS	Revisão Sistemática
RSL	Revisão Sistemática da Literatura
RT	Restrição Técnica
SEI	Software Engineering Institute
Scp	Scopus
SD	Science Direct
SDLC	Software Development Life Cycle
SGBD	Sistema Gerenciador de Banco de Dados
SL	Springer Link
SP	Software Process
SPI	Software Process Improvement
SPO	Software Process Optimization
SQ	Soma dos Quadrados
SQL	Structured Query Language
TOC	Theory of Constraints
TPC	Tambor, Pulmão e Corda
UFRJ	Universidade Federal do Rio de Janeiro
UML	Unified Modeling Language
XP	Extreme Programing

LISTA DE PUBLICAÇÕES

1. RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; Bottleneck Identification in Software Development Processes: A Proposal Based on the Principles of the Theory of Constraints; Proceedings of 2015 IEEE 10th International Conference on Global Software Engineering (ICGSE 2015); IEEE Xplorer; DOI: 10.1109/ICGSEW.2015.16; *Quails B2.*

2. RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; SILVA, M. F.; Research Opportunities on the Application of the Theory of Constraints to Software Development Process; JSW–Journal of Software; Vol. 12 Nor. 04 April, 2017; ISSN 1796-217X; DOI: 10.17706/jsw.12.4.227-239. *Quails B1*.

3. RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; SILVA, M. F; Um Meta Modelo do Processo Unificado Utilizando a Ontologia de Fundamentação Unificada-B (UFO-B); ENCOSIS-Encontro Regional de Computação e Sistemas de Informação; Anais Encosis/SBC - pp. 13-22; Maio, 2017; ISSN 2238-5096.

4. RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; A Síndrome do Deadline: Origem, Causas e Implicações no Processo de Desenvolvimento de Software. iSys – Revista Brasileira de Sistemas de Informação; vol. 10, No. 2, pp. 30-47, Junho de 2017; ISSN 1984-2902; *Qualis B3*.

5. RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; SILVA, M. F; Scheduling of Tasks in a Software Development Environment Using a Dynamic Job Shop Scheduling (DJSS). Proceedings of the XLIX Brazilian Symposium of Operational Research (SBPO); Blumenau – SC; 2017. Anais SBPO/SOBRAPO – ISSN 1518-1731. Agosto de 2017.

6. RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; SILVA, M. F. – Bottlenecks Identification in Software Development Process: A Quali-Quantitative Approach. Proceedings of the *XVI* Brazilian Symposium on Human Factors in Computational Systems (IHC 2017); Anais IHC ISSN 2316-5318; ISBN 978-1-4503-6377-8/17/10; ACM Digital Library; ACM New York – NY- USA; DOI:10.1145/3160504.3160513; October 2017. *Quails B2.*

7. RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; SILVA, M. F.; Revisão da Literatura Sobre a Teoria das Restrições Aplicada em Processo de Desenvolvimento de Software: Revista IEEE América Latina; IEEE Xplore, DOI, ISI, ISSN 1548-0992; Volume 16, Issue X, 2018*; Qualis B4.(Aceito e Previsto para 2018).*

LISTA DE PREVISÃO DE PUBLICAÇÕES

1. RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; SILVA, M. F; Um Modelo Job Shop Dinâmico para Escalonamento de Tarefas em Ambiente de Desenvolvimento de Software. Revista iSYS – Revista Brasileira de Sistemas de Informação*; Qualis B3*. *Submetido em Julho de 2018; Previsto para 2018*

2. RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; SILVA, M. F.; Bottleneck Identification in Software Development Process: An Experiment Report in Software Engineering Learning Environment; Journal of the Brazilian Computer Society; *Quails B1; Submetido em Julho de 2018;Previsto para 2018.*

3. RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; SILVA, M. F. – A Method to Identify Bottlenecks in Software Process Development – ACM Transactions on Software Engineering and Methodology; *Quails A2*; *Submetido em Julho de 2018; Previsto para 2018.*

4. RIBEIRO, S. A.; SCHMITZ, E. SILVA, M. F.; ALENCAR, A. J. S.; A Quantitative View of Bottlenecks in Software Development Process. ACM Transactions on Software Engineering and Methodology; *Quails A2; Submetido em Agosto 2018; Previsto para 2018/19*.

5. RIBEIRO, S. A.; SCHMITZ, E. A.; FALBO, R. A; ALENCAR, A. J. S.; SILVA, M. F.; Applying Unified Foundational Ontology-B (UFO-B) Concepts in the Unified Process: A Conceptual Approach; Intent submit to - Applied Ontology Journal - IOS Press: *Quails B1*; *Previsto para 2018/19*.

1 INTRODUÇÃO

"Os nossos conhecimentos são a reunião do raciocínio e a experiência de numerosas mentes!"

Ralph Waldo Emerson

Este trabalho é o resultado de uma investigação sobre identificação de gargalos no PDS (Processo de Desenvolvimento de Software) com a aplicação da Teoria das Restrições ou TOC em seu acrônimo do inglês *Theory of Constraints*.

A ideia de aplicar a TOC no Processo de Desenvolvimento de Software surgiu a partir dos resultados colhidos em uma Revisão da Literatura (RL) desenvolvida e publicada por (Ribeiro *et al*., 2017-a) e (Ribeiro *et al*., 2017-d), onde foram apresentadas algumas lacunas sobre a temática proposta nesta pesquisa. O estudo secundário revelou a inexistência de trabalhos envolvendo a TOC no processo de produção de software. O estudo mostrou ainda que a TOC é uma metodologia sólida, amplamente difundida e muito aplicada em sistemas de produção e administração industrial.

O entendimento de que o PDS, mesmo com todas as suas particularidades, também é constituído de um processo produtivo, fez com que fosse identificada na TOC uma potencial ferramenta para identificar e tratar gargalos produtivos no âmbito do desenvolvimento de software.

1.1 Contexto: O Processo de Desenvolvimento de Software

A crescente demanda por software nas últimas décadas impulsionou o desenvolvimento na indústria. A tendência é que a demanda continue crescendo, em razão da popularização dos dispositivos computacionais e da alta aplicabilidade de software para auxiliar na solução de problemas do cotidiano.

O aumento da procura por software exige que a indústria acelere o desenvolvimento para atender o mercado e sua grande diversidade de domínios e aplicações. Para atender esta demanda, é preciso que a indústria adote e aplique padrões ao processo produtivo apoiado por metodologias, métodos e processos de desenvolvimento sólidos e confiáveis.

O processo de produção de software apresenta uma diferença fundamental com relação aos processos produtivos industriais – a unidade produtiva de software é organizada para produzir um único produto de cada vez, e em princípio, devendo ser reconfigurada para a confecção de um novo produto. Este cenário exige que o processo de desenvolvimento seja delineado de acordo com a especificidade de cada projeto/produto, como apontam (Pressman e Maxin, 2014), (Pfleeger, 2009), (Sommerville, 2011) e (Schach, 2011).

Para melhor contextualizar e trabalhar estas questões no PDS é preciso buscar um entendimento sobre a abrangência da metodologia, a aplicabilidade do método e a melhor maneira de executar o processo. Em seguida, é necessário ajustar o PDS ao domínio do problema na tentativa de buscar as soluções que atendam as expectativas do cliente e os objetivos da indústria de software.

Segundo o MPS.BR (2016), um processo se define por um conjunto de atividades inter-relacionadas ou interativas, que transforma insumos (entradas) em produtos (saídas). Enquanto que, um método orienta a execução de um processo em conformidade com uma norma ou procedimento e uma metodologia conduz o processo com a aplicação das melhores práticas do desenvolvimento de software.

De acordo com o CMMI/SEI (2016), uma metodologia envolve: modelos, dados, métodos e algoritmos e deve ser utilizada para determinar as necessidades, competências, esforço e custo. Já os métodos definem uma abordagem estruturada, objetiva e formal para o desenvolvimento de software, que inclui: modelos, notações, regras, recomendações e diretrizes para a garantia da qualidade do produto de software. O CMMI/SEI (2016) diz ainda que um processo na área de software representa um conjunto de atividades organizadas lógica e sistematicamente, cujo objetivo é atingir uma meta previamente estipulada.

Conforme Sommerville (2011), uma Metodologia de Desenvolvimento de Software (MDS) é uma representação simplificada de um Processo de Desenvolvimento de Software (PDS), aplicada com um conjunto de boas práticas. Em geral este conjunto de boas práticas é empregado em fases ou passos, que são subdivisões do processo, para melhor ordená-lo, controlá-lo e gerenciá-lo. Ainda segundo Sommerville (2011), um método é uma abordagem estruturada para o PDS com o objetivo de facilitar a produção de software, enquanto que processo de software é um conjunto de atividades e resultados associados

que geram um produto de software. Em outras palavras, mas na mesma linha, Guimarães *et al.* (2012), afirmam que uma metodologia é um roteiro para orientar a condução de um projeto contendo atividades, técnicas, tecnologias, regras, artefatos, recursos e métodos para serem usados no desenvolvimento. Já os métodos definem o ciclo de vida dos processos dividindo-os em etapas para facilitar o gerenciamento e o controle. Com relação a processos, Guimarães *et al.* (2012) define como um conjunto pré-definido de atividades ou comportamentos executados por humanos ou máquinas a fim de alcançar uma ou mais metas.

Para Schach (2011), o processo de software é a maneira pela qual produzimos software e como tal, deve abranger a metodologia com seu ciclo de vida e técnicas subjacentes. Já a metodologia, ainda segundo Schach (2011), é um componente de um processo de software, sendo que a principal metodologia existente atualmente é o Processo Unificado (PU).

Os criadores do Processo Unificado (PU)[1] afirmam que este agrega os três conceitos: (1) como metodologia, o PU define uma estratégia; (2) como método, o PU define os modos em que essa estratégia deve ser realizada; e (3) como um processo, o PU define uma série de passos que podem ser seguidos para atingir a estratégia estabelecida pelos métodos.

Nesta pesquisa, o termo processo será denotado como uma representação abstrata, envolvendo a metodologia e os métodos como instâncias do processo. Desta forma, o PU será tratado neste trabalho como uma ferramenta para o PDS suportado conceitualmente por uma metodologia, estrategicamente definido por um método e estruturalmente dirigido por um processo. O PU foi identificado como a ferramenta ideal para o PDS desenvolvido nesta pesquisa, o qual será utilizado para guiar o processo para a aplicação da TOC na tentativa de identificar as restrições do sistema produtivo e dirimir os gargalos do PDS.

1.2 Caracterização de Gargalo no Processo Produtivo

Segundo Goldratt e Cox (2006), um gargalo é um recurso que possui capacidade menor ou igual à demanda atribuída a este recurso. Enquanto que uma restrição é um fator ou obstáculo que limita um melhor desempenho do sistema em relação às metas

[1] ***Grady Booch:*** *comunicação pessoal via e-mail em 20 de fevereiro de 2016 - Disponível no Apêndice J.*

estabelecidas. Uma restrição quando não tratada pode criar novos pontos de estrangulamento, transferindo o gargalo para outro local no sistema produtivo. O gargalo determina a limitação da produção através de sua taxa de vazão, ou seja, a produção do sistema está condicionada a capacidade do gargalo. Isto implica que todo gargalo possui uma restrição associada a ele.

A Figura 1 representa um processo produtivo linearizado do tipo batelada (lote), destacando o gargalo e sua taxa de vazão, dado por C_{max} = *40 Ton/Dia*. Nessa apresentação é possível observar que o processo produtivo apresenta algumas restrições locais, associadas à capacidade das máquinas em determinadas etapas do processo.

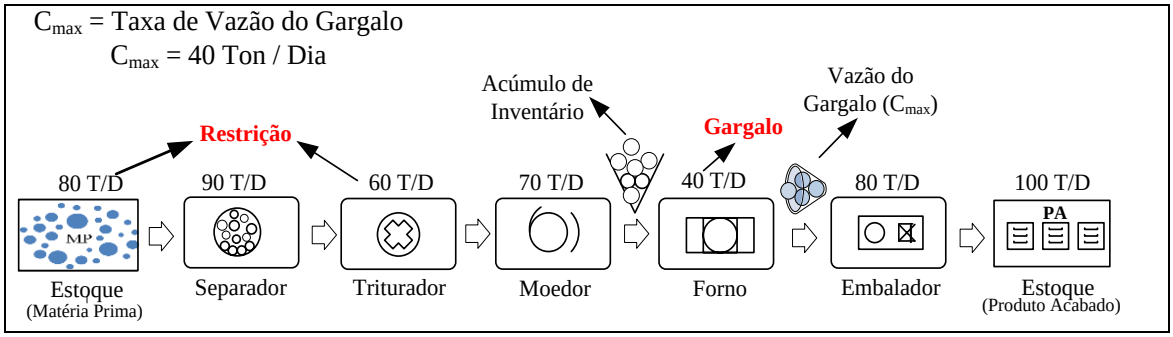


Figura 1: Representação do gargalo no processo produtivo da indústria

É importante conhecer e entender estas restrições para ajustar o processo em função da capacidade máxima do sistema (C_{max} do gargalo), e assim evitar que a produção de inventários seja menor que a capacidade do gargalo impedindo o surgimento de um novo gargalo no sistema. O processo deve também ser ajustado para que a produção de inventários não seja maior que a capacidade do sistema, pois isso gera acúmulo de inventário antes do gargalo e afeta diretamente a maximização do ganho, que é a meta de toda empresa, segundo Goldratt e Cox (2006).

1.3 Caracterização do Gargalo no PDS

Neste trabalho, assumiu-se que uma restrição é qualquer perturbação no Processo de Desenvolvimento de Software (PDS) que possa, quando não tratada, impactar diretamente na produção do software. As restrições geralmente estão associadas às características etnográficas das máquinas (pessoas) na Linha de Produção (LP) e devem ser identificadas e tratadas durante a execução do processo.

Um gargalo, por sua vez, é qualquer elemento restritivo que impede a maximização da produção. Neste trabalho, o gargalo está diretamente associado à capacidade produtiva de uma LP em realizar uma tarefa e consequentemente construir um artefato e é causado por variáveis qualitativas (esforço, dificuldade e capacidade produtiva).

Analogamente ao processo produtivo manufatureiro apresentado na Figura 1, a Figura 2 apresenta um esquema representativo do gargalo no processo de desenvolvimento de software, onde um Recurso com Restrição de Capacidade (RRC) representa uma restrição no sistema produtivo, enquanto que um Recurso sem Restrição de Capacidade (RsRC) representa um recurso com habilidades em resolver tarefas diversas no âmbito do PDS.

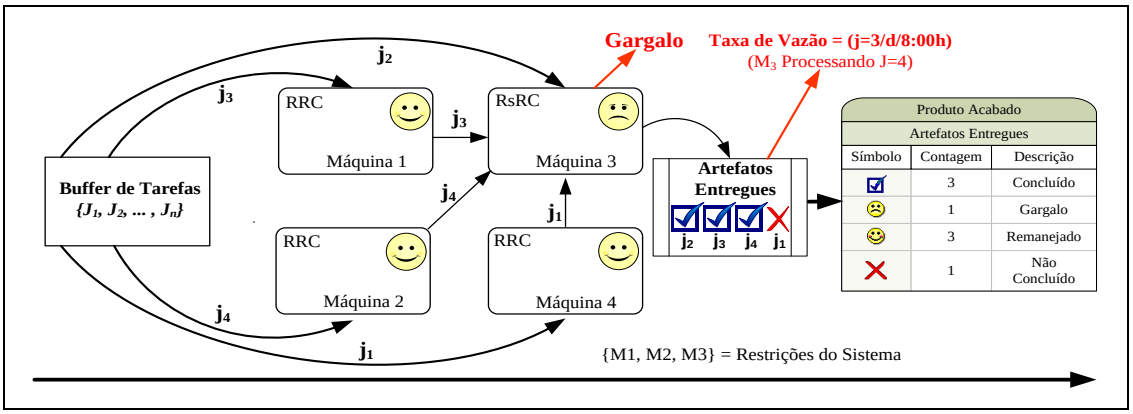


Figura 2. Representação do gargalo no PDS

Os RsRC não possuem restrições técnicas. As restrições são físicas, *i.e*, restrições associadas a seu limite físico no processo laboral. Estas restrições determinam a sua capacidade produtiva ou a sua taxa de vazão em uma determinada fração de tempo, transformando-o no gargalo do sistema. Os RRC possuem alguma dificuldade técnica em concluir ou desenvolver uma tarefa. Quando isto ocorre, geralmente a tarefa é transferida total ou parcialmente para uma máquina RsRC, exigindo mais esforço e provocando a saturação da máquina. Este processo cria um gargalo no sistema, uma vez que a carga de trabalho está acima de sua capacidade produtiva (ou taxa de vazão) e o esforço extra impede a maximização da produção, uma vez que consome recursos que seriam aplicados em outra atividade.

1.3.1 Terminologia do Gargalo no PDS

A terminologia usada para caracterizar um gargalo no PDS envolve três fatores intrinsicamente associados: (1) Esforço – é o tempo total (minutos) gasto por uma

máquina/LP para produzir um determinado artefato; (2) Produção – é o percentual de artefatos produzidos e entregue por uma LP, e (3) Dificuldade – é a dificuldade percentual relatada pelas máquinas/pessoas em cada atividade.

O esforço aplicado para realizar uma atividade afeta diretamente a produção dos artefatos. Quando uma LP consume muitos recursos (tempo e pessoas) em uma atividade, ela sistematicamente está desalocando recursos destinados à outra atividade, impedindo a maximização da produção. Isto porque o orçamento de horas é pré-fixado pela agenda e igual para todas as LPs do *Shop*.

A produção está associada à capacidade técnica, intelectual e física de uma máquina/LP e aumenta ou diminui de acordo com o total de esforço empregado e com o grau de dificuldade encontrada.

A dificuldade geralmente implica em alto esforço ou baixo esforço. Neste último caso, isto é provocado quando a dificuldade é alta e a LP possui pelo menos um RRC, *i.e*, uma máquina/LP com restrição de capacidade para executar determinada tarefa da agenda.

1.4 Motivação

Consensualmente a literatura prega que o PDS deve ser apoiado por ferramentas, métodos e modelos para dirigir o desenvolvimento, obedecendo logicamente às características de cada projeto. Em geral, o processo conduz o desenvolvimento proporcionando a gestão do PDS e em alguns casos específicos promovendo apenas ajustes e melhorias locais e oportunas no ciclo de desenvolvimento do produto de software.

Isto, segundo o CMMI/SEI (2016) ocorre porque existem no mercado modelos de maturidade, padrões, metodologias e diretrizes que auxiliam as organizações na melhoria de seu processo. No entanto o CMMI/SEI (2016) afirma que a grande maioria destas abordagens não contempla toda a extensão do processo, pois o foco é uma parte específica do negócio e não uma abordagem sistêmica em relação aos problemas encontrados no processo de produção de software.

Ao focar na melhoria de uma única área do negócio, esses modelos infelizmente têm ajudado a perpetuar as barreiras e compartimentalizações que existem nas organizações (CMMI/SEI, 2016).

Segundo Dennis e Wixom (2014), o PDS deve ser avaliado e medido em toda sua extensão, seja qualitativamente ou quantitativamente. Na prática, segundo Wheeler (2001), o que é empregado na maioria das vezes são roteiros que guiam o processo, através da aplicação de regras metodológicas que tentam exercer algum tipo de controle sobre o mesmo. Isto não garante a estabilidade do processo, pois não envolve a previsibilidade, a identidade, a capacidade e a mensuração, que são indicadores utilizados para o monitoramento, o controle e o gerenciamento da evolução do processo (Wheeler, 2001).

A percepção destes problemas impulsiona o desejo de inserir a TOC no PDS com o intuito de aumentar a eficiência do desenvolvimento a partir da identificação e do tratamento dos gargalos. Esta questão é fundamental, pois se for possível identificar os pontos de estrangulamentos, será possível também direcionar melhor os recursos utilizados no PDS, e com isso, reduzir o impacto nos projetos de construção de software.

A TOC é uma metodologia sólida, amplamente difundida e aplicada em sistemas de manufatura. Isto foi evidenciado após a análise dos trabalhos de (Almeida, *et al.*, 2013), (Mansourabad, *et al.* 2013), (Rhee, *et al.* 2010), (Baptista, *et al.*, 2013), (Sengupta, *et al.* 2008), (Kim *et al.*, 2008) e (Zou e Rose, 2009) que envolvem a utilização da TOC na otimização de processos industriais. Estes trabalhos mostram como a TOC pode ajudar na identificação e no tratamento dos elementos restritivos do processo produtivo na indústria manufatureira. Uma vez que o software também passa por um processo produtivo, temos indicações que a TOC pode igualmente ser aplicada no PDS, e consequentemente promover melhorias na produção de software.

A literatura da TOC mostra que os gargalos em geral, mesmo os triviais, não obedecem a um padrão específico e nem são comuns a todos os tipos de ambientes. Os gargalos geralmente estão associados aos recursos utilizados, ao ambiente de desenvolvimento e ao produto a ser desenvolvido. O processo por sua vez, sofre perturbações de variáveis internas e externas, impactando diretamente no ciclo de desenvolvimento do produto (Wheeler, 2006).

Uma pergunta que se segue naturalmente é: “Seria plausível a aplicação dos princípios da TOC no processo de desenvolvimento de software?”. Uma resposta positiva a esta pergunta poderia ser de grande aplicação na área de melhoria de processos de desenvolvimento de software através da inserção de novos métodos adaptados a partir daqueles correntemente aplicados na identificação e no tratamento de gargalos nos

processos industriais. Isso possibilitaria uma forma de promover os ajustes necessários no processo de desenvolvimento, voltado não somente para a solução de problemas específicos, como também, para uma melhor utilização dos recursos com a subsequente otimização dos resultados produtivos.

A validade da realização desse estudo veio da constatação da inexistência de trabalhos realizados com esta finalidade após uma revisão da literatura sobre o tema, apresentada no Capítulo 3. A revisão da literatura sobre a aplicação da TOC em processos de software foi iniciada em fevereiro de 2013 e foi finalizada em setembro do mesmo ano. Como toda revisão literária se perde com o tempo em razão do surgimento de novos trabalhos, é necessário que novas buscas sejam sistematicamente repetidas. Neste trabalho, replicamos as buscas a cada seis meses com o intuito de identificar novos trabalhos correlatos. Até o momento da construção deste texto, o único estudo relacionado ao tema proposto foi o artigo de (Ribeiro *et al.* 2015), publicado em Julho de 2015, que é fruto do presente trabalho.

1.5 Proposta: Identificação de Gargalos no PDS

A TOC tem como princípio fundamental a ideia de que todo processo pode ser continuamente melhorado (Goldratt e Cox, 2006), (Goldratt, 1993), (Goldratt, 2006). Para que isso seja realizado, devemos primeiramente identificar os elementos restritivos do processo para, em seguida, aplicar o tratamento adequado de acordo com os princípios da TOC. A aplicação correta das medidas necessárias para ajustar o processo de acordo com o ambiente e com os objetivos do projeto pode permitir uma melhoria do processo e, por conseguinte, do produto de software.

Este trabalho mostra os resultados de uma investigação sobre a existência e a identificação de gargalos no processo de desenvolvimento de software. O tratamento das restrições, bem como os meios aplicados para realizar os tratamentos caracterizam-se como uma proposta complementar.

Como este tipo de estudo requer a observação de várias instâncias de processos de desenvolvimento, a pesquisa foi direcionada a um tipo particular de processo – processo de desenvolvimento de software, executado em um ambiente de aprendizado da disciplina de Fundamentos de Engenharia de Software (FES), do Departamento de Ciência da Computação (DCC) da Universidade Federal do Rio de Janeiro (UFRJ).

O estudo realizado para a confecção deste trabalho envolve uma quase experimentação[2] *in vivo*[3] em um ambiente de aprendizagem de Engenharia de Software, com ensaios *in vitro*[4] em um ambiente controlado, conforme apresentado por Basili (2007). A opção em adotar a quase experimentação se deu pelo fato de que não foi elaborado um grupo de controle para realizar a experimentação.

Nesse ambiente, grupos de alunos trabalharam no desenvolvimento completo, desde a especificação de requisitos, construção/codificação e testes até a entrega de um mesmo produto de software pelas linhas de produção. Os ensaios foram repetidos três vezes, com equipes e domínios diferentes.

Em uma linha de produção fabril o produto final é o resultado de uma sequência de operações executadas por máquinas especializadas. No nosso caso, o produto final (software) também é obtido por uma sequência de operações executadas pelos componentes (alunos) de cada grupo. Para iluminar a analogia, usaremos o termo máquina para denotar cada componente processando um conjunto de operações em uma dada linha de produção (grupo) para a construção das partes (artefatos) que compõe o produto final.

1.6 Objetivos da Pesquisa

Esta pesquisa tem como objetivo principal apresentar um método para identificar quais são os elementos restritivos e os potenciais gargalos encontrados em um PDS. A particularização do método é um objetivo específico desta pesquisa, o qual foi aplicado e executado em um ambiente de aprendizado de Engenharia de Software.

Um objetivo secundário é identificar os elementos restritivos encontrados no PDS e aplicar os tratamentos necessários na eminência de suprimir os gargalos do PDS. O tratamento das restrições está diretamente relacionado ao objetivo principal da pesquisa e a

---

[2] **Quase experimento:** *Segundo Wollin et. al (2007), os quase-experimentos são pesquisas que não possuem distribuição aleatória dos sujeitos ou dos objetos. Os quase experimentos se caracterizam ainda por não possuírem grupos de controle. Já os experimentos são pesquisas que possuem grupos de controle e os sujeitos ou objetos são escolhidos ao acaso (randomicamente).*

[3] ***In vivo**: Estudo que envolve indivíduos em seu próprio ambiente: Em Engenharia de Software o estudo experimental se desenvolve durante o processo de desenvolvimento de software sob condições reais de trabalho. Adaptado de (Travassos et.al, 2002).*

[4] ***In vitro:** Estudos executados em laboratório ou comunidade controlada. Em Engenharia de Software estes experimentos, em sua maioria, são executados em universidades ou entre grupos selecionados de uma organização de desenvolvimento de software. Adaptado de (Travassos e Barros, 2003).*

sua particularização e será aplicado durante a experimentação obedecendo sequencialmente os preceitos da TOC.

1.6.1 Questões de Pesquisa

O trabalho foi iniciado com uma série de questões levantadas sobre o tema proposto. As questões de pesquisa ajudam a delinear e direcionar o trabalho na tentativa de encontrar as respostas necessárias para identificação e tratamento dos gargalos no PDS.

A partir das indagações iniciais, foi elencado um conjunto de 8 questões a serem respondidas com esta investigação. Os tópicos abaixo irão apresentar as questões levantadas na ordem em que deverão ser respondidas.

1. Existem gargalos no processo de desenvolvimento de software?
2. O gargalo é o mesmo para todas as Linhas de Produção?
3. Os gargalos mudam no decorrer do processo de desenvolvimento?

O segundo grupo de questões (4-8) serão respondidas com base na análise dos dados das três primeiras questões. As respostas serão consequência da aceitação ou refutação das hipóteses nula (H_0) e alternativa (H_1) das questões 1, 2 e 3.

4. Existe um gargalo predominante?
5. Existe relação entre restrições (*quali)* e os gargalos (*quanti*)?
6. As intervenções influenciam no tratamento do gargalo?
7. O gargalo requer mais esforço da máquina?
8. O gargalo está sempre associado à mesma máquina?

Ao responder estas questões será possível identificar e tipificar os gargalos no PDS, com vistas a apontar qual o caminho a ser seguido para alcançar as metas de desenvolvimento em um processo produtivo de software.

1.6.2 Hipóteses Levantadas

Para cada questão de pesquisa foram levantadas duas hipóteses (H_0 *e* H_1), onde: H_0 corresponde à hipótese nula e H_1 corresponde à hipótese alternativa. Os tópicos abaixo irão apresentar as hipóteses na ordem que deverão ser respondidas e conforme a disposição das três primeiras questões de pesquisa apresentadas na Seção 1.6.1.

Questão 1 (*Q1*)

H_0 = Não existe gargalo no processo de desenvolvimento de software!

H_1 = Existe gargalo no processo de desenvolvimento de Software!

Questão 2 (*Q2*)

H_0 = O gargalo é o mesmo para todas LPs!

H_1 = O gargalo não é o mesmo para todas LPs!

Questão 3 (*Q3*)

H_0 = O gargalo não muda no processo de desenvolvimento!

H_1 = O gargalo muda no processo de desenvolvimento!

A análise dos dados colhidos durante a experimentação permitirá aceitar ou refutar estas hipóteses para dar sustentação à proposta apresentada ao longo deste texto.

1.7 Nomenclatura e Termos Utilizados

A Tabela 1 apresenta a definição dos termos e expressões utilizados nesta tese.

Tabela 1: Lista de termos e definições utilizados na tese

Termos	Descrição
Artefato	Produto resultante (saída) da execução de uma ou mais atividades[5]. Neste texto, o termo artefato será empregado para referenciar à saída de um conjunto de atividades processadas e agrupadas por tipo (Arquitetura, Banco de dados, Codificação, Documentação e Teste).
Atividade	Conjunto de ações e/ou operações realizadas que produz um resultado (artefato).
Construção *One way*	Termo usado para codificação sem verificação e sem validação (teste V e V).
Exp1, Exp 2 e Exp 3	São as rodadas/ensaios quase-experimentais de cada *shop*.
Gargalo	Limitador da capacidade produtiva das Linhas de Produção.
Linha de Produção (LP)	Composta por um conjunto de componentes/pessoas (máquinas) com *(M ≥ 2, ≤ 4);*
Máquina	Componente de uma linha de produção representado por uma pessoa / aluno.
Mini gargalo	Máquina com restrição de recursos, geralmente capacidades técnicas; Um *mini gargalo* é originado por uma restrição qualitativa, e afeta apenas a linha de produção em que a máquina está associada.
Restrição	Qualquer perturbação na linha de produção ou no *Shop*. Em geral está associada a questões qualitativas.
Resultados produtivos	Resultados gerados em cada *quase-experimento*. Compreende a produção das máquinas/linhas de produção e todo o conteúdo gerado pelos pesquisadores (Planilhas, Textos, Relatórios, Banco de Dados, etc.).
Shop	Corresponde ao ambiente produtivo criado para cada ensaio experimental (Exp1, Exp2 e Exp3). É composto pelo PDS, pelo domínio e pelas linhas de produção/máquinas.
Shop 1, *Shop* 2 e *Shop* 3	Refere-se a cada ambiente produtivo criado para executar os ensaios.
Síndrome do *Deadline*	Fenômeno identificado nos *quase-experimentos* e usado para tipificar

[5] *Um artefato também pode ser uma entrada, necessária para uma atividade produzir outro resultado (artefato).*

	atrasos decorrente da priorização de tarefas pelas máquinas. Uma definição mais completa pode ser encontrada em (Ribeiro *et al.* 2017-c).
Tarefa	Conjunto de 26 unidades (*Jobs*) prescritas a serem escalonadas no *shop* para cumprimento de uma agenda (metas e objetivos) predeterminada.
Operadores e Incógnitas	Todos os operadores e incógnitas matemáticas, bem como caracteres especiais aparecerão neste texto em itálico. Ex. $\in$, α, $\exists$, *(L) (M)*, etc.
Vocábulo Estrangeiro	Todos os vocábulos escritos em idioma estrangeiro aparecerão em itálico ao longo do texto.
Citações	As citações bibliográficas deste texto obedecem ao disposto na ABNT, desta forma, citações diretas e citações referenciais aparecerão entre parênteses, As citações indiretas serão referenciadas iniciando com o nome do autor, seguida da dada/ano entre parênteses.

1.8 Cobertura e Limitações da Tese

Esta proposta limita-se em apontar um caminho para identificar os elementos restritivos no PDS, validar potenciais gargalos e relatar a experiência na expectativa de contribuir com futuras pesquisas relacionadas à identificação e tratamento de gargalos no processo de produção de software.

Por razões ambientais, metodológicas e de direcionamento, muitas questões que envolvem o tema e a pesquisa realizada não serão cobertos nesta tese. Estas questões poderão ser exploradas em trabalhos futuros pelos realizadores desta pesquisa ou por outros pesquisadores que queiram contribuir com o tema proposto.

Os tópicos abaixo apresentam algumas destas questões, as quais não serão tratadas nesta pesquisa, mas que devida à importância, merecem ser exploradas em momentos e oportunidades futuras.

- Existe relação entre o nível de experiência da equipe e os gargalos encontrados?
- A qualidade dos artefatos produzidos tem relação direta com a dificuldade relatada no *survey* pelos participantes?
- O gargalo influencia no custo do projeto?
- Os gargalos encontrados são os mesmos encontrados na indústria de software?
- Os tratamentos aplicados podem ser executados na indústria de software?
- Os gargalos são os mesmos em ambiente distribuído de desenvolvimento?
- É possível simular um ambiente de desenvolvimento baseado em um esquema *job shop* no ambiente real (indústria de software)?
- O PDS adotado pode ser aplicado na indústria?
- Quais os fatores que levaram a desistência de algumas pessoas/máquinas?

- Fatores socioeconômicos, regionais e culturais influenciaram na pesquisa?
- Fatores socioeconômicos, regionais e culturais influenciaram na produção dos artefatos e consequentemente no produto final (software desenvolvido)?

Outras tantas questões poderiam ainda ser identificadas e exploradas. Acredita-se que este é um campo vasto, e que carece de mais estudos exploratórios para que se possa ter um alinhamento fidedigno dos objetivos de quem desenvolve com as necessidades de quem usa o produto (software).

1.9 Organização da Tese

Esta tese está organizada em dez seções temáticas, sendo: 8 capítulos formais (incluindo a introdução e as considerações conclusivas), a relação das fontes consultadas e devidamente referenciadas ao longo do texto e os Apêndices, constituídos de textos conceituais e documentos (Planilhas, Formulários e Modelos) elaborados durante a fase de construção do conhecimento deste trabalho.

A Introdução é o primeiro Capítulo desta tese, e como visto, apresenta a contextualização, a motivação, a proposta, os objetivos, a terminologia usada e a abrangência desta pesquisa.

O Capítulo 2 apresenta a fundamentação teórica das disciplinas envolvidas nesta pesquisa, iniciando pela apresentação dos princípios da TOC e o método do Tambor Pulmão e Corda (TPC), essencial para suportar a metodologia de pesquisa e a construção do processo experimental para identificação e tratamento dos gargalos. A conceituação do PU, os fundamentos da Engenharia de Software Experimental e o problema de *Job Shop Scheduling* (JSS) também são apresentados neste capítulo.

O Capítulo 3 apresenta a revisão da literatura sobre o emprego da TOC no PDS. A revisão da literatura evidenciou várias lacunas sobre o tema, dentre elas o objeto deste estudo, apresentado por (Ribeiro *et al*. 2017-a) e (Ribeiro *et al*. 2018) como uma oportunidade de pesquisa que envolve a aplicação da TOC para identificar gargalos no PDS.

O Capítulo 4 exibe o plano laboral para realização da pesquisa, descrevendo: (1) o planejamento da pesquisa juntamente com os critérios utilizados para mensurar e validar os resultados obtidos; (2) o estudo empírico realizado; e (3) o histórico do desenvolvimento

do trabalho; e (4) o modelo *Job Shop Scheduling Dinãmico* (JSSD), usado para escalonar as tarefas nos *Shops*.

O Capítulo 5 apresenta o método de identificação de gargalos, com destaque para: (1) a particularização do método aplicado aos ensaios experimentais realizados neste estudo; (2) a abordagem qualitativa do método com a aplicação da TOC e do TPC para identificar e tratar as restrições *"quali"*; (3) a abordagem quantitativa do método com a *baseline* e a formulação para os cálculos de esforço, produção e dificuldade; e (4) o modelo algoritmo do método particularizado.

O Capítulo 6 destaca a avaliação qualitativa do experimento. A análise *"quali"* foi utilizada para identificar as restrições no processo de desenvolvimento de software em tempo de execução. O Capítulo apresenta também a tipificação das restrições qualitativas encontradas, classificadas como: (1) restrições de ordem técnica (RT); e (2) restrições de ordem comportamental (RC). Os tratamentos aplicados às restrições durante a execução do experimento também são apresentados neste capítulo.

O Capítulo 7 sinaliza a abordagem analítica quantitativa a partir dos dados colhidos e mensurados nesta pesquisa. Basicamente o capitulo 7 se subdivide em duas seções principais: (1) análise descritiva: que compreende em uma primeira visão sobre os resultados coletados durante a execução dos ensaios; e (2) análise estatística: nesta etapa o foco é responder estatisticamente as questões levantadas sobre a identificação de gargalos no PDS. O Capítulo 7 apresenta ainda uma discussão sobre os gargalos encontrados no PDS.

O Capítulo 8 traz as considerações finais a respeito das discussões em torno do tema abordado e dos resultados colhidos, a relevância da pesquisa e sua contribuição para o processo de desenvolvimento de software. Esta seção desvela ainda a experiência adquirida pelo autor e a motivação para trabalhos futuros na área de processo de software e áreas correlatas ao desenvolvimento de software. A delimitação da pesquisa listada na Seção 1.8 também é ligeiramente abordada nas considerações finais.

As Seções subsequentes (9 e 10) são apresentadas na seguinte ordem: (1) as Referências Bibliográficas, que listam os textos estudados e citados nesta tese. Todas as citações encontradas no texto estão elencadas nesta seção, incluindo as citações presentes nos textos dos apêndices; e (2) os Apêndices, que exibem os textos complementares que compõe a fundamentação multidisciplinar estudada e conceituada durante a fase inicial da

exploração e busca do conhecimento. O Apêndice apresenta ainda os documentos produzidos pelo autor e utilizados nesta pesquisa, como: Planilhas, Listas, questionários, agendas (dos três experimentos), o Template experimental, *Surveys* (pesquisa de dificuldades encontradas), os modelos BPMN (*Business Process Model Notation*) do processo, o *feedback* escrito e o modelo da corrente crítica e do caminho crítico.

2 FUNDAMENTAÇÃO TEÓRICA

"É a teoria que decide o que podemos observar."

Albert Einstein

A primeira seção deste capítulo trata da Teoria das Restrições. A ideia de aplicar a TOC no processo de desenvolvimento de software foi devido a sua alta aplicabilidade em processos de manufatura. O conjunto arquitetural e estrutural da TOC mostra-se muito eficiente para a melhoria continuada do processo (Trojanowsk e Pajak, 2010), através da identificação e do tratamento nos gargalos do ambiente produtivo.

A identificação dos elementos restritivos de um sistema de produção requer que o processo produtivo seja bem definido e repetitivo. O PU (Processo Unificado), objeto da segunda seção, foi escolhido como a ferramenta para guiar o processo de desenvolvimento durante a construção do produto de software, por possuir tais características.

O estudo empírico realizado durante este projeto foi amparado por trabalhos e técnicas amplamente aplicadas no campo da Engenharia de Software Experimental (ESE). A fundamentação do estudo empírico desenvolvido é o terceiro ponto abordado neste capítulo.

O modelo teórico adotado nesta pesquisa teve como base o problema de escalonamento conhecido como *Job Shop Scheduling Problem (JSSP)*. Em virtude do dinamismo do processo de desenvolvimento de software e seu ambiente de execução, o escalonamento linear das tarefas não se aplica e o modelo adotado foi o do problema *Dynamic Job Shop Scheduling Problem* (DJSP).

2.1 Teoria das Restrições: Conceitos Gerais

A Teoria das Restrições ou TOC, acrônimo de *Theory of Constraints* é definida como um tipo de metodologia de melhoria contínua que evoluiu e expandiu sua base metodológica ao longo do tempo (Kim *et al.*, 2008). A proposta, como descrita em (Goldratt e Cox, 2006) teve início na década de 80, como uma filosofia de gerenciamento produtivo que incorpora um aspecto prático na tomada de decisão dentro do ambiente

produtivo, baseando-se no princípio de que qualquer limitador que impeça alcançar a meta desejada sinaliza uma restrição do sistema.

Uma restrição pode ser pensada como sendo o elo mais fraco de uma corrente, ou "o menino gordo", conforme a analogia feita por (Goldratt e Cox, 2006) em seu livro intitulado, "A Meta". Um axioma fundamental da TOC declara que todo sistema possui pelo menos uma restrição ou que não existe qualquer processo que não sofra a influência de pelo menos uma restrição (Goldratt e Cox, 2006).

A TOC é amplamente empregada nos pátios industriais para a melhoria continuada de processos como mostram os trabalhos de (Goldratt e Kishira, 2009), (Rahman, 1998), (Asan, 2009) e (Giutini, 2002). Todos estes textos serviram de inspiração e exemplos para o estudo de aplicação da TOC ao processo de desenvolvimento de software.

Os trabalhos de (Zhou e Rose, 2009), (Hasgul e Kartal, 2007), (Sengupta *et al.*, 2008), (Gupta *et al.*, 2010), (Rahman, 1998), (Kim *et al*. 2008), (Rogers *et al.*, 2006) e (Mabin e Balderstone, 2000), (Balderstone e Mabin, 1998) mostram que a TOC é uma metodologia madura, amplamente empregada nas mais variadas frentes do processo produtivo. As razões para isso incluem o fato de que a TOC é uma metodologia simples, de fácil entendimento, fácil aplicação e extremamente eficaz, como definido por Gupta *et al.*(2010), Kim *et al.* (2008) e Gupta e Boyd (2008).

A observação de um sistema produtivo mostra a presença de restrições devido às influências externas a que o sistema está sujeito. Aliado a isso, o caráter dinâmico do ambiente faz com que as restrições sejam alteradas no decorrer do tempo. De forma análoga, pode-se assumir que o processo de desenvolvimento (produção) de software possui restrições ou sofre influências de restrições, tanto internas como externas. Essas restrições, quando não tratadas, são traduzidas em gargalos no sistema produtivo. Identificar os elementos restritivos do processo de software amparado pela TOC estabelece a motivação principal desta pesquisa, tratá-los é o desejo maior.

A TOC baseia-se em cinco passos básicos para fundamentar o processo de melhoria continua, desde que as metas do sistema estejam bem definidas. Os itens a seguir apresentam os cinco passos da TOC de acordo com as definições de (Goldratt e Cox, 2006).

Passo 1: IDENTIFICAR as restrições do sistema que possam impedir a maximização do ganho;

Passo 2: EXPLORAR as restrições através de um processo decisório para obter a máxima capacidade da restrição, sabendo que as restrições governam a produção;

Passo 3: SUBORDINAR todo o processo produtivo às restrições, inclusive a decisão do passo 2 acima;

Passo 4: ELEVAR as restrições, aumentando a performance do sistema de acordo com a capacidade máxima da restrição;

Passo 5: RETORNAR ao passo 1, caso uma restrição seja quebrada/tratada após a execução dos passos anteriores, para que a inércia não crie uma nova restrição no sistema produtivo.

Os cinco passos da TOC têm como objetivo garantir que os constantes esforços de melhoria sejam centrados nas restrições da organização. Na literatura da TOC isto é referenciado como Processo de Melhoria Contínua (PMC).

2.1.1 Aplicação da TOC ao PDS

Um ponto observado durante a pesquisa sobre a TOC é sua alta adaptabilidade que fez com que enxergássemos na TOC uma forte aliada para identificar e tratar elementos restritivos em um PDS. A adaptação da TOC ao PDS pode ser aplicada através do conceito *Drum, Buffer and Rope* (DBR) para a identificação dos gargalos, e do algoritmo da TOC para o tratamento dos gargalos identificados. A seção seguinte descreve o método do Tambor, Pulmão e Corda (TPC), em sua tradução literal do DBR para o português.

2.2 O Método Tambor, Pulmão e Corda (TPC)

O TPC é um conceito derivado da TOC muito utilizado na manufatura para o planejamento e a programação das tarefas no ambiente de produção. O método é apresentado por Goldratt e Cox (2006) perpetrando uma analogia com um grupo de escoteiros andando em fila em uma trilha. No exemplo, Goldratt e Cox (2006) usa as características físicas dos indivíduos do grupo na tentativa de organizar a caminhada de forma mais harmoniosa, evitando que os garotos mais ágeis disparem na frente e os mais lentos fiquem pelo caminho. O protagonista do livro, Alex Rogo, aplica a ideia na linha de produção da fábrica em que trabalha, e denomina o método de Tambor, Pulmão e Corda

(TPC)[6]. A origem do termo está associada aos mecanismos utilizados para programar a produção e para melhor empregar os recursos no processo produtivo de sua fábrica.

O método TPC é definido por (Goldratt e Cox, 2006) como:

1. **Tambor (Drum):** refere-se ao gargalo (quando conhecido). A denominação se dá pelo fato do ritmo de produção ser ditada pelo gargalo. Ou seja, não é possível produzir mais do que a capacidade do gargalo. Além disso, o gargalo não pode ser interrompido. Caso isso aconteça, toda a produção será afetada. Para garantir que isto não ocorra, o gargalo deve ser alimentado de acordo com sua capacidade;

2. **Pulmão (Buffer):** é o estoque de inventários que alimenta o gargalo. Seu tamanho deve ser gerenciado, de forma a manter continuamente, um estoque mínimo para que o gargalo não pare ou não acumule inventário antes do gargalo;

3. Corda (Rope): mecanismo que gerencia a liberação de material para a linha de produção no ritmo do gargalo. A corda "amarra" a liberação de acordo com a cadência do gargalo. Este processo faz com que não acumule inventário no *buffer*, evitando desta forma, que Recursos sem Restrição de Capacidade (RsRC)[7], anteriores ao gargalo produzam mais do que o gargalo é capaz de consumir.

2.2.1 Exemplo de Aplicação do TPC

Para exemplificar, pode-se usar a definição feita por Woeppel (2008), que afirma que o pressuposto fundamental para a aplicação do TPC é de que dentro de qualquer planta de produção existe um número limitado de recursos escassos, que controlam a produção geral dessa planta. Este limitador de recursos é o tambor, que define o ritmo de produção para a planta independente dos outros recursos.

O tambor é usado para maximizar a produção do sistema produtivo (Pandit, 2009). Desta forma, o planejamento e a execução são focados na exploração do tambor para proteger o sistema contra perturbações que possam afetar a produção da planta. Isto é feito através da utilização de *buffers* de tempo, da subordinação de todos os outros recursos do sistema e das decisões em consonância com o ritmo do tambor por meio de um mecanismo semelhante a uma corda. A corda estabelece a comunicação ou a ligação entre o gargalo e as entradas do sistema.

[6] *Tradução literal do Inglês Drum Buffer and Hope (DBR).*
[7] *Na literatura da TOC o termo é referenciado como recursos não restritivos (non-restrictive resource*)

O TPC é um método que se assemelha a uma corrente onde cada elo é "*linkado*" a um recurso na cadeia produtiva para que um produto ou serviço seja entregue. Os Recursos com Restrição de Capacidade (RRC) são os elos fracos da corrente, portanto são vulneráveis, de tal modo que as máquinas na linha de produção devem produzir conforme a capacidade dos elos fracos.

2.2.2 Aplicação do TPC ao Problema

No presente trabalho, o TPC é utilizado para representar o processo de produção de um produto de software, que embora obedeça a um processo rígido e estruturado na maioria dos casos, o desenvolvimento acontece em um ambiente altamente dinâmico. E por mais que se possa fazer uso da automatização de processos, de padrões e de modelos de desenvolvimento, o ambiente de produção de software difere do processo industrial devido às características do produto a ser gerado. Portanto, o modelo do TPC utilizado neste trabalho é uma adaptação da composição original, mas que mantém a estrutura básica estabelecida pelo tambor, pulmão e corda.

A Figura 3 apresenta o modelo do TPC desenvolvido e aplicado neste trabalho, enfatizando as atividades representadas pelas elipses e suas interações, incrementos e micro incrementos, obedecendo estruturalmente o Processo Unificado.

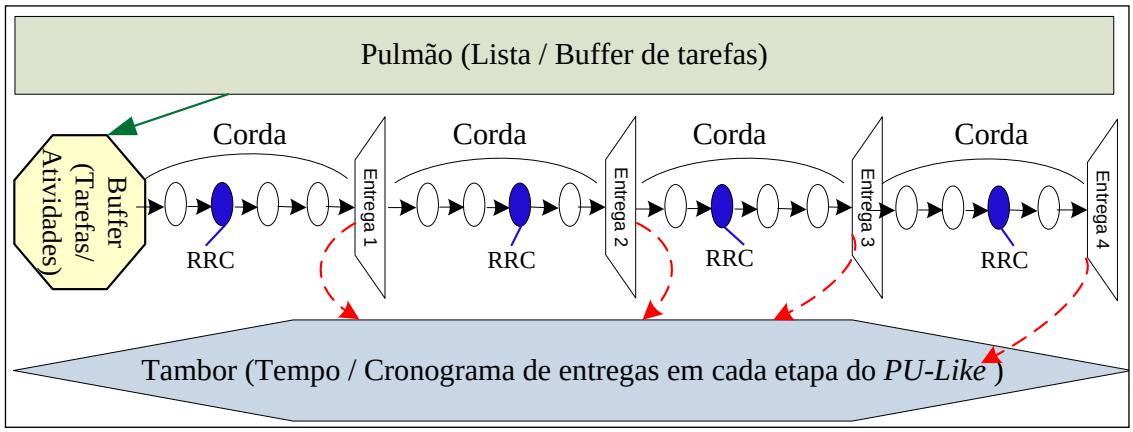


Figura 3: Modelo conceitual do TPC

As elipses em azul representam as restrições ou os recursos com restrição de capacidade. As elipses em branco são as atividades sem limitação de recursos.

Sempre que um Recurso com Restrição de Capacidade (RRC) for identificado, o passo seguinte do algoritmo da TOC (Explorar) deve ser executado. Desta forma, o modelo do TPC proposto e utilizado nesta pesquisa, corresponde à seguinte estrutura.

1. O pulmão é o conjunto de atividades a serem desenvolvidas, essencialmente estabelecidas por uma "lista de tarefas". Neste específico caso, o cronograma de entregas caracteriza-se como uma representação análoga de um processo de manufatura, pois é ele que alimenta as máquinas na linha de produção com a inserção de novas atividades e conforme o ritmo do tambor;
2. A corda é representada pelas interações e incrementos das atividades a serem desenvolvidas, que são liberadas para o desenvolvimento em função da demanda do processo produtivo e de acordo com o ritmo do tambor. As interações e micro interações representam o gatilho que aciona e dispara as próximas atividades da fila de processamento. Este processo é constituído pela estrutura incremental do Processo Unificado (PU);
3. O tambor é caracterizado pelo tempo disponível para cada etapa do PU e pelo cronograma de entregas, que neste caso particular, obedecem a uma programação com dois marcos fundamentais, dada pelo início e o fim de cada entrega. O tambor dita o ritmo da linha de produção, regulando quais atividades devem ser produzidas por cada máquina e em cada fase do PU.

2.3 Processo Unificado (PU): Conceitos Gerais

Segundo Holanda Ferreira (2014) o termo processo[8] pode ser classificado basicamente em duas definições formais: (1) ação continuada ou realização contínua e prolongada de alguma atividade, seguimento ou curso; e (2) sequência contínua de fatos ou operações que apresentam certa unidade ou que se reproduzem com certa regularidade.

Segundo Weiszflog (2009), o termo processo[9] pode ainda ser aplicado a um conjunto de três definições gerais: (1) sucessão sistemática de mudanças numa direção definida; (2) série de ações sistemáticas visando um resultado; e (3) ação ou operação contínua ou série de ações ou alterações que ocorrem de uma maneira determinada**.**

A literatura dispõe de uma ampla fonte referências sobre processos de desenvolvimento de software, em particular sobre Processo Unificado (PU). Alguns

[8] ***Dicionário Aurélio (Holanda Ferreira, 2014):*** *Aqui, apresentam-se apenas duas definições formais suprimidas por conveniência e particular emprego neste trabalho.*
[9] ***Dicionário Michaelis (Weiszflog, 2009):*** *Assim como na nota anterior, a terminologia foi suprimida por interesse particular deste trabalho.*

trabalhos são clássicos, muito citados por outros autores, como é o caso dos trabalhos de (Jacobson *et al.*, 2009a) e (Jacobson *et al.*, 2009b), criadores do PU. Por esta razão, somente serão citados nesta pesquisa estes dois trabalhos sobre o PU, pois estes contemplam toda a estrutura e fundamentação do Processo Unificado, principalmente o trabalho de (Jacobson *et al.*, 1999b).

O Processo Unificado (PU) é um modelo iterativo e incremental de desenvolvimento de software, que combina as atividades necessárias para transformar os requisitos de usuários na construção de um produto de software (Jacobson *et al.*, 1999-a), (Jacobson *et al.*, 1999-b). As etapas do processo são combinadas em ciclos iterativos de desenvolvimento que permitem acompanhar e gerenciar o processo ao longo do seu ciclo de vida. A Figura 4 ilustra o ciclo de uma iteração do Processo Unificado.

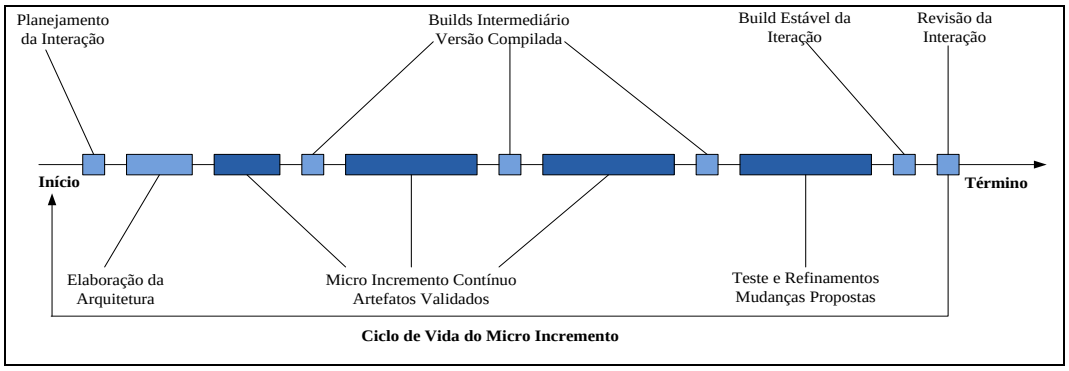


Figura 4: Ciclo de vida do micro incremento do PU. (adaptado de Ambler (2005))

Uma das diretrizes do PU enfatiza a necessidade de atribuir tarefas a grupos ou indivíduos, que necessariamente estejam envolvidos diretamente no desenvolvimento do projeto. Isto é feito através da identificação das etapas, dos marcos e dos objetivos do projeto, assim como dos artefatos que serão gerados e utilizados durante o processo. É imperativo que este procedimento de identificação seja realizado na etapa de concepção a partir da iteração inicial e de seus micros incrementos.

A Figura 5 mostra uma adaptação do esquema geral do PU, e foi desenvolvida para ilustrar os principais aspectos do processo de desenvolvimento de cada fase do Processo Unificado[10].

[10]***Processo Unificado:*** *Em razão do PU ser amplamente difundido e facilmente encontrado na literatura, não será apresentado neste trabalho conceitos e definições detalhadas sobre o PU. O leitor pode encontrar informações mais detalhadas nas fontes consultadas e referenciadas neste trabalho, ou observar as ilustrações que por si só são auto definidas e detalhadas. Os Apêndices A e B, complementam a conceituação e contextualização sobre o Processo de Software e o Processo Unificado.*

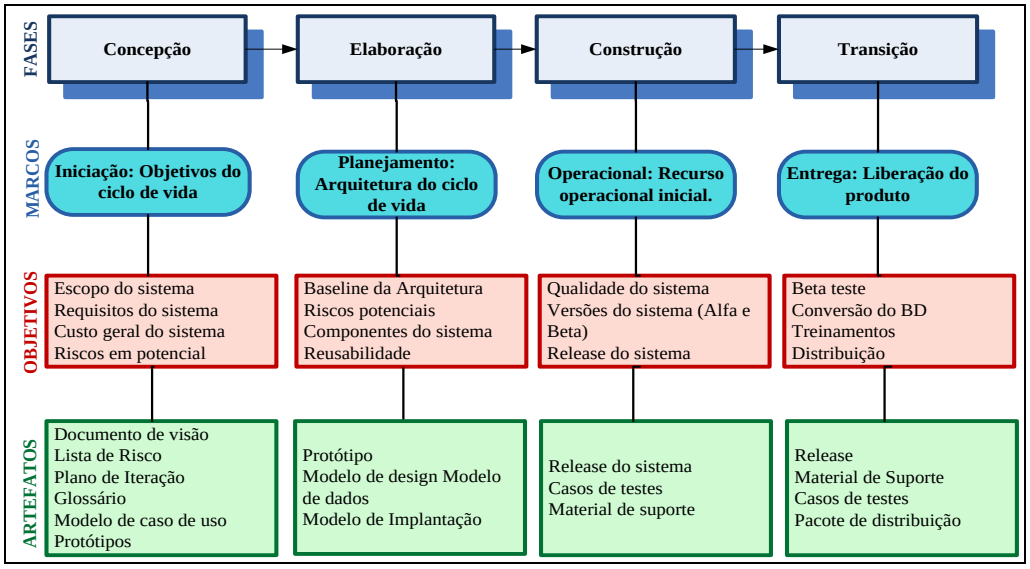


Figura 5: Instanciação do processo unificado (adaptado de (Larman 2004))

A instanciação de um processo baseado no PU permite identificar as atividades que deverão ser realizadas para alcançar os objetivos do projeto, iniciando pela concepção e estendendo-se por todas as outras fases. Isto é feito adaptando o processo ao projeto contratado e considerando as características, o modelo do ciclo de vida adotado e os recursos a serem alocados no mesmo.

A instanciação apresentada na Figura 3 é importante para guiar o desenvolvimento do processo, e consequentemente do projeto de desenvolvimento do software. O esquema mostrado na Figura 3 também servirá de fonte de referência para a navegabilidade e para o rastreamento das atividades durante a execução do processo, de acordo com as atribuições feitas a cada equipe de desenvolvimento alinhada às fases do PU.

2.3.1 Aplicação do PU ao Problema

O Processo Unificado foi adotado como um guia referencial para gerir o processo de desenvolvimento do software e alinhar as equipes de desenvolvimento aos objetivos do projeto, aos recursos disponíveis e aos prazos adotados para a construção e, consequentemente, para a entrega dos artefatos produzidos.

A adoção do PU nesta pesquisa foi impulsionada pela sua estrutura evolutiva. O PU é iterativo, incremental e adaptativo (Jacobson *et al.*, 1999-a), (Jacobson *et al.*, 1999-b) e (Larman, 2004). A adaptação do PU para receber os *"quase-experimentos"* envolveu algumas modificações comportamentais sem que a sua estrutura fosse corrompida. Estas modificações foram feitas estritamente para atender aos interesses desta pesquisa. Em

razão disso, a instanciação do PU utilizada neste trabalho foi denominada de Processo Unificado-*Like* (*PU-Like*).

2.3.2 O *PU-Like*

O *PU-Like* é um modelo de processo construído a partir do Processo Unificado com algumas modificações[11] em suas fases de execução, tais como:

Concepção: O processo foi ajustado para que esta fase fosse iniciada a partir da especificação dos requisitos do domínio, e sem a necessidade de estudos sobre viabilidade técnica, financeira e de planejamento do projeto. Assumiu-se que o projeto é viável tecnicamente e financeiramente, dispensando estudos e apresentações sobre a viabilidade. Foi definido ainda que o planejamento e o gerenciamento do projeto deverão ser conduzidos pelo pesquisador;

Elaboração: Na fase de elaboração, foram eliminadas as atividades de análise de riscos, mesmo considerando que estas atividades são de extrema importância para qualquer projeto, sobretudo para um projeto de desenvolvimento de software. A ideia é focar no processo de desenvolvimento e observar os possíveis gargalos. Assim, a elaboração basicamente limitou-se à construção das "plantas" do sistema, que incluem: a arquitetura, os padrões, os modelos e a prototipação;

Construção: A construção restringiu basicamente a atividade de elaboração e as subatividades correlatas para codificar o produto de software. A construção de versionamentos do sistema (*alpha, beta...*) ficou a cargo de cada equipe com o objetivo apenas de controlar a produção internamente (dentro de cada equipe). Exigiu-se apenas que a versão a ser entregue seja a versão construída em cada fase do *PU-Like*;

Transição: A fase de transição resumiu-se a entrega do produto final, com a apresentação do software rodando, instalado em uma máquina em uma versão *desktop*. Na transição também foi exigido a entrega de toda a documentação, incluindo a análise, os modelos, os códigos, etc. As atividades de treinamento, homologação, implantação e *releases* de sistemas não foram cobrados nesta fase.

---

[11] O PU sofreu algumas adaptações para atender aos interesses desta pesquisa, mas sem que sua estrutura fosse corrompida. Em função disso o PU foi denominado de "PU-Like".

A Figura 1 apresenta um modelo sumarizado do *PU-Like* usando a *Business Process Model Notation* (BPMN).

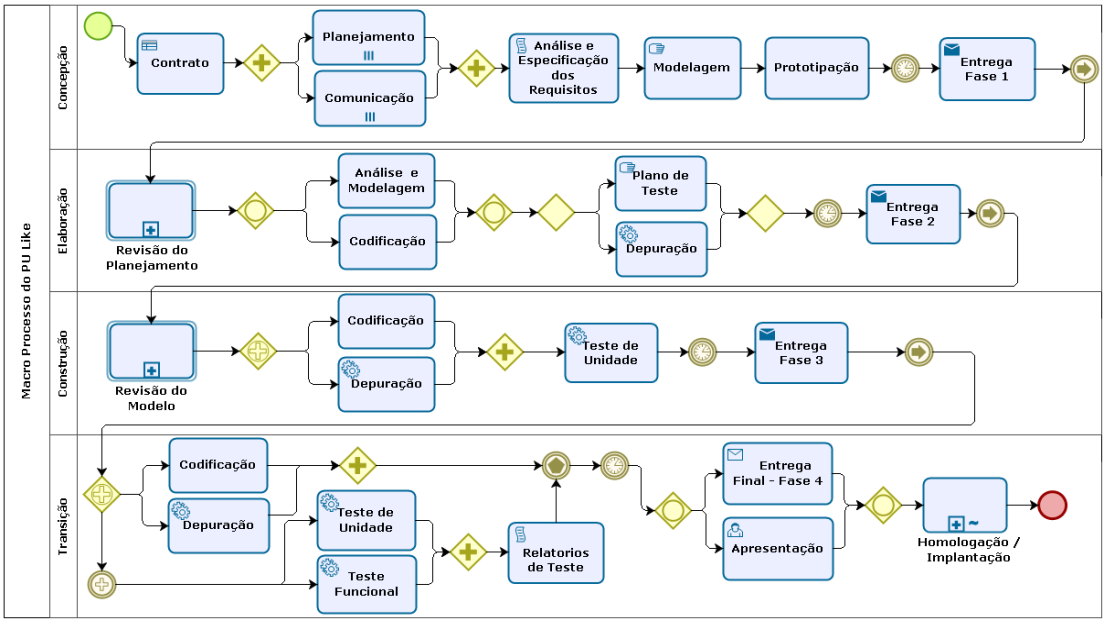


Figura 6: Modelo BPMN do *PU-Like* (Ribeiro *et al.* 2017-b)

As atividades voltadas para as disciplinas de apoio do PU também não foram metodologicamente desenvolvidas. O gerenciamento de mudanças foi cobrado das equipes de desenvolvimento de forma individualizada, observando principalmente os critérios de avalição do produto e a produção da equipe com relação ao seu desempenho acadêmico. Exemplificando, caso um artefato tenha sofrido alteração em algum ponto do desenvolvimento e as atividades correlatas (modelagem, funções do banco de dados, modelo arquitetural, documentação, etc.) não tenham sofrido as alterações/correções devidas, o artefato seria avaliado como incompleto ou incorreto. Essa avaliação foi aplicada tanto par o produto, quanto para a produção acadêmica dos estudantes.

Finalmente, consideramos que os ajustes citados acima não foram tão significativos para comprometer a estrutura básica do PU. A alteração mais incisiva foi feita na fase de concepção do PU, pois como a pesquisa se desenvolve em um ambiente acadêmico e com aprendizes de Engenharia de Software, os conhecimentos requeridos para a execução dessa fase ainda não são dominados pelos alunos.

2.4 Engenharia de Software Experimental

A Engenharia de Software Experimental (ESE) é uma linha de pesquisa relativamente nova, muito discutida atualmente, contando inclusive com eventos e

periódicos específicos sobre o tema. Segundo Basili (1996) e Basili (2013), os primeiros trabalhos envolvendo experimentação em Engenharia de Software (ES) surgiram nos anos 80 e desde então muitas contribuições têm sido publicadas.

Praticamente toda a base para fundamentar a experimentação realizada nesta pesquisa foi extraída das publicações de (Basili *et al.*, 2006) e (Wohlin *et al.*, 2012). Outra fonte bastante consistente sobre ESE e que serviu para confrontar e validar alguns pontos sobre o processo experimental empregado neste estudo foi o livro de (Juristo e Moreno, 2001). Outros textos relevantes e bastante explorados na fase experimental foram (Shull *et.al.* 2008), (Mafra e Travassos, 2006), (Travassos *et al.* 2002), e (Conradi e Wang, 2003). Estes trabalhos serviram de apoio e de referencial teórico para aquisição e aprofundamento do conhecimento sobre experimentação em Engenharia de Software.

Wohlin *et al.*, (2012) afirmam que os experimentos em Engenharia de Software (ES) devem ser conduzidos em cinco etapas básicas.

1. **Escopo:** apresenta a definição do problema e objetivos do estudo experimental;
2. **Planejamento:** determina os passos a serem seguidos no experimento e deve ser construído a partir do escopo e dos objetivos do estudo. O planejamento deve ainda definir a instrumentação e o apoio ferramental a ser utilizado no experimento, além dos métodos e das técnicas que serão utilizados para validação dos resultados;
3. **Execução:** etapa na qual é realizada a experimentação em conformidade com o planejamento do experimento. Esta etapa define também como os dados serão coletados e como serão analisados.
4. **Análise:** Nesta etapa os dados coletados são analisados, interpretados e avaliados. Os resultados colhidos após a análise serão usados para aceitar ou refutar as hipóteses levantadas durante o planejamento e a execução do experimento.
5. **Apresentação:** Uma vez finalizado o experimento, os resultados colhidos deverão ser "empacotados" e apresentados. A apresentação pode ser feita através de um artigo, de um relatório técnico, de um pôster, ou na forma de uma apresentação em um evento (simpósio ou conferência).

Por conveniência, no presente trabalho algumas etapas não obedeceram necessariamente a ordem apresentada por (Wohlin *et al.*, 2012), uma vez que os dados foram coletados e analisados ao longo de cada rodada experimental. No entanto, isto não significa que todas as etapas não foram executadas. O capítulo 4, apresentará de modo

mais detalhado a construção e a condução do experimento realizado, explicitando a execução de cada etapa.

2.4.1 Dificuldades Encontradas na ESE

Todo conhecimento científico tem sua base na experimentação, pois é através desse processo que as hipóteses são testadas e as teorias são confirmadas ou refutadas. Na ciência da computação, sobretudo na Engenharia de Software (ES), isso não deveria ser diferente, mas na prática não é isso que geralmente acontece (Travassos, 2011).

Segundo Juristo e Moreno (2001), esta discussão é um campo vasto e vai além das principais desculpas que tradicionalmente são apresentadas e que levam as pessoas e organizações a não realizar experimentos em Engenharia de Software. Acredita-se que o principal fator continua sendo o cultural, constituindo-se em um paradigma que precisa ser quebrado.

De acordo com Juristo e Moreno (2007), existem algumas razões que dificultam a experimentação no desenvolvimento de software. No entanto, alguns destes argumentos são meramente falácias ou justificativas descontextualizadas para não se realizar experimentação em ESE.

Juristo e Moreno (2007) destacam um conjunto de argumentos e factoides que levam os desenvolvedores a não experimentação em Engenharia de Software. Isto é importante porque endossa a importância de se realizar experimentação em ES, caminho este seguido neste trabalho. A Tabela 2 apresenta resumidamente os principais argumentos extraídos de (Tichy, 1998) *apud* (Juristo e Moreno, 2001) bem como a refutação destes argumentos.

Tabela 2: Resumo das falácias e refutações sobre ESE[12] (Juristo e Moreno 2007)

Falácias	Refutação
Método científico tradicional não é aplicável.	Para entender o processo de informação, cientistas da computação devem observar fenômenos e formular explicações. Este é o método científico.
O atual nível de experimentação é bom o suficiente.	Em relação a outras ciências, os dados mostram que os cientistas da computação validam um pequeno percentual de suas alegações.
Experimentos custam caro.	Experiências significativas podem caber em pequenos orçamentos; experimentos caros podem valer mais do que o seu custo.
Demonstrações são suficientes.	Demonstrações podem fornecer incentivos para estudar melhor o problema. Na maioria das vezes as demonstrações apenas ilustram

[12] ***Referência**: a tabela 2 foi extraída de (Juristo e Moreno, 2001) com tradução livre do autor deste trabalho.*

	um potencial problema.
Há muito ruído no caminho.	Felizmente, as técnicas podem ser utilizadas para simplificar variáveis e responder a perguntas.
Experimentações retardam o progresso.	Aumentando a proporção de documentos com uma validação significativa tem-se uma boa chance acelerar o progresso.
A tecnologia muda muito rápido.	Se uma questão torna-se irrelevante rapidamente, esta também é restritivamente definida e não vale a pena um esforço maior com ela.
Você nunca irá publicá-lo	Pequenos passos ainda são publicáveis, porque melhoram a nossa compreensão e ajudam a levantar novas questões.

A obra de Juristo e Moreno (2007) elenca ainda outros fatores[13] que dificultam a realização de experimentos na Engenharia de Software, tais como:

1. Desenvolvedores de software não são treinados para realizar experimentação;
2. Desenvolvedores de software não sabem como analisar os dados de um experimento;
3. Existe um grande número de variáveis no domínio de ES que precisam ser observadas e tratadas;
4. O fator humano também é uma restrição importante que deve ser observada na experimentação em ES;
5. Os resultados dos experimentos não são independentes de praticantes ou desenvolvedores;
6. A quantidade de dinheiro no mercado de software direcionada para a experimentação ainda é muito pequena.

Embora exista a cultura da não experimentação no desenvolvimento de software, as exigências mercadológicas e o aumento considerável da complexidade dos sistemas computacionais estão demandando esta necessidade e conduzindo a Engenharia de Software a uma nova perspectiva fundamentada e embasada na experimentação.

2.4.2 Experimentação em Ambiente de Aprendizagem de ES

O ambiente de aprendizado em Engenharia de Software foi escolhido para a realização deste experimento devido à possibilidade de executar a mesma atividade de pesquisa em diferentes equipes simultaneamente. Acredita-se que isso seja uma limitação natural em um ambiente real de desenvolvimento, uma vez que em uma "*software house"*,

[13] ***Tabela 9:*** *Recomenda-se a leitura do livro Basics of Software Engineering Experimentation (Juristo e Moreno, 2007) para maior profundidade na extensa argumentação dos autores sobre motivos que dificultam a realização de experimentos em Engenharia de Software.*

dificilmente várias equipes podem trabalhar em um mesmo projeto desde a concepção até a transição com o objetivo de construir um mesmo produto.

O ambiente de ensino de Engenharia de Software não exige muitos recursos para executar o experimento, diminuindo assim, os riscos associados ao projeto e consequentemente a esta pesquisa.

Os trabalhos de (Carver *et al.*, 2003-a), (Carver *et al.*, 2003-b), apresentam considerações relevantes envolvendo benefícios para estudantes, pesquisadores e instrutores em pesquisas realizadas em ambiente de ensino.

A principal desvantagem está na qualificação da equipe, fator que diferencia uma equipe de aprendizes de uma equipe de um ambiente industrial. Em geral, as fábricas de software contam com profissionais graduados, capacitados e treinados em determinados modelos, padrões, processos e ambientes de desenvolvimento. Nesta pesquisa, esta foi a primeira restrição encontrada, já que os desenvolvedores estão em processo de aprendizagem.

Basicamente, a mesma perspectiva apresentada em (Carver *et al.* 2003-a), é encontrada em (Berander, 2004), (Höst *et al.* 2000), (Svahnberg *et.at* 2008), (Maras *et al.* 2015) e (Runeson, 2003). Nestes trabalhos, os autores também utilizam o ambiente acadêmico para simular experimentos, diferenciando-se apenas na definição do objeto de estudo. Estes trabalhos além de contribuírem para a construção do conhecimento, influenciaram e motivaram a condução do experimento usando estudantes de computação em seu próprio ambiente de aprendizagem.

A experimentação em ambiente de ensino não pode ignorar a questão do aprendizado e da formação acadêmica dos alunos. Aliás, este é um princípio ético fundamental neste tipo de experimento (Carver 2002-a) e (Carver 2002-b). Esta questão foi uma preocupação constante e necessária neste experimento, onde se procurou não só levar o conhecimento teórico aos aprendizes de Engenharia de Software, como também o conhecimento prático relacionado ao conteúdo da disciplina. Afinal, a aprendizagem neste tipo de experimento é fundamental para o resultado dos ensaios.

2.5 O Problema *Job Shop Scheduling* (JSSP)

Os problemas de escalonamento (*scheduling*) são aqueles que envolvem alocação de recursos no tempo, com a finalidade de executar uma série de tarefas, atendendo a uma função objetivo, que pode ser: minimizar o tempo de execução, minimizar o custo, maximizar a qualidade, etc. A solução ótima para esses problemas exige um grande esforço computacional, sendo que, o problema de escalonamento *job shop*, é um caso particular da classe *NP - Hard.* ((Blazewicz *et al.* 1996) em (Ribeiro, 2006) e (Ribeiro *et al.* 2012)).

Um *Job Shop Problem* (JSP) clássico é normalmente referenciado como um *n/m JSP*, onde *n* é o número de *jobs* e *m* é o número de máquinas. Um conjunto de *n jobs* $J = \{J_1, J_2,..., J_n\}$ deve ser processado em um conjunto de *m* máquinas $M = \{ M_1, M_2,..., M_m \}$ disponíveis. Cada *job* possui uma ordem de execução específica entre as máquinas, ou seja, um *job* é composto de uma lista ordenada de operações, cada uma das quais definida pela máquina requerida e pelo tempo de processamento da mesma ((Klein, 2000) em (Ribeiro, 2006) e (Ribeiro *et al.* 2012)). No modelo clássico de *job shop* tem-se ainda que um conjunto ***J*** de *jobs* consiste em um conjunto *j* de operações e *k* índices, onde *j* é o número total de operações para cada *job* e *k* é o índice da máquina para processar a "*k-ésima*" operação *j*.

De acordo com Pinedo (1995), as seguintes condições devem ser consideradas para o problema.

- Todas as máquinas são diferentes e suas velocidades de processamento são constantes.
- Operações não podem ser interrompidas;
- Cada *job* é escalonado somente uma única vez;
- Cada máquina pode processar apenas uma operação em cada instante de tempo;
- Cada *job* só pode estar sendo processado em uma única máquina;
- Um *job* pode ser escalonado em qualquer máquina;
- Cada *job* é produzido por uma sequência conhecida de operações;
- Não existe restrição de precedência entre operações de diferentes *jobs.*

O objetivo mais comum empregado em um problema *Job Shop Scheduling* é encontrar uma sequência de *jobs* válida, ou seja, que não viole nenhuma das restrições

acima, e que complete todos os *jobs* com um menor tempo possível. Este objetivo é conhecido como *makespan*, e tem por finalidade minimizar o tempo de finalização das operações (Ribeiro, 2006).

A Tabela 3 apresenta um exemplo de um *job* clássico com 3 *jobs* sendo processados em três máquinas.

Tabela 3: Um *job shop* clássico 3X3 (Ribeiro, 2006)

Representação de um *Job Shop Scheduling* (3 X 3)			
Job	**Máquina Utilizada / Tempo**		
1	1(4)	2(6)	3(5)
2	1(3)	3(6)	2(7)
3	2(4)	1(5)	3(2)

Cada *job* é escalonado em um dado instante de tempo e os valores apresentados entre parênteses indicam o total de unidades de tempo consumido no processamento.

A solução pode ser representada por um diagrama de Gantt, como ilustrado na Figura 7.

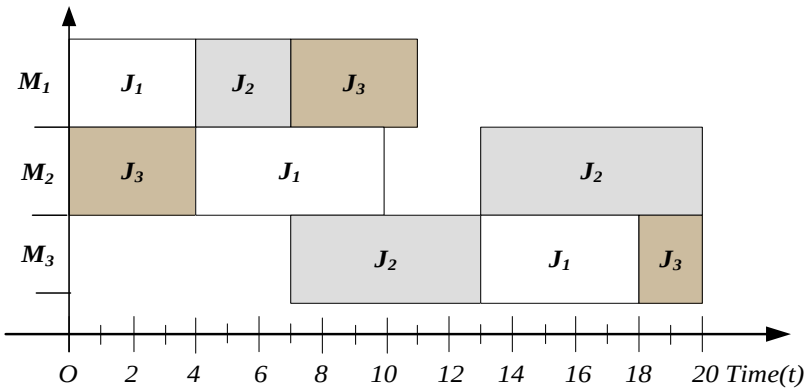


Figura 7: Representação de *job shop scheduling* (Ribeiro, 2006)

Uma vez que as sequências de máquinas de cada *job* são fixadas, o problema a ser resolvido consiste em determinar as sequências dos *jobs* em cada máquina, de forma que o tempo de execução transcorrido, desde o início do primeiro *job* até o término do último seja minimizado. O objetivo é encontrar a solução com menor *makespan* (Ribeiro, 2006).

2.5.1 Métodos de Escalonamento do Tipo JSP

O problema de escalonamento do tipo JSP é um dos três tipos de problemas clássicos de agendamento de tarefas. Os outros dois tipos são: (1) o *Flow Shop Scheduling Problem* (FSSP); e (2) o *Open Shop Scheduling Problem* (OSSP) (Thörnblad, 2013). Os

métodos de Escalonamento do tipo JSP podem ser categorizados como local e global (Casavant e Kuhl, 1988). O método global possui duas classificações: (1) estático, que compreende o modelo clássico do JSP; e (2) dinâmico, que se baseia na distribuição de processos/tarefas entre as máquinas. A classificação dinâmica pode ser ainda determinística ou não determinística.

2.5.2 *Job Shop* Dinâmico (JSD)

O escalonamento JSD é definido em Araújo Jr. (2006) *apud* Rabelo e Klen (2000), como uma ação para adaptar o escalonamento corrente, simultaneamente com sua execução em função da ocorrência de eventos não planejados. Isto deve ser observado, tanto para os escalonamentos provenientes do chão de fábrica, quanto para aqueles do nível de planejamento, de forma que o escalonamento se mantenha realístico, viável e realizável de acordo com as restrições temporais, de capacidade e tecnológicas existentes, e por fim deve ser coerente e sem desvios em relação aos objetivos e as metas traçadas.

Os ambientes reais de produção, em sua maioria sofrem constante mudança e adaptações provocadas por eventos em tempo real. Em geral, as mudanças são necessárias para atender as leis do mercado e/ou para ajustar o ambiente em função de mudanças estruturais e organizacionais, como por exemplo: criação de novos postos de trabalhos, nova aquisição de linhas produtivas, substituição de tecnologia, quebra de máquinas, entre outros. O problema de escalonamento na presença de eventos em tempo real é chamado de escalonamento dinâmico (Ouelhadj e Petrovic, 2009).

Segundo Ouelhadj e Petrovic (2009), o escalonamento dinâmico de tarefas obedece a três políticas: periódica, eventual e híbrida. As políticas periódicas e híbridas também são conhecidas como políticas de horizonte de tempo de rolamento, de modo que o problema de escalonamento ou DSP *(Dynamic Scheduling Problem)* é decomposto em uma série de problemas estáticos (Thörnblad, 2013).

Na política orientada a eventos, um escalonamento é fracionado e reescalonado em respostas aos eventos durante a execução do processo. Esta caraterística faz com que o *Job Shop* Dinâmico (JSD) seja um candidato em potencial para um ambiente de desenvolvimento de software, que tem que lidar com eventos perturbando o processo em tempo real e tarefas sendo escalonadas e reescalonadas até que seja satisfatoriamente completada.

2.5.3 Aplicação do JSD ao Problema

O contexto de desenvolvimento de software lança o processo em um ambiente altamente dinâmico. Assim, as condições do JSP clássico apresentadas por Pinedo (1995) e descritas na Seção 2.5 não são completamente aplicáveis, pois ao contrário do que é estabelecido, temos que:

- As máquinas são diferentes e suas velocidades de processamento não são constantes.
- Operações podem ser interrompidas sempre que necessário e reiniciada;
- Um *job* pode ser reescalonado *n* vezes;
- Cada máquina pode processar várias operações em instantes de tempo diferentes;
- Um mesmo *job* pode ser processado em diferentes máquinas e em diferentes linhas de produção;
- Um *job* necessariamente é escalonado em todas as linhas de produção (caso típico do modelo e ambiente desenvolvido nesta pesquisa);
- Os *jobs* são produzidos por uma sequência de operações convenientes ou de acordo com os objetivos das máquinas de cada linha de produção;
- Em alguns casos, existem restrições de precedência entre operações de diferentes *jobs.*
- A função objetivo é dada pela maximização da produção de uma linha de produção do *shop*, com $\boldsymbol{L}=\{M_1A_{1...An,}\ M_2A_{1...An,\ ...}\ M_nA_{1...}A_n\}$ = 100%, onde ***M e A*** são as máquinas *(M)* produzindo (A_n) artefatos.

Após estudar os tipos de JSP existentes na literatura, delinear o problema central desta pesquisa e elencar os tópicos acima, foi possível identificar as características do modelo a ser adotado e concluir que a melhor alternativa para representar o problema é o escalonamento do tipo JSD determinístico.

A escolha do JSD é principalmente devido às circunstâncias do ambiente produtivo e do "*shop*" preparado para rodar o problema, onde: (1) as tarefas executadas nas máquinas são previamente definidas e determinadas e (2) a ordem de execução é flexível, pois obedece ao plano de desenvolvimento elaborado para Linha de Produção.

3 REVISÃO DA LITERATURA

"Antes do interesse pela escrita, deve haver o interesse pela leitura! (...) Sem ler ninguém escreve!"

José Saramago

Um processo de investigação deve ser iniciado através da identificação de uma oportunidade de pesquisa. A relevância dessa oportunidade deve ser amparada e suportada pela revisão da literatura, pois é ela que dará sustentação ao pesquisador durante a jornada investigativa, através da aquisição e da construção do conhecimento necessário para que a investigação possa ser desenvolvida.

Este capítulo apresenta um mapeamento sistemático da literatura sobre a aplicação da TOC no PDS. Os resultados deste estudo secundário apresentaram lacunas acerca deste tema e evidenciaram algumas oportunidades de pesquisas que impulsionaram a realização deste trabalho.

3.1 Revisão da Literatura sobre a TOC em PDS

A análise do grande de número de trabalhos científicos publicados a cada ano é inviável, mesmo com todos os recursos tecnológicos a disposição. As áreas de desenvolvimento destes trabalhos estão cada vez mais multidisciplinares e abrangentes, tornando difícil e dispendiosa a tarefa de encontrar material de pesquisa correlato com uma área especifica de atuação ou com um tema específico.

Além disso, o mau uso do ferramental tecnológico de apoio pode conduzir a uma pesquisa com erros de processamento da informação coletada (Biolchini, *et al,* 2007) e (Biolchini *et al,* 2005), o que pode levar o pesquisador a: (1) eliminação, dada pela supressão de alguma informação considerada não fundamental para dar lugar à outra considerada fundamental; (2) distorção, que é a substituição ou adequação de informações para ajustá-las a uma determinada necessidade, sem a análise devida desta informação ou das fontes que a cercam, e (3) generalização, que se auto traduz pelo agrupamento de experiências consideradas semelhantes por possuírem algum elemento em comum, mas que nem sempre estão realmente correlacionadas.

Por outro lado, a revisão da literatura é um caminho que pode vir a facilitar e direcionar a busca pela informação correta e adequada para pautar uma pesquisa.

Segundo Kitchenham (2004) e Kitchenham e Charters (2007), um estudo secundário em Engenharia de Software é denominado de Revisão Sistemática da Literatura (RSL) e pode ainda ser tipificado, em: (1) Mapeamento Sistemático – que se refere a uma revisão mais ampla de estudos primários para identificar evidências, lacunas e/ou oportunidades de pesquisas e (2) Revisão Terciária – que é um estudo secundário que tem como objetivo identificar quais Revisões Sistemáticas (RS) foram realizadas sobre um determinado assunto ou área de pesquisa.

Neste trabalho foi realizado um mapeamento sistemático para identificar potenciais oportunidades de pesquisas em processo de desenvolvimento de software com a aplicação da TOC.

3.2 Estrutura Adotada para Revisão da Literatura (RL)

A RL foi construída conforme sugere a literatura consultada (Kitchenham, 2004), (Kitchenham, *et al*., 2007), (Kalus *et al.*, 2005), (Biolchini *et al.*, 2007), (Brereton, *et al.* 2007) e (Sulayamam e Mendes 2009), e foi estruturalmente organizada em: (1) planejamento da revisão; (2) condução da revisão e (3) relatório da revisão.

Planejamento da revisão: O planejamento pode ser subdividido em três etapas (Kitchenham, 2004) e (Islam *et al*. 2012): (1) Protocolo de Revisão – que define o propósito e os objetivos da pesquisa, as fontes consultadas, a estratégias e os critérios adotados para seleção dos documentos, a forma de extração dos dados e prospecção dos resultados; (2) Especificidade da pesquisa – define a área temática de acordo com o objeto fim da pesquisa e (3) Execução da pesquisa – que se traduz pela busca nos repositórios escolhidos.

Condução da revisão: segundo Kitchenham (2004), Kitchenham *et al.*, (2007) e Biolchini *et al.*, (2005) a condução da revisão é a etapa de avaliação do protocolo desenvolvido. Isto se dá através de testes de execução de busca de acordo com os critérios estabelecidos, onde algumas fontes ou repositórios são escolhidos e examinados. Nesta etapa também são selecionados, lidos e analisados os estudos primários, como sugere Biolchini *et al*., (2005) e Biolchini *et al*. (2007). O ciclo se completa com a exclusão ou a aceitação de um documento.

Relatório da Revisão: O relatório da revisão é a forma que deve ser utilizada para apresentar os resultados da revisão. De acordo com Kitchenham (2004) é muito importante comunicar os resultados de um estudo secundário. Conforme Kitchenham e Charters (2007) usualmente as revisões sistemáticas são relatadas em pelo menos dois formatos: (1) em um relatório técnico ou em uma seção de um documento (tese ou dissertação); ou (2) em um jornal ou artigo de conferência.

As três fases mapeadas na Figura 8 representam a estrutura de execução da revisão utilizada neste trabalho.

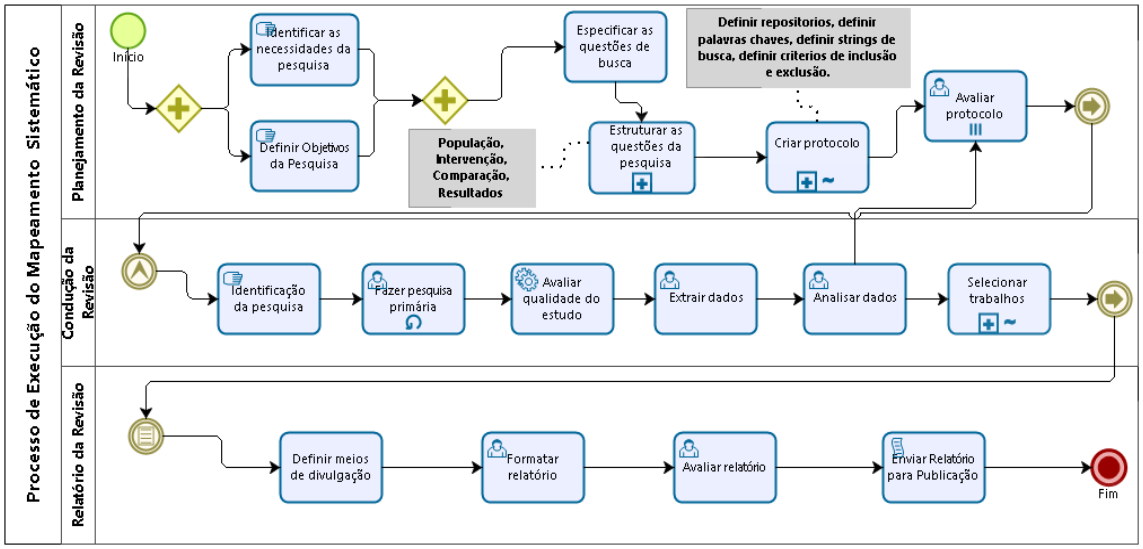


Figura 8: Modelo BPMN para o processo do mapeamento sistemático

3.2.1 Protocolo da Revisão

O protocolo adotado nesta pesquisa segue estritamente o modelo apresentado por (Ribeiro *et al.* 2017-a) e (Ribeiro *et al.* 2018), conforme apresentado na Tabela 4. Portanto alguns itens podem ter sido suprimidos ou aparecer em uma ordem exclusivamente adequada a este trabalho. Ainda assim, o modelo construído representa com certa fidelidade as orientações de (Kitchenham, 2004), (Kitchenham e Charters, 2007), (Kitchenham *et al*, 2009) e (Biolchini *et al*, 2005) para elaboração de protocolos para RSL.

Tabela 4: Protocolo adotado para a revisão

Id	Tópicos do protocolo	Definição
1	Contexto	Elaborar um mapeamento sistemático sobre a TOC em PDS.
2	Objetivos	Analisar estudos primários completos e disponíveis eletronicamente nos repositórios escolhidos com o desígnio de encontrar lacunas e oportunidades de pesquisa sobre o emprego da TOC no PDS.
3	Escopo da Pesquisa	Realizar buscas de artigos completos em bibliotecas digitais através de suas máquinas de busca.
4	Métodos de busca	Busca eletrônica nos repositórios escolhidos, aqui denominados de

		"Alvos de Busca".
5	Questões da pesquisa	Definição das questões primárias (Qp) e secundárias (Qs).
6	Procedimentos de seleção	Os estudos primários devem ser completos e estar disponíveis eletronicamente nos repositórios alvos.
7	Filtros	Os filtros aplicados irão estabelecer os critérios de inclusão e Exclusão.
8	Procedimento de Extração de Dados	Os procedimentos de extração de dados foram realizados em três passos: (1) o preenchimento de um relatório de extração de dados (em forma de tabela) contendo: título, autores, ano de publicação, quantidade de páginas, palavras-chave, repositórios e uma nota para "ranqueamento"; (2) aplicação de filtros de acordo com os critérios de inclusão e exclusão; e (3) "ranqueamento" e análise discursiva sobre os estudos primários selecionados.
9	Procedimento de Análise	Análise correlacional do trabalho com a área temática da pesquisa e subáreas, de acordo com as palavras chave e as *strings* de busca.
10	Empacotamento	Fechamento / Encerramento do trabalho que terá como resultado um relatório técnico contendo todo o processo da revisão o qual deverá ser inserido em uma sessão da tese de doutorado.

As seções a seguir irão apresentar os tópicos do protocolo a partir do *Id 5*. Isto porque os tópicos, 1 – Contexto; 2 – Objetivos; 3 – Escopos da pesquisa e 4 – Métodos de busca, já estão definidos na Tabela 4.

3.3 Alvos de Busca

Definem-se como alvos de busca, ou simplesmente alvos, os repositórios de documentos que serão explorados nesta pesquisa. Os alvos foram selecionados como fontes de pesquisa de dados indexados, categorizados por suas máquinas de busca eletrônica e suas bibliotecas digitais separadas por área de pesquisa, como apresenta a Tabela 5.

Tabela 5: Alvos escolhidos para realizar das buscas

Alvo	Seção	Subseções
ACM Digital Library (ACM)	Computing Literature	Science Computing and Software Engineering
Elsevier-Science Direct (SD)	Physical Sciences and Engineering	Computer Science
Google Scholar[14] (GS)	Computer Science	Software Engineering and Process Optimization
IEEE Xplore (IX)	IEEE Transaction Computers	IEEE Transaction on Software Engineering
Scopus (Scp)	Computer Science	Software Engineering
Springer Link (SL)	Computer Science	Information Systems and Applications

[14] ***GS**: O alvo Google Scholar não possui seções e subseções específicas. Para resolver o problema, foram associadas duas palavras chave para refinamento e definição de área e subárea de busca.*

3.4 Critérios de Busca Adotados

De acordo com Kitchenham, (2004), Biolchini *et al.*, (2007) e Brereton, *et al.*, (2007), o critério mais comum é a adoção de palavras-chave, que chave guiam a pesquisa desde a fase inicial até a fase final, podendo ainda ser utilizada para critério de seleção de documentos através da composição de *strings* de buscas. As *strings* são adotadas para refinar os resultados obtidos durante o processo de busca. As palavras-chave empregadas neste trabalho são apresentadas na Tabela 6.

Tabela 6: Palavras chave usadas nas buscas

Palavras Chave	Expectativa
Process Optimization (PO)	Publicações que envolvem otimização de processos de forma geral com o uso de TOC.
Software Process (SP)	Publicações que associam o emprego da TOC ao processo de software
Software Process Improvement (SPI)	Publicações sobre melhoria de processos de software com foco no uso da TOC
Software Process Optimization (SPO)	Publicações que envolvem otimização de processo de software, preferencialmente usando TOC.
Theory of Constraints (TOC)	Palavra chave usada como referência e composição das buscas associadas às palavras chave acima.

A Tabela 7 apresenta as *strings*[15] de buscas construídas a partir das palavras-chave e dos operadores, *∧ (E) e ∨ (OU)*. Com o intuito de restringir o número de resultados indesejados encontrados durante a busca, foram incluídos dois fechamentos, aspas *{“,”}* e parênteses *{(,)}*, conforme apresenta a Tabela 4.

Tabela 7: Strings de busca utilizadas

String de Busca com o Operador ∧ (AND) e os fechamentos parênteses e aspas.
("Theory of Constraints ∧ Software Process ∧ Software Process Optimization ∧ Software Process Improvement ∧ Process Optimization”)
String de Busca Usando Operador ∨ (OR) e os fechamentos parênteses e aspas.
("Theory of Constraints ∧ Software Process (∨ Software Process Optimization ∨ Software Process Improvement ∨ Process Optimization”))

3.5 Questões de Pesquisa

As questões de pesquisa estabelecem o direcionamento e a identificação das buscas nos repositórios a partir dos objetivos estabelecidos para o estudo secundário. Os objetivos são previamente definidos para alinhar as buscas ao tema proposto.

[15] ***Strings de Busca***: *a construção das strings da Tabela 7 representam uma generalização das mesmas. As strings foram ajustadas de acordo motor de busca dos repositores escolhidos e suas especificidades sem comprometer a sua estrutura geral.*

Foram definidos dois tipos de questões: (1) Questão principal (***Qp***); e (2) Questão secundária (***Qs***). Os tópicos a seguir apresentam as questões elaboradas e utilizadas para delinear este estudo secundário.

- **Qp1** - É possível aplicar a TOC no Processo de desenvolvimento de software?

A questão *Qp1* define e direciona os objetivos da pesquisa, apoiada pelas questões *Qs1, Qs2, Qs3, Qs4 e Qs5*.

- **Qs1** - Que tipo de aplicação podemos ter com a TOC no processo de desenvolvimento de software?
- **Qs2** - É possível aplicar a TOC para identificar gargalos no PDS?
- **Qs3** - Em que o uso da TOC pode ajudar na otimização de processos de Softwares?
- **Qs4** - Em quais ferramentas, técnicas, habilidades, processos e metodologias com foco na otimização e melhoria do processo de software a TOC pode ser aplicada?
- **Qs5** - O emprego da TOC associada a técnicas de otimização em outras áreas com eficácia comprovada pode ser associado ao processo de software?

A questão 4 (Q_s4) procura identificar ferramentas, técnicas e métodos de otimização que podem ser usadas com a TOC, e a questão 5 (Q_s5), procura uma relação direta do uso da TOC em outras áreas de conhecimento que podem ser associadas ou aplicadas na otimização de processo de software.

3.6 Critérios de Inclusão e Exclusão

Todos os trabalhos foram analisados de acordo com o tema central, sua relevância e profundidade. Para isso, foram definidos alguns critérios para considerar o documento relevante para este estudo primário. Os critérios de inclusão e exclusão apresentados a seguir, foram definidos com base nas questões de busca criadas na Seção 3.5.

3.6.1 Critérios de Inclusão

Os trabalhos a serem incluídos na pesquisa devem envolver necessariamente o PDS e serão ordenados de acordo com sua relevância e obediência aos seguintes critérios:

1. Estar relacionado com a TOC (*Theory of Constraints*);

2. Estar relacionado com processos de software;

3. Estar relacionado com otimização de processos de software;

4. Estar completo (disponível integralmente) e publicado em algum dos alvos;

5. Estar publicado no idioma inglês;

6. Conter no título e no resumo alguma relação com o tema central desta pesquisa.

3.6.2 Critérios de Exclusão

A eliminação dos trabalhos não relevantes ocorreu com base em um conjunto de critérios de exclusão, aplicados em três etapas:

1. Exclusão súbita: o trabalho está fora do período adotado (2008 - 2015) ou o trabalho é repetido;

2. Exclusão após análise inicial: O trabalho não possui nenhuma relação com o tema central desta pesquisa ou não está alinhado com o contexto desta pesquisa;

3. Exclusão após análise detalhada: o trabalho não tem profundidade suficiente em relação ao tema proposto, ou com temas correlatos.

3.7 Seleção dos Trabalhos

Os resultados das buscas foram analisados em duas fases: primeiramente, observando o título do trabalho, o resumo, a conclusão e as palavras-chave. Em seguida, os trabalhos foram armazenados para análise detalhada, conforme estabelecido na seção anterior. Durante esta fase os trabalhos selecionados foram ordenados com um *score* (A, B ou C), sendo: *(A> 80%), (B > 65% ∧ < A), e (C > 50% ∧ < B).*

A avaliação foi realizada com base nos critérios de inclusão e exclusão, e com a atribuição de pesos aplicados da seguinte forma: (1) peso 3, para os trabalhos com forte correlação com a *Qp* e pelo menos uma questão secundária; (2) peso 2, para os trabalhos que atendem parcialmente a *Qp e* pelo menos uma questão secun*dária*; e (3) peso 1, aos trabalhos que estejam relacionados com pelo menos uma das questões secundárias (*Qs*) e baixa correlação com a questão primária. Para serem selecionados, todos os resultados deverão passar pelos critérios de inclusão e devidamente "*ranqueados".*

3.8 Resultados da Revisão

De acordo com Biolchini *et al.* (2005) os resultados de uma revisão sistemática podem ser apresentados de várias formas, sendo uma delas no formato de tabela, modelo que será utilizado neste estudo. A Tabela 8 apresenta os resultados gerais da pesquisa, tendo já aplicado os critérios iniciais de análise.

Tabela 8: Resultados iniciais obtidos (1ª fase da análise)

Resultados Geral das Buscas												
Strings de Busca	***Resultado Inicial das Buscas***						***Excluído por Filtragem***					
	GS	***IX***	***SL***	***ACM***	***ESD***	***Scp***	***GS***	***IX***	***SL***	***ACM***	***ESD***	***Scp***
(TOC ∧ SP ∧ SPI ∧ SPO ∧ PO)	58	13	26	33	27	16	31	7	16	21	17	12
(TOC ∧ SP ∨ (SPI ∨ SPO ∨ PO))	27	24	15	32	15	11	9	13	12	11	13	9
Total	**75**	**37**	**41**	**65**	**42**	**9**	**35**	**20**	**28**	**34**	**31**	**21**

A diferença numérica entre os campos "Resultados Iniciais das Buscas" e "Excluídos por Filtragem" da Tabela 8, representa os dados do campo "Pré-selecionados" da Tabela 9, incluídos após a análise inicial. A Tabela 9 também mostra o número de trabalhos excluídos após a análise inicial.

Tabela 9: Análise inicial (2ª fase da análise)

Análise Inicial												
Strings de Busca	***Pré-Selecionados***						***Excluídos Após Análise Inicial***					
	GS	***IX***	***SL***	***ACM***	***ESD***	***Scp***	***GS***	***IX***	***SL***	***ACM***	***ESD***	***Scp***
(TOC ∧ SP ∧ SPI ∧ SPO ∧ PO)	27	6	10	12	10	4	14	4	7	7	6	2
(TOC ∧ SP (∨ SPI ∨ SPO ∨ PO))	18	11	3	21	2	3	6	2	1	13	2	1
Total	**45**	**17**	**13**	**33**	**12**	**7**	**20**	**6**	**8**	**20**	**8**	**3**

Seguindo o mesmo critério de visualização de dados, os valores apresentados no campo "Analisado" da Tabela 10, traduzem a diferença numérica entre os campos "Pré-selecionados" e "Excluídos por Análise Inicial" da Tabela 9. A Tabela 10 mostra o total de trabalhos “Excluídos” após a análise final.

Tabela 10: Análise final (3ª fase da análise)

Análise Final												
Strings de Busca	***Analisados***						***Excluídos Após Análise Final***					
	GS	***IX***	***SL***	***ACM***	***ESD***	***Scp***	***GS***	***IX***	***SL***	***ACM***	***ESD***	***Scp***
(TOC ∧ SP ∧ SPI ∧ SPO ∧ PO)	13	2	3	5	4	2	8	1	1	4	2	2
(TOC ∧ SP (∨ SPI ∨ SPO ∨ PO))	12	9	2	8	0	1	6	8	3	5	0	0
Total	**25**	**11**	**4**	**13**	**4**	**3**	**14**	**2**	**4**	**4**	**2**	**1**

3.8.1 Trabalhos Selecionados

O resultado sumarizado do estudo primário, onde foram selecionados 24 artigos, conforme as questões de buscas e os critérios de avaliação adotados neste trabalho são mostrados na Tabela 11.

Tabela 11: Total de trabalhos selecionados por alvo

Resultado Final: Trabalhos Selecionados								
Strings de Busca	Google Scholar	*IEEE Xplorer*	*Springer Link*	*ACM DL*	*Elsevier Science Direct*	*Scopus*	***Total por String***	***% por String***
(TOC ∧ SP ∧ SPI ∧ SPO ∧ PO)	5	1	2	1	2	0	**11**	**45,83%**
(TOC ∧ SP (∨ SPI ∨ SPO ∨ PO))	6	1	2	3	0	1	**13**	**54,17%**
Total por Alvo	**11**	**2**	**4**	**4**	**2**	**1**	**24**	**100%**
% Selecionado por Alvo	**45,83%**	**8,33%**	**16,67%**	**16,67%**	**8,33%**	**4,17%**	**100%**	

3.8.2 Relação de Trabalhos Selecionados

A Tabela 12 apresenta a lista de trabalhos selecionados, organizados por *score* e alvos de busca. Além disso, a tabela apresenta uma correlação dos trabalhos selecionados com os termos usados na construção das *strings* de busca e as questões de pesquisa.

Tabela 12: Lista de Trabalhos selecionados

Trabalhos Selecionados					
Bases	**Autores**	**Título do Trabalho**	**Termos Associados**	**Questões Relacionadas**	***Score***
Google Scholar	Condori-Fernandez, et.al (2009)	A Roadmap Approach For Implementing Theory of Constraints In Manufacturing Organizations.	TOC, PO	Sq4, Sq5	C
	Ellis (2011)	A Theory of Constraints Service Systems Improvement Method: Case of the Airline Turnaround Problem.	TOC, SPI	Sq4, Sq5	C
	Murauskaite, Adomauskas, (2008)	Bottlenecks in Agile Software Development Identified Using Theory of Constraints (TOC) Principles.	TOC, SP, SPI	Pq, Sq1, Sq2	B
	Feng, Gui, Ling, (2007)	Explorations of Thinking Process based on TOC.	TOC	Sq4, Sq5	C
	Gülsün, *et al* (2012)	Improving System Performance: The Theory of Constraints and an Application in a Production Firm.	TOC, PO	Sq4, Sq5	C
	Staron, Meding (2011)	Monitoring Bottlenecks in Agile and Lean Software Development Projects – A Method and Its Industrial Use.	TOC, SP	Pq, Sq1, Sq2	C
	Rezaie, *et al* (2010)	Theory of Constraints and Particle Swarm Optimization Approaches for Product Mix Problem Decision.	TOC, PO	Sq4, Sq5	C

	Bailey (2009)	The Theory of Constraints: Productivity Metrics in Software Development	TOC, SP, SPI	Pq, Sq1, Sq2	A
	Almeida (2013)	Using the Theory of Constraints to Analyze Bottlenecks in the Freight Transportation System: the Case of the Center-north Corridor in Brazil.	TOC, PO	Sq4, Sq5	C
	Trojanowsk, Pająk (2010)	Using The Theory Of Constraints To Production Processes Improvement.	TOC, PO	Pq, Sq4, Sq5	B
	Bakhtiyari, *et al*. (2013)	Using Theory of Constraints in selecting product mix.	TOC, PO	Sq4, Sq5	C
Scopus	Kasemset, Kachitvichyanukul (2012)	A PSO-based procedure for a bi-level multi-objective TOC-based job-shop scheduling problem	TOC, PO	Sq4, Sq5	C
IEEE Xplorer	Lin (2009)	Using TOC Thinking Process Tools to Improve Safety Performance.	TOC, PO	Pq, Sq4, Sq5	C
	Ribeiro *et al*. (2015)	Bottleneck Identification in Software Development Processes: A Proposal Based on the Principles of the Theory of Constraints – Experimentation Report	TOC, SP, SPI, SPO	Pq, Sq1, Sq2, Sq3,	A
Springer Link	Geri, Ahituv (2008)	A Theory of Constraints approach to interorganizational systems implementation.	TOC, PO	Sq4, Sq5	C
	Rhee, Cho, Bae (2010)	Increasing the efficiency of business processes using a theory of constraints.	TOC, PO	Pq, Sq4, Sq5	B
	Baptista, et.al (2013)	Profit optimization in machining service providers using principles of the Theory of Constraints.	TOC, PO	Sq4, Sq5	C
	Kilger, Wetterauer (2007)	The Selection Process.	TOC, PO	Sq4, Sq5	C
ACM - Digital Library	Zhou, Rose (2009)	A Bottleneck Detection and Dynamic Dispatching Strategy for Semiconductor Wafer Fabrication Facilities.	TOC, PO	Sq4, Sq5	C
	Sengupta, *et al* (2008)	A New Method for Bottleneck Detection.	TOC, PO	Pq, Sq4, Sq5	B
	Hasgul, Kartal (2007)	Analyzing A Drum-Buffer-Rope Scheduling System Executability Through Simulation.	TOC, PO	Pq, Sq4, Sq5	C
	Lemessi, et.al (2012)	Semi-Automatic Simulation-Based Bottleneck Detection Approach	TOC, PO	Sq4, Sq5	C
Elsevier Science Direct	Xiao, *et al*. (2011)	Modified Active Set Projected Spectral Gradient Method for Bound Constrained Optimization.	TOC, PO	Sq4, Sq5	C
	Barreto, *et al* (2008)	Staffing a Software Project: A Constraint Satisfaction and Optimization-Based Approach.	PO	Sq4, Sq5	C

3.8.3 Considerações Sobre os Trabalhos Selecionados

Os trabalhos foram classificados de acordo com os critérios estabelecidos na Seção 3.7. Ao observar a relação entre os termos de busca e as questões de pesquisa, foi possível fazer algumas considerações.

1. **Trabalhos com Score A:** Os trabalhos com *score* A apresentam alguma relação com a (*Qp*) e com as questões (*Qs1 e Qs2*). Além disso, os trabalhos também estão relacionados com os termos (SP) e (SPI). Contudo não exploram a efetiva aplicação da TOC em processo de desenvolvimento de software. O trabalho de Ribeiro *et al*. (2015), é um estudo inicial e reporta a condução de um experimento para identificar gargalos no Processo de Desenvolvimento de Software (PDS). Já o trabalho de Bailey *et al*. (2009), basicamente mostra em quais áreas a TOC pode ser aplicada no processo de desenvolvimento de software, mas não mostra como usar a TOC para esta finalidade. O trabalho sugere ainda que a TOC pode ser usada em medições voltadas para o desenvolvimento ágil, principalmente com Extreme *Programming (XP), Software Development Life Cycle (SDLC), Feature Driven Development (FDD), Scrum and Rapid Application Development (RAD).*

2. **Trabalhos com *Score* B:** Os trabalhos com *score* B apresentam uma baixa correlação com a (Qp), mas acredita-se que os métodos usados no estudo primário, podem ser usados ou aplicados em processo de software. O trabalho de Murauskaite, Aomauskas, (2008), tem forte ligação com o emprego da TOC em processos de software e os termos (SP) e (SPI), mas não apresenta um método ou qualquer medição eficiente para a identificação de gargalos em processos de software ou questões de melhoria e otimização do processo de software. O trabalho também não deixa explicito como o estudo foi realizado, motivo pelo qual foi atribuído peso 2 e *score* B.

3. **Trabalhos com *Score* C:** Em geral, os trabalhos com *score* C não possui uma relação direta com a questão primaria (Qp), mas entende-se que os métodos aplicados podem ser aplicados em processo de desenvolvimento de software. Estes trabalhos estão diretamente relacionados com as questões (Qs4) e (Qs5) e com o termo TOC e PO.

3.9 Lacunas Encontradas com a Revisão

A análise dos resultados identificou, mesmo nos trabalhos de maior pontuação, algumas lacunas envolvendo o tema central desta pesquisa.

As principais lacunas encontradas são apresentadas a seguir.

Uso da TOC em Processo de Software: em geral o assunto é pouco abordado ou discutido, o que se tem são conceituações e contextualizações sobre o uso da TOC sem maiores aprofundamentos. Com relação ao PDS, não foi identificado nenhum trabalho que utilizasse a TOC em diretamente no processo de desenvolvimento de software.

Otimização de Processo de Software: o emprego de técnicas de otimização é largamente explorado em trabalhos envolvendo a TOC na indústria manufatureira, mas sem qualquer relação com processos de desenvolvimento de software. A busca não retornou nenhum resultado, apresentando uma grande lacuna e com ampla oportunidade de pesquisa, principalmente associado ao emprego de métodos quantitativos de otimização.

Identificação de Gargalos: Todos os trabalhos analisados que usam a TOC na identificação de gargalos estão relacionados ao ambiente produtivo. Portanto, uma grande lacuna foi encontrada no tema da aplicação dos métodos de identificação e tratamento de gargalos usados na TOC aos processos de desenvolvimento de software. O trabalho de (Murauskaite e Aomauskas, 2008), faz uma abordagem muito superficial apresentando somente o algoritmo da TOC, mas sem explanar, exemplificar e apresentar os resultados colhidos a partir da aplicação da TOC na identificação de gargalos em processos de desenvolvimento ágeis.

Os princípios da TOC: Os 5 passos da TOC, o método do Tambor, Pulmão e Corda (TPC) e os conceitos de melhoria continuada não são efetivamente explorados nos trabalhos que envolvem o processo de desenvolvimento de software. O emprego desses princípios é, em geral, parcialmente aplicado ou suprimido.

3.10 Oportunidades de Pesquisa

O estudo secundário evidenciou a falta de pesquisas que efetivamente aplicam a Teoria das Restrições em Processo de Desenvolvimento de Software, indicando a existência de uma grande lacuna com significativas oportunidades de pesquisa, tanto no campo teórico, quanto no campo prático (Ribeiro *et al*. 2017-a).

A Tabela 13 apresenta um conjunto de subáreas que podem ser transformadas em oportunidades de pesquisa para pesquisadores que se interessarem por este tema.

Tabela 13: Oportunidades de pesquisa identificadas

Oportunidades de Pesquisa: TOC X PDS			
Áreas	**Tipo de Estudos**		**Status: Situação identificada para a área**
	Teóricos	**Práticos**	
Processo de Software	SIM	SIM	Aberta
Melhoria do processo de Software[16]	SIM	SIM	Parcialmente Aberta
Otimização de Processo de Software	SIM	SIM	Aberta
Desenvolvimento de Software	SIM	SIM	Aberta
Otimização de Processo de Software	SIM	SIM	Aberta
Aplicação na Indústria de Software	SIM	SIM	Parcialmente Aberta
Gestão de projetos de Software	SIM	SIM	Aberta
Aplicação em Processo de Teste de Software	SIM	SIM	Aberta
Qualidade de Software	SIM	SIM	Aberta
Experimentos em Engenharia de Software	SIM	SIM	Parcialmente Aberta

As áreas identificadas como parcialmente abertas indicam que existe algum trabalho que emprega o TOC no PDS. Contudo, estas obras não apresentam solidez em termos de metodologia, método ou aplicação.

[16] *Foi realizada uma segunda busca apresentada em (Ribeiro et.al. 2018), contendo as seguintes bases: (1) Google Acadêmico (Br-Pt); (2) Scielo-Br, (3) BDBComp; e (4) Domínio Publico. Com o objetivo de estender a pesquisa na tentativa de identificar trabalhos (artigos, teses e dissertações) nacionais/português. A busca obedeceu aos mesmos procedimentos apresentados neste capítulo, e retornou o trabalho de (Costa et al. 2013) que é o resultado de uma dissertação de mestrado publicado no Simpósio Brasileiro de Qualidade de Software (SBQS 2013). O trabalho aplica a TOC em melhoria contínua de processos através do processo de raciocínio da TOC (Thinking Process) e apresenta um mapeamento de um processo baseado em Árvore de Realidade Atual (ARA) e Árvore da Realidade Futura (ARF). Embora o título do trabalho remeta a aplicação da TOC em melhoria do processo de software, a pesquisa foi direcionada a um processo específico (Processo de Atendimento de Chamados – apoiado por um software / plataforma) e não a um processo de desenvolvimento de software.*

4 PLANEJAMENTO DO EXPERIMENTO

"O maior inimigo do conhecimento não é a ignorância e sim a ilusão do conhecimento."

Stephen Hawking

Embasado nas principais referências sobre estudos empíricos em Engenharia de Software, este capítulo, irá apresentar a metodologia e o planejamento experimental utilizado para executar os "*quase-experimentos"* que deram origem a este trabalho. O planejamento experimental é o meio pelo qual este estudo será conduzido na tentativa de responder primeiramente a questão central da pesquisa: (1) Existe(m) gargalo(s) no processo de desenvolvimento de software? Respondida essa questão, desdobram-se outras duas questões, a saber: (2) Se existe(m), é possível identificá-lo(s) ou tipificá-lo(s)? (3) Uma vez identificado(s) o(s) gargalo(s) é possível estabelecer algum tipo de tratamento?

4.1 Metodologia

A metodologia adotada foi embasada nos trabalhos de (Wohlin *et al.*, 2012), (Wohlin, 2007), (Basili, *et al.* 2006) e (Juristo e Moreno, 2001). A formatação metodológica segue uma orientação com objetivos específicos para desenvolver a pesquisa simulando um ambiente real de produção de software em um laboratório acadêmico, reportando ao experimento: técnicas, métodos e o processo de produção.

De acordo com Carver *et al.* (2003-a), a pesquisa em ambiente de aprendizado em Engenharia de Software apresenta uma série de benefícios para os pesquisadores, os estudantes, os instrutores e até para a indústria. Carver *et al.* (2003-a), destaca ainda que este tipo de experimento promove uma cooperação técnica entre pesquisadores, indústria e estudantes. Já do lado dos estudantes, o ganho está relacionado ao *Know How* adquirido com a vivência prática e com a simulação do ambiente real, diante da possibilidade de analisar, modelar, implementar e testar um produto/software, como se fosse para um cliente real em uma organização real.

Basicamente, seguiu-se a orientação de Wohlin *et al.*(2012), apresentada na Seção 2.4 para o desenvolvimento da metodologia, o planejamento e a execução do experimento.

Os critérios adotados e os métodos de trabalho também se apoiaram em alguns trabalhos disponíveis na literatura que aplica o uso da TOC na indústria para identificação de gargalos, tais como: (Zughen e Zhu, 2009), (Hong e Yan, 2009), (Sengupta, 2008). Embora sejam trabalhos voltados para ambiente de produção de produtos manufaturados, acredita-se que os mesmos podem suportar o ambiente de produção de software, que é o nosso objetivo. Também serviram de fontes de leitura e estímulo os trabalhos de (Slack *et al.*, 2002), (Christopher, 2012), (Venanzi e Silva, 2013), (Franchi, 2011) e (Ritzman e Krajewski, 2004) sobre o planejamento e o gerenciamento de produção, que contribuíram para uma visão geral do ambiente fabril e dos processos de produção utilizados na indústria manufatureira.

4.1.1 Metodologia do Processo Experimental

A metodologia proposta envolve um conjunto de marcos que foram pré-estabelecidos para orientar o desenvolvimento do processo, a saber:

1. O ambiente externo: envolve dentre outras, a fase de estudos exploratórios e a criação dos times de desenvolvimentos que irão representar as máquinas e as linhas de produção no *shop*;

2. O planejamento do PDS: baseia-se principalmente no *PU-Like* e nas disciplinas elencadas para rodar o processo, além da identificação do caminho crítico e da corrente crítica;

3. A modelagem do processo: é a etapa em que é construído um modelo de execução do projeto envolvendo o PDS, a TOC, o esquema de escalonamento de tarefas (JSD) e a metodologia de desenvolvimento;

4. A execução do processo: que é estabelecida pelas diretrizes de execução do PDS, dado pelo *PU-like* e pelo processo experimental.

5. O gerenciamento e o controle do PDS: composto pelas atividades para guiar o processo acerca dos resultados esperados.

6. Os resultados: que são os artefatos produzidos em cada fase do *PU-Like* e que compõem o produto final.

A Figura 9 apresenta um esboço dos marcos metodológicos para o processo de construção e de execução do experimento. A figura ainda mostra os fluxos que orientam a execução do processo e a expectativa mediante a execução de cada marco/ subprocesso.

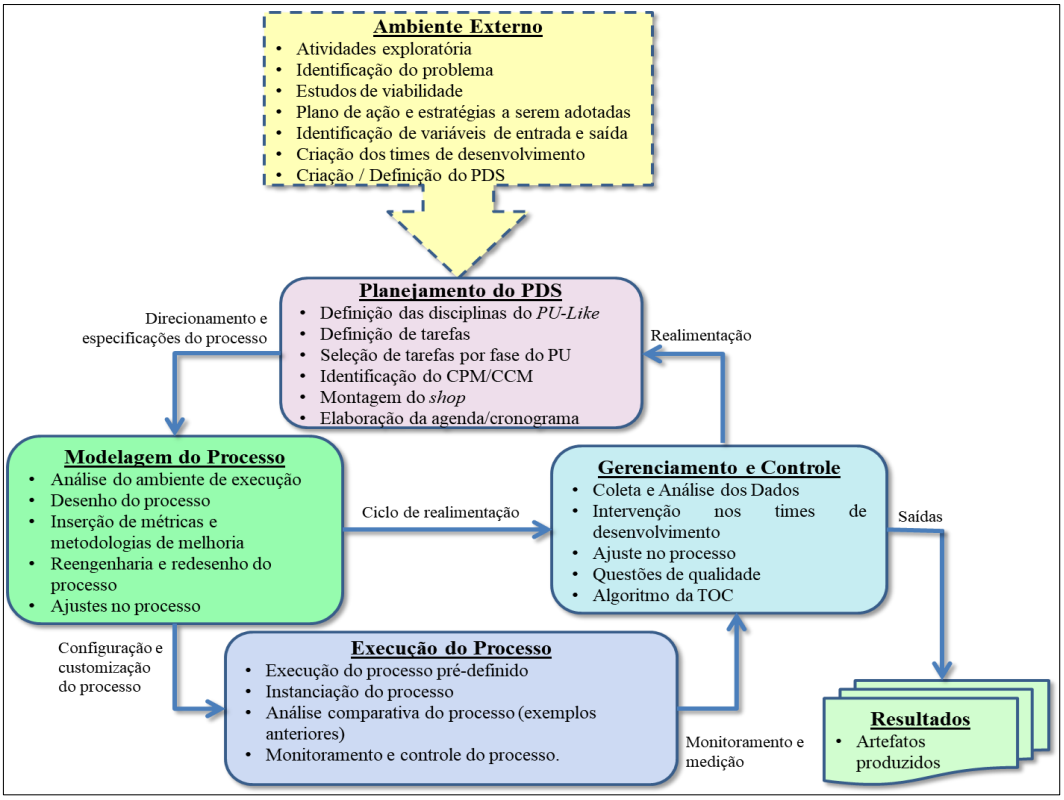


Figura 9: Esquema metodológico do processo de execução do experimento

A metodologia também fundamenta e dá suporte para o modelo do processo da Seção 4.2 e para o método de identificação de gargalos proposto no Capítulo 5.

4.2 O Modelo do Processo

Esta pesquisa envolve um ambiente dinâmico estabelecido pelo PDS, pelas pessoas envolvidas e pelo domínio do problema[17]. Assim, a abordagem *quali-quantitativa* foi explorada para a construção do modelo em uma estrutura envolvendo:

A Fase Exploratória: deve levar em consideração estudos/pesquisas iniciais, o ambiente, a interação entre pesquisador e participantes, o processo e os fluxos de execução de tarefas, os recursos utilizados e as variáveis e evidências encontradas. Tudo isso em detrimento aos objetivos do projeto.

A Construção do Modelo: compreende no desenho e na sequência de passos para executar o PDS. O modelo deve ser visualmente inteligível e ser comum a todas as máquinas e a todas as linhas de produção do *shop* e ser guiado pelo *PU-Like*.

[17] *O domínio do problema descreve as características do produto a ser desenvolvido e exibe uma coleção de funcionalidades que direciona a aplicação para um conjunto de problemas e suas possíveis soluções.*

A Implantação do Modelo: consiste na aplicação do método construído e na validação da proposta, levando-se em conta a identificação das restrições do sistema sem comprometer a avaliação e os resultados a serem alcançados.

A Execução do Experimento: deve ser guiada pela metodologia de desenvolvimento e pelo modelo de processo desenvolvido em consonância com o problema, com o PDS e com os objetos identificados na fase exploratória.

Avaliação dos Resultados: devem ser analisados qualitativamente e quantitativamente em função dos aspectos e objetivos da pesquisa. Os resultados devem ser anotados na fase exploratória, obedecendo ainda os critérios de avaliação e de qualidade previamente definidos.

A partir dos apontamentos metodológicos, foi realizado um mapeamento do processo do projeto onde foram identificados os seguintes processos e subprocessos:

1. **Planejamento:** O planejamento[18] é composto de dois subprocessos, "Exploração" e "Preparação".
2. **Execução:** Este processo é dividido em outros dois processos: O primeiro é o PDS, dado pelo *PU-Like*. O segundo é o processo de execução do projeto, o qual está diretamente alinhado ao PU-Like. O processo de execução do projeto se divide ainda em quatro *milestones*[19], que representa as fases do *PU-Like* e estende-se ao processo de execução do projeto seguindo estruturalmente o *PU-like e suas fases*.
3. **Análise de Dados:** Este processo tem por objetivo guiar a análise e avaliação dos dados colhidos. A análise é feita em duas etapas: (1) Análise qualitativa: Nesta etapa os dados *quali* são coletados, analisados e interpretados em tempo de execução. Isto é feito para que as restrições qualitativas sejam identificadas e tratadas durante a execução do processo a fim de reduzir os impactos na produção. (2) Análise quantitativa: Já os dados quantitativos são coletados ao fim de cada fase e ao final da execução do processo. A análise *quanti* é realizada em cada fase do *PU-LIKE*, e com profundidade ao fim da execução do PDS. Coma a análise é possível identificar os pontos de estrangulamento do processo de produção ou as razões pela qual alguma fase ou tarefa do processo produtivo não foi executada ou foi executada parcialmente, impedindo a maximização da produção.

[18] A elaboração conceitual do planejamento do experimento é detalhada no Apêndice C.
[19] *Milestone são subpartições dentro de um processo na notação BPMN.*

4. **Identificação de Gargalos:** O propósito deste processo é mapear os passos que devem ser seguidos para que os gargalos do PDS possam ser identificados, elucidados e tratados. O processo de identificação de gargalos é o ponto central desta pesquisa. Este processo canaliza e direciona todo o esforço da execução experimental e da execução do processo produtivo.

A Figura 10 apresenta o modelo geral do processo experimental, mapeado e desenhado usando a notação *Business Process Model Notation* (BPMN).

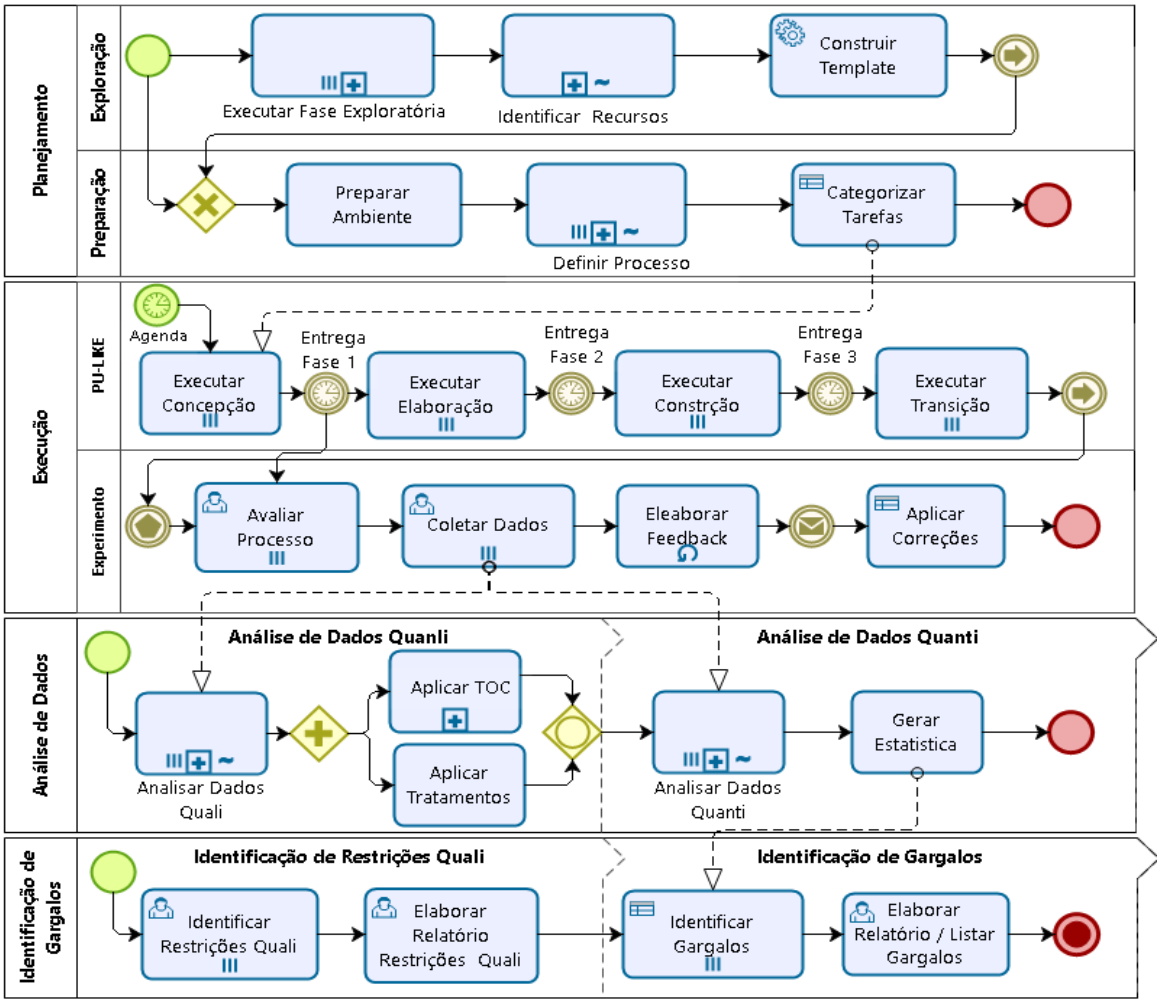


Figura 10: Modelo BPMN do processo de execução do experimento

A Figura 14 apresenta a modelagem do processo contendo os subprocessos e as atividades a serem percorridas para a execução do modelo experimental. O modelo expandido com todas as etapas e suas respectivas atividades podem ser consultados no Apêndice J.

4.3 Escopo

O escopo define o arcabouço do experimento e modela a proposta em função dos objetivos. O propósito do escopo é assegurar que os aspectos importantes do experimento sejam definidos antes do planejamento e da execução (Wohlin *et al.*, 2012). O escopo nasce da ideia geral do experimento e passa por refinamentos sucessivos até que os objetivos sejam alcançados.

A ideia inicial do experimento tem como foco a identificação de elementos restritivos do PDS através da aplicação dos princípios da TOC, que também serão empregados no tratamento das restrições encontradas.

O *PU-Like* foi adotado como o PDS utilizado nesta pesquisa, para guiar o desenvolvimento em função dos objetivos. O arranjo experimental estabelece o plano de trabalho inicial em função do problema e dos objetivos a serem alcançados.

O plano de trabalho foi dividido em quatro estágios envolvendo os recursos ambientais e humanos (pessoas envolvidas). A construção do plano apoiou-se em teorias, métodos, modelos de (Wohlin *et al.*, 2012) e (Wohlin, 2007), e ferramentas / *templates* disponíveis, como: (Basili e Weiss, 1984) e (Schull *et al*., 2008). Algumas características bem específicas foram inseridas para atender aos objetivos do experimento. O plano de trabalho adotado pode ser sumariamente apresentado da seguinte forma: (1) Estágio Experimental 1 – compreende o plano de ação inicial do experimento; (2) Estágio Experimental 2 – representa e define o ambiente e os participantes; (3) Estágio Experimental 3 – estabelece o planejamento e os métodos de controle do experimento e (4) Estágio Experimental 4 – define as metas e os critérios de medição.

4.4 Arranjo Experimental

O arranjo experimental foi elaborado para a pesquisa se desenvolver em um ambiente de aprendizado, onde foram planejadas e inseridas atividades práticas da disciplina de Engenharia de Software de acordo com a seguinte especificação:

1. **Número de repetições:** o experimento foi planejado para ser executado em três momentos diferentes, ou três semestres letivos e foram denominados de Experimento 1, Experimento 2 e Experimento 3.

2. Ambiente de desenvolvimento: laboratório de ensino de Engenharia de Software;

3. Participantes: alunos do curso de Ciência da Computação entre o 5º e o 6º período, o pesquisador e o orientador da pesquisa;

4. Execução: Em cada experimento, foi usado um domínio diferente e novos participantes foram selecionados.

5. Artefatos: Os artefatos são os resultados do processamento de uma dada linha de produção a partir de um conjunto de 27 atividades. Foram definidos 5 artefatos, a saber: Arquitetura, Banco de Dados, Codificação, Documentação e Teste, os quais deverão ser produzidos e entregues a cada fase do processo. A lista de atividades está disponível no Apêndice J desta tese.

6. Avaliação: é a medição dos resultados produtivos. A cada entrega os artefatos devem ser avaliados e medidos, conforme critérios próprios e específicos desenvolvido para esta pesquisa.

7. Template: foi desenvolvido um *template*, baseado em (Basili e Weiss, 1984), (Travassos, 2012), (Wohlin *et al.* 2012) e (Schull *et al*., 2008), para conduzir o experimento de acordo com a orientação das definições do escopo e as questões de pesquisa. Uma representação sumarizada do *template* pode ser encontrada no Apêndice E.

A Figura 11 apresenta um esboço do arranjo experimental desenvolvido e aplicado nos três casos (Exp1, Exp2 e Exp3).

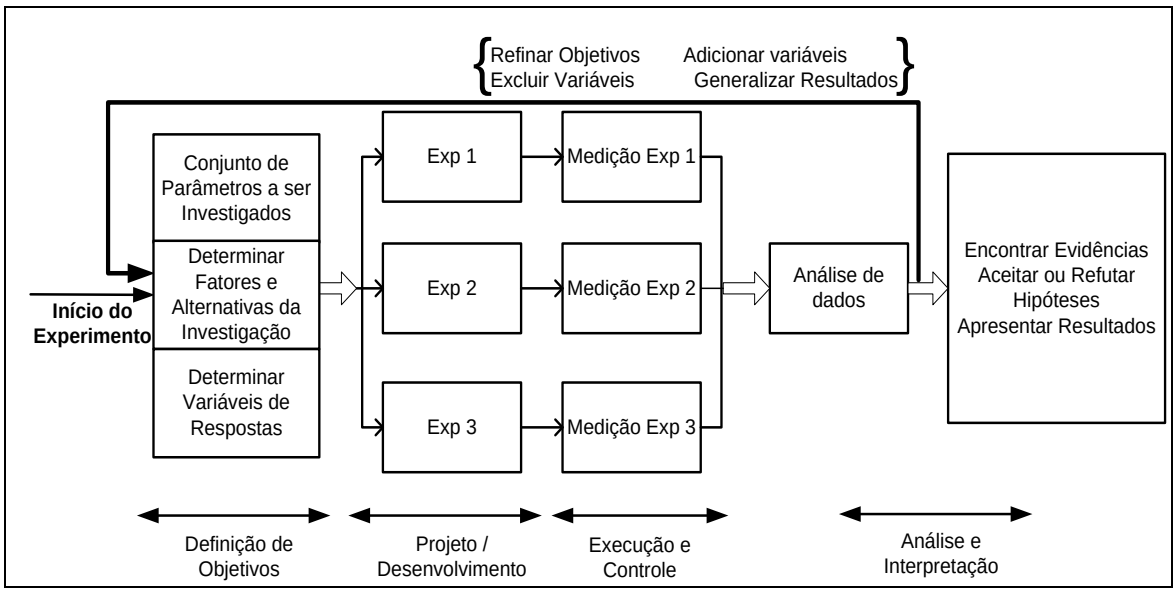


Figura 11: Arranjo experimental construído (adaptado de [Juristo e Moreno, 2002]).

4.5 Planejamento do Experimento

A experimentação é o caminho mais seguro para confrontarmos os problemas e propor soluções para melhorar a produção de software, tanto na academia, quanto na indústria. E todo experimento, deve seguir um plano de trabalho bem definido e bem delineado para que os objetivos da pesquisa sejam atingidos.

4.5.1 Construção do Cenário de Execução do Experimento

A construção do cenário deve envolver o ambiente, as metodologias, o objeto de pesquisa e os *stakeholders*[20] do projeto. O cenário também deve fundamentar a história do experimento para direcionar o pesquisador no que diz respeito às perspectivas e ao futuro, com o objetivo de buscar o que é realmente importante para a pesquisa.

Segundo Lourenço Jr. *et al.* (2010), o cenário é uma ferramenta poderosa para direcionar a pesquisa ao que é fundamentalmente significante e desconhecido, devendo representar histórias plausíveis, pertinentes e alternativas sobre o futuro. O cenário construído neste experimento segue a recomendação da literatura e representa o contexto em que o experimento é estabelecido.

O cenário desenvolvido é composto pelo modelo geral do processo produtivo estabelecido pelo ambiente de execução, o ambiente simulado (indústria), os participantes, os objetivos, as estratégias, a metodologia, a perspectiva e a história que é o enredo que delimita e delineia o objeto a ser estudado.

A Figura 12 traz uma representação do cenário produtivo desenvolvido, onde os recursos a serem transformados são as entradas que alimentam o processo para gerar os artefatos (saídas) em um ciclo contínuo de desenvolvimento, envolvendo: objetivos, estratégias, avaliação e revisão do processo e do projeto.

[20] ***Stakeholders:*** *Refere-se a todo o pessoal estrategicamente envolvido com o projeto, podendo ser uma pessoa ou grupo de pessoas com interesse particular na empresa, no negócio ou em um resultado específico.*

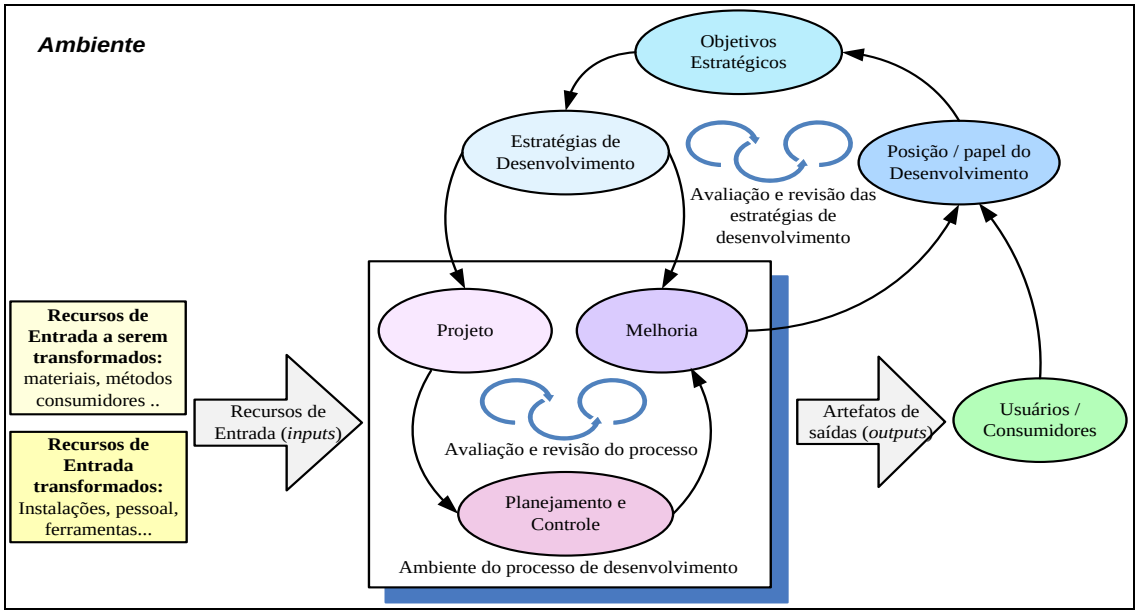


Figura 12: Modelo geral do processo produtivo (adaptado de (Slack *et al.*, 2011)).

4.5.2 Definição do Modelo de Desenvolvimento

O modelo utilizado para realizar este experimento estabelece uma padronização, ligando os cinco[21] elementos de padrões de projeto apresentados por Gamma *et al.* (1995) ao problema na tentativa de buscar uma solução comum. O modelo representa o ambiente produtivo e estabelece o processo de desenvolvimento. A padronização segue o fluxo do modelo e é aplicada para todas as linhas de produção, permitindo o controle e o acompanhamento por parte do pesquisador em torno do processo de desenvolvimento. O PDS, modelado segundo o *PU-like*, estabelece a direção e orienta o processo em cada fase do desenvolvimento do produto.

4.5.3 Definição de Recursos a Serem Utilizados

Os recursos são os elementos e os meios utilizados na experimentação alinhados com os objetivos da pesquisa, e abrangem: pessoas, tempo, materiais, métodos, ferramentas. Os recursos podem ser inseridos ou ajustados durante a execução do experimento em função das perspectivas e das metas a serem alcançadas. Os principais recursos utilizados no desenvolvimento desta pesquisa foram: (1) pessoas; (2) calendário (tempo); (3) ferramentas computacionais (hardware, software, rede); (4) ferramentas de modelagem; (5) métodos de desenvolvimento, modelos e processos e (6) laboratório de ES.

[21] *1. Nome: descreve a essência do padrão; 2. Objetivo: define como o padrão deve atuar; 3. Problema: apresenta o problema a ser aplicado; 4. Solução: apresenta a solução encontrada e 5. Consequências: refere-se aos benefícios da utilização do padrão.*

4.5.4 Definição dos "Entregáveis"

Neste trabalho, utilizou-se o termo "*entregável*" para representar os artefatos produzidos em cada fase do experimento de acordo com a estrutura do *PU-Like*. Os "*entregáveis*" são os mesmos em cada fase do processo e o conjunto dos valores entregues, compõe o produto final, dado pela fase de transição do *PU-Like*.

Esta terminologia é aplicada principalmente na produção parcial das máquinas, que são utilizadas para a medição, o monitoramento e o controle, seguindo necessariamente o modelo experimental proposto para desenvolver e conduzir o experimento. Os entregáveis são os artefatos construídos e são compostos por:

1. **Arquitetura:** A arquitetura define a planta estrutural dos sistemas desenvolvido em cada experimento e é composta pelo modelo MVC (*Model View Control*), pelo diagrama de sequência da aplicação baseado no MVC e o diagrama de pacotes, que representa o modelo decomposto do sistema.
2. **Banco de Dados:** O banco de dados é composto pela documentação do BD, o modelo lógico, os serviços, e os scripts SQL (*Structured Query Language*).
3. **Codificação:** compreende na codificação das funções de software, na produção das telas do sistema, na camada de aplicação do MVC, no "*interfaceamento*" interno.
4. **Documentação:** a documentação contém a especificação do sistema, a análise dos requisitos, os modelos de domínio (diagrama de casos de uso, diagrama de classes, diagrama de sequência e diagrama de atividades). A documentação contempla ainda os modelos lógico e relacional do banco de dados, os planos de testes e os testes realizados.
5. **Teste:** representa o plano de teste, o relatório de teste e as classes de teste unitário e funcional.

4.5.5 Definição de Resultados Produtivos

Resultados produtivos é um termo utilizado neste trabalho para referenciar todo o material produzido durante a elaboração e a execução do experimento, tanto pelo pesquisador quanto pelos participantes. Os resultados produtivos gerados durante a experimentação são: os artefatos de saída (entregáveis e produto final); a documentação elaborada; os relatos das máquinas; os *surveys*; as anotações; as planilhas de dados; os

questionários de auto avaliação do projeto; os formulários; as fórmulas; as hipóteses; os modelos analíticos e de domínio, os diagramas e o processo desenvolvido.

É através dos resultados produtivos que são colhidos os dados experimentais, que serão posteriormente analisados e interpretados em busca das evidências, que irão dar sustentação aos questionamentos da pesquisa.

4.5.6 Métodos e Instrumentos de Coleta de Dados

Em função das características do experimento e de sua pluralidade, optou-se pela abordagem *quali-quantitativa* para coleta, análise e medição. A Tabela 14 apresenta os instrumentos qualitativos utilizados.

Tabela 14: Relação de instrumentos qualitativos

Instrumento	Método de Aplicado
Agendamento das tarefas a serem entregues	Cronograma de entregas de acordo com o PU-Like
Auto avaliação de conhecimento técnico dos participantes.	Aplicação de Questionários
Coleta de documentos e fontes bibliográficas.	Análise de textos e documentos
Percepção da dificuldade de realização de atividades	Aplicação de *Survey* / Questionário
Relatos dos participantes	Breve redação de dificuldades encontradas em tempo real
Observações do livro de registro do experimento	Anotações em tempo de execução

O *Survey* foi o instrumento usado para avaliar a dificuldade das máquinas, onde cada membro dos times de desenvolvimento aponta uma nota com relação à dificuldade encontrada em uma escala de (0 – 3). Sendo: (1) 3,0 a nota máxima, ou seja, a tarefa pontuada com nota 3,0 representa grande dificuldade em desenvolvê-la ou concluí-la; (2) a nota 2,0 reflete em dificuldade moderada; (3) a nota 1,0 indica pouca dificuldade e (4) a nota 0,0 resulta em nenhuma dificuldade na realização da tarefa.

A abordagem quantitativa utilizou os instrumentos apresentados na Tabela 15[22].

Tabela 15: Relação de instrumentos quantitativos

Instrumento	Unidades	Método utilizado
Contagem de experimentos (repetições)	3	Template
Contagem de tempo (totalizado em minutos)	2	Software de acompanhamento
Contagem de máquinas por LP	>= 2, <= 4	Template
Contagem de linhas de produção	10, 9, 6	Template
Contagem máquinas/participantes	19, 28, 31	Template
Contagem de artefatos	5 p/ Grupo	Template
Contagem de tarefas (atividades)	26 p/ Grupo	Template

[22] *Os valores da tabela 9 na coluna Unidade para os itens: Contagens de linhas de produção e Contagem de máquinas são os valores do Exp1, Exp2 e Exp3, obedecendo necessariamente esta ordem.*

Relatórios numéricos	3 p/ Exp.	Template
Seleção de variáveis amostrais.	1 p/Exp	Template
Repositório de dados	1 p/Exp	Base de dados do experimento

A abordagem *quali-quantitativa* corresponde a uma união dos dois métodos e seus instrumentos de coleta, e será aplicada para sustentar as evidências observadas com o estudo.

4.5.7 Formulação de Hipóteses

A hipótese científica deve ser levantada para instigar a experimentação e como consequência, a aceitação ou rejeição de uma tese. Segundo Ferreira (2011) a hipótese científica deve conter uma proposta testável dos objetivos e a metodologia apresentada deve ser capaz de testá-la. Para Wohlin *et al.* (2012), o teste de hipótese e a base da análise estatística de um experimento e deve ser levantado na fase de planejamento e definição do experimento. Duas hipóteses devem ser formuladas: a hipótese **nula** (H_0) e a hipótese **alternativa** (H_1).

As hipóteses (H_0 e H_1) apresentadas nesta pesquisa, mais precisamente no Capítulo 1, na Seção 1.6.2, foram construídas com base nas questões levantadas e apresentadas na Seção 1.6.1. Tanto as hipóteses quantos as questões, serão analisadas e respondidas no Capítulo 7 deste texto. Para todos os experimento realizados, foi estabelecido um nível de confiança, $\alpha=0.05$.

4.5.8 Seleção de Variáveis

A seleção das variáveis experimentais e de respostas é uma etapa importante em qualquer experimento. As variáveis experimentais, comumente denominadas de variáveis independentes, constituem nas variáveis de entrada do estudo primário.

De acordo com Wohlin *et al.* (2012) as variáveis independentes podem ser métodos de desenvolvimento, processos, ferramentas e o próprio ambiente. Enquanto que as variáveis de respostas ou variáveis dependentes são as variáveis de saída. Segundo Juristo e Moreno (2002) as variáveis de resposta são oriundas da matemática e têm por objetivo formular uma função que relaciona a variável de resposta aos fatores que influenciam a variável (entradas).

Wohlin *et al.* (2012) destaca ainda que antes de iniciar qualquer projeto as variáveis dependentes e independentes que serão estudadas, exploradas e analisadas no experimento

devem ser escolhidas. As Figuras 13 e 14 apresentam o esquema de inserção de variáveis independentes no experimento e sua saída em função das variáveis dependentes.

Figura 13: Conceito de variáveis independentes e dependentes (Wohlin *et al.*, 2012).

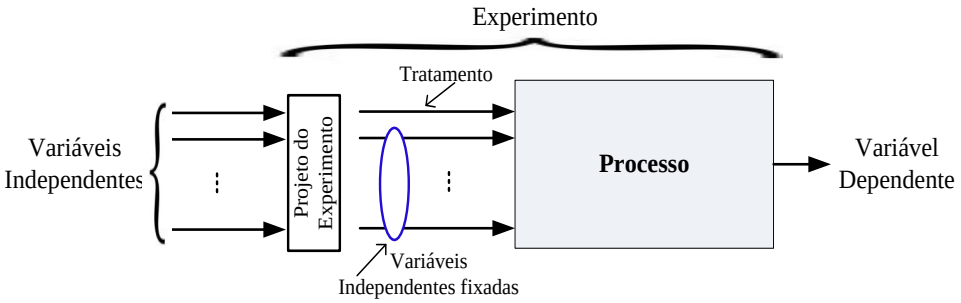


Figura 14: Variáveis independentes e dependentes em um experimento (Wohlin *et al.*, 2012).

Neste estudo, as variáveis independentes selecionadas são: a TOC; o PDS; o PU; as linguagens; as ferramentas; o ambiente; o objeto de estudo e os participantes (alunos) que representam as máquinas na linha de produção.

As variáveis dependentes são os resultados produtivos, dado principalmente pelos artefatos produzidos pelas linhas de produção em cada fase do processo produtivo e de acordo com as entradas/variáveis independentes. Os relatórios, as avaliações de conhecimento, os *surveys* e as anotações do livro de registros dos experimentos (*log book*), também são variáveis dependentes do experimento. Neste caso específico, estes resultados são produzidos pelo pesquisador/observador durante a execução dos ensaios.

Seguindo as notações referenciais, os critérios adotados nesta pesquisa envolvem a análise qualitativa e quantitativa. A parte qualitativa refere-se ao conjunto de informações colhidas do ponto de vista da observação etnográfica (vivência direta da realidade) a partir do objeto de estudo. É o resultado da observação rotineira, da transcrição textual dos resultados dos *surveys* e dos questionários de auto avaliação do conhecimento e dos registros no livro do experimento.

A parte quantitativa envolve a quantificação numérica dos dados colhidos durante os três ensaios experimentais, e compreende: (1) do esforço – quantidade de energia aplicada para produzir os artefatos; (2) da produção – quantidade de artefatos produzidos e (3) da dificuldade – resultados da pesquisa (*survey*) sobre a dificuldade encontrada pelos

participantes. Os dados quantitativos são colhidos ao fim de cada fase do *PU-Like*, mediante as entregas realizadas e a aplicação dos *surveys* para medir a dificuldade do ponto de vista dos participantes em cada tarefa.

4.5.9 Definição dos Critérios de Análise e Interpretação dos Dados

A análise dos dados é um processo complexo que envolve anacronismos entre dados pouco concretos e conceitos abstratos, raciocínio indutivo e dedutivo e descrição e interpretação que levam a entendimentos e a constatação de um estudo (Teixeira, 2002).

De acordo com Newman e Benz (1998) a metodologia qualitativa envolve uma coleção variada de materiais empíricos, como: estudo de caso, experiência pessoal, observações, introspecção, história de vida, entrevista, interações e textos. Já os métodos quantitativos são referenciados a partir das dos testes de hipóteses elaborados em conformidade com as questões de pesquisa do planejamento experimental estabelecidos pelas variáveis dependentes (respostas) através da medição e controle das variáveis independentes (entradas) selecionadas. A metodologia quantitativa envolve dentro outros aspectos, principalmente os métodos estatísticos, probabilísticos, matemáticos, o tratamento dos dados, a análise estatística dos resultados e as evidências (Newman e Benz, 1998).

Nesta pesquisa, a análise dos dados segue uma abordagem "*quali-quantitativa*", ou seja, envolvem os dois métodos. A parte qualitativa refere-se aos resultados colhidos com a observação cotidiana dos participantes e a parte quantitativa refere-se aos resultados numéricos analisados e medidos a partir das variáveis independentes. O método aplicado e os critérios de medição serão apresentados no Capítulo 5.

4.5.10 Interpretação dos Dados

Segundo Wohlin *et al.* (2012), a interpretação dos dados passa por um conjunto de três etapas: (1) Caracterização: os dados são caracterizados utilizando estatística descritiva, que visualizam tendência central, dispersão, etc.; (2) Exclusão: dados anormais ou falsos são excluídos, reduzindo assim, o conjunto de dados a serem analisados; e (3) Análise: os dados são analisados com base nas hipóteses levantadas para o experimento.

A análise "*quali-quanti"*, utilizada neste estudo é obtida pela junção dos métodos **qualitativo e quantitativo**, sendo estes executados independentemente um do outro.

Os dados qualitativos foram colhidos, analisados, interpretados e tratados em tempo real, *i.e*, durante a execução de cada experimento. Isto foi feito para cada linha de produção em cada *shop*. Estes dados foram usados ainda para realizar um cruzamento com os dados quantitativos, que são analisados parcialmente durante as entregas (4 fases do *PU-Like*) e integralmente ao fim de cada rodada experimental. O cruzamento dos dados gerados pelas duas abordagens é necessário para que se possa validar as evidências observadas com o estudo.

4.5.11 Ameaças à Validade

A validade dos resultados obtidos em experimentos depende de fatores que configuram a experiência (Wohlin *et al.*, 2012). Segundo (Wainer 2007), a presença de ameaças à validade depende necessariamente do modelo ou desenho experimental definido pelo pesquisador e está diretamente relacionada com ao nível de confiança adotado durante o período de investigação da pesquisa.

Existem diferentes tipos de validade e estas podem ser definidas ou priorizadas dependendo do objetivo do experimento. Os tipos mais comuns apresentados por (Mafra e Travassos, 2006), (Wohlin, *et al.*, 2012) e (Wainer 2007), são: validade interna, validade externa, validade de construto e validade conclusiva.

As principais ameaças à validade identificadas nesta pesquisa são:

1. **Validade interna:** (1) o fato de o pesquisador interagir diretamente com os objetos da pesquisa; (2) o fato da seleção de variáveis independentes serem as mesmas para os 3 ensaios; (3) o fato de que a seleção dos sujeitos não depende exclusivamente do pesquisador; e (4) o fato de que o ambiente de execução possa influenciar nos resultados.
2. **Validade externa:** (1) a pesquisa ter sido feita somente com alunos de um único departamento de uma única universidade; (2) a pesquisa não ter sido feita em outros ambientes e com outros sujeitos; e (3) o fato de que a pesquisa possa não ter abrangência e com isso não produzir os efeitos necessários para a simulação de uma situação real.
3. **Validade de constructo:** (1) o método desenvolvido pode não ser aplicável ou não produzir perspectivas para a execução do experimento; (2) as observações etnográficas, processual e ambiental e a análise dos entregáveis pode não ser

suficientes para identificar os gargalos; e (3) as técnicas de medição e os indicadores utilizados podem não ser representativos para identificar os gargalos no PDS.

4. **Validade de conclusão:** (1) a falta da execução e observação do trabalho em outros ambientes pode levar a conclusões imprecisas devido à falta de variedade amostral de dados; e (2) os indicadores e a análise empregada podem não ser suficientes para se tenha resultados conclusivos confiáveis.

Estes quatros tipos de ameaças de validade serão explorados com a abordagem *quali-quantitativa* na tentativa de diminuir os riscos associados a esta pesquisa.

4.5.12 Livro de Registro do Experimento

O livro de registro do experimento *(log-book)* contém uma série de registros e históricos de anotações etnográficas sobre a execução do experimento. Neste trabalho os registros e anotações foram pautados com base nos seguintes critérios: (1) opinião dos participantes; (2) correção dos artefatos; (3) revisão dos artefatos (4) avaliação do esforço; (5) avaliação dos entregáveis; (6) exposição/apresentação da produção; (7) conformidades com os padrões; e (8) aplicação/uso correto das ferramentas e metodologias.

Todos estes critérios serão aplicados em cada entrega realizada conforme as fases do *PU-Like,* e naturalmente obedecendo à metodologia, o planejamento e os métodos de pesquisa.

O correto preenchimento do livro de registro foi fundamental para que as informações não fossem perdidas durante a execução dos experimentos. Além disso, com as anotações registradas no livro, foi possível interferir pontualmente em cada experimento e tomar decisões acerca do mesmo para uma melhor condução do experimento, identificação e tratamento das restrições e observação dos resultados.

4.5.13 Definição do Formato e dos Requisitos Mínimos de Cada "Entregável"

Para formatar e convencionar os resultados produtivos a serem entregues pelas máquinas foram adotados os seguintes critérios:

- As entregas devem ser realizadas em conformidade com as fases do *PU-Like*;
- As entregas devem ser realizadas nas datas previstas no calendário/agenda de execução do experimento;

- Os requisitos mínimos dos entregáveis compreende de um conjunto de tarefas que devem ser executas para cada artefato construído;
- As entregas devem cobrir os requisitos mínimos pré-definidos e elaborados seguindo as fases do *PU-like*;
- Os artefatos definidos são: Arquitetura, Banco de dados, Codificação, Documentação e Teste;
- O conjunto de tarefas a serem executadas é fixo e previamente definido;
- As tarefas podem ser processadas pelas linhas de produção quantas vezes forem necessárias para que sua completude seja atingida;
- As tarefas podem ser processadas e reprocessadas por diferentes máquinas na linha de produção nas diferentes fases do *PU-like;*
- Os artefatos são fixados às fases do *PU-Like*, ou seja, em cada fase os cinco artefatos devem ser produzidos e entregues.
- Em cada fase do *PU-Like*, novas atividades são inseridas para a completude dos artefatos a serem entregues.

É importante ressaltar que os mesmos critérios foram empregados nos três experimentos realizados. Com exceção da computação do tempo consumido para produzir cada tarefa, que passou a ser exigido a partir do experimento 2.

4.5.14 Elaboração de Critérios de Avaliação

Avaliar a produção de um ambiente de produção de software não é uma tarefa trivial. Para criar uma avaliação padronizada e que pudesse ser aplicada comumente a cada uma das linhas de produção definiu-se os seguintes critérios, os quais foram aplicados em cada entrega realizada e de acordo com os artefatos produzidos.

1. **Entrega total ou completa:** a linha de produção produziu todas as tarefas e consequentemente entregou todos os artefatos. Neste caso será atribuído peso 1.
2. **Entrega parcial:** a linha de produção entregou uma quantidade *(>50% < 100%)* das tarefas a serem realizadas na fase. Neste caso será atribuído peso *0.5*.
3. **Artefato não entregue:** quando a linha de produção não processou as tarefas ou processou uma quantidade inferior a *50%*. Neste caso o peso atribuído será *0*.

Para efeitos de quantificação dos artefatos entregues e para definir a qualidade do que foi entregue, foi estabelecido um range com 0.0, 0.5 e 1.0, conforme os pesos

atribuídos acima, que correspondem à produção das máquinas para as entregas não realizadas, as entregas parciais e a entrega total.

4.5.15 Critérios de Aceitação do Produto de Software

As normas ISO/IEC 12207:2008, 25010:2011, 25020-24:2010 e 25040-41:2012 (NBR/ISO/IEC 2008; 2010; 2011; 2012) estabelecem alguns critérios de qualidade para aceitação de um produto de software, baseando-se nos seguintes parâmetros: funcionalidade, confiabilidade, usabilidade, eficiência, manutenabilidade e portabilidade. Todos estes parâmetros devem ser avaliados quanto aos recursos utilizados, o ambiente, o processo e o produto de software gerado. A norma ISO/IEC 25010 e 25020-24 recomenda ainda um sistema de "*ranqueamento*" com pontuações que contemplam os seguintes critérios: Satisfatório *{3-Execelente, 2-Bom, 1-Regular}* e Insatisfatório *{0-Ruim e/ou Não entregue ou Não construído}*.

Exclusivamente neste trabalho, para a aceitação dos artefatos entregues em cada fase e do produto final, foi seguido à orientação da ISO/IEC 25010:2011, mas usando somente os parâmetros: funcionalidade, confiabilidade e usabilidade. O peso atribuído seguiu os critérios de avaliação apresentados na Seção 4.4.14 para cada artefato produzido. A média obtida ($\mu \geq 75\%$) estabelece a aceitação do artefato como entregue. O mesmo critério se aplica ao produto final, ou seja, a média obtida somando todos os artefatos produzidos de acordo com os critérios adotados para avaliação e aceitação deve ser maior ou igual a 75%.

4.5.16 Definição dos Meios de Comunicação entre Interlocutores e os Grupos

Para estabelecer a comunicação entre as equipes e os interlocutores, foram definidos três canais de comunicação:

1. ***Site*** **da disciplina**: Local onde disponibilizamos os conteúdos (material de estudo, estudo de caso, observações e algumas exigências).

2. *E-mail*: Meio de comunicação criado e utilizado para enviar os artefatos produzidos em cada fase. O *e-mail* também serviu para o envio de dúvidas e questionamentos sobre o domínio do problema, sobre o calendário das entregas e sobre os artefatos a serem produzidos. Este canal também foi utilizado pelo

interlocutor para responder aos questionamentos e realizar algumas cobranças, em geral direcionadas a algum grupo específico.

3. **Grupo fechado em rede social:** Este canal de comunicação foi usado para questões e dúvidas comuns de todo o grupo, além de servir também como um fórum de debate. Geralmente os debates envolvem apenas os membros do grupo em assuntos e dificuldades específicas encontradas pelos grupos, embora a colaboração seja livre. Neste canal também foram disponibilizados alguns materiais de consulta, como: *templates,* notas de aula e exemplos.

4.6 Escalonamento de Tarefas Usando *Job Shop Scheduling*

A observação nesta pesquisa passa necessariamente pelo ambiente de execução dos ensaios experimentais. Entretanto é importante a adoção de um modelo eficaz para o processamento das atividades, dado pelo conjunto de variáveis independentes envolvidas neste tipo de estudo.

Após desenvolver a metodologia e modelar o processo para executar os experimentos, encontrou-se em um modelo *job shop* a melhor alternativa para modelar o escalonamento das tarefas no PDS.

4.6.1 Visão Geral do Modelo *Job Shop* Adotado

A inserção das tarefas nas máquinas não obedece a um ordenamento fixo ou linear. Além disso, uma tarefa pode ser compartilhada e "*reescalonada*" até que seja concluída ou até atingir a qualidade esperada.

O escalonamento varia em função da disponibilidade e da capacidade das máquinas. Além disso, cada máquina pode estabelecer o que deseja fazer e em que tempo de acordo com a lista dos artefatos a serem produzidos e entregues. Contudo, o PDS impõe a priorização de algumas tarefas para que o produto possa ser construído. Estas tarefas podem ser mapeadas através dos métodos do caminho crítico e da cadeia crítica[23] e foram elencadas como tarefas predecessoras (*w*).

[23] *O mapeamento das 26 tarefas através dos métodos do caminho crítico e da cadeia crítica é apresentado nos Apêndice J.*

Algumas tarefas dependem ainda de aprimoramento do conhecimento técnico. Isto é feito através da inserção de conteúdo teórico da disciplina de Fundamentos de Engenharia de Software (FES), o que corrobora com o aspecto não linear do escalonamento das tarefas *shop*.

Um *job (j)* escalonado em uma dada máquina pode transitar para outra máquina ou até mesmo ter o processamento compartilhado, estabelecendo assim um processamento dinâmico das tarefas. Em virtudes destas características, foi escolhido o modelo *Dynamic Job Shop Scheduling* (DJSS). Na Figura 15, é possível observar que os *Jobs(j)* navegam entre as máquinas em um processo de troca/reescalonamento ou compartilhamento para conclusão da tarefa.

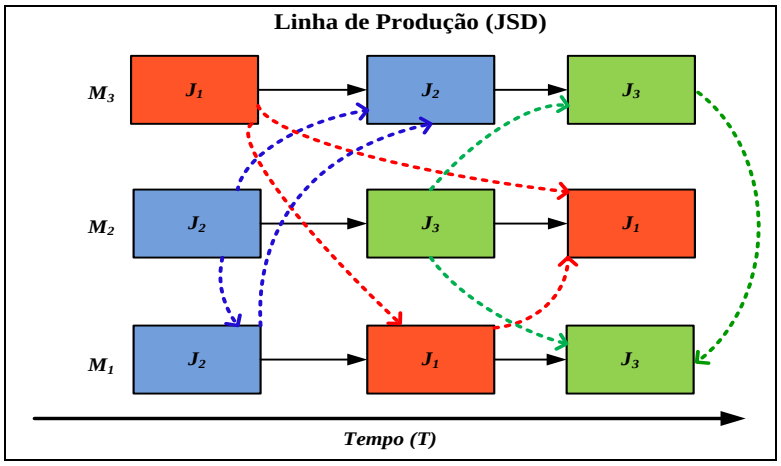


Figura 15: Modelo esquemático do *Job Shop* Dinâmico

4.6.2 O Modelo *Job Shop* Dinâmico (JSD)

Para representar o processo produtivo e o escalonamento das tarefas no ambiente de produção, foi desenvolvido um modelo baseado na composição de um sistema *job shop* dinâmico com as seguintes características:

- Um *shop S* é composto por um conjunto *L* de Linhas de Produção;
- Uma Linha de Produção *L* é composta por um conjunto *M* de máquinas;
- Cada máquina M_i é representada por um indivíduo *i*;
- Uma tarefa *j* corresponde a um *job, onde j = 1..26;*
- Uma relação *W(a,p)* denota que a atividade *a* é predecessora de *p*;
- Um conjunto de artefatos é denotado por *A de j, com A= {1..5};*
- Uma operação *O* é uma quádrupla denotada por *O (Mi, Aj, t, int*), onde M_i são as máquinas da LP processando A_j artefatos iniciados em um tempo *t* de um intervalo *(int)*;

- Cada *shop* $S \supset L$ componentes processando A_j $Artefatos_{tarefas}$;
- $D_A = \{A_1, A_2, ..., A_n\}$, com $n = i..5$, são os conjuntos de artefatos produzidos e entregues em uma F_f, onde *f é uma das fases do PU-like;*
- Cada L deve produzir e entregar um conjunto D_A de artefatos ao fim de cada fase do *PU-Like*;
- Uma tarefa *j* pode ser reescalonada até que seja concluída ou até atingir a qualidade esperada, independente da fase do *PU-Like.*

A partir das definições dos parâmetros e das variáveis que compõe o modelo *job shop* dinâmico, o problema pode ser modelado pela seguinte função objetivo:

$$Max\ (D_A) = Max\ (Aval\ (A_j)), \qquad \forall j = 1..5$$

Onde: $Aval(A_j)$ é a avaliação do percentual $(j \rightarrow 0..100)$ da quantidade do tipo de Artefato (A_j) entregue em cada uma das fases (F_f) do *PU-like.*

Sujeito a:

1. $(A_i, A_j) \in W \rightarrow \forall t \in T \ \neg\ [(\exists (O\ (Mi, Aj, t, int)) \wedge \exists (O\ (Mk, Aj, t, int))]$

*/**duas máquinas não podem trabalhar em uma mesma atividade de um artefato (A_j) num dado instante do tempo.**/*

2. $t \in T\ \neg\ [(\exists\ (O\ (Mi, Aj, t, int)) \wedge \exists\ (O\ (Mi, Ak, t, int))]$

*/**uma máquina não pode trabalhar em dois artefatos distintos num dado instante do tempo**/*

3. $T_{M_k} = \sum int_k < Orc(k)$

*/**o somatório dos intervalos (int_k) de tempo usados para M_k são restritos ao seu orçamento Orc(k) de tempo disponível.**/*

4.6.3 Especificação do Problema (JSD)

O problema em torno do JSD consiste em identificar um ou mais linhas de produção do *shop* com restrições de capacidade em tempo de execução.

Uma restrição de capacidade é qualquer elemento impeditivo que perturba o processo produtivo, fazendo com que uma LP possua entrega (D_A) menor que 100% dos artefatos, dado pelo conjunto de *jobs (j)* escalonados. Os recursos com restrição de capacidade limitam a produção das LPs, que quando não tratados, acarretam um gargalo no processo produtivo.

Um gargalo por sua vez, é denotado pela limitação de produção de uma máquina causada por uma ou mais restrições de capacidade que impendem à máxima produção, definida por: $D_A=\{(M_i\ A_j) = 100\%\}$. Neste específico modelo, a produção máxima se dá pelo conjunto de tarefas processadas em cada uma das fases conforme estabelece o desenvolvimento baseado no *PU-Like* e conforme o calendário produtivo, o qual delimita o prazo de cada entrega e estabelece o orçamento de horas para o *shop*.

O conjunto de tarefas escalonadas e devidamente processadas em todas as fases do *PU-Like* define a produção total de cada máquina e consequentemente o produto final a ser entregue.

A Figura 16 a seguir apresenta um caso simples do ambiente *job shop* modelado para uma linha de produção (LP) rodando com 2 máquinas $L = \{M_1, M_2\}$, processando duas tarefas *j*, escalonadas em um dado tempo *T* em uma fase do *PU-Like* e de acordo com a relação de precedência *W*.

- Caso simples:
  a) $L = \{M1,M2\}$;
  b) $W= \{(A4,A1),(A3,A1),(A2,A1),(A5,A2)\}$;
  c) $T = T_{M_1} + T_{M_2}$;
  d) $F_f =\{1..4\}$.

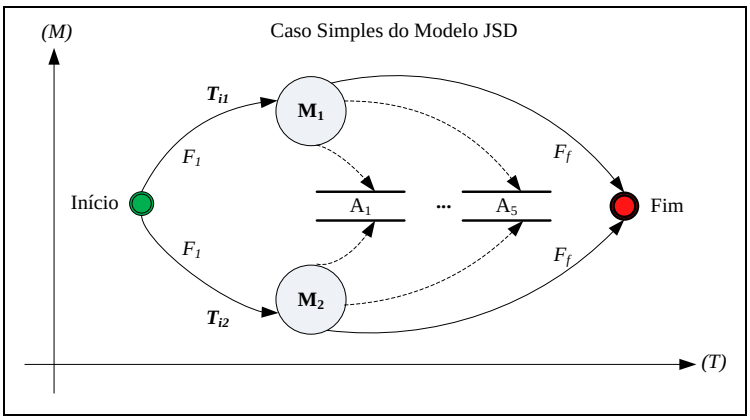


Figura 16: Caso simples do *Job shop* Dinâmico modelado

A Figura apresenta um caso simples do JSD em um *shop* composto por 2 máquinas (M_k), com $k = 1, 2$, processando duas tarefas em um intervalo de tempo T_t. O somatório dos intervalos (int_k) de tempo para M_k são restritos ao orçamento $Orc(k)$ de tempo disponível para uma dada máquina M_k. As linhas tracejadas indicam um fluxo de entregas parciais realizadas pelas nas máquinas em tempo de execução.

4.6.4 Especificação do Caso Geral

No caso geral, *L* representa uma linha de produção do *shop* e M_i são as máquinas de cada LP, com $(i \geq 2 \leq 4)$, e $A = \{A1,A2..A5\}$ é o conjunto de tipos de artefatos a serem produzidos a partir de um conjunto *j* resultante da execução de 26 tarefas. Cada LP deve produzir e entregar $D_A = 100\%$ ao final de cada rodada.

A Figura 17 é uma representação do caso mais geral do modelo *JSD* proposto. Para simplificar e torná-la mais intuitiva, assim como no caso simples, a figura representa uma *LP* com duas máquinas *(M)* processando, um conjunto de Artefatos *(A)*, composto por j_n tarefas, nas 4 fases do *PU-Like*, sabendo-se que um *shop (S)* é dado por conjunto de linhas de produção *(L)* tal qual descrito acima.

- Caso geral:
  a) $L = (M_i, A_j)$;
  b) $W = \{(A4,A1),(A3,A1),(A2,A1),(A5,A2)\}$;
  c) $T = T_{(M_1A_1} + T_{M_2A_1}(F_1) + T_{M_1A_2} + T_{M_2A_2}(F_2) + (\ldots) + T_{M_1A_n} + T_{M_2A_n}(F_4)$;
  d) $F_f = \{1..4\}$.

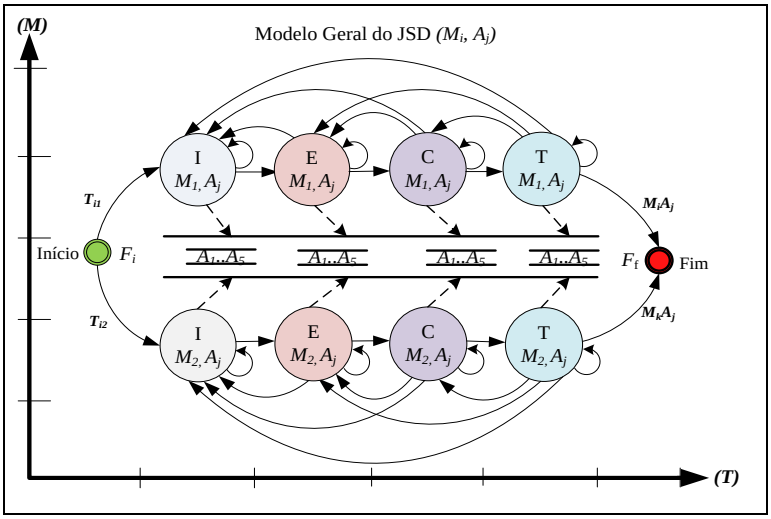


Figura 17: Representação do caso geral do JSD (Adaptado de (Ribeiro *et al.* 2017-e))

No exemplo mostrado, o *shop* contém uma única LP, composta por 2 máquinas M_1,M_2, processando j_n tarefas em um tempo *T*. O *shop* é instanciado com as tarefas obedecendo a relação de precedência (*W*), onde: *A1=Arquitetura, A2 = Banco de Dados, A3 = Codificação, A4 = Documentação e A5 = Teste*. O *Shop* é realimentado com as demais tarefas ao longo do ciclo e sempre em razão de (*W*). Os fluxos de retorno indicam que a tarefa foi reescalonada para reprocessamento após a avaliação. Isto pode ocorrer em

todas as fases do *PU-Like* (*I=Inception*[24]*, E=Elaboração, C=Construção e T=Transição*). As linhas tracejadas indicam um fluxo parcial de entregas de acordo com as fases do *PU-Like*.

O esquema de escalonamento de tarefas por meio de um *Job Shop* Dinâmico (JSD). foi elaborado em conjunto com a TOC e o PDS (*PU-Like*) e foi utilizado para programar as tarefas no shop de acordo com o método do TPC (Tambor, Pulmão e Corda). O JSD permite caracterizar o ambiente produtivo através do mapeamento e escalonamento das tarefas durante todo o ciclo de desenvolvimento e assim representar um ambiente fabril. Além disso, facilita a alocação das tarefas de acordo com a ordem estabelecida pelas disciplinas as fases do *PU-Like*.

4.7 Planejamento da Execução do Experimento

A experimentação foi planejada e executada em conformidade com os preceitos éticos sugeridos por Singer e Vinson, (2002), obedecendo à metodologia, o processo experimental e o processo de software definido, conforme recomenda Easterbrook, *et al*., (2008) e Schull *et al.* 2008, e foi organizado conforme o planejamento estabelecido no arranjo experimental e no *template* de desenvolvimento. Assim, foram definidas um total de três rodadas experimentais, denominadas de experimento 1 (Exp1), experimento 2 (Exp2) e experimento 3 ((Exp1).

Os participantes / alunos foram divididos em grupos de trabalhos os quais representaram as linhas de produção no *shop*. Cada linha de produção executou o mesmo domínio de problema, *i.e*, desenvolveram um software com os mesmos artefatos de entrada e saída, com o mesmo conjunto de tarefas a serem realizadas, com as mesmas regras de negócio e utilizaram as mesmas ferramentas e recursos (listados no Apêndice D). Todas as linhas de produção fizeram uso da metodologia do processo unificado com algumas modificações e ajustes para delinear todas as fases de desenvolvimento e para controlar as entregas em todas as fases. O desenvolvimento seguiu estritamente o modelo do *PU-Like* desenvolvido alinhado com o modelo do processo de execução do experimento. Em cada rodada experimental foi desenvolvido um software para um domínio diferente e com regras de negócios específicas aplicadas aos três experimentos.

[24] *Foi usada a denominação em inglês (I=Inception) da primeira fase do PU-Like na figura 17 para evitar conflito com a nomenclatura usada na terceira fase (C=Construção).*

5 O MÉTODO DE IDENTIFICAÇÃO DE GARGALOS

"A solução de problemas só restaura a normalidade. As oportunidades significam explorar novos caminhos para obter resultados melhores".

Peter Drucker

Neste Capítulo será apresentado o método proposto para identificação de gargalos no PDS. O método foi desenvolvido para suportar a execução do experimento, a análise e a medição dos dados em função do planejamento apresentado no Capítulo 4 e de acordo com a formalização metodológica apresentada na Sessão 4.1.1.

Na primeira parte será apresentado o método genérico para identificação de gargalos, envolvendo: (1) as características do ambiente de execução; (2) o planejamento experimental; (3) o objeto de estudo; e (4) os sujeitos/participantes. Em seguida, o método foi particularizado para atender as especificidades das abordagens *quali-quantitativas* deste estudo.

5.1 O Método de Identificação de Gargalos

O método de identificação de gargalos proposto é estabelecido com base em três indicadores:

1. **Análise do Esforço:** que é o tempo gasto por cada LP para produzir os artefatos;
2. **Análise da Produção:** que refere-se ao percentual produzido e entregue de cada artefato pelas LPs;
3. **Análise da Dificuldade**: que é a análise dos apontamentos feitos no *survey* pelas máquinas em todas as tarefas e ao fim de cada fase do *PU-Like;*

A observação rotineira (*in loco*) das máquinas e consequentemente das LPs no *shop* em tempo de execução é fundamental para a qualidade dos dados destes indicadores.

5.1.1 Direcionamento do Método

As características do PDS, do processo experimental e do tipo de produção (*software*), apontam para dois tipos de restrições: qualitativas e quantitativas, implicando na derivação do método para atender essas duas frentes.

Restrições Qualitativas: as restrições qualitativas estão relacionadas aos problemas etnográficos, uma vez que as máquinas das linhas de produção são constituídas por pessoas. Estas restrições são medidas e tratadas em tempo de execução e se dividem em: (1) Restrições Comportamentais (RC) – de caráter individual, moldada pelos aspectos sócios culturais, pela formação (história de vida do indivíduo) e pela filogenética e (2) Restrições Técnicas (RT) – caracterizadas pelo conhecimento tácito e o conhecimento técnico do indivíduo nas questões relacionadas ao âmbito do PDS.

Restrições Quantitativas: as restrições quantitativas estão associadas aos recursos disponíveis / utilizados e aos resultados produtivos alcançados, que são medidos ao fim de cada fase do *PU-Like* e mais acuradamente ao fim da execução do Processo de Desenvolvimento de Software. É através dos resultados quantitativos que é feito as medições / análises dos dados para identificar os gargalos produtivos no PDS.

Quando não tratadas, tanto as restrições qualitativas quanto as restrições quantitativas podem produzir gargalos produtivos no PDS. Para identificar as restrições "*quali-quanti"* devem ser observados os tópicos a seguir.

5.1.2 Critérios de Análise e Medição

As restrições *"quali-quantitativas"* devem ser medidas a partir dos três indicadores estabelecidos (Esforço, Produção e Dificuldade) e analisadas seguindo estruturalmente os critérios de alocação de recursos, de escalonamento das tarefas no *shop* e os critérios para o monitoramento da execução. Para a aferição dos dados e a análise dos resultados na busca por evidências de gargalos produtivos, pode se fazer uso das seguintes ferramentas: controle estatístico do processo, cartas de controle ou a utilização de uma linha de base. Neste trabalho foi adotada uma linha de base construída a partir das médias (μ) e do desvio padrão (σ) dos resultados colhidos para os três indicadores.

Para a aferição e analise conclusiva dos resultados, deve-se fazer uso de métodos estatísticos para analisar os dados coletados.

5.1.3 Alocação e Exploração dos Recursos

Os cronogramas de execução bem como a alocação dos recursos devem seguir uma agenda pré-definida para este tipo de experimento, como pode ser observada no Apêndice J. A alocação do tempo e a distribuição das tarefas entre as máquinas ficam a cargo de cada

linha de produção obedecendo ao orçamento de horas pré-estabelecido na agenda e o escalonamento inicial, dado pelas tarefas predecessoras. As tarefas podem ser alocadas e realocadas quantas vezes forem necessárias para que o artefato gerado seja contemplado com a completude e a qualidade esperada.

5.1.4 Escalonamento das Tarefas

O escalonamento das tarefas no s*hop* é comum a todas as linhas de produção. A ordem de execução pode variar dependo da estratégia de desenvolvimento adotada pela LP. O controle de quais máquinas irão executar determinadas tarefas, também fica a cargo da linha de produção, podendo uma tarefa ser iniciada em uma máquina e transferida para outra máquina, objetivando a minimização do consumo de tempo ou a maximização da qualidade.

Uma mesma tarefa também pode ser escalonada por mais de uma máquina ao mesmo tempo, além de poder também ser reescalonada quantas vezes for necessário para ser concluída dentro dos quesitos de qualidade estabelecidos.

5.1.5 Monitoramento da Execução

O monitoramento é uma atividade realizada por um avaliador para identificar as restrições qualitativas no *shop*. Neste caso específico, pelo pesquisador/interlocutor. Na indústria, sugere-se que este papel possa ser realizado por um especialista que esteja à frente do projeto e consequentemente do processo de desenvolvimento do software, como: o gerente de projetos, um analista pleno ou um arquiteto de software. Isto é importante porque preserva a estrutura desenvolvida para um ambiente de aprendizagem, no caso de uma aplicação deste método na indústria de software.

5.1.6 A Linha de Base

Devido às características do ambiente produtivo e dos dados coletados neste tipo de estudo, recomenda-se a adoção de uma linha de base para analisar e medir o consumo de recursos por uma tarefa / atividade no *shop* em tempo de execução. A linha de base deve ser gerada em cada fase do PU com o objetivo de encontrar / identificar restrições (*RC* ou *RT*) e assim, aplicar as devidas correções seguindo a orientação do algoritmo da TOC.

A linha de base é uma forma simples e eficiente para medir o desempenho de um *shop*, como pode ser visto em: (1) Sengupta *et al.* (2008), que usa um *Baseline Model*

baseado na performance (Tempo X Capacidade) das máquinas de uma linha de produção para detectar gargalos produtivos; e (2) Genbrugge *et al*. (2006) que propõe o uso de linhas de base como unidade de medida para indicadores estatísticos de desempenho em *grid's* computacionais de alta performance.

Necessariamente, a linha de base deve ser construída a partir dos indicadores quantitativos de: (1) Esforço – que reflete no tempo médio aplicado em cada artefato por cada LP no *shop*; (2) Dificuldade – que são resultados medidos (médias) dos apontamentos feitos pelas máquinas de cada LP na execução de cada atividade; e (3) Produção – que corresponde ao total de artefatos produzidos e entregues por cada LP de acordo com os critérios de avaliação da qualidade e quantidade adotados.

5.1.7 O Modelo BPMN da Linha de Base

Para definir estruturalmente a linha de base e consequentemente o que deve ser medido e avaliado, foi e modelado um processo BPMN para ilustrar graficamente o procedimento, como pode ser observado na Figura 18.

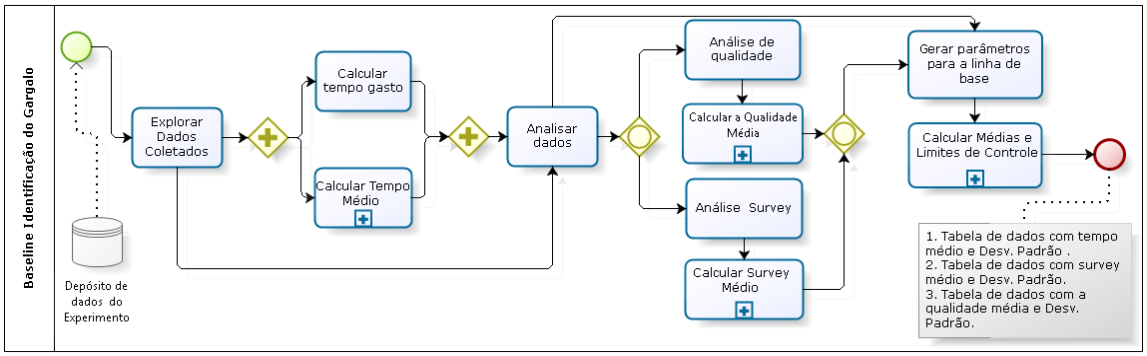


Figura 18: Linha de base para identificação de gargalo no PDS

O *framework* da linha de base é constituído em três partes básicas: (1) exploração dos dados coletados com a execução do subprocesso para calcular o tempo médio gasto; (2) Exploração e análise dos dados coletados com a execução dos subprocessos para calcular média da qualidade e calcular o *survey* médio; e (3) geração dos parâmetros da linha de base com a execução do subprocesso para calcular a média (μ_T, μ_Q, μ_S) da linha de base e desvio padrão $(\sigma_T, \sigma_Q, \sigma_S)$ para determinar os limites de controle de X' e X''.

5.1.8 Passos para Execução do Método

De forma a elucidar o método de identificação de gargalos e sua aplicação, foi elaborado um conjunto de passos mapeados usando a notação BPMN (*Business Process*

Model Notation), conforme apresentado e descrito na Sessão 4.2 e no modelo ilustrativo apresentado no Apêndice I.

Passo 1: Planejamento Experimental – consiste do plano de ação a ser elaborado e adotado de acordo com a agenda e o *template* de execução experimental.

Passo 2: Execução Experimental – este passo é organizado e executado de acordo com os marcos iniciais e finais das fases do *PU-Like* e estendem-se ao processo de execução do projeto experimental de acordo com o plano de ação construído no passo 1.

Passo 3: Análise de Dados – o passo 3, tem por objetivo guiar o processo de análise e avaliação dos dados coletados durante a execução do passo 2 e envolve as duas abordagens (análise qualitativa e análise quantitativa).

Passo 4: Identificação de Gargalos no PDS – a identificação de gargalos é o ponto central do método proposto e tem como propósito mapear os passos a serem seguidos para que os gargalos do PDS possam ser identificados, estudados e tratados.

5.1.9 Aplicação do Método

A aplicação do método tem por objetivo a maximização da produção, ou seja, fazer com as LPs possa produzir e entregar o máximo de artefatos com o menor custo (tempo) e de acordo com os critérios de qualidade definidos no planejamento do experimento, apresentados no Capítulo 4. Este objetivo pode ser alcançado com identificação e tratamento das restrições qualitativas e com a identificação dos gargalos produtivos, feita através da análise quantitativa dos dados sobre os artefatos que restringiram a maximização da produção, ou seja, que impactaram na produção de cada LP no *shop.*

A linha de base deve ser usada como indicador para a avalição quantitativa destes artefatos, desvelando as seguintes observações:

1. **Artefato com alto consumo de tempo:** o artefato está com consumo consideravelmente acima do tempo médio indicado pela linha de base. Neste caso existe a possibilidade da existência de problemas durante a execução/construção do artefato. A tarefa, o processo, e a LP devem ser avaliados concomitante com as métricas de produção e dificuldade (*Survey*).
2. **Artefato com baixo consumo de tempo:** o artefato apresenta consumo abaixo da média estipulada pela linha de base. Implica que o artefato não foi construído ou

não foi totalmente concluído. A entrega completa do artefato deve ser avaliada, observando também as medidas de qualidade e dificuldade. Neste caso, é importante ainda relatar os motivos que levaram a LP a não entregar o artefato ou efetuar a entrega incompleta. Isto é importante para que o processo possa ser ajustado.

3. **Artefato com baixa qualidade:** o artefato está com a qualidade abaixo da média fixada pela linha de base. Existe o indicativo de que o artefato possui erros de construto ou não foi entregue com completude: Se a baixa qualidade foi causada pela entrega incompleta, deve-se adotar os procedimentos do item 2. Caso sejam problemas de construto, deve-se avaliar a dificuldade relatada pelas máquinas para identificar a origem do problema.

4. **Artefato com dificuldade (*Survey)* alta:** o artefato possui um índice de dificuldade acima da média fixada pela linha de base. Isto indica que a LP teve dificuldade na interpretação e no entendimento do que deveria ser feito para realizar a entrega. Outro indicativo é a falta de qualificação técnica das máquinas para desenvolver as tarefas associadas ao artefato.

O método apresentado nesta Sessão corresponde a um modelo geral, o qual pode ser aplicado em qualquer ambiente de desenvolvimento de software. No entanto, para que o método atenda a particularidade deste estudo, e possa ser aplicado em ambiente de ensino de Engenharia de Software, foi necessário realizar alguns ajustes em função das características ambientais, do PDS adotado e dos times de desenvolvimento. Os aspectos culturais e regionais onde o PDS foi estabelecido, bem como a experiência dos participantes e o conhecimento técnico e tácito em desenvolvimento de software regido por um processo, também foram considerados.

Desta forma, as sessões seguintes irão apresentar a particularização do método proposto para as duas abordagens (*quali-quanti*), obedecendo ao plano experimental elaborado e aos critérios estabelecidos para a execução do processo e para a análise de dados para identificar os gargalos no PDS adotado nesta pesquisa.

5.2 Método de Identificação de Gargalos: Abordagem Qualitativa

A identificação de gargalos no PDS determina que o processo seja acompanhado e medido qualitativamente em tempo real. Isto é importante para que as restrições

qualitativas possam ser identificadas e tratadas / ajustadas, evitando assim o surgimento de gargalos produtivos. A análise qualitativa desenvolvida e aplicada neste estudo envolve os itens 5.1.3 e 5.1.5 do método de identificação de gargalos e é feito por meio dos seguintes passos:

1. **Observação e Monitoramento:** acompanhamento e controle dos times de desenvolvimento em tempo de execução. Sempre que um problema qualitativo for identificado, uma reunião com o time de desenvolvimento ou com o participante que desenvolveu tal tarefa deve ser disparada.
2. **Reuniões com os times de desenvolvimento:** reuniões previamente agendadas para avaliação do progresso do projeto.
3. **Reuniões pontuais com determinadas equipes:** reuniões específicas com as equipes que apresentaram problemas anotados nos passos 1 e 2.
4. **Reuniões pontuais com determinados membros de equipe:** reuniões específicas e restritas com participantes que tenham apresentado RTs e/ou RCs;
5. **Leitura do Livro de Registos do Experimento:** Leitura das anotações registradas no *"log book"* ao fim de cada fase do *PU-Like*;
6. **Análise parcial de dados em tempo de execução:** o objetivo da análise parcial é verificar a corretude e a completude do software de acordo com os resultados produtivos, como: identificação de um modelo mal construído, um código mal feito ou um banco de dados com problemas estruturais. A tabulação e a análise destes dados são feitas ao fim de cada fase do *PU-Like.*

Estes pontos definem o método de identificação de restrições "*quali"*, o qual estabelece ainda as seguintes abordagens para tratamento das restrições encontradas.

1. **Aulas expositivas:** As aulas expositivas são ações preventivas, ou seja, são aplicadas para nivelamento do conhecimento e antes da execução de uma tarefa ou etapa de desenvolvimento; Na indústria, as aulas expositivas podem ser substituídas por treinamentos ou instruções ministrados por profissionais mais qualificados e experientes.
2. **Visita *in loco*:** a visita *in loco* é uma visita feita a uma linha de produção *on site* e em tempo de execução. A visita *in loco* é feita tanto para identificar e mapear um problema/restrição, quanto para tratá-lo;

4. **Reunião de alinhamento:** trata-se de uma reunião com as máquinas da linha de produção onde o problema foi identificado. O objetivo é sempre tratar a restrição e não apontar ou acusar a origem do problema;
5. ***Feedback* expositivo:** O feedback é uma exposição do problema juntamente com uma orientação para um possível tratamento e deve ser apresentado para todas as linhas de produção/máquinas. A ideia é evitar que um mesmo problema aconteça nas demais LPs, antecipando o tratamento pró-ativamente.
6. ***Feedback* escrito:** É um registro escrito em cada fase do *PU-Like* que deve ser submetido a LP que apresentou algum tipo de restrição ou problemas na construção do software. Isto é importante, pois evita ruídos no *shop*, uma vez que os problemas ficam restritos e centrados em cada LP.
7. **Uso da TOC:** Aplicação dos 5 passos da TOC para avaliação do processo com o objetivo de eliminar as restrições encontradas.

5.2.1 Identificação e Tratamento das Restrições Qualitativas com a TOC

A fundamentação lógica da TOC está baseada nos cinco passos que determinam a concentração do esforço para melhorar a capacidade produtiva do *shop*. O TPC é um método de programação de produção que foi desenvolvido para dar suporte a esta fundamentação lógica e tem por finalidade "proteger" um Recurso com Restrição de Capacidade (RRC). A identificação e o tratamento das restrições qualitativas proposta neste trabalho passa por este arranjo para melhorar a capacidade produtiva das máquinas, principalmente quando aplicado aos passos 1 e 4 da TOC.

Desta forma, o tratamento é feito mediante o apontamento (identificação) dos problemas de construto e seguem rigorosamente os 5 passos da TOC.

1. **Identificação** e avaliação da restrição;
2. **Explorar** a restrição para transformar/modificar este "elo fraco" da corrente;
3. **Subordinar** os demais recursos à restrição, como realocar a tarefa em outras máquinas;
4. **Elevar** a restrição, que neste caso implica em alternativas para reduzir o impacto da restrição;
5. **Evitar a inércia** ao reavaliar o processo, as máquinas, as restrições encontradas e o risco de surgimento de novas restrições.

A Seção a seguir irá fundamentar a aplicação do item 7 (5 passos da TOC) concomitante ao TPC (Tambor, Pulmão e Corda).

5.2.2 Implementação da Solução: Aplicação dos passos da TOC com o TPC

A solução proposta deve necessariamente e oportunamente executar os 5 passos da TOC e envolver a análise dos resultados de acordo com a metodologia desenvolvida e o processo modelado. O TPC deve ser usado para sincronizar o escalonamento de tarefas com os objetivos/metas de desenvolvimento. Os tópicos a seguir definem e constrói a solução adotada.

5.2.3 Identificação das Restrições do Sistema

Para identificar as restrições, o primeiro passo é identificar o tambor. O tambor é tipicamente o recurso ou o centro de trabalho mais carregado do *shop* (Woeppel, 2008). Isto pode ser feito analisando a demanda, a capacidade produtiva das LPs e o tempo de execução das tarefas no *shop* por cada LP.

A identificação do tambor geralmente é feita antes da programação da produção, no caso de uma linha de produção fabril, como pode ser visto em (Yan *et al.* 2010). Aqui, a identificação foi feita durante o planejamento e confirmado durante execução da primeira fase com a análise parcial dos resultados.

O tambor identificado para os *shops* dos experimentos (*Exp1, Exp2 e Exp3*) é o tempo disponível para cada LP, dividido entre as 4 fases do *PU-Like* e organizado de acordo com a relação atividade-tempo estabelecida pelo cronograma. O tempo é o recurso que define o total de unidades que deve ser entregue em cada fase, marcando assim o ritmo da produção de acordo com a demanda produtiva e a capacidade das LPs.

A capacidade é mensurada ao fim de cada entrega, através da análise parcial dos resultados. De certa forma, isto remete a um problema, pois o ritmo do tambor não pode ser ajustado em função da capacidade antes da execução, ou seja, antes da capacidade ser medida.

Isso geralmente causa impactos na produção da segunda fase, como pode ser observado na Figura 19 (a) e (b) apresenta o escalonamento das tarefas e a capacidade medida das LPs do *shop* do *Exp.1*, com impactos significantes na segunda fase do processo.

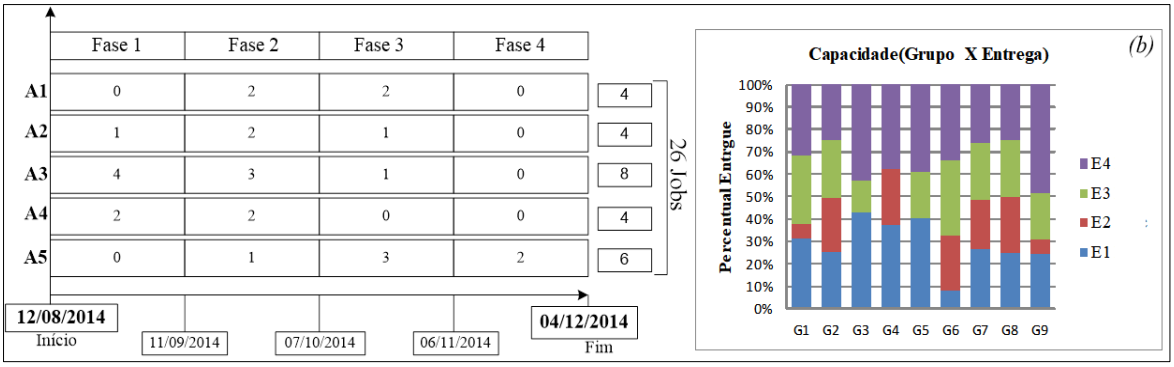


Figura 19: Identificação do Tambor: (a) Demanda de tarefas e (b) Capacidade produtiva

A Figura 19 (a) apresenta as 26 tarefas escalonadas no *shop* em consonância com a agenda e as fases do *PU-Like* e de acordo com a precedência. O Conjunto $A=\{A_1,..,A_5\}$ refere-se a um composto de artefatos, separados por tipo. A Figura 19 (b) apresenta o percentual do trabalho entregue por cada linha de produção[25] em cada entrega. Cada entrega (*E1, E2, E3 e E4*) representa a produção das LPs de acordo com as 4 fases do *PU-Like*.

Estes problemas são mitigados nas outras fases, uma vez que já é possível mapear a capacidade produtiva das LPs e ajustar o ritmo do tambor. Isso foi feito ajustando / flexibilizando o cronograma de entregas. Na indústria, esta flexibilização pode ser feita ajustando as variáveis de folga do contingenciamento, por exemplo.

5.2.4 Explorando as Restrições

Uma vez identificado o tambor, é preciso explorar as restrições e detectar quais os recursos (Tempo e Máquinas) estão disponíveis e o que é preciso fazer para atender a demanda ou suavizar o impacto na LP.

Portanto, deve-se realizar um cronograma ajustado em função da demanda e da capacidade das máquinas. Em conjunto com o cronograma, deve ainda ser elaborado o esquema de reprogramação da produção, observando as tarefas predecessoras e a corrente crítica. O objetivo é diminuir a carga nos recursos com restrição de capacidade (RRC) e com isso satisfazer as necessidades do *shop*, distribuindo a demanda em função da capacidade das máquinas.

A Figura 20 apresenta a capacidade produtiva do *shop* do experimento 1.

[25] *Dados extraídos do shop 1 / Entrega 1.*

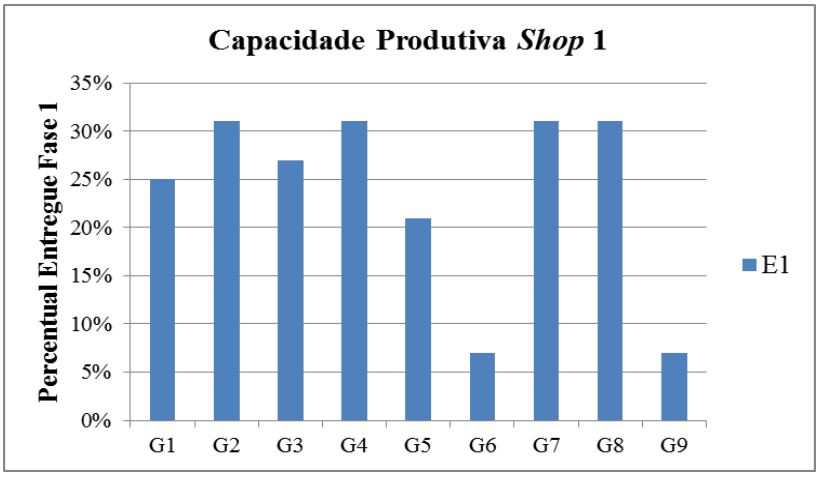


Figura 20: Capacidade produtiva do *shop* 1

A Figura 20 apresenta o percentual dos artefatos entegus na fase 1. A demanda de 35% refere-se ao total dos artefatos demandados para a fase. Neste exemplo, as LPs *6 e 9* apresentaram um rendimento muito abaixo do esperado.

Diferentemente da manufatura, este é um procedimento de maior complexidade de ser realizado devido às características individuais das máquinas (pessoas), ou no caso dos times de desenvolvimento na indústria de software. Além disso, as máquinas podem oscilar em relação à capacidade produtiva, aumentando ou diminuído a vazão de artefatos. Neste caso a observação deve ser efetiva e continua com o intuito de dirimir os surgimentos de novas restrições locais que possam perturbar toda a linha de produção.

Neste trabalho, procuramos efetuar uma redistribuição de tarefas quando, detectado a continuidade do problema, alocando nas máquinas com RRC as tarefas menos complexas ou que não possuíam tarefas predecessoras. Explorando assim a restrição e evitando que Recursos sem Restrição de Capacidade (RRsC) sofressem maior impacto ou sobrecarga.

Para simplificar, tomaremos como exemplo único a reprogramação das tarefas da LP9 do Experimento 1 apresentado na Tabela 16, onde os valores fechados com parênteses indicam a nova distribuição das tarefas.

Tabela 16: Reescalonamento das tarefas

Reescalonamento de Tarefas (LP 9)			
Job	***Job's Reescalonado nas Máquinas (M)***		
1	3(1)	2(3)	3 (3)
2	1 (1)	3(3)	2 (2)
3	2 (3)	1(1)	3 (2)

A Figura 21 representa o mapeamento da redistribuição das tarefas entre as máquinas da LP9. A ideia é explorar os RRC e reprogramar as tarefas para reduzir os impactos já causados na produção.

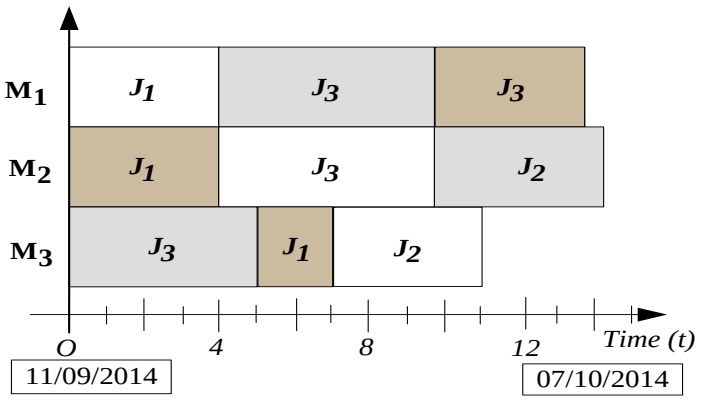


Figura 21: Exemplo de reprogramação das tarefas demandas

A reprogramação das tarefas efetuadas na LP (G9) do *Shop* 1, representado na Figura 20, acontece após a análise dos dados e verificação da capacidade dado pelo percentual entregue. Neste cenário, um *job* é compartilhado entre duas máquinas (RsRC) em instâncias de tempo iguais ou diferentes. As datas de início e fim indicam que a reprogramação acontece na fase seguinte do *PU-Like*. Desta forma, além das tarefas reprogramadas da fase 1, a LP deve produzir as novas tarefas inseridas na fase 2 conforme a programação original, salvo no caso de tarefas não predecessoras, que foram ajustadas ou reprogramadas. Inevitavelmente, a reprogramação irá causar perturbações na produção da segunda fase. Isto implica que o tambor (se houver possibilidade), deve ser novamente ajustado o monitoramento nestas máquinas devem ser intensificados.

5.2.5 O TPC: Subordinando e Sincronizando as Restrições

O pulmão é qualquer contingência de recurso para proteger o tambor. Ele compensa a variação do processo, e torna o *DBR Schedule* muito estável e imune à maioria dos problemas (Woeppel, 2010).

Avocamos que o pulmão é o conjunto de tarefas a serem processadas, mais os recursos a elas associados. Principalmente o recurso de tempo das tarefas não predecessoras ou precursoras, que possuem folgas ou flexibilidade para ajustes da programação do *shop*. Desta forma, estas tarefas são usadas para proporcionar uma ressincronização do trabalho através da subordinação das mesmas às maquinas com recursos disponíveis (RsRC).

O Tambor é o tempo disponível para cada LP do *shop*. O orçamento de tempo é estabelecido pelo cronograma e estabelece o início e o fim do processamento das tarefas em razão das etapas do *PU-Like.*

A corda é o mecanismo de sincronização da produção do *Shop*. Ela auxilia na programação liberando as tarefas de acordo com o ritmo do tambor e a disponibilidade do pulmão. Definiu-se a corda nesta pesquisa como as interações e micro interações existente entre as tarefas e as atividades do PDS, dado pelo *PU-like* e a ordem de precedência.

Durante a execução dos três experimentos, as tarefas sem restrição de precedência e seus recursos foram usados para ajustar a programação, alocando e "desalocando-as" no *shop.* Tirando com isso, estas tarefas dos RsRC e transferindo-as aos RRC em um processo de subordinação. A liberação das tarefas é conectada (amarrada) ao tambor em função da disponibilidade do pulmão. Desta forma as tarefas são liberadas na mesma proporção em que são produzidas e entregues em um processo de subordinação e sincronização, como pode ser observado na Figura 22 (a) e (b) e na Figura 23.

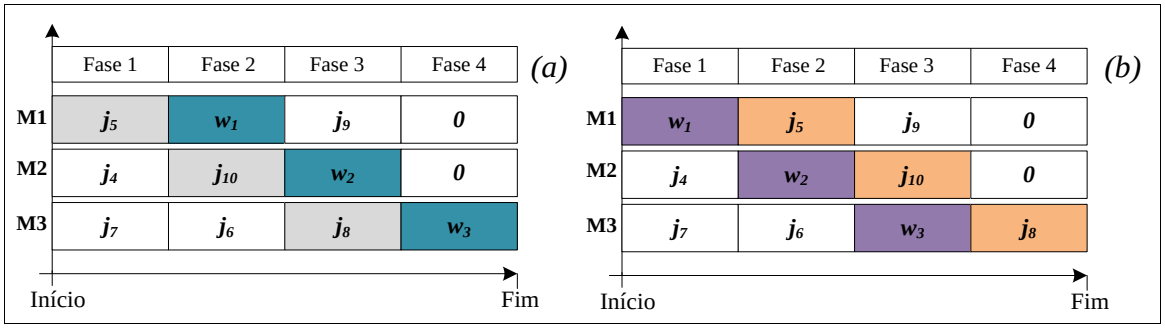


Figura 22: (a) (b) Subordinação das tarefas

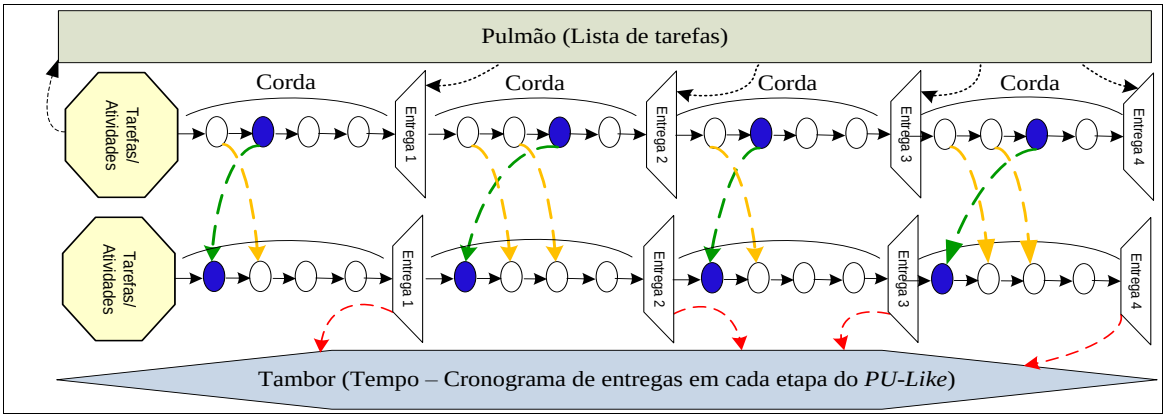


Figura 23: Sincronização das tarefas por meio do TPC

A Figura 22 (a) apresenta o escalonamento de 10 *jobs*, onde: *w1, w2 e w3* são tarefas predecessoras escalonadas em máquinas RRC, em determinadas fases do processo.

A máquina 1 possui restrição de capacidade com $w_1 \rightarrow$ *fase 2*. O *buffer* libera a tarefa w_1 para a *fase 1* e a tarefa j_4 é realocada na fase seguinte. O mesmo acontece com w_2 e w_3. A Figura 22 (b) apresenta o novo arranjo do *shop* com as tarefas não-predecessoras alocadas em função da capacidade das máquinas em suas respectivas fases. Na Figura 23, temos o esquema do TPC com a troca das tarefas RRC predecessoras (elipses azul) com tarefas não-predecessoras e sem restrição de recurso (elipses brancas). As linhas verdes e amarelas representam o processo de alocação e realocação de acordo com a liberação do pulmão no ritmo do tambor, sincronizados pela corda, dado pelas interações do *PU-Like*.

O pulmão tem o efeito de adicionar capacidade nas LPs e eliminar a saturação das tarefas programadas, permitindo assim um escalonamento menos robusto no *shop* e consequentemente suavizando o trabalho das LPs. Isto é importante diante das características dinâmicas das LPs e do *shop*.

5.2.6 Elevando a Restrição

Elevar uma restrição é aumentar a capacidade da restrição. Como as restrições neste trabalho estão intrinsicamente associadas às máquinas (pessoas), o processo de elevação da capacidade das restrições foi feito aplicando orientações e treinamentos específicos em determinadas frentes de trabalho, como: aulas expositivas e especificas sobre o tema ou ferramenta[26] e atribuição de novas responsabilidades para explorar a capacidade produtiva da restrição, como: (1) alocação das tarefas não predecessoras à restrição; (2) atribuição de revisão das tarefas, como modelos e depuração de código, e (3) atribuição de tarefas secundárias, necessárias para a produção dos artefatos pela LP, como instalação e configuração de ferramentas.

5.2.7 Retornar: Evitando a Inércia

O algoritmo da TOC impõe que seja estabelecido um *loop* contínuo para evitar que a inércia se instale no sistema produtivo e com isso surjam novas restrições ou restrições que haviam sido tratadas volte a impactar a produção.

Especificamente neste trabalho, este passo foi feito através do monitoramento contínuo das LPs de cada *shop*.

[26]*Em função do calendário acadêmico e do cumprimento da ementa da disciplina de Engenharia de Software, o treinamento em algumas ferramentas e ambientes de desenvolvimento não foram aplicados. Como por exemplo: IDE Eclipse e o MySQL e o GIT.*

Esta tarefa foi atribuída estritamente ao pesquisador, com o acompanhamento rotineiro *in loco* e o gerenciamento do processo.

Uma observação importante é que as técnicas empregadas nesta pesquisa, para identificação e tratamento de restrições qualitativas, também pode ser aplicada em outros domínios de problemas e em outros ambientes, inclusive na indústria, pois possuem caráter genérico e as restrições qualitativas, de forma geral, são associadas às pessoas/máquinas e aos times de desenvolvimento (linhas de produção).

5.3 Método de Identificação de Gargalos: Abordagem Quantitativa

Nesta pesquisa, assumimos que um gargalo no processo produtivo é qualquer elemento restritivo que possa impedir a maximização da produção, dado pelo alto consumo de recursos e a baixa qualidade do artefato produzido.

Consideramos ainda que se um artefato consumiu recursos além do necessário para sua conclusão, então existe um indicativo da existência de restrições associadas à tarefa de origem. E isto possivelmente irá impactar no desenvolvimento completo e na qualidade de outro artefato no *shop*, pois haverá menos recursos para serem aplicados entre as tarefas no *shop*, devido à agenda de tempo e o número de máquinas por LP serem pré-fixados e iguais para todas as LPs do *Shop*. Como consequência, todas as fases subsequentes do *PU-Like* serão afetadas, inclusive as tarefas do *buffer* que estão na fila de espera para serem escalononadas, já que o orçamento de horas e o número de máquinas de cada linha de produção são previamente definidos e fixados.

O método atribuído para a identificação de gargalos está necessariamente alinhado aos critérios estabelecidos para detectar as restrições qualitativas apresentado na Seção 5.4, e pelo método de identificação de gargalos quantitativos, através da construção de uma linha de base, para medir o desempenho de cada LP no *shop* em tempo de execução.

5.3.1 Construção da Linha de Base

Este trabalho usou três indicadores para medir a produção das LPs nos *shops* (Esforço, Produção e Dificuldade). Estes mesmos indicadores foram usados para a construção da linha de base [27]através do cálculo das médias (μ_T, μ_P, μ_E) e assim determinar os seguintes parâmetros de medição: (1) $\boldsymbol{\mu_T}$: Média do Esforço / Tempo: Indica o tempo

[27] O *Apêndice G apresenta a linha de base construída / calculada para os Shops 2 e 3.*

médio consumido por artefato no *shop*; (2) $\boldsymbol{\mu_P}$: Média da Produção: é a média da qualidade aferida (corretude e completude) a partir da pontuação aplicada, conforme especificado nas Seções 4.4.14 e 4.4.15; e (3) $\boldsymbol{\mu_E}$: Média da Dificuldade:[28] a média do *Survey* aplicado para medir as dificuldades de cada máquina em realizar uma determinada tarefa dentro de uma fase do *PU-Like*. A construção da linha de base utilizou ainda o desvio padrão *(σ_T, σ_Q, σ_E)* das médias amostrais para determinar a dispersão (variação) dos dados nos três *shops*.

5.3.2 Cálculo do Esforço Médio (μ_T)

O esforço médio é calculado para cada artefato em uma dada fase do processo, e é obtido pela equação (1).

$$\mu_T = \left(\frac{T_{M_1\,A_1} + T_{M_1\,A_2} + \ldots + T_{M_nA_n}}{L_{\mathrm{M}}}\right) = \left(\frac{1}{M}\right)\left(\frac{1}{26}\right)\sum_{j=1}^{26} T_{M_iA_j} \qquad (1)$$

Onde: μ_T é o tempo médio gasto por cada máquina da Linha de Produção *(L)*; *T* é o tempo total consumido pela máquina em cada artefato; M_i = é o número de máquinas por LP, sendo $M_i = \{\geq 2, \leq 4\}$; $j=26$ é o número total de tarefas escalonadas o *shop*; A_i são os artefatos entregues por máquina e L_M é o número de máquinas ativas na linhas de produção.

5.3.3 Cálculo da Produção Média (μ_P)

A produção média *(μ_P)* é obtida a partir da equação (2).

$$\mu P = \left(\frac{(Aval(A_1) + Aval(A_2) + .. + Aval(A_n))}{n}\right) = \left(\frac{\sum_{i=1}^{n}\left(Aval\ (A_j)\right)}{n}\right) \qquad (2)$$

Onde: μ_P é a média da produção obtida sobre os tipos de artefatos entregues por cada Linha de Produção (*L*); *Aval*(A_i) é a avaliação do artefato A_i de acordo com os pontos atribuídos no *rank* de qualidade estabelecido nas seções 4.4.14 e 4.4.15 e n o número de máquinas da LP.

5.3.4 Cálculo da Dificuldade Média (μ_E) Percebida

A dificuldade média *(μ_E)* é obtido a partir da equação (3).

[28] *A Dificuldade será representada pelo termo E (Entrevista). Isto porque o termo D, corresponde a entrega (Delivery) e S (Survey) corresponde aos shops.*

$$\mu_E = \left(\frac{Dif(M_1) + Dif(M_2) + \cdots Dif(M_n)}{L_M} \right) = \left(\frac{\sum_{i=1}^{n} Dif(M_i))}{L_M} \right) \quad (3)$$

Onde: μ_E é a média ponderada da dificuldade (*Dif*) percebida pelas máquinas de uma LP obtida pelos resultados das entrevistas *(E)* usando os pesos atribuídos de acordo com as notas aplicadas em cada tarefa durante a entrevista (*survey*), como foi definido na Seção 4.4.6 e M_i corresponde às máquinas de uma dada LP.

5.4 Especificação do Método de Identificação de Gargalos

A identificação de gargalos envolve todo o processo produtivo. Isto é uma imposição do tipo de estudo, do ambiente e do PDS. O método será apresentado na forma de algoritmo, e para torná-lo mais compreensível, ele foi dividido em três partes: (1) preparação; (2) medição; e (3) avaliação e saídas, como apresentado nas seções seguintes.

5.4.1 Parte 1: Preparação

A primeira parte do algoritmo define os passos iniciais para execução do processo, e envolve o planejamento, o ambiente, o PDS e o processo de execução do experimento, bem como a definição das entradas e saídas.

Algoritmo do Método - Parte 1: Preparação

Entradas: *Processo (p), Tarefas (j), Linhas de Produção (L), Shop (s), PU fases (f), (w) Predecessoras.*
Saída: *Tarefas escalonadas.*

```
Início
    Analisar e Avaliar Processo(p);          // Ajuste do processo conforme o domínio e ambiente.
    Definir Agenda (a);                      // Definição do cronograma e distribuição do tempo.
    Preparar Shop(s);                        // Definir equipes (LPs) e máquinas.
    Definir Linhas de Produção (L);          // Alocar máquinas nas LPs.
        lp[Mi]=(⌊2⌋, ⌈4⌉)                    // Atribuição de maquinas as LPs (≥2, ≤4).
    PU fases ← agenda;                       // Programas fases de acordo com agenda.
    Definir tarefas (j) para cada Fase (f);  // Definir as tarefas de cada fase do PU-Like.
        W ← 0; j ← 0; f ← 0;                 // Inicialização das tarefas w, j e fase (f) do PU.
    Para j ← 1 até 26                        // Rotina de preparação para o escalonamento.
      Definir tarefas predecessoras (w);     // Mapeamento das tarefas iniciais (predecessoras).
      Mapear a corrente crítica;             // Mapeamento da corrente crítica.
     Faça                                    // Rotina para escalonar tarefas no shop.
       Para cada L[Mi] no shop(s) faça       // Rotina para escalonar tarefas por fase do PU.
         fase f([i]);                        // Identificação das fases do PU.
         Se Fase (f) = 1 e tarefa = w;       // Iniciação do escalonamento fase=1 do PU.
           Então Escalonar tarefa (w) no shop (s);   // Programação das tarefas predecessoras (w).
           Senão Escalonar tarefas (j) no shop (s);  // Programação das tarefas predecessoras (j).
         Fim Se
       Fim para
     Até que fase (f) = 4;
    Fim Para
Fim
```

5.4.2 Parte 2: Computação e Parametrização

A segunda parte do algoritmo realiza os cálculos dos indicadores (Tempo, Produção e Dificuldade) em dois estágios: (1) cálculo (soma) do tempo consumido em cada tarefa / artefato pelas LPs, cálculo da dificuldade *(survey)*, cálculo da produção; e (2) cálculo das médias dos parâmetros *(μ_T, μ_Q, μ_E)* para serem usados na composição da linha de base.

Os dados sobre a Produção e a Dificuldade (*Survey)* dependem de uma execução externa. Assim, esta segunda parte do algoritmo apresentará apenas a função para cálculo, assumindo que os dados já foram colhidos, analisados e inseridos.

Algoritmo do Método - Parte 2: Computação e Parametrização

Entradas: *Tarefas (j), Fases(f), Tempo (T), Produção (P), Dificuldade (E) Linhas de Produção (lp), Shop (s), PU.*
Saídas: *Esforço Total, Esforço Médio, Produção Média, Dificuldade Média.*

Início
Calcular o tempo total consumido ( );
T ← 0; P ← 0; E ← 0; // *Inicialização das variáveis: Tempo, Qualidade e Survey.*
Para *j ← 1 até 26* ***faça*** // *Rotina para somar/acumular o tempo gasto.*
Faça //*Rotina para navegar nas fases do PU.*
Para *L[M_i] (≥2, ≤ 4)* ***faça*** // *Rotina para somar o tempo total por maquina/tarefa.*
Tarefa (W) ← Tempo [f]; //*Atribuição do tempo das tarefas predecessoras.*
Tarefa (j) ← Tempo [f]; //*Atribuição do tempo das tarefas não predecessoras.*
Tempo_total ← Tempo(tarefa [W],[j]); //*Tempo gasto nas tarefa (w) e (j).*
T ← Tempo_total; // *Tempo total em T.*
Fim Para;
j ← j+1; // *Incrementador*
Até que *fase (f) = 4;*
T ← T + tarefas (t); // *Tempo total acumulado sobre (t) tarefas.*
Fim Para;
μ_T ← 0; μ_P ← 0; μ_E ← 0 //*Inicialização das médias*
Calcular parâmetros (μ_T, μ_P, μ_E); // *função para cálculo das médias.*
Para *fase (1) ← 1 até 4* ***faça*** // *Rotina para percorrer as fases do PU.*
Artefato (A) ← tarefa (w + j) // *Agrupamento das tarefas w e j.*
Para *Artefato (A) ← 1 até 26 faça;* // *Rotina para percorrer as 26 tarefas.*
Calcular tempo médio ( ); // *Função para calcular o tempo médio.*
$Tempo\ Médio = \left(\frac{\sum T_{M[i]+A[j]}}{L_M}\right)$; // *Cálculo da média de tempo.*
μ_T ← μ_T + Tempo Médio; // *Atribuição do tempo médio aferido.*
Calcular a Produção média ( ); //*Função para calcular (μ_P).*
$Produção\ Média = \left(\frac{\sum_{i=1}^{n}\left(Aval\ (A_j)\right)}{n}\right)$ // *Cálculo da Produção Média(μ_P).*
μ_P ← μ_P + Produção Média; //*Atribuição da qualidade média aferida.*
Calcular o Dificuldade Média ( ); //*Função para calcular a dificuldade média.*
$Dificuldade\ Média = \left(\frac{\sum_{i=1}^{n} Dif(M_i))}{L_M}\right)$; // *Cálculo do μ_E.*
μ_E ← μ_E + Dificuldade Média; //*Atribuição da μ_E aferido.*
Fim Para;
Retorne *(μ_T, μ_P, μ_E);* // *Saída / Retorno das médias.*
Fim Para;
Fim.

5.4.3 Parte 3: Mecanismos de Avaliação e Saída

A terceira parte do algoritmo estabelece a linha de base com os critérios para avaliação e apresentação dos resultados a partir dos parâmetros de medição. O desvio padrão das médias *(σ_T, σ_Q, σ_E)* é calculado para delimitar a região de análise e direcionar a busca por evidências.

Algoritmo do Método - Parte 3: Avaliação e saída

Entradas: *Média (μ_T, μ_Q, μ_E), Shop(s), Grupo(g), Linhas de Produção (lp), Tarefas(t), Artefatos(A), Fases(f).*
Saídas: *Linha de Base (μ_T, μ_P, μ_E), Desvio Padrão (σ_T, σ_P, σ_E), resultados (r).*

```
Início
    σT ← 0; σP ← 0; σE ← 0;                    // Inicialização dos parâmetros dvp.
    Calcular o desvio padrão (σT, σP, σE );     // Função para cálculo do desvio padrão.
  Para (μT, μP, μE) faça                        // Rotina para calcular o desvio padrão.
    Calcular desvio de μT ( );                  // Função para calcular o dvp da média de tempo.
    dvpT ← SQRT (1/n Σ1n μT (xi− μT )2 / j=26);  // Cálculo do desvio do tempo médio.
        σT ← σT + dvpT;                         // Atribuição do dvp aferido.
    Calcular desvio de μP ( );                  //Função para calcular o desvio da produção média.
    dvpQ ← SQRT (1/n Σ1n μQ (xi− μQ )2 / j=26);  // Cálculo do dvp da produção média.
        σQ ← σP+ dvpQ;                          // Atribuição da dvp aferido.
    Calcular o Survey Médio ( );                // Função para calcular o desvio da dificuldade médio.
    dvpE ← SQRT (1/n Σ1n μE (xi− μE)2 / j=26);   // Cálculo do desvio da dificuldade médio.
        σE ← σE + dvpE;                         // Atribuição da dvp aferido.
  Fim Para;
        Retorne (σT, σP, σE);                   // Saída / retorno dos dvp obtido.
      Para (μT, μP, μE) faça
        Definir a linha de base ( );            // Função para criação da linha de base
        Baseline ← (μT, μP, μE);                // Atribuição das médias para linha de base.
      Fim para;
    Para lp ← 1 até n faça                      // Rotina para percorrer as linhas de produção.
      Para j← 1 até 26 faça                     // Rotina para comprara tempo(T) das tarefas na baseline.
        Se (T[j], P[j], E[j]) < Baseline (μ) e (T[j], P[j], E[j])  // Condição de avaliação1.
          então Avaliar Artefato ( Gargalo);    // Artefato é um potencial gargalo.
            Senão descartar ( );                // Artefato não é gargalo.
          Se (T[j], P[j], E[j]) > Baseline (μ) e (T[j], P[j], E[j])  // Condição de Avaliação 2.
            então Avaliar Artefato (Gargalo );  // Artefato é um potencial gargalo.
            Senão descartar ( );                // Artefato não eh gargalo.
          Fim se;
        Fim se;
          Lista Gargalo ← Gargalo + Gargalo;    // Totalização de gargalos encontrados.
          J ← j + 1;                            // Incremento de j.
      Fim para;
        lp ← lp + 1;                            // Incremento para percorrer as lps.
    Fim para;
        Gerar análise descritiva ( );           // Gerar analise descritiva para cada um dos parâmetros.
        Definir estatística ( );                // Definir a estatística para analise dos dados.
        Gerar análise estatística dos dados ( );  //Aplicar os testes estáticos
  Retorne (Lista de Gargalo);                   // Saída - Resultado: lista de gargalo(s) identificados.
Fim.
```

Por conveniência alguns laços e passos do algoritmo foram suprimidos ou agrupados em uma única rotina já que a intenção é apresentar o método e enfatizar os procedimentos utilizados para a computação e avaliação dos dados.

O passo seguinte à execução do algoritmo é a definição dos métodos estatísticos a serem aplicados para a análise dos dados, *i.e*, para aferir se as evidências encontradas são estatisticamente significantes ao ponto de serem apontadas e categorizadas como gargalos produtivos no PDS. Este passo é importante para que se possa responder as questões levantadas e tipificar o(s) gargalo(s) identificado(s).

6 AVALIAÇÃO QUALITATIVA

"Deixe o futuro dizer a verdade, e avaliar cada um de acordo com seus trabalhos e suas conquistas."

Nikola Tesla

A pesquisa foi fundamentada e baseada nas questões levantadas durante a elaboração do plano experimental. As questões determinam o foco e os objetivos do experimento e modelam as hipóteses construídas para validar ou refutar a teoria a partir dos dados da pesquisa. Tanto as questões quanto as hipóteses apontadas neste estudo foram apresentadas na Seção 1.6. Os métodos qualitativos concentram-se em descobrir e compreender as experiências, perspectivas e os pensamentos dos participantes (Harwell, 2011). Em outras palavras, Harwell (2001) afirma que a pesquisa qualitativa explora o significado, o propósito ou a realidade de uma atividade situada que consiste em um conjunto de práticas interpretativas e materiais que tornam um mundo visível.

A avaliação qualitativa dos dados deve seguir um conjunto de critérios de validade, já apontados por Newman e Benz (1998), como: neutralidade, engajamento prolongado *on-site*, observação persistente e consistente; interrogação/inquérito de pares (*peer debriefing*), triangulação, checagem de membros, material referencial, relacionamentos estruturais, amostragem teórica, generalizabilidade, credibilidade, legitimidade, resignação e auditoria de rastreamento.

A avaliação qualitativa empregada nesta pesquisa obedece aos critérios de validade com o emprego da neutralidade, da observação, do engajamento duradouro/prolongado no local *(on-site)* e principalmente com a generalizabilidade e a legitimidade do estudo empírico.

6.1 Identificação Qualitativa de Restrições no Ambiente Produtivo

A identificação dos gargalos é o resultado da execução de um conjunto de atividades que inclui: (1) a definição do ambiente de desenvolvimento; (2) a criação do modelo *job shop* para o escalonamento das tarefas; (3) a definição de um método para identificar os gargalos e (4) a análise *quali-quantitativa* dos dados.

A busca por evidências de gargalos no ambiente de desenvolvimento de software

iniciou-se a partir das observações etnográficas anotadas no *livro*[29] *de registro* do experimento. Estas observações mostraram que os participantes podem representar uma ou mais restrições no sistema. Estas restrições devem ser identificadas, caracterizadas, analisadas e tratadas a fim de evitar ou coibir um gargalo produtivo (quantitativo).

A observação rotineira do comportamento das equipes no ambiente produtivo levou à identificação de alguns problemas. No caso em que estes problemas traduziram-se em restrições no ambiente de desenvolvimento, passaram a ser denominados *elementos restritivos* ou *restrições locais* (mini gargalos) em função de suas características individualizadas.

Diferentemente de uma máquina em um sistema de manufatura, as máquinas usadas neste experimento são constituídas por pessoas, fazendo com que os problemas identificados no PDS estejam diretamente ligados às equipes de desenvolvimento (linhas de produção). Em geral, os elementos restritivos não são comuns a todas as máquinas, ou seja, variam entre os grupos e de indivíduo para indivíduo e na maioria das vezes podem ser sanados ou atenuados com a interferência dos interlocutores / pesquisadores.

Os elementos restritivos são muito específicos e refletem, na maioria das vezes, as características individuais de cada componente do grupo. A pesquisa evidenciou que as restrições locais são caminhos que podem levar à identificação dos gargalos no processo produtivo de software. Devido ao fato de ser baseada em técnicas etnográficas, acredita-se que esta mesma observação possa ser levada para outros ambientes e até mesmo para a indústria de software.

De acordo com o plano experimental descrito no Capítulo 4, foram executados três *"quase-experimentos"*, os quais são apresentados nas seções seguintes.

6.1.1 Exp1 (Shop 1)

O *Exp1* foi a primeira a rodada experimental. Os dados experimentais da primeira rodada são mostrados na Tabela 17.

[29] *O livro de registro de cada ensaio será completamente digitalizado e quando finalizado pretende-se gerar um relatório técnico sobre as anotações da execução dos experimentos e disponibilizá-lo como fonte de consultas para estudantes e pesquisadores.*

Tabela 17: Variáveis do Experimento 1

Dados do *Exp1*		
Domínio do Problema: Sistema de Controle de Aluguel de Veículos		
Id	**Variáveis**	**Valores**
1	Período de realização	12 /08/2014 – 04/12/2014
2	Duração em semanas	17
2	Número total de alunos	35
3	Número de alunos de outras áreas	03
4	Número de Linhas de Produção – LP (grupos)	10
5	Número de LP defeituosas	01
5	Número máximo de máquinas / LP	04
6	Número mínimo de máquinas / LP	02
7	Número de aulas teóricas	17
8	Número de aula práticas	17
9	Quantidade de visitas *in-loco*	4 p/LP
10	Quantidade de Seções de *feedback*	4

6.1.2 Exp2 (*Shop* 2)

Na segunda rodada foi introduzido um novo domínio para o sistema a ser desenvolvido com um novo grupo de alunos. A Tabela 18 apresenta as variáveis e seus respectivos valores usados no *Exp2*.

Tabela 18: Variáveis do Experimento 2

Dados do *Exp2*		
Domínio do Problema: Sistema de Controle de Acadêmico		
Id	**Variáveis**	**Valores**
1	Período de realização	17 /03/2015 – 16/07/2015
2	Duração em semanas	18
2	Número total de alunos	32
3	Número de alunos de outras áreas	02
4	Número de Linhas de Produção – LP (grupos)	09
5	Número de LP defeituosas	01
5	Número máximo de máquinas / grupo	04
6	Número mínimo de máquinas / grupo	03
7	Número de aulas teóricas	18
8	Número de aula práticas	18
9	Quantidade de visitas *in-loco*	4 p/LP
10	Quantidade de Sessões de *feedback*	4

6.1.3 Exp3 (*Shop* 3)

Para a terceira rodada foi definido um novo grupo de alunos para participar do experimento que envolve um sistema para controle de venda de passagens. A Tabela 19 apresenta as variáveis e seus respectivos valores usados no *Exp3*.

Tabela 19: Variáveis do Experimento 3

Dados do *Exp3*		
Domínio do Problema: Sistema de Controle de Vendas de Passagens		
Id	**Variáveis**	**Valores**
1	Período de realização	13 /10/2015 – 17/03/2016
2	Duração em semanas	19
2	Número total de alunos	18
3	Número de alunos de outras áreas	0
4	Número de Linhas de Produção – LP (grupos)	06
5	Número de LP defeituosas	02
5	Número máximo de máquinas / LP	04
6	Número mínimo de máquinas / LP	02
7	Número de aulas teóricas	18
8	Número de aula práticas	19
9	Quantidade de visitas *in-loco*	4 p/LP
10	Quantidade de Sessões de *feedback*	4

6.2 Restrições Detectadas no PDS

As restrições encontradas foram separadas em dois grupos: (1) restrições de ordem comportamental (RC); e (2) restrições de ordem técnica (RT).

As restrições apontadas na Tabela 20 representam as restrições qualitativas de ordem comportamental encontradas nos *shops* dos Experimentos 1, 2 e 3.

Tabela 20: Restrições de ordem comportamental do PDS

Restrições de Ordem Comportamental (RC)				
Id	***Restrições***	***Exp 1***	***Shop 2***	***Shop 3***
RC1	Comunicação entre os grupos	Sim	Sim	Não
RC2	Relacionamento interno (Conflitos)	Sim	Sim	Sim
RC3	Pró-atividade e Passividade	Sim	Sim	Sim
RC4	Comprometimento com o Projeto	Sim	Sim	Sim
RC5	Desvio de foco: Impacto externo de outras atividades curriculares	Sim	Sim	Não

As restrições de ordem técnica estão diretamente relacionadas ao conhecimento técnico e o conhecimento tácito (Ribeiro *et al*. 2017-d). Estas restrições mostraram ser as principais candidatas causadoras de gargalos no ambiente produtivo. Outro fator relevante é que as restrições de ordem técnica tem um potencial para provocar restrições de ordem comportamental.

A Tabela 21 a seguir apresenta as restrições qualitativas de ordem técnica, encontradas nos *shops* dos *Experimentos* 1, 2 e 3.

Tabela 21: Restrições de ordem técnicas do PDS

Restrições de Ordem Técnica (RT)				
Id	***Restrições***	***Shop 1***	***Shop 2***	***Shop 3***
RT1	Maturidade sobre o domínio	Sim	Sim	Sim
RT2	Documentação do sistema	Sim	Sim	Sim
RT3	Conhecimento da linguagem UML e seu emprego notacional correto	Sim	Sim	Sim
RT4	Conhecimento das ferramentas utilizadas	Sim	Sim	Sim
RT5	Processo unificado	Sim	Sim	Sim
RT6	Modelo arquitetural (MVC)	Sim	Sim	Sim
RT7	Estratégias de teste	Sim	Sim	Sim

6.3 Identificação e Tratamento das Restrições

Os tópicos seguintes apresentam a identificação, a descrição e o tratamento dado às restrições listadas nos tópicos anteriores, seguindo o método de identificação de gargalo apresentado na Seção 5.4.

6.3.1 Restrições Comportamentais (RC)

As restrições serão identificadas pelo ID da Tabela 13, formado pela sigla que tipifica a restrição (RC) e por um numeral sequencial.

1. RC1: Comunicação entre os grupos

Identificação: Durante a execução dos Exp1 e Exp2 foram detectados problemas de comunicação envolvendo uma linha, em cada um dos experimentos. A descoberta nas duas situações foi feita através da informação provida por um dos membros dessas equipes, após uma visita *in loco* onde foi cobrado maior comprometimento, pontualidade e completeza na entrega dos artefatos.

No primeiro caso, o grupo deixou de entregar um dos artefatos. A justificativa foi que a produção estava em posse de um dos componentes e que este teria viajado para um evento sem enviar a produção, impedindo que a equipe efetuasse a entrega. No segundo caso, a justificativa pela falta foi que um dos membros da equipe estava incomunicável, impedindo também a entrega de parte da produção.

Problemas de comunicação deste tipo impactam diretamente nos resultados parciais, que são as entregas de acordo com as fases do *PU-Like*, mas não comprometem a entrega final, desde que a equipe consiga desenvolver as atividades de acordo com o nível mínimo de qualidade exigido. O impacto maior está relacionado ao *score* acadêmico

obtido na disciplina pelo aluno, que não é uma questão da pesquisa embora seja uma preocupação dos pesquisadores.

Fases de ocorrência: Exp1 (terceira fase); Exp2 (terceira e quarta fase).

Grupos de ocorrência: *Exp1* (G3 e G9). *Exp2* (G1).

Tratamento: o tratamento empregado ficou no campo da orientação e da punição, com relação à nota parcial da disciplina. A orientação tentou mostrar a importância em estabelecer canais de comunicação sólidos e do acompanhamento destes canais.

Comentário: O *Exp3* não apontou nenhum problema de comunicação entre as equipes.

2. RC2: Relacionamento interno (Conflitos)

Identificação: Os três experimentos apresentaram problemas de relacionamento envolvendo conflito de ideias e de interesses. Estes problemas são provocados por falha na comunicação, cobrança entre os membros, deficiência técnica e por influências dos elementos mais proativos.

O comprometimento também é um fator que contribui para a instauração de conflitos entre os componentes e é evidenciado através dos relatórios acadêmicos de frequência e pelo relato de membros da equipe. O desconforto e a insatisfação por parte dos membros das equipes também ficam evidente nos casos de falta de comprometimento e foram explorados pelo interlocutor para descobrir o problema causador.

Os conflitos internos são fatores amplamente discutidos e estudados no ambiente de desenvolvimento de software e são apresentados inclusive na literatura clássica, como (Pressman e Maxin, 2014), (Sommerville, 2011), (DeMarco e Lister, 1999) e o (PMBOK/PMI, 2007). Os conflitos são restrições sérias e pertinentes para o PDS e podem impactar fortemente no processo e no projeto como um todo.

Fases de ocorrência: Exp1 (terceira fase); Exp2 (segunda, terceira e quarta fase); *Exp* 3 (terceira fase).

Grupos de ocorrência: *Shop* 1 (G3); *Shop* 2 (G1), *Shop* 3 (G2).

Tratamento: O tratamento para este tipo de problema requer uma certa habilidade do líder de projetos. No caso específico deste estudo, o problema foi mediado pelo interlocutor, com as seguintes medidas:

- Conversa reservada com a equipe;
- Conversa individualizada com os envolvidos;

- Orientação e motivação quanto à resolução do conflito;
- Reapresentação da importância do projeto e do objetivo comum de todos os envolvidos;
- Ameaça de penalidades: todo o grupo seria penalizado por não conseguir cumprir com as etapas do projeto.

Comentário*:* Como é um problema estritamente local, ou seja, afeta apenas aquele grupo ou determinada pessoa, é possível que os grupos estejam resolvendo os conflitos internamente sem deixar que os mesmos sejam externados. Outro ponto importante é que não foram observadas mais restrições desta natureza a partir dos tratamentos aplicados, talvez pela inibição ou pelo fato da equipe assumir certas responsabilidades para não serem prejudicados em termos da nota acadêmica.

3. RC3: Pró-atividade e passividade

Identificação: Uma característica comum evidenciada nos três experimentos foi a relação entre a passividade de alguns indivíduos e a pró-atividade de outros participantes dos grupos. Isto não é uma surpresa, pois ambas as características estão presentes no biotipo de cada indivíduo e fazem parte da personalidade de cada pessoa.

As anotações do livro de registro do experimento mostraram que os indivíduos proativos assumem um papel de liderança, mesmo que não formalmente e atacam o problema de frente tentando resolvê-lo, seja pela busca por novas ferramentas ou por novas fontes de consulta. Os proativos tentam implementar / desenvolver o projeto como um todo, pesquisando e procurando enxergar além do lugar comum.

O impacto negativo observado neste experimento é que, quase sempre, o indivíduo proativo acaba assumindo um papel e responsabilidades que deveriam ser divididas entre a equipe, e com isso acumula tarefas que deveriam ser desenvolvidas por outros membros. Como consequência, toma para si uma quantidade de tarefas acima de sua capacidade produtiva e com isso, afetando a qualidade e a quantidade dos artefatos entregues.

Neste experimento, foi possível notar que os passivos são em geral observadores e acabam acatando as decisões e imposições dos proativos quanto: (1) aos problemas que devem ser atacados e resolvidos primeiro; (2) ao emprego de metodologias e ferramentas; (3) a estrutura do projeto; (4) ao que fazer primeiro; e (5) todas as tomadas de decisões importantes para realizar o trabalho.

Os únicos pontos positivos observados nos indivíduos passivos foram suas observações e questionamentos pontuais. Isto talvez, porque exercem um papel mais de observador do que de construtor. Por outro lado, deixam a desejar no plano laboral.

Fases de ocorrência: A RC3 foi notada nos três experimentos, a partir da segunda fase.

Grupos de ocorrência: *Shop* 1 (G2, G4, G7, e G8). *Shop* 2 (G4, G8 e G9). *Shop* 3 (G1, G4, G5)

Tratamento: Redistribuição mais igualitária das tarefas e maior motivação e cobrança sobre os passivos no plano laboral.

Comentário: A vivência do pesquisador em projetos de software permite sugerir que duas frentes podem ser exploradas. A primeira é a motivação, que deve ser aplicada aos passivos provocando e despertando interesse para com as decisões. A segunda é a contenção, que deve ser aplicada aos proativos, com muito cuidado, pois pode influenciar diretamente no grupo, no projeto e no aprendizado dos alunos. Ainda assim, a orientação passada aos grupos foi que fizessem uma melhor distribuição das tarefas envolvendo uma maior participação dos passivos.

4. RC4: Comprometimento com o projeto

Identificação: Esta restrição foi detectada a partir das observações rotineiras no ambiente laboral e por delação por parte de membros de alguns grupos, devido aos conflitos de datas de entregas com atividades paralelas. Os problemas encontrados com este tipo de restrição foram pontuais, locais e individuais, existindo praticamente em todas as fases de todas as rodadas experimentais. Isso já se esperava em função do ambiente onde a pesquisa foi executada.

A RC4 é provocada principalmente pelos seguintes "*mini gargalos*": (1) falta de entendimento do domínio do problema; (2) falta de conhecimento técnico e tácito e (3) objetivo voltado apenas para o resultado acadêmico (notas). Com os tratamentos qualitativos, os grupos em sua maioria conseguiram realizar as entregas, mesmo que a qualidade aferida não tenha apresentado o melhor resultado em alguns casos.

Fases de ocorrência: A RC4 foi observada nos três experimentos, a partir da segunda fase do *PU-Like*.

Grupos de ocorrência: *Shop* 1 (G3). *Shop* 2 (G1). *Shop* 3 (G2)

Tratamento: Cobrança contínua dos grupos e instrução sobre a importância do experimento.

5. RC5: Desvio de foco: Impacto externo de outras atividades curriculares

Identificação: o primeiro experimento evidenciou este problema, principalmente na segunda entrega/fase, onde a maioria dos artefatos foi entregue de forma incompleta ou mal construídos por praticamente todos os grupos. Ao investigar o fenômeno, foi constatado que a segunda entrega coincidia com a semana de provas do curso de ciência da computação.

Primeiramente os grupos não externaram nenhum tipo de reclamação, porém, após a publicação dos resultados da avaliação dos resultados produtivos desta fase, houve uma queixa generalizada sobre o agendamento das entregas parciais (2ª fase). A agenda coincidente causava uma mudança de foco por parte dos participantes, que procuraram cumprir com os compromissos das outras disciplinas, deixando o desenvolvimento do software em segundo plano.

Fases de ocorrência: Exp1 (Fase 2 e Fase 3)**,** Exp2 (Fase 2 e Fase 3).

Grupos de ocorrência: *Shop* 1 (todas as LPs). Especificamente a LP (G3), onde os dois componentes se ausentaram por uma semana para participar de um congresso técnico. A LP (G4), também apresentou este problema, mas somente um componente foi ao evento, o que provocou um impacto menor nos resultados produtivos da LP. No *Shop* 2, foi detectado a LP (G2), com essa restrição já que. dois de seus participantes tem um emprego fora da universidade e não tem mostraram interesse com o projeto.

Tratamento: A restrição foi tratada para terceira e quarta entrega do *Shop* 1 com o ajuste do calendário de entregas. A terceira entrega coincidia com a semana da jornada científica e foi postergada uma semana, o mesmo acontecendo com a quarta entrega.

Comentário: Durante o segundo e o terceiro experimentos este problema já não foi identificado, isto porque foi construído um calendário de entregas observando as datas das obrigações curriculares no calendário acadêmico.

As anotações sobre esta restrição (desvio de foco) evidenciaram um fenômeno que foi denominado de síndrome do *deadline*[30]. A síndrome do *deadline* é uma variação da

[30] *A sídrome do deadline é uma fenômeno que remete as pessoas e os times de desenvolvimento a um plano de trabalho que pode ser entendido como: "Não importa o processo, enquanto tiver tempo para a tarefa corrente, priorize as tarefas de menor deadline!" (Ribeiro et al., 2017).*

Síndrome do Estudante, proposta por (Goldratt, 2006) e da "Lei de Parkinson" apresentada por (Parkinson e Osborn, 1957). A síndrome do Deadline foi adotada nesta pesquisa para identificar as perturbações antes de cada entrega em um processo de desenvolvimento de software, causada principalmente pela priorização de tarefas (Ribeiro *et al*., 2017-b).

6.3.2 Restrições Técnicas (RT)

Seguindo a mesma formalização adotada na seção anterior, as restrições técnicas também serão identificadas pelo ID (Tabela 14), formado pela sigla que tipifica a restrição (RT) e por um numeral sequencial.

1. RT1: Maturidade sobre o domínio do problema

Identificação: Inicialmente as equipes tiveram muita dificuldade no entendimento do domínio do problema. A falta de compreensão gerou questionamentos não somente sobre o domínio como também sobre o processo construtivo para cumprir com as exigências impostas pelo domínio de problema e suas regras de negócio.

Fases de ocorrência: Fases 1 e 2 em todos os experimentos

Grupos de ocorrência: todos os grupos, exceto o G8 e G2 no *Shop* 1, o G4 no *Shop* 2 e G2 no *Shop* 3.

Tratamento: Elaboração de uma pró-análise e uma pró-especificação formal e direta dos requisitos, contendo:

- Identificação das principais funcionalidades e classes do sistema;
- Esclarecimento de dúvidas genéricas (para todos os grupos – exposição em sala);
- Esclarecimentos de dúvidas específicas (para cada grupo – visita *in loco*);
- Exposição sobre o processo de desenvolvimento;
- Revisão de aulas teóricas.

2. RT2: Documentação do sistema

Identificação: Em todo início de cada um dos três *"quase-experimentos"*, foram detectadas dúvidas generalizadas sobre a elaboração da documentação do sistema. De fato, foi notada uma deficiência geral nas equipes quanto à proficiência nas técnicas de elaboração da documentação.

Fases de ocorrência: Fases 1, 2 e 3 de todos os experimentos.

Grupos de ocorrência: todos os grupos de todos os experimentos.

Tratamento: Elaboração de um roteiro apoiado por aulas expositivas explicando a forma de construção da documentação.

3. RT3: Conhecimento da UML (*Unified Modeling Language*) e seu emprego notacional correto

Identificação: Apesar da maioria dos participantes admitirem ter conhecimento sobre a UML (questionário de nivelamento), foi observada uma grande dificuldade para representar, nessa linguagem, os artefatos requeridos pelo processo. Foi identificado ainda que a falta de prática com a linguagem de modelagem, acentuada pelo esquecimento dos elementos notacionais, levou a uma grande dificuldade no emprego da mesma.

Fases de ocorrência: fases 1 e 2 do *Exp1*,*fase* 2 do *Exp2* e fases 1, 2 e 3 do *Exp3*.

Grupos de ocorrência: todos os grupos dos experimentos 1 e 2, exceto o G8 do *Exp1* 1 e G4 do *Exp*2. No experimento 3 (G1) e (G2).

Tratamento: Foram aplicadas cinco aulas teóricas expositivas sobre a UML, envolvendo os elementos da UML e os diagramas definidos para compor a planta do software. Além disso, foi mostrado um estudo de caso de um domínio similar para aumentar a capacidade e habilidade na abstração dos requisitos.

Comentários: A aplicação do tratamento promoveu uma melhora significativa em todos os grupos com relação à ferramenta e seu emprego correto. A visão foi aumentada e o domínio do problema passou a ser explorado na prática e corretamente modelado.

4. RT4: Conhecimento das ferramentas utilizadas

Identificação: O conhecimento dos times de desenvolvimento nas ferramentas adotadas[31] é muito diversificado e com grande variação entre os grupos, tanto do conhecimento técnico, quanto do conhecimento tácito adquirido ao longo de sua vida acadêmica.

Isto foi detectado pela análise dos questionários de auto avaliação do conhecimento e dos *surveys* aplicados. Esta é uma observação importante e com forte poder de impacto para com os resultados produtivos e que deve ser atacada sistematicamente. A existência

[31] ***Principais Ferramentas Utilizadas:*** *Linguagem de Modelagem Unificada (UML) Processo Unificado (PU), Linguagem de Programação Orientada a Objeto-Java (LPOO-Java), SGDB My SQL, IDE (Eclipse e Net Beans), Versionadores (GIT Hub) e Testadores (JUnit e DBUnit).*

do problema foi confirmada com a análise das entregas e com o monitoramento rotineiro das equipes em tempo de execução.

Para exemplificar, podem ser destacadas algumas das variações de conhecimento como: (1) Técnico: alguns participantes conhecem mais LPOO-Java, outros mais UML, outros mais Banco de Dados (BD) e assim por diante e (2) Tácito: que representa conhecimento e experiência em: interpretação de procedimentos e processos e habilidades na solução de problemas.

Fases de ocorrência: Fases 1, 2 e 3 em todos os experimentos.

Grupos de ocorrência: todos os grupos do experimento 1, e todos os grupos do experimente 2, exceto a LP (G4). No experimento 3, a LP (G2) apresentou esta restrição.

Tratamento: Tentativa de nivelar os grupos com instruções específicas sobre cada frente de trabalho; incentivo em explorar as habilidades de cada indivíduo dentro do grupo e orientação em redistribuir as tarefas internamente de acordo com as habilidades e conhecimentos individuais.

Comentário: Não foi possível identificar se esta heterogeneidade provoca ou é uma restrição para a máquina. Mas é sabido, que as linhas que melhor trabalharam a especificidade de cada máquina tiveram desempenho superior do que as linhas onde as tarefas foram distribuídas aleatoriamente ou quantitativamente.

5. RT5: Conhecimento do Processo Unificado

Identificação: Nas fases iniciais foram encontrados problemas de entendimento do funcionamento do Processo Unificado, que por sua vez, prejudicavam o progresso do desenvolvimento, pois afetavam a distribuição de esforço da equipe na construção dos artefatos.

Fases de ocorrência: Fases 1 e 2 em todos os experimentos.

Grupos de ocorrência: todos os grupos em todos os experimentos.

Tratamento: Transmissão de conhecimento através de aulas expositivas e práticas e acompanhamento durante as visitas *in loco*.

Comentário: O PU é usado pelo interlocutor para controlar e gerenciar o desenvolvimento, mas a falta de entendimento afeta as linhas de produção, por este motivo o conhecimento é levado a todos os participantes.

6. RT6: Modelo arquitetural MVC

Identificação: Alguns grupos encontravam dificuldades (detectado inicialmente) na implementação das operações/métodos em Java segundo a metodologia MVC. Apesar de já conhecerem a técnica e terem sido instruídos quanto à mesma, este foi um problema recorrente, que envolveu todas as três rodadas, mas não se aplicou a todos os grupos.

Fases de ocorrência: Fases 2, 3 e 4 em todos os experimentos.

Grupos de ocorrência: todos os grupos do Exp1, exceto (G2, G8, e G7) e todos os grupos do Exp2, exceto o G4. No Exp3 somente a LP (G1) apresentou esta restrição.

Tratamento: Apresentação de modelos e exemplos *in loco* para os grupos com dificuldade; explicação técnica sobre o modelo e seu emprego correto; e, exigência da modelagem do MVC com o diagrama de sequência da UML para maior compreensão do mesmo.

7. RT7: Estratégias de teste

Identificação: Definiu-se que todos os testes unitários e funcionais seriam aplicados na modalidade 'caixa preta'. Como parte da documentação foi exigido o desenvolvimento de um plano de teste com planilha e relatórios sobre o emprego das estratégias de testes sugeridas.

A falha na produção dos artefatos de Teste foi um problema genérico, impactando todas as linhas nos dois primeiros experimentos, causados tanto pela falta de conhecimento técnico como pela baixa importância dada a esta atividade.

Fases de ocorrência: Fases 3 e 4 no Exp1 e 2; e Fase 2, 3 e 4 no Exp3.

Grupos de ocorrência: todas as LPs do *Shop* 1, exceto (G8); todas as LPs do *Shop* 2; e as LPs (G2, G5) do *Shop* 3.

Tratamento: Aulas expositivas sobre os testes solicitados, Apresentação de exemplos in loco para os grupos com dificuldade em realizar os testes; explicação técnica sobre a aplicação correta dos testes. No Shop 3, para efeito de tratamento, foi antecipado o conteúdo didático sobre testes e antecipado a exigência do plano de teste já para a segunda entrega (fase 2) do experimento.

Comentários: No Exp3 foi executado uma estratégia diferente para tentar diminuir os impactos no Shop 3 em dois momentos: (1) a antecipação do escalonamento de algumas tarefas do artefato Teste no shop e (2) a antecipação das aulas teóricas sobre o conteúdo relacionado a teste de software. Durante a execução do processo foi possível identificar

quais as restrições que impactaram na atividade de teste. Duas frentes destacaram-se: (1) falta de conhecimento técnico para realizar a tarefa e (2) a síndrome do *deadline* (Ribeiro et al. 2017-b).

6.3.3 Aplicação da TOC

A TOC foi aplicada nas restrições técnicas. Primeiro com a IDENTIFICAÇÃO das restrições, e em seguida com as DECISÕES de como tratar as restrições a partir da intervenção do interlocutor frente às equipes de desenvolvimento. A SUBORDINAÇÃO das restrições foi aplicada com a inserção de conhecimento teórico e prático, forçando as máquinas a resolver o problema técnico para continuar o desenvolvimento. As restrições identificadas em uma fase foram todas ELEVADAS na fase seguinte, com a cobrança da corretude e completude dos artefatos, além da inserção de novas atividades no *Shop*. A cada fase, o processo era reiniciado, VOLTANDO ao passo 1, para identificar novas restrições e seguir aplicando o algoritmo.

6.4 Considerações sobre a Análise Qualitativa

Observando o comportamento de cada linha de produção ou mais especificamente de cada máquina e a independência funcional das linhas, é possível ver que existem alguns problemas locais ou falhas, os quais foram denominados de "*mini gargalos*". Em geral, isso está relacionado a uma atividade específica associada a uma máquina com restrição de capacidade ou recurso com restrição de capacidade (RRC).

Quando isso acontece, temos uma restrição na linha de produção que deve ser tratada, executando as 5 etapas da TOC. O fracasso em realizar o tratamento, pode provocar a criação de um gargalo no sistema produtivo que irá afetar diretamente o fluxo do processo de produção e, consequentemente, limitar a produção em relação à qualidade e a quantidade de artefatos a serem produzidos.

Embora à primeira vista pareça ser uma platitude, os resultados da análise qualitativa (Restrições Comportamentais e Restrições Técnicas) sugere que o gargalo em um processo de software está associado à capacidade de produção de cada indivíduo e consequentemente de cada Linha de Produção. Esta observação vai de encontro com a resposta da nossa pergunta principal, que deverá ser confirmada após a análise quantitativa dos dados, que será apresentada no Capítulo 7.

7 ANÁLISE QUANTITATIVA

"Existem muitas hipóteses em ciência que estão erradas. Isso é perfeitamente aceitável, elas são a abertura para achar as que estão certas".

Carl Sagan

A pesquisa quantitativa se enquadra na categoria de estudos empíricos ou estudos estatísticos, que inclui: estudos experimentais, *quase-experimentos*, pré-testes e pós-testes (Newman e Benz, 1998). Segundo Harwell (2011), o objetivo da aplicação dos métodos quantitativos é a maximização da replicabilidade e generalização dos resultados. Harwell (2011) afirma ainda, que o particular interesse da pesquisa quantitativa está na predição e não na expectativa do pesquisador em relação a suas experiências, percepções e preconceitos. Em resumo, o método quantitativo foca na objetividade da realização do estudo e nas conclusões tiradas do mesmo.

A análise dos dados é uma etapa fundamental em qualquer experimento. É ela que dá sentido e valida o experimento e os resultados obtidos. A análise dos resultados permite ao pesquisador o entendimento dos problemas inerentes aos objetivos da pesquisa e a construção das respostas para as questões levantadas.

Neste capítulo será reportado a análise quantitativa dos dados colhidos e os resultados conclusivos obtido com os *"quase-experimentos"* realizados.

7.1 Ensaios Experimentais

Os ensaios realizados neste estudo serviram de fonte de coleta de dados para a análise quantitativa, assim como também foram a fonte para as observações etnográficas da análise qualitativa.

Como visto no Capítulo 6, foram realizados três *"quase-experimentos"*, denominados de Exp1, Exp2 e Exp3. Porém, diferentemente do Capítulo 6, nesta seção somente serão usados os dados dos ensaios 2 e 3 para aceitar / refutar as hipóteses levantadas. Isto porque no Exp1, não foi computado o esforço / tempo consumido pelas LPs para produzir os artefatos e consequentemente o software proposto em cada ensaio experimental.

7.1.1 Quase Experimento 1 (*Shop 1*)

O primeiro quase experimento (*Exp1 – Shop 1*) contou com 9 linhas de produção e serviu para: (1) colher as informações acerca dos problemas do processo de desenvolvimento de software; (2) para a análise qualitativa das linhas de produção e (3) para ajustes da própria pesquisa e seu processo de execução. O registro histórico e a experiência adquirida no primeiro experimento também foram usados como balizadores (ajustes e controle) para a execução dos outros dois experimentos. Mesmo sendo o projeto piloto, o primeiro experimento[32] foi essencialmente importante para fazer algumas correlações com os demais experimentos sobre a produção das LPs e as dificuldades encontradas para produzir os artefatos entregues.

7.1.2 Quase Experimento 2 (*Shop 2*)

O segundo quase experimento (*Exp2 – Shop 2*) também contou com 9 linhas de produção. No entanto, com a desistência da LP6, somente foram computados e utilizados os dados de 8 linhas de produção. A filosofia e o processo de execução seguiram os mesmos padrões do Exp1, com a execução e entregas programadas dentro de um agendamento dividido em 4 etapas de acordo com as fases do *PU-Like*. O produto final resultou em produtos de software completos, construído a partir da produção dos artefatos (subprodutos) desenvolvidos em cada entrega.

7.1.3 Quase Experimento 3 (*Shop 3*)

O terceiro quase experimento contou com 6 Linhas de Produção (LPs), mas somente 4 LPs concluíram os trabalhos. Por este motivo, os dados das LPs (*LP3 e LP6*) não foram tabulados e utilizados nas análises "*quali-quanti*". O experimento seguiu ordenadamente a estrutura adotada e aplicada nos outros dois experimentos, onde todas as LPs rodaram o mesmo processo e executaram as mesmas atividades do experimento anterior, mudando apenas o domínio de problema. No terceiro experimento o processo já estava razoavelmente maduro, não havendo a necessidade de ajustes maiores. Isto facilitou o gerenciamento do processo e o atendimento das solicitações das LPs / máquinas.

---

[32] O primeiro quase-experimento foi usado na íntegra para identificação e tratamento de restrições qualitativas e serviu ainda como calibrador para os quase-experimentos 2 e 3. No entanto não foi computado o esforço (tempo de processamento das tarefas no shop) do Exp1, motivo pelo qual os dados deste experimento não serão usados na análise quantitativa.

Vale ressaltar que nos três casos, foram utilizados três domínios de problemas diferentes e com linhas de produção (times de desenvolvimento) diferentes.

7.1.4 Definição de Gargalo

O gargalo foi definido segundo a terminologia apresentada na Seção 1.3.1 e de acordo com o modelo *Job Shop* apresentado na Seção 4.6 para o PDS nos três casos. Desta forma, um gargalo no PDS se caracteriza por: um artefato consumidor de recursos (tempo) que impede a maximização da produção, ou seja, impede que uma determinada LP tenha produção (C_{Max} (D_A) = 100%).

7.2 Questões do Experimento

A pesquisa foi fundamentada e baseada nas questões levantadas durante a elaboração do plano experimental. As questões de pesquisa determinam o foco e os objetivos do experimento e modelam as hipóteses construídas para validar ou refutar a teoria a partir dos dados da pesquisa (Harwell, 2011).

Tanto as questões quanto as hipóteses apontadas neste estudo foram apresentadas na Seção 1.6 deste texto. Todavia, para facilitar a leitura, as questões e hipóteses serão novamente apresentadas nas próximas seções.

7.2.1 Questões, Hipóteses e Respostas

O esforço desta pesquisa se concentra em responder as três primeiras questões (*Q1, Q2 e Q3*) através da análise de suas hipóteses (H_0 e H_1). As respostas das demais questões dependem dos resultados desta análise e serão respondidas indutivamente após as conclusões sobre as questões 1, 2 e 3.

7.2.2 Questão 1 (*Q1*)

A primeira questão e consequentemente suas hipóteses: (1) nula (H_0); e (2) alternativa (H_1), questiona a existência de gargalos no PDS.

Q1: Existe gargalo no processo de desenvolvimento de software?

H_0 = Não existe gargalo no processo de desenvolvimento de software!

H_1 = Existe gargalo no processo de desenvolvimento de Software!

Intuitivamente pode-se assumir que a hipótese alternativa (H_1) responde a questão (Q1), uma vez que Goldratt (2006) afirma categoricamente que todo sistema produtivo possui pelo menos um limitador de produção. No entanto é necessário avaliar os dados coletados para verificar se a afirmativa de (Goldratt e Cox, 2006) também se aplica ao processo de desenvolvimento de software.

7.2.3 Questão 2 (Q2)

A avaliação qualitativa mostrou que as restrições variam entre as máquinas e consequentemente entre as LPs do *shop*, principalmente devido ao conhecimento tácito e técnico de cada indivíduo (máquina) e pelo tratamento dado à cada restrição. No entanto é imperativo saber se a diversidade de conhecimento das máquinas e o tratamento aplicado favorecem para a alternância do gargalo no *shop*, ou ainda se isso pode fazer com que outros gargalos possam surgir no *shop*.

Q2: O gargalo é o mesmo para todas LPs?

H_0 = *O gargalo é o mesmo para todas LPs!*

H_1 = *O gargalo não é o mesmo para todas LPs!*

A existência da questão 2 está condicionada à validade da questão 1. A investigação aponta para a existência de gargalos no PDS. Para dizer se os gargalos são os mesmos, pode-se fazer uso da análise qualitativa, onde foi detectado que os elementos restritivos ou "mini gargalos" são específicos e inerentes a determinadas máquinas. A avaliação quantitativa poderá validar a percepção qualitativa e consequentemente refutar a hipótese nula (H_0) ou indicar que o gargalo é o mesmo entre as LPs do Shop, corroborando com as definições de gargalo de Goldratt e Cox (1996).

7.2.4 Questão 3 (Q3)

A Questão 3 sugere um cenário diferente para o PDS ao indagar se o gargalo muda durante o processo de desenvolvimento de software. No ambiente fabril, geralmente o gargalo é conhecido e é o mesmo para todas as linhas de produção no *shop*, desde que estas possuam as mesmas características produtivas. Enquanto que no PDS o gargalo é desconhecido e as LPs não possuem as mesmas características produtivas devido aos fatores etnográficos das máquinas que compõe as linhas de produção.

Q3: O gargalo muda no decorrer do processo de desenvolvimento?

H_0 = O gargalo não muda no decorrer do processo de desenvolvimento!

H_1 = O gargalo muda no decorrer do processo de desenvolvimento!

Com a análise quantitativa, pretende-se responder estas questões objetivamente, deixando o campo da percepção e da subjetividade para o campo da prova experimental através da análise e significância estatística. Como consequência, a análise quantitativa deverá conduzir as respostas para aceitação ou refutação das respectivas hipóteses nula dos três casos a serem analisados.

7.3 Análise Descritiva

Nesta seção será apresentada uma análise descritiva sobre os três indicadores (Esforço, Produção e Dificuldade) usados para avaliar o desempenho das máquinas e consequentemente das LPs nos *Shops 2 e 3*. Primeiramente, será feito uma análise geral dos resultados produtivos dos dois casos (*Shop* 2 e *Shop* 3). Na sequência, estes três indicadores serão avaliados para cada um dos tipos de artefato produzidos.

7.3.1 Análise Geral: Esforço, Produção e Dificuldade.

Com o objetivo de mapear o desempenho das LPs em cada *shop* e visualizar um caminho para conduzir a investigação na busca pelos gargalos, foram extraídos os resultados produtivos totalizados para cada indicador, onde:

Esforço: total de tempo (expressos em minutos) consumido pelas LPs para produzir o software;

Produção: medida do percentual do produto construído, entregue, avaliado e aceito;

Dificuldade: valor médio da dificuldade percebida pelas LPs para a produção do software, conforme apontamentos indicados no *survey*.

A Tabela 22 apresenta os resultados produtivos e a estatística descritiva do *Shop* 2.

Tabela 22: Resultados produtivos do *Shop* 2

Resultados Produtivos (*Shop 2*)			
LPs	***Esforço***	***Produção***	***Dificuldade***
LP1	17.590	58,33%	0,35%
LP2	8.718	76,08%	0,26%
LP3	7.560	61,11%	0,51%
LP4	17.889	80,56%	0,36%
LP5	8.386	59,26%	0,20%

LP7	19.233	62,96%	0,24%
LP8	15.222	76,36%	0,36%
LP9	13.583	80,52%	0,27%
Estatística Descritiva – *Shop* 2			
Estatística	***Esforço***	***Produção***	***Dificuldade***
Média	13.523	0,690	0,318
Mediana	14.403	0,698	0,310
Desvio padrão	4.723	0,105	0,097
Mínimo	7.560	0,567	0,200
Máximo	19.233	0,807	0,510

A Tabela 23 mostra os resultados produtivos das LPs para o *Shop* 3.

Tabela 23: Resultados produtivos do *Shop* 3.

Resultados Produtivos – *Shop* 3			
LPs	***Esforço***	***Produção***	***Dificuldade***
LP1	16.759	87%	0,29%
LP2	9.128	53%	0,44%
LP4	16.069	90%	0,39%
LP5	21.236	78%	0,43%
Estatística Descritiva – *Shop* 3			
Estatística	***Esforço***	***Produção***	***Dificuldade***
Média	15.798	0,770	0,387
Mediana	16.414	0,822	0,410
Desvio padrão	5.001	0,167	0,068
Mínimo	9.128	0,534	0,290
Máximo	21.236	0,904	0,440

A grande variação de consumo de tempo entre as LPs expõe o dinamismo do PDS e evidenciam problemas de construto nos *shops*. Os registros do "*log-book"* mostraram que as LPs com alto consumo de tempo, em geral apontam para problemas técnicos ou comportamentais, como apresentado no Capítulo 6. Estes problemas (Restrições Técnicas (RT) e Restrições Comportamentais (RC)), na maioria dos casos são transformados em gargalos produtivos.

Outro dado interessante sobre os resultados gerais obtidos nos dois s*hops* é que o maior esforço nem sempre resulta em maior produtividade. Em relação à dificuldade, os dois *shops* apresentaram as seguintes discrepâncias: (1) uma LP com muita dificuldade, aplicou muito esforço, mas produziu pouco; (2) uma LP com muita dificuldade e baixo esforço e, por consequência, produziu pouco e (3) uma LP com muita dificuldade, aplicou muito esforço e produziu satisfatoriamente.

7.3.2 Esforço por Tipo de Artefato (*Shop* 2 e *Shop* 3)

O esforço desprendido pelas LPs na produção de cada tipo de artefato, é um dado interessante porque permite visualizar quais são os artefatos onde as LPs empregaram mais energia na construção do software.

A Tabela 24 apresenta o esforço, medido em minutos, das LPs do *Shop* 2 para cada tipo de artefato juntamente com suas estatísticas descritivas.

Tabela 24: Esforço por tipo de artefato - *Shop* 2

Esforço das LPs por Artefato – *Shop* 2					
LPs	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
LP1	1.359	1.760	4.958	7.336	2.177
LP2	441	884	2.836	3.140	1.417
LP3	56	684	3.626	3.121	73
LP4	920	1.247	9.989	5.124	609
LP5	175	2.048	5.082	773	308
LP7	1.125	4.033	7.459	6.011	605
LP8	1.449	2.965	8.049	2.519	240
LP9	1.459	1.033	3.432	2.677	4.982
Estatística Descritiva – Esforço do *Shop* 2					
Estatística	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Média	873	1.832	5679	3.838	1.301
Mediana	1.021	1.504	5020	3.131	607
Desvio padrão	576	1.157	2.552	2.141	1.644
Mínimo	56	684	2.836	773	73
Máximo	1.459	4.033	9.989	7.336	4.982

A Tabela 25 mostra os resultados obtidos para o *Shop* 3.

Tabela 25: Esforço por tipo de artefato - *Shop* 3

Esforço das LPs por Artefato – *Shop* 3					
LPs	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
LP1	1.762	1.959	8.179	3.878	981
LP2	412	1.672	4.539	2.505	0
LP4	1.870	2.474	5.739	3.682	2.304
LP5	490	2.075	12.668	4.470	1.533
Estatística Descritiva – Esforço do *Shop* 3					
Estatística	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Média	1.134	2.045	7.781	3.634	1.205
Mediana	1.126	2.017	6.959	3.780	1.257
Desvio padrão	790	332	3.593	824	969
Mínimo	412	1.672	4539	2.505	0
Máximo	1.870	2.474	12.668	4.470	2.304

7.3.3 Produção das LPs por Tipo de Artefato (*Shop* 2 e *Shop* 3*)*

A Produção é um indicador que permite avaliar o desempenho das LPs em tempo de execução (qualitativamente) ou (quantitativamente) ao final de cada fase do *PU-Like*. Isto é importante porque pode evidenciar as restrições ou até mesmo o gargalo do PDS antes mesmo da entrega final (software concluído).

Neste trabalho, foi usado um critério de medição simples, porém confiável, para avaliar a produção. A aferição da produção foi efetuada a partir dos valores computados em cada entrega/fase do *PU-Like* e pontuados em um sistema de ranquemento conforme apresentado na Seção 4.1.14 e 4.1.15.

A Tabela 26 exibe a produção total de cada artefato e a estatística descritiva para o software produzido no *Shop* 2, enquanto que a Tabela 27 mostra os dados correspondente ao *Shop* 3.

Tabela 26: Produção por tipo de artefato – *Shop* 2

Produção das LPs por Artefato – *Shop* 2					
LPs	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
LP1	50%	88%	88%	63%	13%
LP2	75%	89%	75%	88%	50%
LP3	13%	75%	100%	100%	0%
LP4	100%	100%	100%	100%	13%
LP5	25%	100%	88%	63%	25%
LP7	63%	63%	38%	100%	38%
LP8	88%	100%	100%	100%	13%
LP9	63%	88%	88%	88%	75%
Estatística Descritiva – Produção do *Shop* 2					
Estatística	*Arquitetura*	*Banco de Dados*	*Codificação*	*Documentação*	*Teste*
Média	0,593	0,876	0,844	0,875	0,281
Mediana	0,625	0,880	0,875	0,939	0,187
Desvio padrão	0,297	0,134	0,209	0,164	0,248
Mínimo	0,125	0,625	0,375	0,625	0,000
Máximo	1,000	1,000	1,000	1,000	0,751

Tabela 27: Produção por tipo de artefato - *Shop* 3

Produção das LPs por Artefato – *Shop* 3					
LPs	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
LP1	88%	100%	100%	63%	100%
LP2	33%	75%	33%	50%	63%
LP4	88%	100%	75%	88%	100%
LP5	75%	88%	63%	75%	88%
Estatística Descritiva – Produção do *Shop* 3					
Estatística	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Média	0,708	0,906	0,677	0,688	0,875
Mediana	0,813	0,938	0,688	0,688	0,9375

Desvio padrão	0,257	0,120	0,277	0,161	0,177
Mínimo	0,333	0,75	0,333	0,5	0,625
Máximo	0,875	1,00	1,00	0,875	1,00

A análise do percentual produzido por tipo de artefato permite avaliar o desempenho das LPs através da relação (Esforço X Produção).

7.3.4 Dificuldade das LPs por Tipo de Artefato (*Shop* 2 e *Shop* 3)

Uma maior dificuldade em realizar uma tarefa, geralmente implica em um maior esforço. E em alguns casos, isto pode afetar outras máquinas da LP, pois a tarefa que apresenta dificuldade é redistribuída ou reescalonada, perturbando as outras atividades do *shop*. A consequência disso, são as interrupções no processo produtivo e/ou esforço demasiado de determinadas máquinas (RsRC) para suprir as deficiências dos Recursos com Restrição de Capacidade (RRC) para que a agenda seja cumprida.

A dificuldade foi calculada de acordo com os critérios estabelecidos na Seção 4.4.6 e conforme a nota (*0 – 3*) atribuída pelas máquinas de cada LP no *Survey*. A Tabela 28 apresenta a dificuldade média das LPs do *Shop* 2 e a estatística descritiva resultante da analise dos dados.

Tabela 28: Dificuldade média por tipo artefato – *Shop* 2

Dificuldade Média Aferida – *Shop* 2					
LPs	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
LP1	36%	35%	46%	41%	23%
LP2	13%	35%	53%	25%	17%
LP3	56%	51%	56%	64%	34%
LP4	38%	40%	55%	46%	13%
LP5	15%	22%	33%	16%	21%
LP7	24%	23%	43%	33%	6%
LP8	44%	59%	55%	32%	14%
LP9	31%	28%	45%	29%	14%
Estatística Descritiva sobre a Dificuldade Shop 2					
Estatística	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Média	0,319	0,367	0,483	0,356	0,177
Mediana	0,333	0,354	0,496	0,323	0,153
Desvio padrão	0,146	0,130	0,080	0,147	0,084
Mínimo	0,125	0,223	0,328	0,164	0,064
Máximo	0,556	0,590	0,563	0,642	0,343

A Tabela 29 apresenta a dificuldade média e a estatística descritiva aferida para as LPs do *Shop* 3.

Tabela 29: Dificuldade média por tipo artefato – *Shop* 3

Dificuldade Média Aferida – *Shop* 3					
LPs	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
LP1	18%	26%	35%	37%	25%
LP2	46%	54%	61%	41%	29%
LP4	42%	35%	48%	38%	34%
LP5	41%	33%	46%	47%	46%
Estatística Descritiva sobre a Dificuldade Shop 3					
Estatística	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Média	0,365	0,370	0,475	0,404	0,319
Mediana	0,413	0,340	0,469	0,390	0,289
Desvio padrão	0,127	0,118	0,108	0,044	0,084
Mínimo	0,177	0,263	0,349	0,370	0,251
Máximo	0,460	0,538	0,613	0,465	0,458

A visualização da dificuldade encontrada pelas LPs na construção do software apresenta fortes indícios de que os gargalos no PDS estão relacionados com as restrições Técnicas. Porém, ainda é preciso investigar a correlação da dificuldade com os outros indicadores para que se possa efetivamente validar esta informação.

7.3.5 Esforço Total por Entrega X Tipo de Artefato (*Shop* 2 e *Shop* 3)

A medição do esforço total aplicado sobre o conjunto de artefatos e em cada fase do *PU-Lke* permite quantificar a distribuição de energia aplicada pelas LPs no *shop*. Esta aferição é importante, pois permite a mensurar o comportamento dos gargalos ao longo do processo desenvolvimento.

A Tabela 30 mostra a totalização do esforço aplicado em cada artefato de acordo com as entregas realizadas pelas LPs no *Shop* 2. A Tabela 31 inclui ainda a estatística descritiva deste conjunto de dados. A Tabela 30 apresenta os dados referente ao *Shop* 3.

Tabela 30: Esforço total (Entrega X Artefato) – *Shop* 2

Esforço Total por Entrega do *Shop* 2					
Entregas	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
1	1.375	5.265	7.333	13.688	0
2	1.207	5.561	11.176	9.685	272
3	3.022	2.837	13.443	5.500	3.171
4	1.380	991	13.479	1.828	6.968
Estatística Descritiva					
Estatística	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Média	1.746	3.664	11.358	7.675	2.602
Mediana	1.378	4.051	12.310	7.593	1.722
Desvio padrão	854	2.160	2.891	5.135	3.245
Mínimo	1.207	991	7.333	1.828	0
Máximo	3.022	5.561	13.479	13.688	6.968

Tabela 31: Esforço total (Entrega X Artefato) – *Shop* 3

Esforço Total por Entrega do *Shop* 3					
Entregas	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
1	1.392	1.651	3.699	5.228	0
2	530	4.131	6.534	3.930	0
3	1.404	1.552	7.615	2.056	2.284
4	1.208	846	13.277	3.321	2.534
Estatística Descritiva					
Estatística	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Média	1.134	2.045	7.781	3.634	1.205
Mediana	1.300	1.602	7.075	3.626	1.142
Desvio padrão	412	1.436	4.018	1.318	1.394
Mínimo	530	846	3.699	2.056	0
Máximo	1404	4131	13.277	5.228	2.534

7.3.6 Produção das LPs por Entrega X Tipo de Artefato (*Shop* 2 e *Shop* 3)

A Tabela 32 apresenta a produção das LPs organizada por entrega e por tipo de artefato, assim como a exibição da estatística descritiva na segunda parte da tabela.

Tabela 32: Produção por Entrega – *Shop* 2

Produção X Entrega – *Shop* 2						
	LPs	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Entrega 1	LP1	20,00%	40,00%	40,00%	0,00%	0,00%
	LP2	14,29%	28,57%	0,00%	57,14%	0,00%
	LP3	0,00%	25,00%	25,00%	50,00%	0,00%
	LP4	20,00%	20,00%	20,00%	40,00%	0,00%
	LP5	0,00%	25,00%	25,00%	50,00%	0,00%
	LP7	28,57%	14,29%	0,00%	57,14%	0,00%
	LP8	20,00%	20,00%	20,00%	40,00%	0,00%
	LP9	11,11%	22,22%	22,22%	44,44%	0,00%
Entrega 2	LP1	14,29%	28,57%	28,57%	28,57%	0,00%
	LP2	14,29%	28,57%	28,57%	28,57%	0,00%
	LP3	0,00%	14,29%	28,57%	57,14%	0,00%
	LP4	20,00%	20,00%	20,00%	40,00%	0,00%
	LP5	0,00%	40,00%	20,00%	40,00%	0,00%
	LP7	12,50%	25,00%	12,50%	50,00%	0,00%
	LP8	20,00%	20,00%	20,00%	40,00%	0,00%
	LP9	11,27%	11,27%	22,54%	21,13%	33,80%
Entrega 3	LP1	10,53%	10,53%	21,05%	42,11%	15,79%
	LP2	16,55%	8,97%	16,55%	33,10%	24,83%
	LP3	0,00%	14,29%	28,57%	57,14%	0,00%
	LP4	20,00%	20,00%	20,00%	40,00%	0,00%
	LP5	11,76%	23,53%	23,53%	23,53%	17,65%
	LP7	11,76%	11,76%	11,76%	47,06%	17,65%
	LP8	17,39%	17,39%	17,39%	34,78%	13,04%
	LP9	9,30%	18,60%	9,30%	34,88%	27,91%

Entrega 4	LP1	12,50%	25,00%	12,50%	50,00%	0,00%
	LP2	15,38%	15,38%	15,38%	30,77%	23,08%
	LP3	11,11%	22,22%	22,22%	44,44%	0,00%
	LP4	17,39%	17,39%	17,39%	34,78%	13,04%
	LP5	11,76%	23,53%	23,53%	23,53%	17,65%
	LP7	10,00%	10,00%	10,00%	40,00%	30,00%
	LP8	11,11%	22,22%	22,22%	44,44%	0,00%
	LP9	15,69%	15,69%	15,69%	29,41%	23,53%
Estatística Descritiva (Artefato X Entrega) – *Shop* 2						
Estatística		***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Média		0,141	0,206	0,194	0,392	0,081
Mediana		0,148	0,200	0,200	0,400	0,000
Desvio Padrão		0,075	0,075	0,081	0,124	0,113
Mínimo		0,000	0,090	0,000	0,000	0,000
Máximo		0,250	0,400	0,400	0,571	0,338

A Tabela 33 exibe a produção por entrega e por tipo de artefato das LPs no *Shop* 3.

Tabela 33: Produção por Entrega – *Shop* 3

Produção X Entrega – *Shop* 3						
	LPs	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Entrega 1	LP1	18%	18%	18%	18%	27%
	LP2	0%	29%	0%	29%	43%
	LP4	15%	15%	15%	31%	23%
	LP5	15%	15%	15%	31%	23%
Entrega 2	LP1	18%	18%	18%	18%	27%
	LP2	0%	40%	0%	0%	60%
	LP4	11%	22%	11%	22%	33%
	LP5	11%	22%	11%	22%	33%
Entrega 3	LP1	10%	20%	20%	20%	30%
	LP2	20%	20%	20%	40%	0%
	LP4	15%	15%	15%	31%	23%
	LP5	15%	15%	15%	31%	23%
Entrega 4	LP1	15%	15%	15%	31%	23%
	LP2	15%	15%	15%	31%	23%
	LP4	17%	17%	8%	33%	25%
	LP5	17%	17%	8%	33%	25%
Estatística Descritiva (Artefato X Entrega) – *Shop* 3						
Estatística		***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Média		0,125	0,197	0,121	0,263	0,277
Mediana		0,154	0,174	0,154	0,308	0,250
Desvio Padrão		0,057	0,065	0,062	0,094	0,123
Mínimo		0,000	0,154	0,000	0,000	0,000
Máximo		0,2	0,4	0,2	0,4	0,6

7.3.7 Correlação Entre os Indicadores

A relação entre os indicadores (Esforço, Produção e Dificuldade) pode ser avaliada pelos coeficientes de *Spearman* sobre os resultados totalizados exibidos na Seção 7.3.1.

A Tabela 34 apresenta os coeficientes de correlação tanto para o *Shop* 2 quanto para o *Shop* 3.

Tabela 34: Matriz de correlação dos *Shops* 2 e 3

Matriz de Correlação do *Shop* 2		
Indicadores	***Coeficiente de Correlação***	***P-valor***
Esforço X Produção	0,214	0,619
Esforço X Dificuldade	-0,072	0,866
Produção X Dificuldade	0,204	0,629
Matriz de Correlação do *Shop* 3		
Esforço X Produção	0,206	0,917
Esforço X Dificuldade	-0,415	0,750
Produção X Dificuldade	-0,802	0,333

Os resultados do *Shop* 2 apresentaram uma correlação positiva baixa e estatisticamente não significativa entre (*Esforço X Produção*). Para (*Esforço X Dificuldade*) os resultados indicaram uma correlação negativa moderada e estatisticamente não significativa. Entre (*Produção X Dificuldade*), a correlação foi negativa baixa e estatisticamente não significativa. Em relação ao *Shop* 3, os resultados mostraram o seguinte comportamento: correlação positiva baixa e estatisticamente não significativa para (*Esforço X Produção*); correlação negativa moderada e estatisticamente não significativa entre (*Esforço X Dificuldade*); e correlação negativa alta e estatisticamente significativa (*Produção X Dificuldade*).

7.4 Análise Estatística dos Dados

A natureza deste estudo exige que os dados sejam analisados estatisticamente para que se possa verificar a existência e as características do(s) gargalo(s) no processo produtivo de software, e então, confrontá-los com a inferência feita sobre a análise descritiva da Seção 7.3.

A ANOVA (*Analysis of Variance*) e o teste Tukey HSD (*Tukey Honest Significant Difference*) foram utilizadas para responder aos questionamentos da pesquisa. A ANOVA foi aplicada para os três indicadores (Esforço, Produção e Dificuldade). No entanto, a estatística sobre a Produção e a Dificuldade será apresentada somente no Apêndice F.

As subseções seguintes irão apresentar os critérios e procedimentos para testar as hipóteses, a análise estatística realizada e as conclusões sobre as questões que nortearam esta pesquisa.

7.4.1 Critérios para os Testes de Hipóteses

A Tabela 35 apresenta os critérios para os testes a serem aplicados às questões (*Q1, Q2, e Q3*) e suas respectivas hipóteses nulas (H_0) e alternativa (H_1).

Tabela 35: Formulação de critérios para análise de hipóteses

Critérios para Análise das Hipóteses				
Q1: Existe gargalo no processo de desenvolvimento de software?				
H_0 = *Não existe gargalo no processo de desenvolvimento de software!*				
H_1 = *Existe gargalo no processo de desenvolvimento de Software*				
Hipóteses	***Formulação / Realidade***		***Decisão***	
	Verdadeira	Falsa	Aceitar H_0	Rejeitar H_0
Nula (H_0)	H_0: $D_A = 100\%, \forall L_M \in S$	H_0: $D_A<100\%$	*Se H_0 = Verdade*	*Se H_1 = Verdade*
Alternativa (H_1)	H_1: $D_A < 100\%, \exists g:L_M(g)$	H_1: $D_A=100\%$	*H_1 = Falsa*	*H_0 = Falsa*
Q2: O gargalo é o mesmo para todas LPs?				
H_0 = *O gargalo é o mesmo para todas LPs!*				
H_1 = *O gargalo não é o mesmo para todas LPs!*				
Hipóteses	***Formulação / Realidade***		***Decisão***	
	Verdadeira	Falsa	Aceitar H_0	Rejeitar H_0
Nula (H_0)	H_0: $\exists g:L_M \mid A*(1)=A*(2)...A*(n), \forall L_M \in S$	H_0: $\alpha \leq 5\%$	*Se H_0 = Verdade*	*Se H_1 = Verdade*
Alternativa (H_1)	H_0: $\exists g:L_M \mid A*(1)\neq A*(2)...A*(n), \forall L_M \in S$	H_1: $\alpha >5\%$	*H_1 = Falsa*	*H_0 = Falsa*
Q3: O gargalo muda no decorrer do processo de desenvolvimento?				
H_0 = *O gargalo não muda no decorrer do processo de desenvolvimento!*				
H_1 = *O gargalo muda no decorrer do processo de desenvolvimento!*				
Hipóteses	***Formulação / Realidade***		***Decisão***	
	Verdadeira	Falsa	Aceitar H_0	Rejeitar H_0
Nula (H_0)	H_0: $\exists g:L_M(i) \mid A*(i,1) = A*(i, 2)...A*(i,n), \forall A_i \in f_{(1..4)}$	H_0: $\alpha \leq 5\%$	*Se H_0 = Verdade*	*Se H_1 = Verdade*
Alternativa (H_1)	H_1: $\exists g:L_M(i) \mid A*(i,1) \neq A*(i,2)...A*(i,n), \forall A \in f_{(1..4)}$	H_1: $\alpha >5\%$	*H_1 = Falsa*	*H_0 = Falsa*

Onde:

- H_0: é a hipótese de igualdade ($H_0 = C_{Max}(D_A) = 100\%$);
- H_1: é a hipótese de pesquisa (operacional) que se deseja comprovar ($\exists g:L_M(i)$);
- D_A: é o total de artefatos entregues por uma LP no *Shop*;
- ($\exists g:L_M(g)$): significa que existe pelo menos um "*g*" (gargalo) tal que L_M (*g*) é verdadeiro para $L_M \in S$;
- α: é valor de *p* (significância estatística) fixado em 5% ou 0,05;

- L_M*:* conjunto de linhas de produção contendo *M* máquinas de um *Shop S;*
- A_i*:* conjunto de artefatos, *A={Arquitetura, Banco de Dados, Codificação, Documentação, Teste}*, produzidos e entregues em cada fase do *PU-Like*;
- *f:* fases do *PU-Like*, sendo *f = {1, 2, 3 e 4}*;

Para a aplicação dos testes de hipóteses é preciso definir quais os tipos de dados que serão analisados e o tipo de análise que deve ser aplicada sobre o conjunto amostral. Desta forma, foram definidos os seguintes procedimentos analíticos:

1. **Questão 1 (*Q1*): Existe gargalo no processo de desenvolvimento de software?**
   - Avaliar a produção geral das LPs para cada *shop*, *i.e*, verificar a capacidade produtiva das LPs com base no parâmetro (C_{Max} (D_A) *= 100%)*;
   - Dados a serem analisados: estatística descritiva do Esforço e da Produção da LPs nos *shops* 2 e 3;
   - H_0: A hipótese nula é que todas as LPs possuem (C_{Max} (D_A) *= 100%);*
   - Tipo de análise: Inferência Estatística.
2. **Questão 2 (*Q2*): O gargalo é o mesmo para todas LPs?**
   - Encontrar para cada $L_M(i)$ qual artefato *A*(i)* é o maior consumidor de recursos;
   - H_0: A hipótese nula é que *A*(1)=A*(2)...A*(n)*;
   - Dados a serem analisados: Esforço total gasto na produção de cada um dos tipos de artefatos em cada uma das LPs;
   - Estatística utilizada: (1) ANOVA: a aplicação do teste ANOVA permitirá dizer se existe alguma diferença significativa *(α = 0.05)* entre pelo menos um par de linhas de produção; e (2) Teste Tukey HSD: a aplicação do teste Tukey HSD permitirá dizer quais tratamentos (diferenças de esforço) encontrados pela ANOVA são estatisticamente diferentes a um nível de significância *(α = 0.05)*.
   - Tipo de Análise: significância estatística com ANOVA e Tukey HSD.
3. **Questão 3 (*Q3*): O gargalo muda no decorrer do processo de desenvolvimento?**
   - Encontrar para cada $L_M(i)$ qual tipo de Artefato $(A_{i,j})$ é o maior consumidor de recursos em cada fase $f_j = A^*(i, j)$;
   - Dados a serem analisados: esforço total gasto na produção de cada um dos tipos de artefatos $(A_{i,j})$ por cada $L_M(i)$ em cada uma das fases (f_j) do *PU-Like*;

- H_0: A hipótese nula é que para cada $L_M(i)$: $A^*(i,1)=A^*(i,2)...A^*(i,n)$;
- Estatística utilizada: (1) ANOVA: a aplicação do teste ANOVA permitirá verificar se existe diferença significativa ($\alpha = 0.05$) entre pelo menos um par de artefatos durante as fases do *PU-Like*; e (2) Teste Tukey HSD: para verificar se as diferenças de esforço apontada pela ANOVA são estatisticamente diferentes a um nível de significância $(\alpha=0.05)$ durante as quatro fases do processo.
- Tipo de Análise: significância estatística com ANOVA e Tukey HSD.

7.4.2 Análise das Hipóteses

Para aceitar ou refutar as hipóteses (H_0 *e* H_1) foram analisados os dados apresentados na Seção 7.3 (Análise Descritiva).

Q1: Existe gargalo no processo de desenvolvimento de software?

Hipótese Nula: H_0= verdade para Produção = (C_{Max} (D_A = 100%)), $\forall L_M \in S$.

Argumentação: A análise descritiva das Tabelas 21 e 22 mostram que nenhuma das LPs entregou 100% dos artefatos nos dois *shops*. Logo, se nenhuma LP obteve Produção = $(C_{max}\ (D_A = 100\%))$, então $\exists g{:}L_M(g)$ no PDS.

Decisão: Rejeita-se H_0 e aceita-se H_1.

Resposta (Q1) = Sim! As restrições qualitativas observadas em tempo de execução apontaram para eventuais problemas no PDS, que se traduziram em gargalos produtivos identificados e categorizados na análise quantitativa. Esta afirmativa é importante, pois corrobora com a afirmação de (Goldratt e Cox, 2006) de que "todo sistema possui limitadores que impedem a maximização da produção".

Q2: O gargalo é o mesmo para todas LPs?

A Tabela 36 imprime um extrato das Tabelas 23 e 24 e mostra que o artefato com o maior consumo de esforço em ambos os shops é a Codificação, seguido da Documentação.

Tabela 36: Extrato do Esforço Médio

Estatística Descritiva – Esforço do *Shop* 2					
Estatística	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Média	873	1.832	5.679	3.838	1.301
Estatística Descritiva – Esforço do *Shop* 3					
Estatística	***Arquitetura***	***Banco de Dados***	***Codificação***	***Documentação***	***Teste***
Média	1.134	2.045	7.781	3.634	1.205

Em princípio, estes resultados classificam a "*Codificação*" e a "*Documentação*" como candidatos ao gargalo em ambos os *shops*. Um resultado mais preciso pode ser obtido pela aplicação do teste ANOVA de fator duplo sem repetição sobre o total do esforço (%) para cada uma das linhas de produção e para cada um dos artefatos. Os resultados do tratamento são exibidos na Tabela 37.

Tabela 37: ANOVA - Fator duplo sobre o Esforço (%) – (Grupo X Artefato)

ANOVA *Shop* 2						
Fonte da variação	***SQ***	***gl***	***MQ***	***F***	***P-valor***	***F-crítico***
LP	0,0002	7	0,0003	0,0001	1	2,354
Tipo	0,7429	4	0,1857	14,0451	2,1E-06	2,714
Erro	0,3702	28	0,0132	-	-	-
ANOVA *Shop* 3						
Fonte da variação	***SQ***	***gl***	***MQ***	***F***	***P-valor***	***F-crítico***
LP	1,11E-16	3	3,7E-17	8,52E-15	1	3,490
Tipo	0,479	4	0,119	27,596	5,7E-06	3,259
Erro	0,052	12	0,004	-	-	-

A hipótese nula para o teste ANOVA de fator duplo determina que não existe diferença estatística tanto entre as colunas (tempo gasto por tipo de artefato) como entre as linhas (tempo gasto por linha de produção) sobre o esforço aplicado nos tipos de artefatos pelas diversas LPs de cada *shop.*

Os resultados da Tabela 36 mostraram que: (1) a hipótese da igualdade do tempo gasto nos tipos de artefatos (colunas) deve ser rejeitada ao nível de significância ($\alpha=0,05$), com *p-valor = 2,1E-06* e (2) a hipótese sobre de que o esforço (%) gasto é igual para todas LPs não pode ser rejeitada ao nível de significância ($\alpha=0,05$), com *p-valor=1*.

As Figuras 24 e 25 mostram graficamente os resultados acima discutidos.

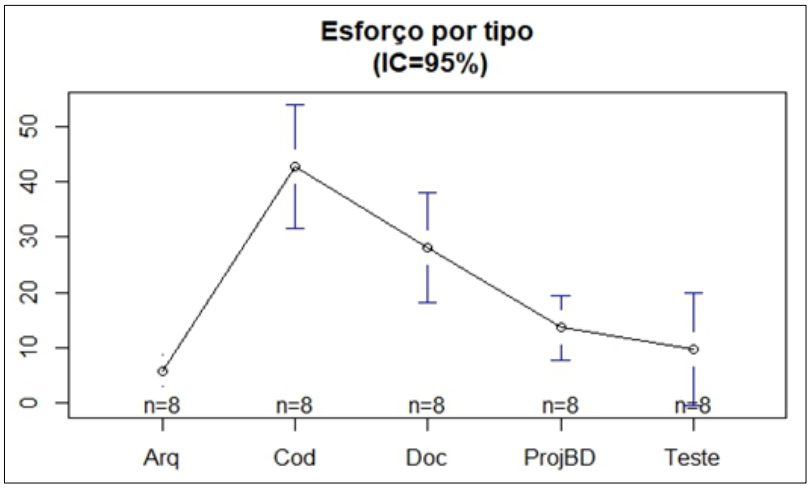


Figura 24: Esforço por tipo de artefato – *Shop* 2

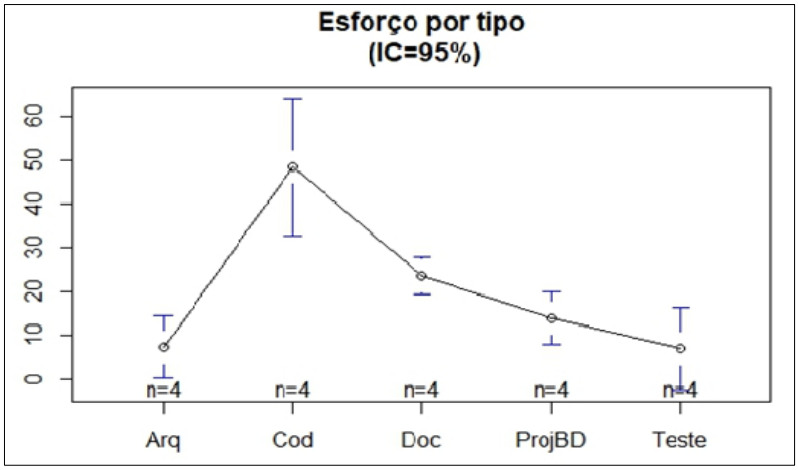


Figura 25: Esforço por tipo de artefato – *Shop* 3

A aplicação do teste Tukey HSD permite verificar se as diferenças entre as médias dos esforços consumidos pelos artefatos são estatisticamente distintas. Os resultados dos testes para os *shops* 2 e 3 são mostrados nas Tabelas 38.

Tabela 38: Teste Tukey HSD sobre o Esforço (%) – (Esforço X Artefato)

Teste Tukey HSD - *Shop* 2				
Pares /Artefatos	***Diferença***	***Menor***	***Maior***	***P-valor (adj)***
Cod – Arq	36.9	22.1	51.683	0.00
Doc – Arq	22.2	7.4	36.975	0.00
ProjBD – Arq	7.7	-7.1	22.507	0.57
Teste – Arq	3.9	-10.9	18.687	0.94
Doc – Cod	**-14.7**	**-29.5**	**0.077**	**0.05**
ProjBD – Cod	-29.2	-44.0	-14.390	0.00
Teste – Cod	-33.0	-47.8	-18.211	0.00
ProjBD – Doc	-14.5	-29.3	0.318	0.06
Teste – Doc	-18.3	-33.1	-3.502	0.01
Teste – ProjBD	-3.8	-18.6	10.965	0.94
Teste Tukey HSD - *Shop* 3				
Pares	***Diferença***	***Menor***	***Maior***	***P-valor (adj)***
Cod – Arq	41.23	28.4	54.1	0.00
Doc – Arq	16.39	3.5	29.3	0.01
ProjBD – Arq	6.55	-6.3	19.4	0.54
Teste – Arq	-0.39	-13.3	12.5	1.00
Doc – Cod	**-24.84**	**-37.7**	**-12.0**	**0.00**
ProjBD – Cod	-34.68	-47.5	-21.8	0.00
Teste – Cod	-41.62	-54.5	-28.8	0.00
ProjBD – Doc	-9.84	-22.7	3.0	0.18
Teste – Doc	-16.78	-29.7	-3.9	0.01
Teste – ProjBD	-6.94	-19.8	5.9	0.48

Argumentação: Com esta análise, é possível verificar que a diferença entre as médias do esforço empregados na "*Codificação*" e na "*Documentação*" é significativa tanto para o *Shop* 2, com *p-valor* ($\alpha = 0.05$), quanto para o *Shop* 3 que apresentou um *p-valor* igual a 0 (zero).

Decisão: Aceita-se H_0 e rejeita-se H_1.

Resposta (Q2) = Sim! As evidências apontadas na análise quantitativa apontaram o artefato "*Codificação*" como o **gargalo** do PDS para os ensaios (*Exp 1* e *Exp 2*) realizados nesta pesquisa, com uma pequena diferença para o artefato "*Documentação*".

Q3: O gargalo muda no decorrer do processo de desenvolvimento?

Para responder a *Q3* é necessário analisar se houve variação do gargalo nos *shops* de acordo com o esforço aplicado pelas LPs durante as entregas, *i.e*, durante as fases do *PU-Like*.

A Tabela 39 apresenta os resultados da análise de variância sobre o esforço aplicado pelas LPs em cada entrega e para cada tipo de artefato para os *Shops* 2 e 3.

Tabela 39: ANOVA Fator Duplo – Esforço (Entrega X Artefato)

Anova: Fator Duplo sem Repetição do Esforço (Artefato X Entrega) – *Shop* 2						
Fonte da variação	***SQ***	***gl***	***MQ***	***F***	***P-valor***	***F crítico***
Entregas	0	3	0	0	1	3,490
Tipo	0,354	4	0,089	4,930	0,014	3,259
Erro	0,215	12	0,018			
Anova: Fator Duplo sem Repetição do Esforço (Artefato X Entrega) – *Shop* 3						
Fonte da variação	***SQ***	***gl***	***MQ***	***F***	***P-valor***	***F crítico***
Entrega	-0,000	3	-0,000	- 0,000	1	3,490
Tipo	0,446	4	0,112	8,266	0,002	3,259
Erro	0,233	12	0,029	-	-	-

Argumentação: Os resultados do teste ANOVA da Tabela 39 mostraram que: (1) existe diferença ($\alpha = 1$) entre as médias do esforço aplicado pelas LPs em cada artefato e em todas as entregas, tanto para o *Shop* 2 quanto para o *Shop* 3, corroborando com a hipótese nula. (2) os resultados desta análise obrigam a aceitar a hipótese nula de que os tempos gastos em cada tipo de artefato são iguais para todas as entregas.

Decisão: Aceita-se H_0 e rejeita-se H_1.

Resposta (Q3) = Não! Os resultados da Q2 mostraram que a atividade de Codificação é o gargalo quando consideramos a fração do esforço total empregado por cada LP na confecção de cada um dos artefatos, independentemente da fase do processo.

Pode-se concluir então que a *Codificação* é o gargalo não somente do processo como um todo, mas, também para cada uma das fases do desenvolvimento.

7.5 Discussão Sobre os Resultados Encontrados

Com a análise "*quali-quanti"* realizada especificamente para o PDS adotado nesta pesquisa foi possível: (1) Identificar os artefatos gargalos; (2) Identificar os artefatos com tendência a serem gargalos; (3) Identificar os artefatos "*não-gargalos*" e (4) Identificar, caracterizar e tipificar as atividades que provocaram os gargalos.

7.5.1 Quais são os Gargalos do PDS?

Os tratamentos quantitativos indicaram que os gargalos no PDS são:

Codificação: A codificação foi identificada como o principal gargalo do PDS. Três motivos principais levaram a esta conclusão: (1) Esforço: dado pelo alto consumo de tempo; e (2) Dificuldade: dado pelo alto índice de relatos de dificuldades em realizar as tarefas deste tipo de artefato; e (3) deficiência técnica das máquinas no entendimento e na construção das funções de software na linguagem programação adotada.

Documentação: foi o segundo artefato com maior incidência de problemas no PDS. Os motivos que levaram a eleição da Documentação como um segundo gargalo do PDS foram: (1) esforço excessivo das máquinas nas tarefas que compõe a documentação; (2) alto índice de dificuldade*; (3)* deficiência técnica das máquinas na linguagem de modelagem utilizada (percepção anotada na Análise Qualitativa); *(4)* Falta de experiência prática em especificar modelar e documentar requisitos.

7.5.2 Quais os Artefatos Identificados como Potenciais Gargalos?

Esta pesquisa identificou dois artefatos como potenciais gargalos no PDS: (1) Arquitetura; e (2) Testes. Os motivos que levaram a esta conclusão foram:

1. Entrega incompleta: provocado pela interferência dos gargalos (Codificação e Documentação) no orçamento de tempo e na quantidade de máquinas envolvidas com os gargalos, pela Síndrome do *Deadline* (priorização de tarefas) e pela falta de conhecimento técnico para aplicar testes e/ou modelar e implementar o modelo arquitetural.

2. Produção baixa: a produção sofre influências do esforço e da dificuldade. Também é interessante pontuar o aspecto comportamental (compromisso) das Máquinas/LPs com o projeto. Comportamento este, que é traduzido pela falta de interesse em realizar e concluir as atividades relacionadas a estes artefatos e pelo

pré-julgamento da importância dos mesmos na produção do software, implicando em entrega abaixo dos requisitos mínimos e abaixo dos valores de referência da linha de base.

O artefato Teste foi identificado como o mais suscetível aos problemas detectados no PDS. Contudo, a aplicação do tratamento foi bastante eficaz, colocando inclusive o artefato em posição de destaque no *Shop* 3, com a maior média de produtividade. A aplicação do tratamento adequado durante o processo de desenvolvimento foi determinante para a construção correta e completa do artefato.

7.5.3 Quais os Artefatos Não São Gargalos?

O Artefato identificado no PDS identificado como "*não-gargalo*" foi o Banco de Dados (BD). O BD por ser predecessor da Codificação, exigiu que a entrega total fosse cumprida conforme determina a agenda (até a segunda fase do *PU-Like*). Nas demais fases, as atividades associadas ao BD resumem-se a apenas tarefas rotineiras de manutenção e ajustes.

Outro ponto considerado é que o artefato BD não sofreu interferência dos artefatos gargalos, a não ser pelo fato da exigência da Codificação em sua completude e corretude. Estas razões levaram a conclusão de que o Banco de Dados e as atividades que o compõem não são gargalos produtivos no PDS. Apesar de algumas LPs, relatarem dificuldades em realizar algumas atividades do BD, o artefato foi entregue na maioria dos casos com completude e com produção média acima dos valores referenciados na linha de base[33].

7.5.4 Quais Foram as Atividades que Provocaram o(s) Gargalo(s)?

A identificação dos gargalos no PDS exige outra frente de investigação! Identificar quais são os componentes de um tipo de artefato responsáveis por transformá-lo em um gargalo produtivo.

A Tabela 40 mostra a fração do esforço que foi consumido por cada uma das LPs nos componentes do artefato gargalo (*Codificação*) para os dois experimentos.

[33] *A Linha de base foi usada para acompanhar a evolução do processo produtivo. A baseline foi calculada a partir de cada entrega do PU-Like e ao final do processo. Os valores da linha de base dos Shops 2 e 3 estão disponíveis no Apêndice H, correspondem aos valores calculados após a entrega e avaliação final.*

Tabela 40: Esforço por atividade

Esforço por atividade (Codificação - *Shop* 2)					
LP	***Atividades***	***Esforço***	***LP***	***Atividades***	***Esforço***
LP1	4.1-CDUC	32.6	LP5	4.1-CDUC	45.7
LP1	4.2-CE(Telas)	40.3	LP5	4.2-CE(Telas)	40.1
LP1	4.3-CAMVC	16.0	LP5	4.3-CAMVC	7.1
LP1	4.4-DC	11.1	LP5	4.4-DC	7.1
LP2	4.1-CDUC	51.6	LP7	4.1-CDUC	16.0
LP2	4.2-CE(Telas)	30.0	LP7	4.2-CE(Telas)	44.5
LP2	4.3-CAMVC	15.3	LP7	4.3-CAMVC	18.6
LP2	4.4-DC	3.0	LP7	4.4-DC	20.8
LP3	4.1-CDUC	36.0	LP8	4.1-CDUC	26.2
LP3	4.2-CE(Telas)	35.5	LP8	4.2-CE(Telas)	30.3
LP3	4.3-CAMVC	23.4	LP8	4.3-CAMVC	25.0
LP3	4.4-DC	5.1	LP8	4.4-DC	18.5
LP4	4.1-CDUC	4.3	LP9	4.1-CDUC	14.9
LP4	4.2-CE(Telas)	59.2	LP9	4.2-CE(Telas)	27.5
LP4	4.3-CAMVC	13.2	LP9	4.3-CAMVC	34.5
LP4	4.4-DC	23.3	LP9	4.4-DC	23.1
Esforço por atividade (Codificação - *Shop* 3)					
LP	***Atividade***	***Tempo***	***LP***	***Atividade***	***Tempo***
LP1	4.1-CDUC	30.5	LP4	4.1-CDUC	9.8
LP1	4.2-CE(Telas)	30.3	LP4	4.2-CE(Telas)	28.1
LP1	4.3-CAMVC	26.1	LP4	4.3-CAMVC	54.7
LP1	4.4-DC	13.2	LP4	4.4-DC	7.4
LP2	4.1-CDUC	14.7	LP5	4.1-CDUC	18.8
LP2	4.2-CE(Telas)	32.2	LP5	4.2-CE(Telas)	44.5
LP2	4.3-CAMVC	35.1	LP5	4.3-CAMVC	27.7
LP2	4.4-DC	17.9	LP5	4.4-DC	9.0

O teste ANOVA com fator duplo foi aplicado aos resultados da Tabela 39 onde a hipótese nula determina que não existe diferença entre os esforços gastos nos componentes, tanto para as LPs como para as atividades que compõe o tipo de artefato.

A Tabela 41 exibe os resultados do teste ANOVA sobre o esforço aplicado pelas LPs nas atividades que produzem os componentes do tipo de artefato "*Codificação*".

Tabela 41: Anova sobre o esforço por atividade

Anova: Fator Único (Esforço X Atividade) – *Shop* 2					
Fonte da variação	***Df***	***SQ***	***MQ***	***F***	***P-valor***
Atividade	3	927	7,28	-	0,00093
Residuais	28	127	-	-	-
Anova: Fator Único (Esforço X Atividade) – *Shop* 3					
Fonte da variação	***Df***	***SQ***	***MQ***	***F***	***P-valor***
Atividade	3	1646	6,73	-	0,0065
Residuais	12	978	-	-	-

Os resultados do teste ANOVA rejeitam a hipótese nula de que o esforço envolvido nas diferentes atividades de Codificação é igual. Um resumo dos resultados obtidos com o tratamento é apresentado graficamente nas Figuras 26 e 27.

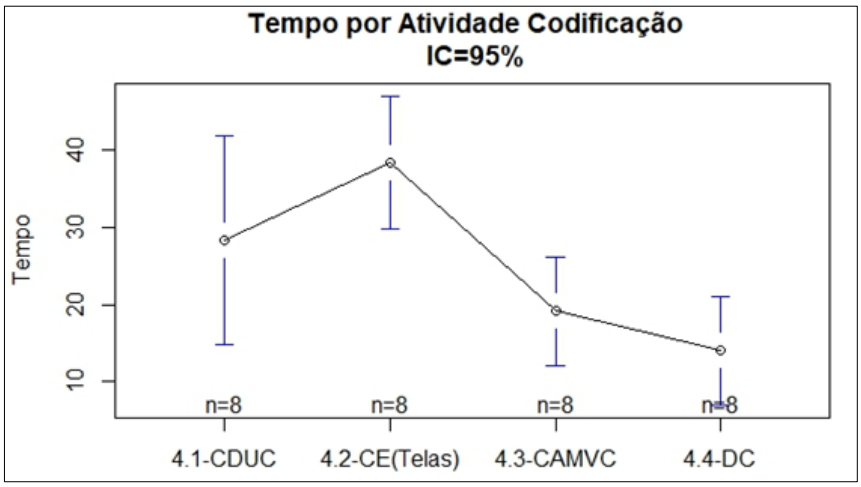


Figura 26: Esforço sobre a atividade *Shop* 2

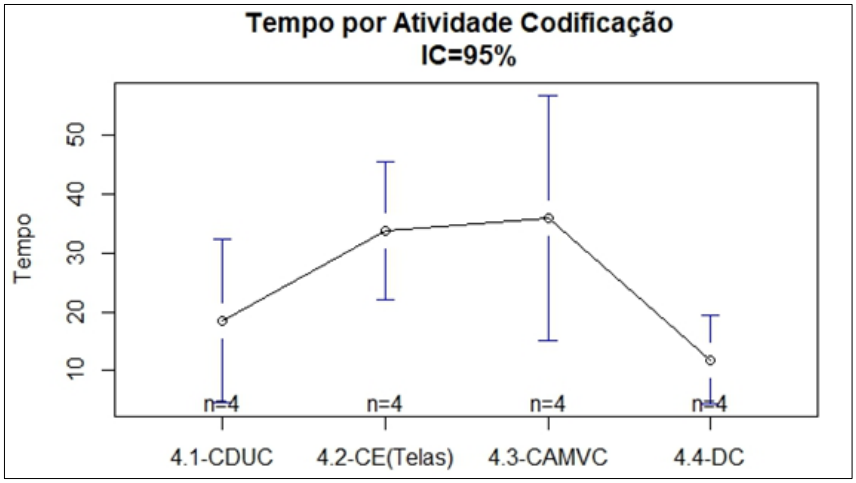


Figura 27: Esforço sobre a atividade do *Shop* 3

A identificação da atividade consumidora de recurso com maior incidência de esforço (tempo) é outra questão importante e que precisa ser respondida. Para tanto, o teste Tukey HSD com Intervalo de Confiança (IC) de 95%, permite verificar quais diferenças de esforços são estatisticamente significativas para os tratamentos aplicados nos dois *shops*. Os resultados dos testes estão dispostos na Tabela 42.

Tabela 42: Teste Tukey HSD sobre o esforço por atividade

TukeyHSD: Esforço X Atividade – *Shop* 2				
Pares (Atividade)	***Diferença***	***Menor***	***Maior***	***P-valor (adj)***
4.2-CE(Telas)-4.1-CDUC	10.0	-5.4	25.42	0.31
4.3-CAMVC-4.1-CDUC	-9.3	-24.7	6.13	0.37
4.4-DC-4.1-CDUC	-14.4	-29.8	0.99	0.07
4.3-CAMVC-4.2-CE(Telas)	**-19.3**	**-34.7**	**-3.89**	**0.01**

4.4-DC-4.2-CE(Telas)	-24.4	-39.8	-9.03	0.00
4.4-DC-4.3-CAMVC	-5.1	-20.5	10.26	0.80
Tukey HSD: Esforço X Atividade – *Shop* 3				
Pares (Atividade)	***Diferença***	***Menor***	***Maior***	***P-valor (adj)***
4.2-CE(Telas)-4.1-CDUC	15.3	-3.6	34.3	0.13
4.3-CAMVC-4.1-CDUC	17.4	-1.5	36.4	0.07
4.4-DC-4.1-CDUC	-6.6	-25.6	12.3	0.73
4.3-CAMVC-4.2-CE(Telas)	**2.1**	**-16.8**	**21.1**	**0.99**
4.4-DC-4.2-CE(Telas)	-21.9	-40.9	-3.0	0.02
4.4-DC-4.3-CAMVC	-24.1	-43.0	-5.1	0.01

Para o *Shop* 2, a atividade da Codificação que apresentou o maior esforço foi CE (Telas)[34], contra CAMCV, com *P-valor* igual a 0.01. No *Shop* 3, não se pode rejeitar a hipótese de que CE(Telas) e CAMCV são os maiores consumidores de recursos. Ainda assim, CE (Telas) e CAMVC destacam-se contra a atividade 4.4 DC.

7.6 Respostas para as Demais Questões (4 – 8)

Este trabalho elaborou um conjunto de oito indagações sobre a existência de gargalos no PDS, sua tipificação e suas características. Nesta seção, serão apresentadas as respostas elaboradas a partir da análise *quali-quantitativa*. As primeiras três questões definiram os objetivos e nortearam a pesquisa na busca de evidências de gargalos no PDS. As respostas destas três primeiras questões foram apresentadas na Seção 7.4 e servirão como base para promover as respostas das outras 5 questões.

A validade das questões (*Q1, Q2 e Q3*) é uma prerrogativa para as repostas das demais questões (*Q4, Q5, Q6, Q7 e Q8*). As respostas produzidas para este segundo grupo de questões foram elaboradas a partir das observações realizadas e anotadas no *log-book* dos experimentos (*Exp1, Exp2* e *Exp3*), da análise qualitativa e da análise quantitativa sobre os dados dos experimentos 2 e 3.

7.6.1 Questão 4 (Q4): Existe um gargalo predominante?

Resposta (Q4): Sim!

Argumentação: Os resultados estatísticos mostraram que o artefato gargalo predominante é a Codificação e em segundo lugar, aparece o tipo de artefato Documentação.

---

[34] *CE (Telas): Codificação de Entrada (Telas); CAMCV: Codificação Arquitetura MVC (Model View Controller); CDUC: Codificação de Casos de Uso; DC: Depuração de Código.*
Observação: *A lista completa com o nome de cada atividade encontra-se no Apêndice J.*

7.6.2 Questão 5 (Q5): Existe relação entre restrições *(quali)* e os gargalos (*quanti*)?

Resposta (Q5) = Sim!

Argumentação: Conforme os registros do "*log-book"* e a análise *quali-quantitativa*, as dificuldades das máquinas em executar determinadas tarefas provocaram distúrbios no processo produtivo, acarretando em consumo exacerbado de recursos (tempo e pessoas) e com isso estabelecendo os gargalos do PDS.

7.6.3 Questão 6 (Q6): As intervenções influenciam no tratamento do gargalo?

Resposta (Q6) = Sim!

Argumentação: A análise *"quali-quanti"* mostrou que os artefatos "*Arquitetura* e *Teste*" sofreram o maior número de intervenções. Os resultados quantitativos também mostraram uma evolução na qualidade destes artefatos na 3ª e 4ª entrega, ou seja, após as principais intervenções que aconteceram entre as fases 2 e 3 do *PU-Like*. As intervenções realizadas sobre o artefato documentação, principalmente nas atividades relacionadas a análise e modelagem foram fundamentais para o progresso das LPs, nos dois *shops*.

7.6.4 Questão 7 (Q7): O gargalo requer mais esforço da máquina?

Resposta (Q7) = Sim!

Argumentação: Os resultados mostraram que os artefatos com tarefas gargalos consumiram mais recursos (Tempo e Pessoas), ou seja, mais máquinas trabalharam nas tarefas gargalos e por mais tempo até sua conclusão. É importante colocar que a exigência de mais esforço das máquinas, geralmente implica em esforço extra de toda linha de produção. Isto se verificou analisando quantitativamente quais máquinas trabalharam nas atividades gargalo.

7.6.5 Questão 8 (Q8): O gargalo está sempre associado à mesma máquina?

Resposta (Q8) = Não!

Argumentação: O gargalo está associado com os tipos de atividades que compõe o artefato produzido. Haja vista a ocorrência de tarefas gargalos de mesmo tipo em várias linhas de produção e com diferentes máquinas. Os registros do *log-book* mostraram que as restrições técnicas e as restrições comportamentais, tiveram forte influência em determinar os gargalos do PDS. E isto é mais um fator que corrobora com a negativa da questão Q8.

7.7 Considerações sobre a Análise Quantitativa

Os resultados obtidos com a análise quantitativa confirmaram as evidências sobre a existência de gargalos no processo de software, e que estes, estão diretamente relacionados com a capacidade de produtiva das equipes (*LPs*). Os resultados mostraram ainda, que os gargalos independem do domínio, da complexidade e das pessoas que compõe os times de desenvolvimento. Isto porque o estudo não revelou a existência de diferenças estatísticas nos resultados dos dois ensaios analisados, uma vez que nos dois casos foram trabalhados com equipes, domínios e complexidade diferente.

A constatação de que os gargalos são comuns entre as linhas, ou seja, se mantém entre as diferentes LPs do *shop* e entre as fases do *PU-Like* é outro fator importante, que pode colaborar com os projetistas e pesquisadores no delineamento e no escalonamento das tarefas em detrimento do gargalo. Isto é importante porque tratamentos específicos podem ser aplicados e com isso diminuir o impacto dos artefatos gargalos no processo produtivo de software.

8 CONSIDERAÇÕES FINAIS

"Quando pensamos, fazêmo-lo com o fim de julgar ou chegar a uma conclusão! Quando sentimos, é para atribuir um valor pessoal a qualquer coisa que fazemos."

Carl Jung

Um processo de software bem delineado é primordial para o desenvolvimento de um produto de software que atenda a todas as exigências e ao seu propósito. Encontrar o caminho para atingir estes objetivos, não é uma tarefa trivial, todavia, é um caminho possível e que pode ser iniciado através da identificação dos pontos de estrangulamentos no processo de desenvolvimento de software.

Entretanto, como recomenda o CMMI/SEI (2016), é fundamental que o PDS busque apoio em metodologias, normas, especificações técnicas e procedimentos que permitam gerenciar o processo em todos os níveis, garantindo assim a sua adaptabilidade, estabilidade e maturidade ao longo de um ciclo. Como consequência, obtém-se um produto final com qualidade[35] e que realmente atenda as expectativas de quem desenvolve e às necessidades de quem usa o software.

Esta pesquisa mostrou que é possível identificar e tipificar os gargalos no PDS e com isso estabelecer um caminho para direcionar o processo produtivo. Contudo, o tratamento dos elementos restritivos encontrados no PDS é essencial para que estes direcionamentos promovam melhorias que sejam de fato aplicadas e traduzidas em qualidade, robustez e confiabilidade do software.

A aplicação da metodologia usada neste estudo na indústria de software é um trabalho árduo, difícil, dispendioso e duradouro devido à complexidade envolvida, o dinamismo do ambiente e a necessidade de repetições para se ter resultados conclusivos mais confiáveis. Além disso, o custo adicionado ao projeto para identificar e tratar os gargalos produtivos talvez não agregue valor comercial ou vantagem competitiva suficiente ao produto de software, para que os gerentes de software adotem este caminho na indústria.

[35] *A qualidade referenciada ao longo desta pesquisa faz menção ao software construído de acordo com seu projeto, o que inclui: o contrato, o escopo, o plano, o processo, as especificações técnicas, as funcionalidades e a usabilidade. Ou seja, a qualidade aqui tratada refere-se apenas ao que é tangível.*

Porém, a replicação deste estudo em outros ambientes, (*p.ex. na indústria de software*) e a diferentes domínios, pode gerar um registro de informações ou conjunto de conhecimentos com a capacidade de direcionar melhor os esforços e atenuar as dificuldades encontradas no processo de desenvolvimento de software, e desta forma, gerar ou agregar valor ao produto final/software.

8.1 Considerações Sobre o Estudo Realizado

Embora esta pesquisa tenha sido apoiada no bom senso, na multidisciplinaridade e na observação contínua, mesmo assim deixou de cobrir vários pontos, como mostra o Capítulo 1, Seção 1.8, limitando e direcionando a pesquisa somente para a identificação e tratamento de gargalos no PDS, e executado em um ambiente acadêmico específico.

No entanto, julga-se que os resultados colhidos foram bastante satisfatórios com a exposição de problemas praticamente inexplorados no contexto do processo de desenvolvimento de software.

8.1.1 Visão Geral sobre a Pesquisa e os Resultados Observados

As medições "*quali-quantitativas"* e as anotações no livro de registros (*logbook)* dos *"quase-experimentos"* foram fundamentais para as percepções e as conclusões sobre os problemas do PDS. Isto implica que os objetivos principais da pesquisa foram atendidos, uma vez que foi possível identificar, tipificar e categorizar os gargalos de um sistema produtivo de software e através disso promover as respostas para as questões de pesquisa.

Nos três casos (*Exp1 e Exp2 e Exp3*), a TOC foi aplicada para corrigir as restrições locais, evitar o surgimento de novos gargalos em tempo de execução. Mas diferentemente de um sistema manufatureiro é inconcebível fazer com que o sistema opere de acordo com a vazão do gargalo. Isto porque as máquinas num *shop* produtivo de software além de não produzirem quantidades pré-definidas de artefatos sofrem muitas influências, tanto externas como internas, que podem ser diferentes para cada Linha de Produção em função das máquinas que ali operam.

As anotações no *logbook* apontaram diversos problemas envolvendo as máquinas/LPs dos três *Shops* revelados na Análise Qualitativa (*Capítulo 6*), principalmente em relação ao conhecimento técnico, que se mostrou muito diversificado e pouco

consolidado. Isto é até natural, quando considerado o ambiente em que os ensaios experimentais foram executados.

A agenda estabelecida, o tamanho das equipes, os domínios aplicados e o apoio ferramental utilizado são fatores que podem ter influenciado no desempenho e afetado os resultados observados. Contudo, a observação prolongada e repetida mostrou uma tendência da existência de determinados tipos de gargalos no PDS e elencou a TOC como uma ferramenta que pode contribuir com a identificação dos problemas que em torno do desenvolvimento de software e seu processo.

A análise de outros dados e o cruzamento das informações podem responder outras questões e evidenciar outros elementos restritivos/gargalos que restringem a máxima capacidade produtiva das LPs, inclusive se observados em outro cenário. Contudo, análises complementares e a realização de novos experimentos serão matérias para investigações futuras.

8.1.2 Relação entre as Abordagens *Quali-Quantitativa*.

A análise descritiva mostrou que a produtividade das LPs está condicionada às Restrições Técnicas (RT) e Comportamentais (RC), apresentadas no Capítulo 6. As restrições qualitativas estão associadas às máquinas e incidem diretamente nos resultados produtivos das LPs, implicando no surgimento de gargalos no PDS.

Embora as LPs tenham necessariamente que seguir o escalonamento das tarefas, dado pelo modelo *job shop* empregado, estas possuem autonomia na elaboração dos artefatos e no gerenciamento de seu tempo, respeitando obviamente a agenda de entregas imposta pelo *PU-Like*. Portanto, é natural que as restrições qualitativas provoquem diferentes perturbações no sistema, mesmo que as LPs atuem sobre um único domínio e um único processo. No entanto, a variedade de restrições *"quali"* não faz com que o gargalo flutue entre as LPS do *shop* e nem mesmo entre as entregas, como visto na análise das questões (*Q2 e Q3*). Isto porque os gargalos são categorizados neste estudo como entidades consumidoras de recursos e embora sejam provocados pelas questões qualitativas, a que as LPs estão sujeitas, o uso dos recursos são direcionados para os artefatos com maior tangibilidade, como o caso da Codificação e Documentação.

Outro fator importante observado foi à alta dispersão sobre os dados, principalmente sobre o Esforço. O esforço foi perturbado pela Dificuldade em dois

sentidos: Alto Esforço – algumas LPs relataram dificuldades e empregaram mais energia para a conclusão dos artefatos e Pouco Esforço – neste último caso, algumas LPs empregaram pouco esforço devido à dificuldade encontrada ou devido às deficiências técnicas. Para a maioria das LPs, observamos que a variação do esforço incide diretamente na produção, porém, mesmo em alguns *outliers* foi encontrado casos onde determinadas LPs empregaram muito esforço e teve produtividade baixa, como pode ser observado na correlação exibida na Seção 7.3.7 (análise descritiva) e no mapeamento produtivo (Análise Empírica) apresentado no Apêndice H, evidenciando a baixa correlação entre Esforço, Produção e Dificuldade.

Além das restrições internas, o PDS está sujeito também a perturbações externas. As perturbações externas promovem a "Síndrome do *Deadline* (Ribeiro *et al.*, 2017-b), que também interferem diretamente na produção das LPs" devido à priorização de outras tarefas em detrimento às tarefas do PDS. A Síndrome do *Deadline*, não gera necessariamente um gargalo no PDS, pois sua existência está condicionada ao ambiente de desenvolvimento e seu tratamento é um problema de gestão de projetos.

As anotações no *logbook* apontaram diversos problemas envolvendo as máquinas/LPs dos três *Shops* apontados na Análise Qualitativa (*Capítulo 6*), principalmente em relação ao conhecimento técnico, que se mostrou muito diversificado e pouco consolidado. Isto é até natural, quando considerado o ambiente em que os ensaios experimentais foram executados.

A agenda estabelecida, o tamanho das equipes, os domínios aplicados e o apoio ferramental utilizado são fatores que podem ter influenciado no desempenho e contribuído para os resultados observados. Contudo, a observação prolongada e repetida mostrou uma tendência da existência de determinados tipos de gargalos no PDS e elencou a TOC como uma ferramenta que pode contribuir com a identificação dos problemas em torno do desenvolvimento de software e seu processo.

A análise de outros dados e o cruzamento das informações podem responder outras questões e evidenciar outros elementos restritivos / gargalos que limitam a capacidade produtiva das máquinas, inclusive se observados em outro cenário. Contudo, análises complementares e a aplicabilidade de novos estudos serão matérias para investigações futuras.

8.2 Contribuições da Pesquisa

Esta pesquisa aplicou uma abordagem multidisciplinar no contexto do processo de desenvolvimento de software e apresentou um método para detectar a existência de gargalos no PDS. A pesquisa procurou ainda direcionar o esforço para que os resultados produzissem contribuições científicas e acadêmicas.

8.2.1 Contribuições Técnico-Científicas

O desafio de aplicar a experimentação em Engenharia de Software exigiu desta pesquisa, a criação de um método específico e inédito para identificar os gargalos no processo de desenvolvimento de software. O método desenvolvido é sem dúvida a contribuição técnica mais importante deste trabalho, pois explora tanto a parte qualitativa do PDS, quanto os resultados obtidos através de uma abordagem quantitativa. Além disso, o método em sua forma geral é aplicável a qualquer PDS e em qualquer metodologia (ágil ou tradicional / estruturada), isto porque o método se isenta da mandatoriedade do uso de uma determinada metodologia ou ferramentas de desenvolvimento.

A particularização do método proposto com enfoque voltado para atender as características e as necessidades desta pesquisa é outro fator que deve ser igualmente enfatizado, pois estruturalmente, também pode ser aplicado em outro PDS e inclusive em outro ambiente, ainda que seja necessário aplicar pequenos e devidos ajustes.

Além do método desenvolvido e aplicado nesta pesquisa, os resultados colhidos na análise "*quali-quanti"* mostrados nos Capítulos 6 e 7, apresentaram algumas contribuições técnicas significativas. Mesmo que esta pesquisa ainda careça de estudos complementares, acredita-se que os apontamentos e a tipificação dos gargalos encontrados são contribuições efetivas para o processo de desenvolvimento de software. Além disso, tanto o método quanto os resultados exibidos, podem contribuir para o direcionamento e para a condução de novas pesquisas sobre os problemas existentes na produção de software.

As lacunas identificadas na revisão da literatura e apresentadas na Seção 3.9, bem como as oportunidades de pesquisas listadas na Seção 3.10, expõem uma ampla frente de trabalho envolvendo a TOC com possibilidades de aplicação no PDS, sobretudo na identificação e tratamento de gargalos. O estudo secundário mostrou ainda a inexistência de trabalhos combinando o processo de software com outras áreas, abrindo ainda mais o leque de oportunidades de pesquisa no âmbito da Engenharia de Software.

Outro ponto importante a ser destacado, é que o esforço desprendido nesta pesquisa permitiu ainda a publicação de vários artigos em periódicos e conferências (listados no pré-texto) contribuindo cientificamente para a área de Engenharia de Software (ES).

8.2.2 Contribuições Didático-Pedagógicas

Este estudo desde o começo se preocupou prioritariamente com o aprendizado e o desempenho acadêmico dos alunos. Além disso, os experimentos produziram resultados satisfatórios para:

1. A melhoria da compreensão dos alunos sobre os pontos abordados na disciplina de Fundamentos de Engenharia de Software;
2. Uma melhor compreensão dos alunos sobre como construir um produto de software seguindo alguns padrões industriais;
3. O entendimento via observação experimental da existência de restrições no PDS e seus possíveis gargalos;
4. A visualização de alguns potenciais gargalos no processo de desenvolvimento de software;
5. Entendimento dos alunos sobre como empregar a Engenharia de Software na prática;
6. A inserção dos alunos de graduação em um ambiente de pesquisa e experimentação;
7. Observação de novos caminhos para estudos empíricos em Engenharia de Software.

8.3 Evidências Observadas

Durante o desenvolvimento da pesquisa e após a análise dos dados algumas evidências foram anotadas.

8.3.1 A Síndrome do Deadline

A síndrome do *deadline* foi primeiramente apresentada por (Ribeiro *et.al* 2017-b) a partir de observações feitas em ambiente de ensino de ES. Ainda assim, é seguro dizer que este fenômeno também se aplica em outros ambientes, como as fábricas de software, pois o ambiente e o PDS estão sujeitos a influências externas. A diferença pode estar no tipo de impacto que a síndrome pode causar no processo e no produto, uma vez que ela é

gerenciável, desde que os elementos causadores sejam conhecidos. Este fenômeno ainda requer mais estudos e mais observações, principalmente em situações observadas na indústria de software e com a influência de outros fatores externos.

Outro ponto a ser observado é que o gerenciamento nos ambientes industriais é feito com mais rigor porque além de disporem de mais recursos para executar o projeto possuem um processo definido e maturado, com equipes treinadas e pessoas estritamente dedicadas ao gerenciamento do processo e dos riscos associados.

A Síndrome do *Deadline* é um problema de gestão de projetos e seus efeitos dependem muito do planejamento, da execução e da equipe de desenvolvimento. Foi observado neste estudo que os problemas causados por este fenômeno tendem a ser diluídos com a maturidade do processo e com a experiência do responsável pela condução do projeto.

8.3.2 O(s) gargalo(s) do PDS

Os gargalos no processo de software estão diretamente relacionados com a capacidade de produção de cada equipe, (tratados no texto como linhas de produção), ou mais especificamente pelo conhecimento técnico e tácito das pessoas (referenciadas no texto como máquinas no processo produtivo) em ferramentas e metodologias adotadas em cada processo.

Esta constatação parece ser bastante óbvia, mas não pode ser afirmada com fidúcia, já que é muito difícil de ser observada olhando somente para uma equipe em um projeto de software. É preciso aplicar o domínio em várias equipes, e se possível for, replicar os ensaios para validar esta asseveração. Isto porque a capacidade na sua essência não é criteriosamente mensurada e sim somente avaliada de "longe" e de acordo com o andamento do processo. Além disso, a capacidade de produção das equipes são impactadas pelas restrições locais (*mini gargalos*), as quais, se não forem tratadas, podem provocar perturbações que impactam diretamente na produção.

As restrições locais não são necessariamente gargalos do processo de software. A experimentação mostrou que os "*mini gargalos*" afetam localmente e quando identificados, tratados e elevados causam pouco ou nenhum impacto no PDS.

8.3.3 Aplicação da TOC

Um dos pontos fortes desta pesquisa foi evidenciar a possibilidade de aplicação de uma metodologia amplamente usada na Engenharia de Produção ao processo de produção de software. A TOC foi aplicada em duas frentes: (1) o algoritmo da TOC – aplicado para identificação, controle e tratamento das restrições encontradas no PDS; e (2) o método TPC (Tambor, Pulmão e Corda) da TOC – aplicado juntamente com o algoritmo da TOC e o esquema de escalonamento de tarefas (DJSS) para controlar e coibir o surgimento de novos pontos de estrangulamento no sistema.

8.4 Trabalhos Futuros

A continuidade deste trabalho é essencial para que novas contribuições sobre a identificação de gargalos no PDS sejam produzidas. Portanto, é intensão dos pesquisadores realizar novos estudos envolvendo o tema abordado e convidar interessados em contribuir com pesquisas originais e correlatas com a temática desta tese.

A elaboração de um estudo que possa apontar os caminhos para o tratamento do(s) gargalo(s) com base nos resultados da análise quantitativa é uma necessidade primária que será trabalhada no sequenciamento desta pesquisa. Este próximo trabalho deverá compreender o conjunto de passos trilhados a partir da constatação do(s) gargalo(s) encontrados, das restrições (RC e RT) e dos tratamentos aplicados na análise qualitativa. A ideia é aplicar este conjunto de tratamentos em um novo ensaio, onde se espera obter resultados sobre como corrigir ou atenuar os efeitos do gargalo no PDS.

Acredita-se que este estudo tenha potencial para ser explorado em outros cenários, partindo principalmente dos pontos não cobertos listados na Seção 1.8. Outra possibilidade seria refazer os experimentos na tentativa de elucidar e corroborar com as evidências aqui encontradas.

Acredita-se que a principal lacuna deixada nesta pesquisa seja a não replicabilidade em outros ambientes, sobretudo na indústria de software, o que permitiria correlacionar às evidências encontradas nos dois ambientes (acadêmico e industrial) e apontar com mais segurança e precisão o(s) gargalo(s) do PDS. Além disso, a replicação dos ensaios em ambientes distintos abriria a oportunidade para a criação de um repositório de dados ou um *benchmark* para apoiar novos estudos na exploração dos gargalos no PDS.

Finalmente, pretende-se ainda, explorar as lacunas encontradas na Revisão da Literatura apresentadas no Capítulo 3 e disponibilizadas como oportunidades de pesquisa por (Ribeiro *et al*, 2017-a) e (Ribeiro *et al*, 2018).

8.5 Experiências Pessoais e Conhecimentos Adquiridos

Por mais de 20 anos tenho atuado profissionalmente em projetos de desenvolvimento de software como desenvolvedor, analista e consultor de sistemas. Há 16 anos que também atuo como docente em cursos de Ciências da Computação e Sistemas de Informação ministrando disciplinas como: Engenharia de Software, Projeto de Software e Análise de Sistemas. Durante todo este tempo vivenciei situações em que a equipe ou processo/projeto eram sacrificados em detrimento de uma necessidade particular do plano (ou da falta dele), do cliente ou da empresa desenvolvedora. Em nenhum dos casos vivenciados o problema era tratado ou investigado como deveria, pelo menos na minha percepção, e sempre surgiam questionamentos diante de um cenário que se fazia caótico, gradativamente, como: (1) O que provocou? (2) Quais as razões? (3) O que está errado no planejamento do projeto? (4) Por que as metas de desenvolvimentos não podem ser cumpridas?

Quando procurei o PPGI/UFRJ com o desígnio de ingressar no curso de doutorado, levei comigo estas indagações como proposta, pois sabia o que queria investigar, só não sabia como.

Ao participar de um simpósio sobre engenharia de produção (SIMPEP) em 2012, tive contato com uma metodologia amplamente aplicada para melhorar os resultados produtivos da indústria manufatureira. A Teoria das Restrições (TOC) surgiu então como uma boa e consistente alternativa para auxiliar na investigação sobre os problemas encontrados no PDS, pois além de elucidar o caminho a ser percorrido, a TOC também se apresentava como uma possibilidade real de atacar a raiz do problema e não somente os vários questionamentos encontrados na produção de software. Já que a TOC é uma ferramenta que exige que o curso da investigação seja direcionado para as causas (gargalos do sistema) e não para os "porquês" que surgem frente aos problemas nos sistemas produtivos.

Ao apresentar a proposta de investigar os gargalos do PDS apoiada pela TOC ao meu orientador, a primeira sugestão foi: fazer um estudo preliminar para saber se algum

trabalho correlato já não tinha sido protagonizado, dado a TOC ser muito estudada e aplicada em vários problemas, sistemas e ambientes.

A inexistência de registros envolvendo a TOC no PDS, observado nos estudos iniciais, fez com que direcionássemos os esforços para uma investigação maior – transformando-se em uma extensa revisão da literatura (que foi apresentada por (Ribeiro *et al.*, 2018)) sobre a TOC aplicada ao processo de desenvolvimento de software.

A realização deste estudo secundário proporcionou-me além da experiência em executá-lo, uma visão clara e direta da importância deste tipo de estudo em uma pesquisa, além de apontar várias frentes de investigação (Ribeiro *et al*. 2017-b) e (Ribeiro *et al*. 2018) relacionadas com a minha proposta inicial.

Após a execução do estudo secundário, o passo seguinte foi aprofundar os estudos sobre a TOC e sua aplicabilidade no PDS. Esta segunda fase da pesquisa revelou algumas exigências impostas pela TOC, evidenciado a multidisciplinariedade da pesquisa.

Isto foi muito importante porque pude aplicar alguns conhecimentos e experiências e ainda buscar coisas novas, como: (1) Engenharia de Software Experimental (ESE); (2) sistemas e processos produtivos da indústria de transformação; (3) a organização estrutural da TOC e sua aplicabilidade; (4) estudar uma nova vertente do modelo *job shop scheduling*, em sua variação dinâmica (JSD); (5) realizar experimentação em ambiente de desenvolvimento de software e (6) realizar o estudo em ambiente de ensino (com estudantes de Ciência da Computação).

Este último ponto, foi sem dúvida o desafio que me proporcionou mais conhecimento e experiência, uma vez que ao mesmo tempo em que pude investigar os problemas que cercam o PDS em um ambiente de desenvolvimento de software, pude também aplicar a docência e trocar conhecimento e experiências com os participantes. Esta experiência dual promoveu a fusão das duas frentes de atuação (acadêmica e técnica) as quais tenho me dedicado ao longo dos anos, doando meu esforço pessoal e profissional. Tudo isso me deixa uma grande certeza: continuarei explorando as novidades e oportunidades da academia e me esforçarei para aplicá-las tecnicamente na indústria de software.

8.6 Reconhecimento aos Colaboradores

É desejo particular encerrar este texto expondo minha gratidão às turmas: 2014/2, 2015/1 e 2015/2 da disciplina de Fundamentos de Engenharia de Software (FES) do Curso de Ciência da Computação do Departamento de Ciência da Computação (DCC) do Instituto de Matemática (IM) da Universidade Federal do Rio de Janeiro (UFRJ), por aceitarem participar desta pesquisa, pelo comprometimento, pelo empenho, pelo companheirismo e pela troca de experiência durante todo este processo.

REFERÊNCIAS BIBLIOGRÁFICAS

"Se Eu vi mais longe, foi por estar de pé sobre ombros de gigantes."

Isaac Newton

[1] ABNT (Associação Brasileira de Normas Técnicas). NBR/ISO 9000:2000 – Sistemas de gestão da qualidade e garantia da qualidade – Fundamentos e Vocabulário. Rio de Janeiro: ABNT/NBR, 2001.

[2] AMBLER, S. W.; A Manager's Introduction to The Rational Unified Process (RUP), Ambysoft; Toronto, Ontario; 2005.

[3] ARAÚJO JR, L. O.; Método de Programação de Sistemas de Manufatura do Tipo Job Shop Dinâmico Não Determinístico; Tese de Doutorado; Programa de Engenharia Mecânica; Escola Politécnica; Universidade de São Paulo – USP; São Paulo – SP; 2006.

[4] ASAN; S. S.; A Methodology Based on Theory of Constraints' Thinking Processes for Managing Complexity in the Supply Chain; Master Science Thesis; Technischen Universität Berlin; Berlin – DE; 2009.

[5] BABAR, M. A.; GORTON, I.; Software Architecture Review: The State of Practice, Computer, vol. 42, no. 7, pp. 26-32, July 2009.

[6] BALDERSTONE, S. J. ; MABIN, V. J.; A review of Goldratt's theory of constraints (TOC): lessons from the international literature", Proceedings of the 33rd Annual Conference of the Operational Research Society of New Zealand, Auckland - NZ, 1988.

[7] BAPTISTA, E. A.; LUCATO, W. C.; FORTUNATO, F, A P. S.; Profit optimization in machining service providers using principles of the Theory of Constraints; Technical Paper in Journal of the Brazilian Society of Mechanical Sciences and Engineering; November 2013, Volume 35, Issue 4, Springer Link; 2013.

[8] BASILI, V. R.; A Personal Perspective on the Evolution of Empirical Software Engineering, in Perspectives on the Future of Software Engineering, Jurgen Munch, Klaus Schmid; Eds. Springer-Verlag; 2013.

[9] BASILI, V. R.; The Role of Controlled Experiments in Software Engineering Research; Empirical Software Engineering Issues, LNCS 4336; Eds. Springer-Verlag, 2007.

[10] BASILI, V. R; ROMBACH, H. D.; SCHNEIDER, K.; KITCHENHAM, B.; PFAHL, D.; SELBY, R. W.; Empirical Software Engineering Issues: Critical Assessment and Future Directions, LNCS 4336, Springer, 2006.

[11] BASILI, V. R.; SELBY, R. HUTCHENS, D.; Experimentation in Software Engineering; IEEE Transactions on Software Engineering; (invited paper) vol. 12 no. 7; 1986.

[12] BASILI, V. R.; WEISS, D. M.; A Methodology for Collecting Valid Software Engineering Data; IEEE Transactions on Software Engineering; Vol. SE-10, No 6, pp 728-738; November-1984.

[13] BERANDER, P.; Using students as subjects in requirements prioritization. In Proceedings of the 2004 International Symposium on Empirical Software Engineering (ISESE'04), pages 167–176, Washington DC, 2004. IEEE Computer Society.

[14] BIOLCHINI, J. C. A.; MIAN , P. G.; NATALI, A. C. C.; CONTE, T. U.; TRAVASSOS, G. H.; "Scientific Research Ontology to Support Systematic Review in Software Engineering," *Advanced Eng. Informatics,* vol. 21, no. 2, pp. 133-151, Apr. 2007.

[15] BIOLCHINI, J.; MIAN, P.; NATALI, A.; TRAVASSOS, G. Systematic review in software engineering. Technical report, COPPE / UFRJ, Rio de Janeiro, 2005.

[16] BLAZEWICZ, J.; ECKER, K.; SCHIMIDT G.; WEGLARZ, J.; Scheduling in Computer and Manufacturing Systems; Ed. Springer – Verlag; Berlin; 1996.

[17] BRERETON, P.; KITCHENHAM, B. A.; BUDGEN, D.; TURNER, M.; KHALIL, M. Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain. Technical report, 2007.

[18] BURBECK, S; Applications Programming in Smalltalk-80™: How to use Model-View-Controller (MVC), 1987/1992.

[19] CARVER, J., JACCHERI, L., MORASCA, S., SCHULL, F.; Issues in using students as subjects in empirical studies in software engineering education. In Proceedings of the Ninth International Software Metrics Symposium (METRICS'03); IEEE Computer Society; Washington - DC, 2003(a).

[20] CARVER, J., JACCHERI, L., MORASCA, S., SCHULL, F.; Using empirical studies during software courses; Experimental Software Engineering Research Network, LNCS; 2003(b).

[21] CASAVANT, T. L.; KUHL, J. G.; A Taxonomy of Scheduling in General Purpose Distributed Computing Systems, IEEE Transactions on Software Engineering, Vol..14, no. 02; 1998.

[22] CHRISTOPHER, M.; Logistics and Supply Chain Management; 4th Edition (Update Kindle Edition); Ed. Financial Times Series/Prentice Hall; New York-NY; 2012.

[23] CMMI - Capability Maturity Model Integration; CMMI V1.3; SEI – Software Engineering Institute; 2010; Overview and Reprint 2016; CMU-Carnegie Mellon University; Pittsburgh-PA / USA.

[24] CONRADI. R., WANG, A.I.; Empirical Methods and Studies in Software Engineering, Experiences from ESERNET, Lecture Notes in Computer Science LNCS 2765, Ed. Springer Verlag; Berlin – Germany; 2003.

[25] DENNIS, A.; WIXOM, B.H.; Análise e Projeto de Sistemas; segunda edição; Ed. LTC; Rio de Janeiro- RJ; 2014.

[26] EASTERBROOK, S.; SINGER, J.; STOREY, M.; DAMIAN, D.; Selecting Empirical Methods for Software Engineering Research," Guide to Advanced Empirical Software Eng., pp. 285-311, Springer, 2008.

[27] FALBO, R. A. Integração de Conhecimento em um Ambiente de Desenvolvimento de Software. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro - RJ, 1998.

[28] FALBO, R.; BERTOLLO, G. A Software Process Ontology as a Common Vocabulary About Software Processes. International Journal of Business Process Integration and Management (IJBPIM); 2009.

[29] FERREIRA, P. V.; Estatística Experimental; Cap.02 Etapas de um de Experimento; CECA; UFAL; Maceió - AL; 2011.

[30] FRANCHI, C. M.; Controle de Processos Industriais: Princípios e Aplicações; 1ª. Ed.; Editora Érica; São Paulo - SP, 2011.

[31] GENBRUGGE, D.; EECKHOUT, L.; BOSSCHERE, K.; Accurate Memory Data Flow Modeling in Statistical Simulation; Proceedings of the 20th annual international conference on Supercomputing (ICS'06); Queensland, Australia, 2006; SIGARCH ACM Special Interest Group on Computer Architecture; ACM Digital Library; New York, NY, USA ; 2006.

[32] GIUNTINI, N.; DI GIORGI, W. A. B.; PIZOLATO, C. L.; PENTEADO, A.; XAVIER, J, S.; Teoria das restrições: uma nova forma de "ver e pensar" o gerenciamento empresarial. In: 2º Seminário de Contabilidade, 2002, Anais Eletrônico; São Paulo-SP; 2002.

[33] GOLDRATT, E. M.; A Corrente Crítica; Ed. Nobel, São Paulo - SP, 2006.

[34] GOLDRATT, E. M.; COX, J.; *A Meta*: Um processo de Melhoria Contínua; 2ª. Ed.; Editora Nobel; São Paulo; 2003 Reimpressão 2006.

[35] GOLDRATT, E. M.; It's Not Luck, Ed. The North River Press, 1993.

[36] GOLDRATT, E. M.; KISHIRA, Y.; CCPM e TOC Uma Revolução no Japão; Revista Mundo PM, Ed. 28, 2009.

[37] GUIMARÃES, S.K. C.; SOARES, U. S.; CORRÊA, D. C. L.; FIGUEIREDO, M. T. F.; SANTOS, R. C.; Metodologia de gestão de processos; Tribunal Superior Eleitoral; Editoração, Impressão e Distribuição e Publicações SGI (Seprov/Cedip/SGI); Brasília DF; 2012.

[38] GUPTA, M. C.; BOYD, L. H.; Theory of constraints: a theory for operations management; IJOPM - International Journal of Operations e Production Management, Vol. 28 Issue: 10, pp.991 – 1012; 2008.

[39] GUPTA, A. BHARDWAJ, A., KANDA, A.; Fundamental Concepts of Theory of Constraints: An Emerging Philosophy; World Academy of Science, Engineering and Technology 2010.

[40] HASGUL, S.; KARTAL, Z.; Analyzing A Drum-Buffer-Rope Scheduling System Executability Through Simulation; Proceedings in of the 2007 Summer Computer Simulation Conference (SCSC 2007), San Diego, California - US; 2007.

[41] HARWELL, M.R. Research design in Qualitative, quantitative, and mixed methods (Chap 10) in The Sage handbook for research in education: Pursuing ideas as the keystone of exemplary inquiry; 2nd *Edition; SAGE Publications; Thousand Oaks, CA;* 2011.

[42] HOLANDA FERREIRA, A. B.; Dicionário Aurélio da Língua Portuguesa, Editora Positivo, Curitiba-PR; 2014.

[43] HÖST, M., REGNELL, B., wsz\, C.; Using students as subjects - a comparative study of students and professionals in lead-time impact assessment. Empirical Software Engineering - an International Journal, 5(3):201–214, 2000.

[44] ISLAM, A. M. M.; GORSCHEK, T.; UNTERKALMSTEINER, M., FELDT, R.; PERMADI, R. B.; CHENG, C. K.; "Evaluation and Measurement of Software Process Improvement: A Systematic Literature Review," IEEE Transactions on Software Engineering, Vol. 38 Nr.02, 2012.

[45] JACOBSON, I.; BOCH, G.: RUMBAUGH, J.; The Unified Process; IEEE Software, vol. 16, no. 3, pp. 96-102, May-June 1999(a).

[46] JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. The Unified Software Development Process. Reading: Addison-Wesley, 1999(b).

[47] JURISTO, N.; MORENO, A. M.; Basics of Software Engineering Experimentation; Universidad Politécnica de Madrid; Publisher Springer-Verlag; New York-US; 2001.

[48] KALUS, G.; KUHRMANN, M.; Criteria for Software Process Tailoring: A Systematic Review; Proceedings of the ICSSP 2013 - International Conference on Software and System Process; pp. 171-180; ACM New York, NY; 2013.

[49] KIM, S.; MABIN, V. J.; DAVIES, J.; The theory of constraints thinking processes: retrospect and prospect; International Journal of Operations e Production Management; Vol. 28, No. 2 pp. 155-184, Emerald Group Publishing Limited; Wellington, New Zealand; 2008.

[50] KLEIN, R.; Scheduling of Resource Constrained Projects; Ed. Kluwer Academic Publisher; Norwell – Massachusetts, 2000.

[51] LARMAN, C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition; Ed. Addison Wesley Professional; 2004.

[52] LOURENÇO JR., A.; OLIVEIRA, L. C. V.; KILIMNIK, Z. M.; O Planejamento De Cenários como Aprendizado; SRJ - Future Studies Research Journal; vol. 2, nr. 1; São Paulo - SP; 2010.

[53] MAFRA, S. N.; TRAVASSOS, G. H. Estudos Primários e Secundários apoiando a busca por Evidência em Engenharia de Software: Relatório Técnico (RT-ES-687/06); Programa de Engenharia de Sistemas e Computação; COPPE/UFRJ. Rio de Janeiro - RJ; 2006.

[54] MABIN, V.; BALDERSTONE, S.; The World of the Theory of Constraints: A Review of the International Literature; St. Lucie Press/APICS Series on Constraints Management; Boca Raton – Florida; 2000.

[55] MANSOURABAD; I. B.; DANESHI; A.; PIZARD, A.; Using Theory of Constraints in selecting product mix; European Online Journal of natural and Social Sciences; Vol. 2; Nr.1 pp. 146-155; 2013.

[56] MARAS, J.; ŠERIĆ, L.; ŠTULA, M; UKIĆ, N.; Combining education, industry, and empirical studies in Software Engineering: an experience report; ECSAW '15 - Proceedings of the European Conference on Software Architecture Workshops; ACM Publication; New York, NY; 2015.

[57] MPS.BR - Melhoria de Processo do Software Brasileiro; Guia Geral MPS de Software - Modelo de Referência MPS para Software (MR-MPS-SW); SOFTEX; 2016.

[58] MURAUSKAITE, A.; AOMAUSKAS, V.; Bottlenecks in Agile Software Development Identified Using Theory of Constraints (TOC) Principles; Master thesis in Applied Information Technology; ISSN: 1651-4769; Department of Applied Information Technology or; Department of Computer Science; IT University of Gothenburg; Gothenburg - Sweden; 2008.

[59] NEWMAN, I.; BENZ, C.R; Qualitative-Quantitative Research Methodology: Exploring the Interactive Continuum; Southern Illinois University Press; Carbondale - IL; 1998.

[60] NBR/ISO/IEC 14598:2001; Software engineering - Product evaluation - Part 6: Documentation of evaluation modules; NBR/ISO/IEC; 2001; Reviewed and Confirmed in 2008.

[61] NBR/ISO/IEC 12207:2008; Systems and software engineering - Software life cycle processes; NBR/ISO/IEC; 2008.

[62] NBR/ISO/IEC 25010:2011; Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models; NBR/ISO/IEC; 2011.

[63] NBR/ISO/IEC 25020:2007-25024:2012; Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Measurement reference model and guide; NBR/ISO/IEC; 2007-2012.

[64] NBR/ISO/IEC 25040:2011-25041:2012; Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Evaluation process; NBR/ISO/IEC; 2011-2012.

[65] OUELHADJ, D.; PETROVIC, S.; A survey of dynamic scheduling in manufacturing systems, Journal of Scheduling, Vol. 12; 2009.

[66] PANDIT, S; Application Of Theory Of Constraints On Scheduling Of Drum-Buffer-Rope System; Second International Conference on Emerging Trends in Engineering (SICETE); IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE), 2009.

[67] PEDREIRA, O.; PIATTINI, M.; LUACES, M. R.; BRISABOA, N. R.; A Systematic Review of Software Process Tailoring; ACM SIGSOFT Software Engineering Notes; Volume 32 Issue 3; ACM; New York, NY; 2007.

[68] PFLEEGER, S. L.; Engenharia de Software: Teoria e Prática; 2ª edição; Ed. Prentice Hall, 2009.

[69] PINEDO, M.; Scheduling – Theory, Algorithms and Systems; Ed. Prentice Hall; Englewood Cliffs – NJ; 2002.

[70] PMI/PMBOK; Guia de Conhecimentos em Gerencia de Projetos (*PMBOK), 3ª edição. 2007 Project Management Institute*, Four Campus Boulevard, Newtown Square, PA 19073-3299 EUA.

[71] PRESSMAN, R. S.; MAXIN, B. R.; Software Engineering: A Practitioner's Approach; Ed., McGraw-Hill/Makron Books do Brasil, São Paulo, 2014.

[72] RAHMAN, S.; Theory of Constraints: A review of the philosophy and its applications, Perth, Australia International Journal of Operations e Production Management; vol.08 nr.04 pp336-355 1998.

[73] RHEE, S-H.; CHO, N. W.; BAE, H.; Increasing the efficiency of business processes using a theory of constraints; inf Sys Front; Springer-Science + Business Media, LLC; 2010.

[74] RIBEIRO, S. A.; Sistema Imune Artificial para O Problema de Escalonamento Job Shop; Dissertação de Mestrado. PPGI/CT/UFES, Vitória-ES; 2006.

[75] RIBEIRO; S. A.; ALVARENGA, A. G.; AHONEN, H.T.; Artificial Immune System to Job Shop Scheduling Problem; Proceedings in Brazilian Production Engineering Symposium; XIX SIMPEP; 2012.

[76] RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A.J; Bottleneck Identification in Software Development Processes: A Proposal Based on the Principles of the Theory of Constraints; Proceedings In the ICGSE'15; International Conference on Global Software Engineering; IEEE Xplorer Conference Proceedings; 2015.

[77] RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S. M.; SILVA, M. F.; Research Opportunities on the Application of the Theory of Constraints to Software Development Process; JSW - Journal of Software; IAP - International Academy Publishing. JSW-Vol 12. Nr. 4.; 2017-a.

[78] RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S. M.; SILVA, M. F.; A Síndrome do Deadline: Origem, Causas e Implicações no Processo de Desenvolvimento de Software; vol. 10, No. 2, pp.30-47; ISYS – Revista Brasileira de Sistema de Informação - SBC-Sociedade Brasileira de Computação; Rio de Janeiro-RJ; 2017-b.

[79] RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S. M. SILVA, M. F.; Um Metamodelo do Processo Unificado Utilizando a Ontologia de Fundamentação Unificada-B; ENCOSIS-Encontro Regional de Computação e Sistemas de Informação; Anais Encosis / SBC - pp. 13-22; Maio'17; Manaus-AM; 2017-c.

[80] RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; SILVA, M. F; Scheduling of Tasks in a Software Development Environment Using a Dynamic Job Shop Scheduling (DJSS). Proceedings of the XLIX Brazilian Symposium of Operational Research (SBPO); Blumenau – SC; 2017. Anais SBPO/SOBRAPO – ISSN 1518-1731; Agosto de 2017-d.

[81] RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; SILVA, M. F. – Bottlenecks Identification in Software Development Process: A Quali-Quantitative Approach; Proceedings of the *XVI* Brazilian Symposium on Human Factors in Computational Systems (IHC 2017); Anais IHC ISSN 2316-5318; ISBN 978-1-4503-6377-8/17/10; ACM Digital Library; ACM New York – NY- USA; DOI:10.1145/3160504.3160513; October 2017-e.

[82] RIBEIRO, S. A.; SCHMITZ, E. A.; ALENCAR, A. J. S.; SILVA, M. F.; Revisão da Literatura Sobre a Teoria das Restrições Aplicada em Processo de Desenvolvimento de Software: Revista IEEE América Latina; IEEE Xplore, DOI, ISI, ISSN 1548-0992; Volume 16, Issue X, 2018.

[83] RITZMAN, Larry P; KRAJEWSKI, Lee J.; Administração da produção e operações; Edição de Reimpressão; Ed. Prentice Hall, São Paulo – SP; 2004.

[84] ROGERS, P.; REIS, E. A.; SECURATO, J. R.; Teoria das Restrições e Decisões de Longo Prazo: O Caminho para a Convergência; Revista de Negócios, Blumenau, v. 11, n. 4, p.83-99, outubro/dezembro 2006.

[85] RUNESON, P.; Using Students as Experimental Subjects an Analysis of Graduate and Freshmen Student Data. In Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering (EASE 2003), pages 95–102, 2003.

[86] SCHACH, S. R.; Object-Oriented and Classical Software Engineering, 8th. Edition; Published by McGraw-Hill Press; New York – NY; 2011.

[87] SENGUPTA, S., DAS, K., VANTIL, R.; A New Method for Bottleneck Detection; Proceedings of the 2008 Winter Simulation Conference. IEEE; 2008.

[88] SCHULL, F., SINGER, J., SJØBERG, D.I.K; Guide to Advanced Empirical Software Engineering. Springer, London ; 2008.

[89] SINGER, J., VINSON, N.G.: Ethical issues in empirical studies of software engineering. IEEE Transaction on Software Engineering; 2002.

[90] SLACK, N., CHAMBERS, S., JOHNSTON, R; Administração da Produção; 2ª Edição; Editora Atlas; São Paulo – SP; 2002.

[91] SOMMERVILLE, I.; Engenharia de Software; 6ª Edição; Ed. Addison-Wesley, Londres, 2011.

[92] SULAYAMAM, M.; MENDES, E.; A Systematic Literature Review of Software Process Improvement in Small and Medium Web Companies; Advances in Software Engineering Communications in Computer and Information Science; Volume 59, 2009, pp 1-8 Springer Link; 2009.

[93] SVAHNBERG, M.; AURUM, A.; WOHLIN, C. Using students as subjects - an empirical evaluation; Proceeding ESEM'08 Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement; ACM Publications; New York - NY; 2008.

[94] TEIXEIRA, E. B.; A Análise de Dados na Pesquisa Científica: importância e desafios em estudos organizacionais; Revista Desenvolvimento em Questão; Editora Unijuí; Ano 1; número 2; Edição Semestral (Julho / Dezembro); 2003.

[95] TRAVASSOS, G. H; Experimentação em Engenharia de Software: Fundamentos e Conceitos; Anais do X Simpósio Brasileiro de Qualidade de Software; Curitiba – PR; 2011.

[96] TRAVASSOS, G. H.; Engenharia de Software Experimental / Templates - Notas de Aula; Experimental Software Engineering Group; COPPE/UFRJ; Copyright ESE Group; Rio de Janeiro-RJ; 2012.

[97] TRAVASSOS, G. H., GUROV, D.; AMARAL, E. A. G.; Introdução à Engenharia de Software Experimental: Relatório Técnico (RT-ES-590/02) Programa de Engenharia de Sistemas e Computação; COPPE/UFRJ; Rio de Janeiro – RJ; 2002.

[98] TRAVASSOS, G. H., BARROS, M.; Contributions of In Virtuo and In Silico Experiments for the Future of Empirical Studies in Software Engineering. In:

Workshop on Empirical Software Engineering: The Future of Empirical Studies in Software Engineering, Roma. 2003.

[99] THÖRNBLAD, K.; Mathematical Optimization in Flexible Job Shop Scheduling: Modelling, Analysis, and Case Studies; PhD Thesis; Department of Mathematical Sciences - Division of Mathematics; Chalmers University of Technology and University of Gothenburg ; Göteborg; Sweden – 2013;

[100] TROJANOWSK, J.; PAJĄK, E.; Using The Theory Of Constraints To Production Processes Improvement; Retrieved from 7th International DAAAM Baltic Conference" Industrial Engineering; 2010.

[101] WAINER, J; Métodos de pesquisa quantitativa e qualitativa para a ciência computação; Anais da XXVI Atualização em Informática da SBC - Sociedade Brasileira de Computação, 2007.

[102] WEISZFLOG, W.; Michaelis Moderno Dicionário da Língua Portuguesa, Editora Melhoramentos; São Paulo (SP), 2009.

[103] WHEELER, D. J.; Entendendo a Variação: A chave para administrar o caos; Ed. Qualimark; Rio de Janeiro – RJ; 2001.

[104] WHEELER, D. J.; EMP III - Evaluating the Measurement Process: Using Imperfect Data; Ed. Statistical Process Controls (SPC Press); Knoxville – TN; 2006.

[105] WOEPPEL, M. Introduction to Drum Buffer Rope (DBR);2008 pag. 1-6; <<http://pinnacle-strategies.com/articles/DBR.pdf>> Ultimo Accesso (Novembro, 2015).

[106] WOHLIN, C.; "Empirical Software Engineering: Teaching Methods and Conducting Studies", in Empirical Software Engineering - Dagstuhl Seminar Proceedings (LNCS 4336), pp. 135-142, edited by V. Basili, D. Rombach, K. Schneider, B. Kitchenham, D. Pfahl and R. Selby, Springer Verlag; 2007.

[107] WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A.; Experimentation in Software Engineering; Ed. Springer/Springer Science e Business; London-UK, 2012.

[108] VENANZI, D., SILVA, O. R.. Gerenciamento da produção e operações; 1ª. Edição; Ed. LTC; Rio de Janeiro – RJ; 2013.

[109] ZHOU, Z.; ROSE, O.; A Bottleneck Detection and Dynamic Dispatching Strategy for Semiconductor Wafer Fabrication Facilities; Proceedings of the 2009 Winter Simulation Conference (WSC); IEEE Publisher; 2009.

APÊNDICES

Apêndice A: Processo de Software: Conceitos Gerais

Um processo é um sequenciamento de eventos temporais com uma direção determinada, guiado por um modelo visando atingir um objetivo específico. Esta definição também pode ser usada para contextualizar o processo de software, obedecendo logicamente a suas especificidades.

Seguindo esta mesma linha de entendimento, Fuggetta (2000) define processo de software como um conjunto coerente de atividades, políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para desenvolver, obter e manter um produto de software.

Enquanto Falbo (1988) diz que um processo de software pode ser visto como o conjunto de atividades, métodos, práticas e transformações que guiam pessoas na produção de software, envolvendo: (1) análise de requisitos; (2) planejamento e execução do projeto; (3) codificação, (4) teste; (5) instalação; e (6) outras atividades correlacionadas que possa ser adotada na execução do projeto. Ainda segundo Falbo (1988), outra questão que envolve a elaboração de um processo de software é a sua adequação ao domínio de aplicação e ao projeto.

A cada projeto, o processo de software deve ser ajustado às especificidades da aplicação, a tecnologia a ser utilizada na sua construção e ao grupo de desenvolvedores. Acrescento os consumidores do produto final, se observamos as questões ambientais e os contextos regionais e culturais.

“Um processo eficaz deve, claramente, considerar as relações entre as atividades, os artefatos produzidos no desenvolvimento, as ferramentas e os procedimentos necessários, a habilidade, o treinamento e a motivação do pessoal envolvido” (Falbo, 1998).

Todo processo é essencialmente um consumidor de recursos, uma vez que as atividades dependem de recursos como, software, hardware e pessoas para serem operacionalizadas. Todas as saídas são produzidas executando estes recursos a partir de algum artefato de entrada. Desta forma, é importante que o processo seja conduzido com

base em um modelo robusto, estruturado, obedecendo a um ciclo de desenvolvimento pré-definido.

De acordo com as observações de Falbo (1998), a escolha de um modelo de ciclo de vida ou modelo de processo é o ponto de partida para a definição de um processo de desenvolvimento de software. Um modelo de ciclo de vida deve organizar as macro atividades básicas, estabelecendo precedência e dependência entre as mesmas.

O modelo do processo define o ciclo de desenvolvimento e sua aplicação bem como o conjunto de atividades a serem sequenciadas, estabelecendo assim uma comunicação e um inter-relacionamento entre as atividades formando um ciclo iterativo e incremental. A Figura 1 a seguir, ilustra um modelo de processo genérico cíclico e incremental, tal qual representado pelo processo unificado, proposto por (Jacobson et.al 2009a) e (Jacobson et.al. 2009b). Este modelo fornece o arcabouço do processo desenvolvido nesta pesquisa.

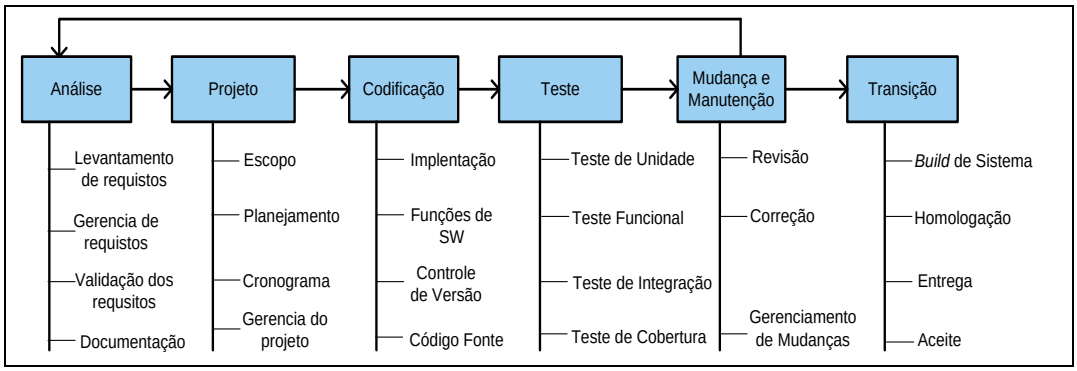


Figura 1: Exemplo de processo de software

Apêndice B: Contextualização do Processo Unificado

A literatura dispõe de uma ampla fonte de referências sobre o Processo Unificado (PU). Alguns trabalhos são clássicos e fortemente citados por outros autores, como é o caso dos trabalhos de (Jacobson et.al, 2009a) e (Jacobson et.al, 2009b), criadores do PU. Por esta razão, somente serão citados nesta proposta estes dois trabalhos sobre o PU, pois os mesmos apresentam toda a estrutura e fundamentação do Processo Unificado, principalmente (Jacobson *et.al*, 1999b) que é o livro do PU.

O Processo Unificado (PU), é um modelo iterativo e incremental de desenvolvimento de software, que combina as atividades necessárias para transformar os requisitos de usuários na construção de software (Jacobson, *et.al*, 1999a), (Jacobson, *et.al*,

1999b). As etapas do processo são combinadas em ciclos iterativos de desenvolvimento que permite acompanhar e gerenciar o desenvolvimento ao longo do ciclo de vida do processo.

De forma geral o Processo Unificado (PU) é representado por um conjunto de fases e disciplinas, como pode ser observado na Figura 2, abaixo.

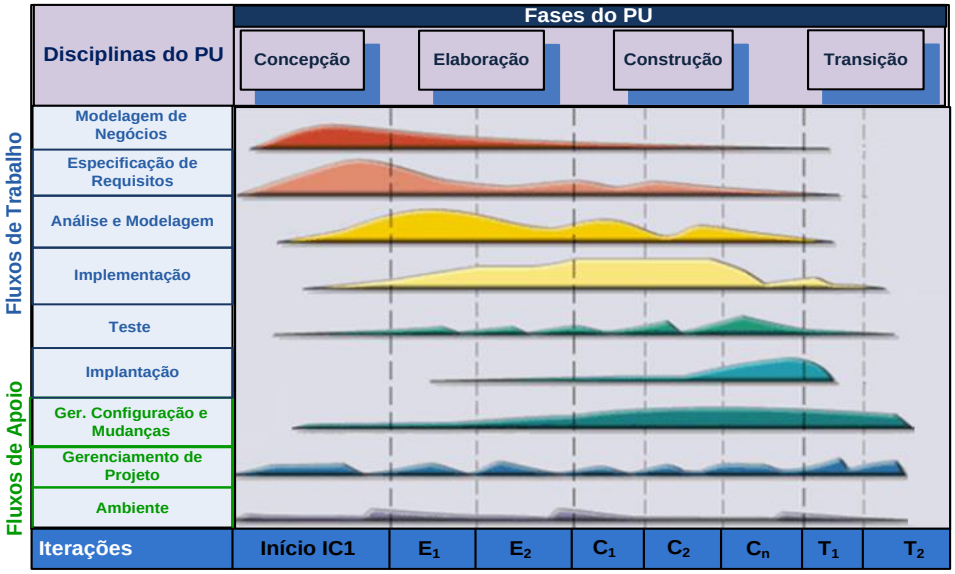


Figura 2: Esquema geral do Processo Unificado (Adaptado de Jacobson, et.al, 1999b).

Cada uma destas fases envolve toda a extensão do desenvolvimento determinado por três características básicas e fundamentais (Jacobson, *et.al*, 1999b), como: (1) ser orientado por casos de uso: Os casos de uso são utilizados para capturar e definir os requisitos funcionais do sistema, além de definir as metas e guiar o trabalho em cada micro interação; (2) ser centrado na arquitetura: A arquitetura é modelada através dos aspectos estáticos e dinâmicos do processo, fornecendo uma estrutura que permite gerir o processo em cada interação; e (3) ser iterativo e incremental: cada micro incremento correspondem a uma iteração e resulta em um ciclo incremental no desenvolvimento do software. As iterações referem-se aos passos e a evolução do produto a ser desenvolvido em cada fase do processo.

A principal finalidade do PU é evidenciar a necessidade de atribuições de tarefas a grupos ou indivíduos que necessariamente estejam envolvidos diretamente no desenvolvimento do projeto. Isto é feito através da identificação das etapas / iterações, dos marcos do projeto, dos objetivos do projeto e dos artefatos que serão gerados e utilizados durante o processo. É imperativo que este procedimento de identificação seja realizado na

etapa de concepção a partir da interação inicial e seus micros incrementos, dos quais se constituem os principais aspectos do processo de desenvolvimento em cada fase do Processo Unificado[36].

Apêndice C: Elaboração Conceitual do Planejamento do Experimento

C.I. Definição do Escopo

Como observado anteriormente, o escopo define o arcabouço do experimento e modela a proposta em função dos objetivos. O propósito do escopo é assegurar que os aspectos importantes do experimento sejam definidos antes do planejamento e da execução (Wohlin *et al.*, 2012). Portanto é necessário que um *template* seja criado para conduzir o experimento de acordo com a orientação das definições do escopo.

O escopo nasce da ideia geral do experimento que passa por refinamentos sucessivos até que os objetivos a serem alcançados sejam identificados. A Figura 3 a seguir, ilustra o processo de construção do escopo.

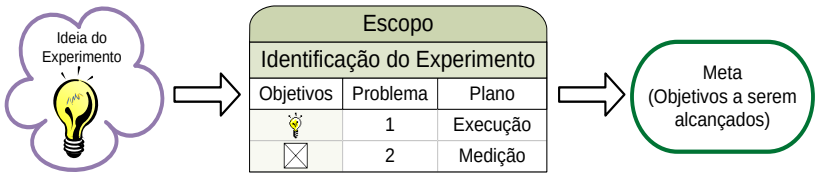


Figura 3: Processo de construção do Escopo (adaptado de (Wohlin *et al.* (2007))

Conforme, Wohlin *et al.* (2012) e Carver *et al.* (2003a), a definição do escopo é uma responsabilidade do pesquisador, principalmente quando se trata de pesquisa em ambiente de aprendizagem. O pesquisador deve elaborar o escopo a partir de uma ideia inicial formulando questões e afirmações sobre o problema a ser atacado, sobre os planos de ações a serem desenvolvidos e levando em conta ainda o ambiente em que será executado o experimento. O escopo do experimento proposto foi elaborado em conformidade com os itens apresentado na Seção 4.1 e descritos nas seções a seguir.

[36] *Processo Unificado: Em razão de o PU ser amplamente difundido e facilmente encontrado na literatura, não será apresentado neste trabalho conceitos e definições detalhadas sobre o PU. O leitor pode encontrar informações mais detalhadas nas fontes consultadas e referenciadas neste trabalho, ou observar as ilustrações que por si só são auto definidas e detalhadas.*

C.II Definição da Proposta

Com esta proposta pretende-se simular um ambiente real de desenvolvimento de software em um ambiente de aprendizagem de ES. O cenário e os participantes escolhidos foram os alunos da disciplina de Fundamentos em Engenharia de Software (FES) do Curso de Ciência da Computação do Departamento de Ciência da Computação (DCC) da Universidade Federal do Rio de Janeiro (UFRJ). É importante ressaltar que a ideia de realizar este estudo, nasceu a partir de uma forte revisão sistemática envolvendo a Teoria das Restrições (TOC) aplicada ao Processo de Desenvolvimento de Software (PDS).

Com a RS foram identificadas várias lacunas envolvendo esta área que viera a implicar na realização desta proposta. Identificou-se na TOC uma potencial metodologia para identificar e tratar os gargalos no PDS a partir do algoritmo da TOC e da metodologia *Drum, Buffer and Rope* (DBR). O processo foi conduzido seguindo a notação do PU.

C.III Definição dos Objetivos

O objetivo primário deste trabalho é definir um método para identificar gargalos em processo de desenvolvimento de software e promover ações para tratar os gargalos identificados, quando possível. Com isso, tem-se o intuito de maximizar a produção em relação à quantidade de artefatos a serem produzidos pelos times de desenvolvimento. Um segundo objetivo é a redução do *lead time*[37], através de um processo, ajustado, melhorado e maduro. Este objetivo poder ser alcançado através das observações e das análises realizadas dos dados coletados. Formular um problema que oriente a produção e contribua para a melhoria do processo de software, também é um desejo que pretende-se alcançar com esta pesquisa.

C.IV Definição do Objeto de Estudo

De acordo com Wohlin *et al.* (2012) o objeto de estudo é uma entidade a ser estudada em um experimento. O objeto de estudo pode ser um produto, um processo, um recurso, um modelo, uma métrica ou uma teoria.

O objeto de estudo deste experimento é o Processo de Desenvolvimento de Software (PDS). Para observar, avaliar e mensurar o PDS foi elaborado um ambiente para

[37] *Lead time: tempo necessário para que um produto evolua da concepção ao lançamento, do pedido à entrega (...) inclui também o tempo de processamento e o tempo de fila (Christopher, 2012).*

simular uma linha de produção de uma fábrica, onde cada linha é responsável pelo desenvolvimento de um produto em série, ou seja, o mesmo produto foi desenvolvido por todas as linhas de produção, seguindo as mesmas regras, o mesmo processo e o mesmo modelo.

A aplicação da TOC no PDS também é uma frente de estudo a qual aliou-se ao *PU-Like*, que foi o modelo adotado. A Figura 4 apresenta o modelo do processo desenvolvido para o PDS.

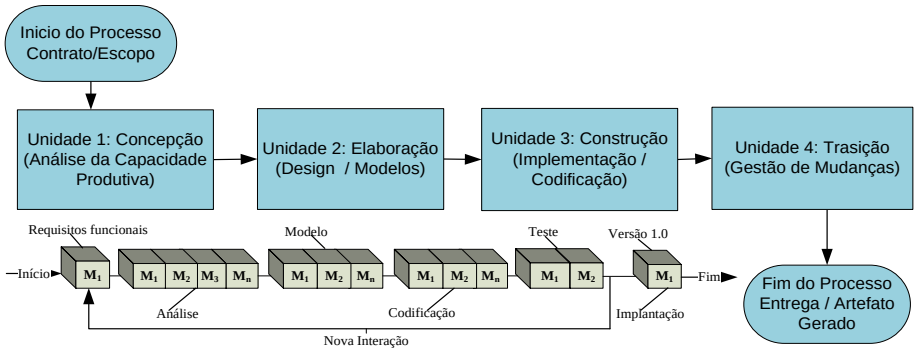


Figura 4: PDS desenvolvido a partir do PU.

Os estágios representam necessariamente as fases do *PU-Like*, destacado em azul na Figura 9, enquanto que os artefatos são as entradas e saídas do sistema produtivo em cada estágio, representados pelas formas em branco, conforme ilustrado pela Figura 5.

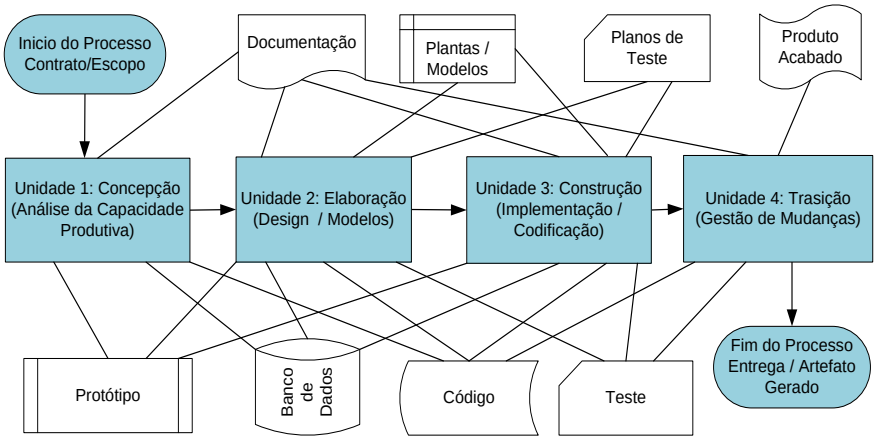


Figura 5: Ambiente de simulação do PDS

Diferentemente do processo de produção de uma fábrica, o PDS é altamente dinâmico, com artefatos sendo consumidos e gerados em vários estágios do processo. Este dinamismo requer maior observação e controle da produção e é provocado devido às perturbações as quais o ambiente está sujeito e pelas variáveis independentes que alimentam o processo.

C.V Perspectivas

A perspectiva reflete o ponto de vista do pesquisador em relação ao experimento e em consonância com o que foi planejado, observado, coletado e interpretado. A perspectiva em volta desta pesquisa é que a partir da observação, da análise e da interpretação dos dados seja possível identificar os gargalos no processo de desenvolvimento de software, tratá-los e reportar os resultados obtidos.

C. VI Contexto

A contextualização define o ambiente em que experimento será executado, apresentando resumidamente as peças utilizadas no estudo, como: número de participantes, artefatos utilizados e produzidos, tamanho das equipes, domínio da aplicação e outras variáveis que possam ser identificadas e contextualizadas. Para melhor apresentar os recursos e o formato utilizado, apresenta-se abaixo uma tabela construída com as informações contextuais identificadas.

C.VII Desenvolvimento do Processo de Produção

O processo é um instrumento de gerenciamento das atividades desenvolvidas que integram o produto, neste caso em especial o produto de software. O processo foi moldado a partir do escopo e do cenário com a finalidade de guiar o desenvolvimento do experimento em buscas dos resultados em volta da teoria levantada. Para desenhar um processo que representa o ambiente produtivo de uma fábrica, foi preciso buscar o conhecimento na literatura sobre administração e processos de produção.

C.VIII Execução do Experimento

A experimentação foi executada em conformidade com a metodologia, o processo experimental, o método e o processo de software definido e foi organizado conforme o planejamento estabelecido no *template* de desenvolvimento. As seções a seguir apresentam os principais itens observados como necessários para rodar o experimento.

C.IX Definição das Equipes de Desenvolvimento

Neste trabalho, cada equipe corresponde a uma máquina, em uma analogia ao processo produtivo de uma fábrica. As máquinas são constituídas de peças produtivas, que são as pessoas/participantes que irão executar as tarefas. Cada equipe (máquina) recebe os

mesmos ativos de entradas (matéria prima e recursos a serem consumidos) e deve produzir um conjunto de artefatos similares, que compreende a totalidade, a especificidade e a qualidade do produto. A união de todos os artefatos completam o produto acabado ou software desenvolvido após o cumprimento de todas as etapas, segundo a metodologia adotada e guiada pelo PU.

A montagem das equipes de desenvolvimento ficou a cargo dos participantes com a obediência ao tamanho máximo de 4 componentes e o mínimo de 2. A flexibilidade na quantidade de componentes que compõe cada equipe, com variação de até dois estudantes foi intencional e utilizada para refutar ou comprovar que o tamanho da equipe necessariamente não é um gargalo. Em todas as três rodadas, as máquinas as máquinas processaram o mesmo tipo de tarefa para gerar o mesmo produto de software, diferenciando apenas o domínio do problema.

O *shop* do experimento 1 foi desenvolvido com 9 máquinas/equipes, com um número máximo 4 componentes e um número mínimo de 2.

Na segunda rodada do experimento, o shop construído teve um total de 7 máquinas. As equipes foram montadas com o número máximo de 4 componentes e um número mínimo de 3 componentes. Isto foi importante porque quebrou a regra do experimento 1 com variação de apenas um elemento em relação ao tamanho das equipes e isso foi usado para sabermos se o total de indivíduos realmente é um elemento restritivo para o processo de desenvolvimento de software.

O *shop* montado para o experimento 3, contou com 4 máquinas sendo três máquinas com 4 componentes e 1 uma máquina com 3. Esta composição será importante para validar a observação sobre a influencia do tamanho das máquinas levantado no experimento 2.

O tamanho do shop não ameaça a validade desta pesquisa, pois cada máquina é única no *shop* e a produção é avaliada individualmente.

C.X Definição de Itens Estruturais para o Desenvolvimento

Como a ideia é representar a linha de produção de uma fábrica, com um conjunto específico de máquinas processando o mesmo material, mas sem desconsiderar o ambiente dinâmico que envolve o processo de desenvolvimento de software, foi preciso usar uma metodologia para guiar o desenvolvimento.

O PU foi eleito para desempenhar esta tarefa. Foram promovidas algumas alterações e adaptações no PU para atender as necessidades desta pesquisa, como vimos na subseção 2.3.2, o qual foi denominado de *UP-Like* (*Unified Process Like*). As mudanças não foram tão significativas, *i.e*, a estrutura do PU não foi alterada, somente ajustada em cada fase para atender a necessidade particular desta pesquisa.

A mudança mais "drástica" foi na fase de concepção do PU, pois como a pesquisa se desenvolve em um ambiente acadêmico com aprendizes de Engenharia de Software, é necessário que algum conteúdo seja ministrado antes do experimento ser iniciado. Desta forma, foi dispensado análise de custo e de viabilidade do projeto conforme recomenda (Alencar e Schmitz, 2012). Assim, partiu-se do princípio que o projeto é economicamente viável e que existem os recursos necessários para desenvolvê-lo respeitando os conceitos técnicos e de qualidade adotados para o projeto.

A qualidade do projeto idealizado e consequentemente do software a ser produzido está associado ao processo de desenvolvimento e ao conjunto de artefatos gerados em cada fase do processo pelas máquinas até o fim do ciclo de desenvolvimento, com a documentação do sistema, o projeto arquitetural (modelagem com UML) e o MVC *(Model View Controller)*, a codificação em JAVA, o Banco de Dados (com o MySQL) e os testes unitário e funcional realizados.

Ao modelo proposto, foram inseridos ainda os 5 passos da TOC, a fim de revisar, identificar e corrigir os problemas, em um ciclo melhoria contínua. Embora o conteúdo da TOC tenha sido ministrado regularmente durante o experimento, os cincos passos da TOC foram utilizados como mecanismo de controle do experimento e foi aplicado pelo pesquisadores/avaliadores e sem conhecimento das equipes de desenvolvimento.

C. XI Direcionamento do Estudo e dos Trabalhos da Equipe.

Organizado e delineado pelo *template*, todos os experimento seguiram necessariamente a mesma ordem de execução.

Todos os experimentos foram executados ainda com os mesmos requisitos e os mesmos critérios de construção e avaliação, como definido nos tópicos a seguir:

Apresentação do Estudo de caso: Apresentação expositiva e escrita do estudo de caso com o domínio do problema e as regras básicas de construção;

Apresentação do PDS: Apresentação do processo de desenvolvimento, metodologias e ferramentas;

Apresentação das ferramentas: apresentação e sugestão das a serem usadas para construir os artefatos a serem entregues e consequentemente o produto;

Aplicação do questionário de FCS: foram aplicados em cada experimento dois questionários de avaliação do conhecimento em momento distintos com a intenção de obter o grau de conforto/conhecimento dos participantes com relação à metodologia e ferramentas a serem utilizadas.

Direcionamento das entregas: cada grupo foi orientado com relação ao que fazer o que entregar e como entregar os artefatos em cada fase;

Apresentação do Calendário: O calendário foi elaborado em função do calendário acadêmico, e norteia o desenvolvimento com os prazos pré-definidos de cada entrega;

Apresentação dos critérios de avaliação: apresentação dos critérios compreende da avaliação do material produzido, em relação à qualidade e completude e o os critérios de avaliação acadêmica para compor a nota dos alunos na disciplina de fundamentos de ES.

C.XII Definição dos Entregáveis

Os entregáveis corresponde a um conjunto de artefatos que devem ser construídos pelas máquinas. Os artefatos definidos são: Arquitetura; Banco de dados; Codificação; Documentação e Testes. Os artefatos são compostos por um conjunto de 27 tarefas, conforme apresentado no Apêndice J, definidos principalmente pelos itens descritos a seguir.

- Plantas UML (*Unified Modelling Language*): são os diagramas de domínio modelados com a UML. Definimos apenas os diagramas diagrama de caso de uso, de classes, sequência e atividades, pois entendeu-se que estes são suficientes para representar o domínio de negócio, especificamente e necessariamente para este estudo. Os diagramas devem ser acompanhados de uma descrição detalhada dos mesmos para compor a documentação do sistema;
- Arquitetura: O modelo desenvolvido com base no MVC *(Model, View, Controller)*, o Diagrama de sequência do MVC, apresentado a troca de mensagens através das

operações entre as camadas do sistema, o diagrama de classes do MVC e o diagrama de pacotes da arquitetura decomposta do sistema;

- Banco de Dados: construção de um banco de dados em My SQL de acordo com o domínio especificado, contendo: o modelo lógico/relacional, os serviços de configuração e administração do BD, As tabelas de dados, os *scripts* SQL, o DAO (*Data Access Object*), que integra a camada de aplicação com os serviços, estabelecendo a comunicação entre sistema e BD;
- Aplicação em JAVA: a aplicação envolve o conjunto de telas, classes, funções do sistema, e o motor MVC obrigatoriamente desenvolvido em JAVA para rodar em *desktop;*
- Teste de Software: os testes adotados foram o teste de unidade ou unitário e o teste funcional. Para cada caso de teste as equipes devem implementar as classes testadoras utilizando o *Junit* e o *DBUnit*, respectivamente ou com outro *framework* similar.
- Documentação do software: é um relatório técnico contendo todo o material produzido pelas máquinas bem como a descrição de cada peça produzida conforme os itens acima.

C.XIII Ferramentas é Técnicas Adotadas

Para definir os meios técnicos e ferramentas aplicadas ao desenvolvimento do produto, foi realizado um questionário de FCS, para saber o nível de conhecimento das equipes. Os resultados foram usados na tomada de decisão dos recursos que seriam utilizados para a construção do software proposto em cada um dos três experimentos. Abaixo apresenta-se estes recursos.

Versionador: Cada grupo usou o versionador que conhece decidido internamente pelo grupo. Só estou cobrando a versão final em cada fase.

Linguagem de Programação: Todos os grupos desenvolveram em Java, com a IDE Eclipse.

Banco de dados: todos os grupos usaram o banco de dados sugerido. DB My SQL (versão *free*). Alguns grupos começaram e tentaram implementar o BD em outro ambiente/ferramenta. Houve interferência do Interlocutor e cobrou-se o uso da mesma ferramenta / ambiente.

Ambiente de Modelagem (UML): Foi sugerido uma ferramenta *free* e de fácil uso (ARGO UML), embora com limitações de recursos. Praticamente todos os grupos estão desenvolvendo o modelo neste ambiente. Alguns grupos desenvolveram em ambientes que já estão familiarizado. Não houve interferência neste aspecto.

Testador: Foi sugerido e instruído o uso do *Junit* com o Eclipse para os teste de Unidade. Porem um grupo sugeriu usar uma ferramenta/ambiente que já conhece e inclusive já tinha aplicados alguns teste já nesta primeira fase. Não houve objeção quanto ao uso por parte do interlocutor. Os demais grupos, acataram o uso do *Junit*.

Testador de BD: Foi sugerido e instruído o uso do *DBUnit*:

C.XIV Apresentação do Trabalho

Apresentação envolve os meios e os critérios de divulgação do experimento e dos resultados colhidos. O objetivo com a apresentação do experimento e seus resultados é contribuir com a comunidade acadêmica e industrial. A própria tese desenvolvida e documentada nesta pesquisa é um meio de apresentação. Outros meios de apresentação inclui: divulgação em seminários, simpósios e colóquios científicos e publicação de artigos em anais e periódicos.

C.XV Definição de Meios de Formatação da Apresentação do Experimento

É importante que a apresentação do trabalho seja formalmente definida e formatada conforme recomenda as boas práticas de construção de relatórios, teses, artigos e apresentações. Neste específico caso, os meios de formatação utilizados foram às normas técnicas da ABNT[38] e o modelo desenvolvido pelo PPGI/URFJ[39]. Com relação aos artigos construídos, os meios de formatação e apresentação seguiram os *templates* de cada periódico.

C.XVI Relato das Evidências Observadas

O relatório final deve cobrir todas as evidencias observadas e os resultados que levaram as mesmas, contendo ainda o arranjo textual teórico e explanatório das evidências e dos dados respectivamente. Os resultados bem como as evidências devem ser

[38] *ABNT – Associação Brasileira de Normas Técnicas*
[39] *PPGI/UFRJ – Programa de Pós Graduação em Informática da UFRJ – Universidade Federal do Rio de Janeiro*

formatados conforme a necessidade do objeto de apresentação ou conforme a melhor disposição com o intuito de melhor visualização pelos leitores ou expectadores.

C.XVII Meios de Apresentação dos Resultados

Em geral os meios de apresentação utilizados para reportar um experimento científico são os periódicos, o que inclui principalmente os jornais e revistas especializadas. A apresentação em conferências, congressos e simpósios também é vista como uma boa forma de divulgação dos resultados de uma pesquisa.

Toda a produção e os resultados gerados neste trabalho serão divulgados usando tanto os periódicos quanto os eventos técnicos relacionados com a temática da pesquisa. O resultado final será apresentado ainda para uma comissão técnica de avaliadores e aberta ao público. Desta forma, espera-se que a apresentação da pesquisa tenha um alcance satisfatório e com isso contribuir com pesquisadores, estudantes de graduação e pós-graduação e com a indústria.

Apêndice D: Apoio Ferramental E Recursos Utilizados no PDS

Para a construção do produto de software e para apoiar o processo de desenvolvimento, foram utilizados alguns recursos técnicos, conforme apresentado nas seções seguintes.

D.I Linguagem de Programação

Adotou-se a linguagem JAVA. O Objetivo desta escolha está diretamente relacionado ao produto de software a ser desenvolvido em cada rodada do experimento. A escolha também foi direcionada devido ao fato da linguagem ser comumente conhecida e difundida no meio acadêmico, por ser portável e por possuir padrões de desenvolvimentos de código bem definido. Acredita-se que esta ultima característica é importante para o objetivo desta pesquisa que é identificar gargalos no processo de software.

D.II Linguagem de Modelagem

Para modelar e gerar as plantas do software foi utilizado a UML. A escolha da UML está diretamente associada as suas características e a forte correlação com a linguagem JAVA através do paradigma de orientação a objetos.

D.III Ambiente de Desenvolvimento do Software:

Foi adotado a IDE (*Integrated Development Environment*) Eclipse: A escolha desta IDE está associada diretamente aos recursos que a mesma possui para o desenvolvimento de código JAVA. Além disso, o Eclipse possui um conjunto de *plugins* que auxiliam e facilitam a construção do sistema. Por mais que algumas equipes tenham sugerido o uso de outras IDE, o desenvolvimento foi fixado no Eclipse[40] por questões de padronização, correção e controle do desenvolvimento.

Serviços de Banco de Dados: Para organizar e armazenar os dados, foi adotada a versão freeware do MySQL. Como o desenvolvimento do experimento foi realizado em um ambiente acadêmico, adotou-se esta ferramenta afim de não onerar o projeto. Outro fator importante é que o MySQL atende suficientemente os objetivos desta proposta e dos projetos desenvolvido nas 3 rodadas do experimento. O MySQL é uma ferramenta robusta que suporta inclusive projetos maiores que requerem maiores recursos de armazenamento e administração de dados.

D.IV Ferramentas de Teste

Para aplicar o Teste Unitário sugeriu-se o *JUnit*. Estas ferramentas, são facilmente integráveis com a IDE Eclipse, principalmente o *JUnit* que é plug-in desenvolvido para a IDE Eclipse, mas que também hoje já está integrada com outras IDE's, como o *Net beans*, por exemplo.

D.V Arquitetura do Software

Como arquitetura do projeto de software foi definido o MVC (*Model, View, Controller*). O MVC é um modelo amplamente usado para formatar uma arquitetura no desenvolvimento de software. Inicialmente proposto por (Burbek 1987/1992)[41] o arranjo estrutural do MVC permite separar o modelo de representação da informação da forma de interação com usuário e o software em si. O modelo (*model*) é o conjunto dados da

[40] *Eclipse é marca registrada da IBM Corporation protegida por copyright. JAVA e MySQL são marca registrada da Oracle Corporations protegidas por copyright.*

[41] ***Burbek 1987-1992:*** *O artigo original foi publicado em 1987 e republicado em 1992. Em 2012 o artigo foi publicado eletronicamente em formato .xml no repositório de domínio público da CSI - Computer Science Illinois / Illinois University - Champaign-Urbana. O artigo é protegido por Copyright © 1987, 1992 by S. Burbeck, permission to copy for educational or non-commercial purposes is hereby granted.*

aplicação, as regras de negócios, o arranjo lógico e funções que compõe o cerne arquitetural do software.

A visão (*view*) representa todos os conjuntos de entrada e saída de dados, como telas, tabelas, diagramas ou até imagens/gráficos. Já o controlador (*controller*) é o motor do sistema na arquitetura MVC. O controlador é responsável pela mediação dos eventos de entrada, que são convertidos em comandos ou solicitações para o modelo ou para a visão. Em outras palavras, o controlador realiza a interface entre a visão e o modelo.

Uma das principais vantagens de se usar o MVC está diretamente relacionada à reusabilidade de código e a separação de conceitos e regras de desenvolvimento, o que contribui para o desenvolvimento e a manutenção do software, já que o MVC direciona o desenvolvimento para uma arquitetura fracamente acoplada ou de baixo acoplamento.

A Figura 6 a seguir, apresenta uma o modelo padrão do MVC e as relações simplistas entre a visão, o controlador, e o modelo.

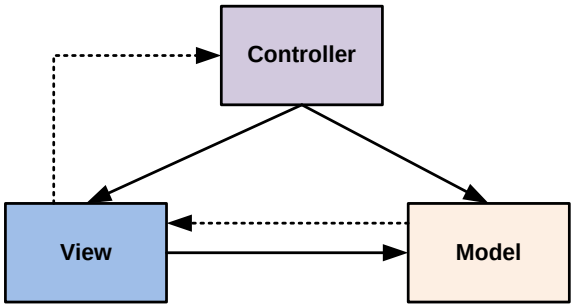


Figura 6: Arquitetura básica do *Model View Controller* (MVC) (Burbek 1987/1992).

A ilustração do modelo MVC apresenta um diagrama simples exemplificando a relação entre *Model, View e o Controller*. As linhas sólidas indicam associação direta e as tracejadas indicam associação indireta.

O apoio ferramental foi eleito após uma análise prévia envolvendo o ambiente de desenvolvimento e o tipo de projeto. Levando em conta ainda o conhecimento e experiências já adquiridas dos. Outro fator determinante para a escolha do aparato ferramental foi à necessidade padronizar o desenvolvimento e com isso obter maior controle no experimento por parte dos executores desta pesquisa.

Apêndice E: Template do Experimento

Um *template* é um modelo documental para endereçamento de conteúdo com apresentação visual e instruções sobre a construção de um determinado trabalho. A literatura de ESE recomenda a elaboração de um *template* para condução do experimento. Mas segundo (Wohlin *et.al* 2012), a construção deste guia não é mandatória, embora seja uma referência confiável para a execução do experimento.

Apesar da não "mandatoriedade", a elaboração do *template* é importante para conduzir a pesquisa no caminho dos objetivos e metas traçadas durante a construção do escopo do projeto. Por esta razão, foi elaborado um *template* para auxiliar na construção e condução do experimento. O *template* construído foi baseado principalmente no *framework* apresentado por (Basili, *et.al*, 1986), mas também se apoiou nas contribuições de (Wohlin *et.al* 2012), (Juristo e Moreno, 2002) e (Travassos *et.al*, 2002). Para sintetizar, será apresentado na Tabela 1 um esboço do *template* construído envolvendo os principais campos do documento.

Tabela 1: Representação sumarizada do *template* construído

***Template* do Estudo Experimental**		
Id	***Conteúdo***	***Descrição***
1. IDENTIFICAÇÃO	Título	Identificação de Gargalos em Processo de Software
	Tema	Processo de Desenvolvimento de Software
	Área Técnica	Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais (UFRJ) e Coordenação de Automação Industrial (CEFET-RJ)
	Autor	Sildenir Alves Ribeiro
	Afiliação	CEFET-RJ / Coordenação de Automação Industrial
	Instituições parceiras	UFRJ e CEFET-RJ
	Local	Cidade do Rio de Janeiro, estado do Rio de Janeiro, país Brasil
	Período	Julho de 2012 – Julho de 2016
2. CARACTERIZAÇÃO	Tipo da Pesquisa	Experimentação
	Domínio	Desenvolvimento de software
	Área de Concentração	Engenharia de Software, Engenharia de Software Experimental
	Subáreas	Teoria das Restrições, Processo Unificado, *Job Shop Scheduling* Dinâmico
	Ambiente de Execução	Laboratório de Engenharia de Software do Curso de Ciência da Computação da UFRJ
	Cenário	Envolve: as pessoas, os processos, o produto, os métodos, os materiais e as ferramentas utilizadas (inclui Hardware e software).
	Idioma	Português

3. ESCOPO	Definição do Escopo	Identificação do objeto da pesquisa estudo e o conjunto de ações
	Introdução	Aspectos gerais e conceituação genérica do experimento
	Objetivos Específicos	Identificar e tratar os gargalos no PDS
	Objetivos Gerais	Colaboração com a academia e indústria de software
	Objeto de Estudo	Produção de Software
	Métodos de trabalho	Métodos de trabalho adotados para atingir os resultados esperados
4. DESENVOLVIMENTO/PLANEJAMENTO	Questões	Questões feitas sobre o experimento a partir dos objetivos específicos
	Hipóteses	Hipóteses levantadas a partir das questões que devem ser respondidas
	Medição	Critérios de medição adotados para avaliar os resultados do produtivos e os dados colhidos durante a execução do experimento.
	Análise	Critérios e métodos analíticos aplicados ao experimento de acordo com os dados selecionados para serem analisados
	Cobertura	Abrangência do estudo experimental
	Lacunas	Questões em aberto ou não respondidas durante ou após a experimentação
	Empacotamento	O empacotamento é usado para armazenar as experiências numa base de conhecimento e de registros e histórico / lições aprendidas
	Apresentação	Meios de apresentação e divulgação da pesquisa e consequentemente dos resultados
	Relatórios conclusivos	Elaboração de relatórios e documentos sobre a execução e os resultados obtidos durante a pesquisa

Apêndice F: Análise Estatística com ANOVA (Artefato / Entrega)

F.I ANOVA: Produção (Artefato X Entrega) – *Shop2*

Tabela 2: ANOVA Fator Duplo - Produção (Entrega X Artefato) – *Shop* 2

Anova: fator duplo sem repetição da Produção (Entrega X Artefato) – *Shop* 2						
Análise de Colunas						
Entregas	*Contagem*	*Soma*	*Média*	*Variância*	*Observações*	
1	5	1,000	0,200	0,024	*Anova Shop 2 para Q3 (Linhas) - Entregas*	
2	5	1,000	0,200	0,014		
3	5	1,000	0,200	0,011		
4	5	1,000	0,200	0,009		
Análise das Linhas						
Arquitetura	4	0,529	0,132	0,000	*Anova Shop 2 para Q3 (Colunas) - Artefatos*	
Codificação	4	0,769	0,192	0,001		
Documentação	4	1,546	0,386	0,001		
Projeto BD	4	0,802	0,200	0,001		
Teste	4	0,354	0,089	0,006		
ANOVA						
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Linhas	0	3	0	0	1	3,490
Colunas	0,207	4	0,052	25,287	9E-06	3,259
Erro	0,025	12	0,002			

F.II ANOVA: Dificuldade (Artefato X Entrega) - *Shop* 2

Tabela 3: ANOVA Fator Duplo - Dificuldade (Entrega X Artefato) – *Shop* 2

Anova: fator duplo sem repetição da Dificuldade (Entrega X Artefato) – *Shop* 2						
Análise das Colunas						
Entregas	*Contagem*	*Soma*	*Média*	*Variância*	*Observações*	
1	5	1	0,2	0,029	*Anova Shop 2 para Q3 (Linhas) - Entregas*	
2	5	1	0,2	0,016		
3	5	1	0,2	0,002		
4	5	1	0,2	0,003		
Análise das Linhas						
Arquitetura	4	0,590	0,147	0,000	*Anova Shop 2 para Q3 (Colunas) - Artefato*	
Codificação	4	0,899	0,225	0,000		
Documentação	4	1,357	0,339	0,011		
Projeto BD	4	0,684	0,171	0,001		
Teste	4	0,470	0,118	0,013		
ANOVA						
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Linhas	0,000	3	0,000	0,000	1,000	3,490
Colunas	0,122	4	0,030	4,892	0,014	3,259
Erro	0,075	12	0,006			

F.III ANOVA: Produção (Artefato X Entrega) - *Shop* 3

Tabela 4: ANOVA Fator Duplo - Produção (Entrega X Artefato) – *Shop* 3

Anova: fator duplo sem repetição Produção (Artefato X Entrega) Shop 3						
Análise das Colunas						
Entregas	*Contagem*	*Soma*	*Média*	*Variância*	*Observações*	
1	5	1,000	0,200	0,005	*Anova Shop 3 para Q3 (Linhas) - Entregas*	
2	5	1,000	0,200	0,010		
3	5	1,000	0,200	0,003		
4	5	1,000	0,200	0,006		
Análise das Linhas						
Arquitetura	4	0,555	0,139	0,000	*Anova Shop 3 para Q3 (Colunas) - Artefatos*	
Codificação	4	0,549	0,137	0,001		
Documentação	4	1,043	0,261	0,004		
Banco de Dados	4	0,761	0,190	0,001		
Teste	4	1,091	0,273	0,004		
ANOVA						
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Linhas	0,000	3	0,000	0,000	1,000	3,490
Colunas	0,067	4	0,017	6,616	0,005	3,259
Erro	0,030	12	0,003			

F.IV ANOVA: Dificuldade (Artefato X Entrega) - *Shop* 3

Tabela 5: ANOVA Fator Duplo - Dificuldade (Entrega X Artefato) – *Shop* 3

Anova: fator duplo sem repetição Dificuldade (Artefato X Entrega) Shop 3						
Análise das Colunas						
Entregas	*Contagem*	*Soma*	*Média*	*Variância*		
1	5	1	0,200	0,008	*Anova Shop 3 para Q3 (Linhas) - Entregas*	
2	5	1	0,200	0,004		
3	5	1	0,200	0,007		
4	5	1	0,200	0,004		
Análise das Linhas						
Arquitetura	4	0,569	0,142	0,000	*Anova Shop 3 para Q3 (Colunas) - Artefatos*	
Codificação	4	0,760	0,190	0,001		
Documentação	4	1,288	0,322	0,001		
Banco de Dados	4	0,604	0,151	0,000		
Teste	4	0,779	0,195	0,000		
ANOVA						
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Linhas	0	3	0	0	1	3,490
Colunas	0,083	4	0,021	36,630	1E-06	3,259
Erro	0,007	12	0,001			

F.V ANOVA: Fator duplo sobre o Esforço (Grupo X Artefato) - *Shop* 2

Tabela 6: Fator duplo sobre o Esforço (%) – (Grupo X Artefato) – Shop 2

ANOVA: Fator Duplo Sem Repetição – *Shop* 2						
Análise das Colunas					**Observações**	
LPs	***Contagem***	***Soma***	***Média***	***Variância***	*1. Anova usada para responder a Q2;* *2. Análise das colunas / Artefatos.*	
LP1	5	1	0,200	0,021		
LP2	5	1	0,200	0,019		
LP3	5	1	0,200	0,052		
LP4	5	1	0,200	0,051		
LP5	5	1	0,200	0,059		
LP7	5	1	0,200	0,024		
LP8	5	1	0,200	0,039		
LP9	5	1	0,200	0,014		
Análise das Linhas					**Observações**	
Arquitetura	8	0,469	0,059	0,001	*1. Anova usada para responder a Q2;* *2. Análise das linhas / LPs*	
Banco de Dados	8	1,086	0,136	0,005		
Codificação	8	2,244	0,280	0,014		
Documentação	8	3,420	0,428	0,018		
Teste	8	0,469	0,059	0,001		
ANOVA						
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Linhas	0,0002	7	0,0003	0,0001	1	2,354
Colunas	0,7429	4	0,1857	14,0451	2,1E-06	2,714
Erro	0,3702	28	0,0132			

F.VI ANOVA: Fator duplo sobre o Esforço (Grupo X Artefato) - *Shop* 3

Tabela 7: ANOVA Fator duplo (Esforço – Grupo X Artefato) – *Shop* 3

ANOVA: Fator Duplo Sem Repetição – *Shop* 3						
Análise das Colunas					**Observações**	
LPs	*Contagem*	*Soma*	*Média*	*Variância*	*Anova fator duplo sobre o esforço das LPs no Shop 3. Análise das colunas / Artefatos*	
LP1	5	1,000	0,200	0,030		
LP2	5	1,000	0,200	0,040		
LP4	5	1,000	0,200	0,009		
LP5	5	1,000	0,200	0,054		
Análise das Linhas					**Observações**	
Arquitetura	4	0,290	0,072	0,002	*Anova fator duplo sobre o esforço das LPs no Shop 3. Análise das linhas / LPs*	
Banco de Dados	4	0,552	0,138	0,001		
Codificação	4	1,939	0,485	0,010		
Documentação	4	0,945	0,236	0,001		
Teste	4	0,274	0,069	0,003		
ANOVA						
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*P-valor*	*F crítico*
Linhas	1,11E-16	3	3,7E-17	8,52E-15	1	3,490
Colunas	0,479	4	0,119	27,596	5,7E-06	3,259
Erro	0,052	12	0,004			

F.VII ANOVA: Fator duplo sobre o Esforço (Entrega X Artefato) - *Shop* 2

Tabela 8: ANOVA Fator Duplo – Esforço (Entrega X Artefato) – Shop 2

Anova: Fator Duplo sem Repetição do Esforço (Artefato X Entrega)						
Análise de Colunas						
Entregas	*Contagem*	*Soma*	*Média*	*Variância*	*Observações*	
1	5	1	0,200	0,039	*Anova Shop 2 para Q3 (Linhas) - Entregas*	
2	5	1	0,200	0,031		
3	5	1	0,200	0,026		
4	5	1	0,200	0,047		
Análise das Linhas						
Arquitetura	4	0,257	0,0642	0,001	*Anova Shop 2 para Q3 (Colunas) - Artefatos*	
Banco de Dados	4	0,531	0,1328	0,006		
Codificação	4	1,693	0,4232	0,015		
Documentação	4	1,112	0,2781	0,033		
Teste	4	0,405	0,1014	0,017		
ANOVA						
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Linhas	0	3	0	0	1	3,490
Colunas	0,354	4	0,089	4,930	0,014	3,259
Erro	0,215	12	0,018			

F.VIII ANOVA: Fator duplo sobre o Esforço (Entrega X Artefato) - *Shop* 3

Tabela 9: ANOVA Fator Duplo Sobre Esforço (Entrega X Artefato) – *Shop* 3

Anova: Fator Duplo sem Repetição do Esforço (Artefato X Entrega)						
Análise das Colunas						
Entregas	*Contagem*	*Soma*	*Média*	*Variância*	*Observações*	
1	5	1	0,200	0,030	*Anova Shop 3 para Q3 (Linhas) - Entregas*	
2	5	1	0,200	0,032		
3	5	1	0,200	0,031		
4	5	1	0,200	0,059		
Análise das Linhas						
Arquitetura	4	0,303	0,076	0,001	*Anova Shop 3 para Q3 (Colunas) - Artefatos*	
Banco de Dados	4	0,555	0,139	0,010		
Codificação	4	1,878	0,470	0,018		
Documentação	4	0,991	0,248	0,019		
Teste	4	0,273	0,068	0,006		
ANOVA						
Fonte da variação	*SQ*	*gl*	*MQ*	*F*	*valor-P*	*F crítico*
Linhas	-0,000	3	-0,000	- 0,000	1	3,490
Colunas	0,446	4	0,112	8,266	0,002	3,259
Erro	0,233	12	0,029			

Apêndice G: A Linha de Base

G.I Construção da Linha de Base

A linha de base é o indicador usado nesta pesquisa para parametrizar, avaliar e aferir as evidências encontradas nos shops e identificar potenciais gargalos. A *baseline* envolve os resultados das tuplas {*Esforço, Produção e Dificuldade*}, como especificado na Seção 5.1.6 e 5.3.1.

G.II Linha de Base do *Shop* 2

A Tabela abaixo apresenta a linha de base para os artefatos do *Shop* 2.

Tabela 10: Linha de Base do *Shop* 2

Linha de Base *Shop* 2										
Artefatos										
Métricas	***Arquitetura***		***Banco de Dados***		***Codificação***		***Documentação***		***Teste***	
$\mu(\overline{X})$, $\sigma(\overline{X})$	*μ*	*(σ)*	*(μ)*	*(σ)*	*(μ)*	*(σ)*	*(μ)*	*(σ)*	*(μ)*	*(σ)*
Esforço	*873*	*538,33*	*1.832*	*1082,65*	*5.679*	*2387,05*	*3.838*	*2002,96*	*1.301*	*1537,60*
Dificuldade	*0,95*	*0,13*	*1,10*	*0,21*	*1,44*	*0,40*	*1,07*	*0,27*	*0,53*	*0,38*
Produção	*0,59*	*0,34*	*0,89*	*0,21*	*0,85*	*0,28*	*0,86*	*0,26*	*0,29*	*0,39*

G.III Linha de Base do *Shop* 3

O mesmo processo foi repetido para o *Shop* 3 e a Tabela a seguir apresenta a linha de base correspondente.

Tabela 11: Linha de Base do Shop 3.

Linha de Base *Shop* 3										
Artefatos										
Métricas	***Arquitetura***		***Banco de Dados***		***Codificação***		***Documentação***		***Teste***	
$\mu(\bar{X})$, $\sigma(\bar{X})$	*(μ)*	*(σ)*	*(μ)*	*(σ)*	*(μ)*	*(σ)*	*(μ)*	*(σ)*	*(μ)*	*(σ)*
Esforço	*1.134*	*789,95*	*2.045*	*332,39*	*7.781*	*3592,65*	*3.634*	*823,68*	*1.606*	*664,51*
Dificuldade	*1,07*	*0,38*	*1,12*	*0,35*	*1,42*	*0,32*	*1,20*	*0,13*	*0,98*	*0,27*
Produção	*0,72*	*0,26*	*0,90*	*0,12*	*0,70*	*0,28*	*0,68*	*0,16*	*0,86*	*0,18*

Os valores fixados nas linhas de base sobre o esforço das LPs, apresentam variações consideráveis, o que contribui para a alta dispersão (σ), destacando-se os artefatos Codificação e Documentação nos *shops* 2 e 3 e o artefato Teste no *shop* 2

Apêndice H: Análise Empírica

H.I Mapeamento Produtivo do *Shop* 2 e *Shop* 3

A partir dos resultados apresentados na Seção 7.3 (*7.3.1 – 7.3.7*) foi possível fazer uma análise sobre o comportamento das LPs em relação ao Esforço, Produção e Dificuldade. Os resultados desta análise podem ser visualizados na Tabela 12 a seguir, onde se apresenta um mapeamento dos resultados produtivos do *Shop* 2 e 3, respectivamente.

Tabela 12: Mapeamento produtivo – Shop 2 e 3

Mapeamento Produtivo (MP) – *Shop* 2																
LPs	***P_L***	***Arquitetura***			***Banco de Dados***			***Codificação***			***Documentação***			***Teste***		
		A	**M**	**B**	**A**	**M**	**B**	**A**	**M**	**B**	**A**	**M**	**B**	**A**	**M**	**B**
LP1	**E**		✓			✓		✓			✓				✓	
	P		✓		✓				✓			✓				✓
	D		✓			✓		✓			✓				✓	
LP2	**E**			✓			✓		✓		✓				✓	
	P		✓			✓			✓		✓					✓
	D			✓			✓	✓			✓				✓	
LP3	**E**			✓		✓		✓			✓					✓
	P			✓	✓				✓		✓					✓
	D		✓			✓			✓		✓				✓	
LP4	**E**			✓		✓		✓				✓				✓
	P	✓			✓			✓			✓					✓
	D		✓				✓	✓				✓			✓	
LP5	**E**			✓		✓		✓					✓			✓
	P			✓		✓			✓			✓				✓
	D			✓			✓		✓			✓			✓	
LP7	**E**		✓		✓			✓					✓			✓
	P			✓			✓			✓	✓					✓
	D			✓			✓		✓		✓					✓

LP8	**E**		✓			✓		✓					✓			✓
	P		✓			✓			✓			✓				✓
	D		✓		✓				✓		✓					✓
LP9	**E**		✓				✓	✓				✓		✓		
	P			✓		✓			✓		✓				✓	
	D			✓			✓		✓		✓					✓

Mapeamento Produtivo – *Shop* 3

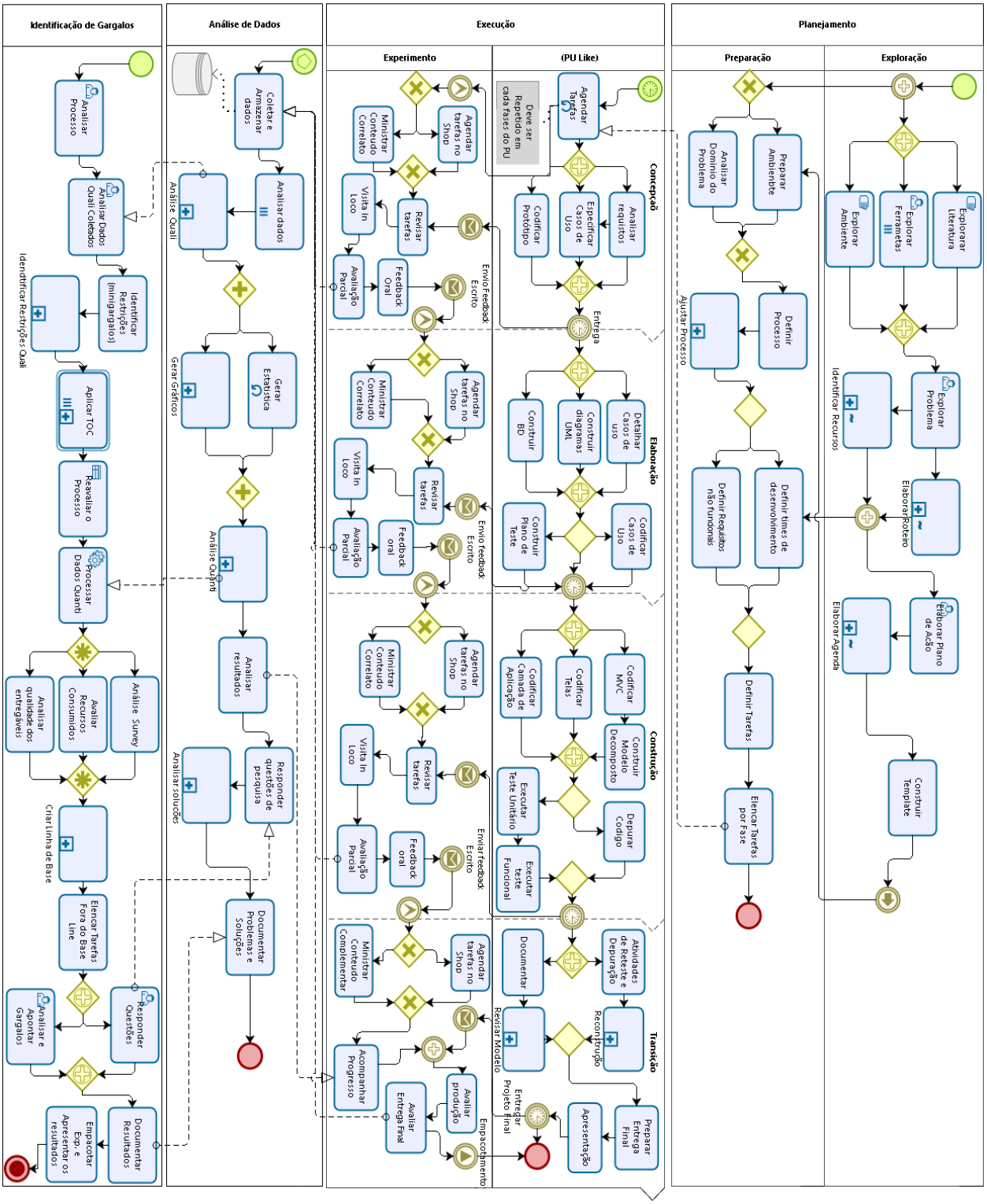
LPs	***Id***	***Arquitetura***			***Banco de Dados***			***Codificação***			***Documentação***			***Teste***		
		A	**M**	**B**	**A**	**M**	**B**	**A**	**M**	**B**	**A**	**M**	**B**	**A**	**M**	**B**
LP1	**E**	✓			✓			✓				✓		✓		
	P	✓			✓			✓				✓			✓	
	D			✓		✓		✓			✓			✓		
LP2	**E**			✓		✓		✓				✓				✓
	P			✓		✓				✓			✓		✓	
	D			✓	✓			✓			✓			✓		
LP4	**E**		✓			✓		✓			✓			✓		
	P	✓			✓			✓			✓			✓		
	D	✓			✓			✓			✓				✓	
LP5	**E**	✓				✓		✓			✓				✓	
	P	✓			✓			✓			✓			✓		
	D	✓				✓			✓		✓			✓		

Legenda: *E = Esforço; P = Produção; D = Dificuldade; A = Alto; M = Moderado; B = Baixa.*

A análise empírica exibida na Tabela 34 corresponde a um intervalo percentual extraído do valor máximo obtido em cada *shop* para Esforço, Produção e Dificuldade. Onde: *Baixo = (0 – 33%), Moderado = (34 – 66%) e Alto = (67 – 100%).* Este mapeamento é interessante, pois facilita a compreensão sobre as restrições qualitativas e seus impactos nos resultados quantitativos.

Apêndice I: Mapeamento do Processo Experimental

I.I Modelagem BPMN do Processo e Método Experimental



I.II Processo Planejamento

O planejamento é composto de dois subprocessos, “Exploração” e “Preparação” com os seguintes passos:

***Swimlane* Exploração**

Passo 1: Explorar Literatura, Ferramentas e Ambiente
Passo 2: Explorar Problema, Identificar Recursos Necessários
Passo 3: Elaborar Roteiro
Passo 4: Elaborar Plano de Ação e Elaborar Agenda
Passo 5: Construir o Template de Execução do Experimento

***Swimlane* Preparação**

Passo 6: Preparar o ambiente de execução;
Passo 7: Analisar domínio do problema
Passo 8: Definir o processo a ser usado
Passo 9: Ajustar o processo em função do problema
Passo 10: Definir times de desenvolvimento
Passo 11: Definir requisitos não funcionais
Passo 12: Definir tarefas
Passo 13: Elencar tarefas por fase (em função do PU-Like).

I.III Processo Execução

Este processo é dividido em outros dois processos: O primeiro é o PDS, dado pelo *PU-Like*. O segundo é o processo de execução do projeto, o qual está diretamente alinhado ao PU-Like. O processo de execução do projeto se divide ainda em quatro *milestone*, que representa as fases do *PU-Like*.

Swimlane PU-Like

Passo 1: Iniciar o processo segundo agenda/cronograma
Passo 2: Agendar tarefas (fase de concepção do PU-Like)
Passo 3: Analisar Requisitos
Passo 4: Especificar Casos de Uso
Passo 5: Codificar Protótipo

Passo 6: Finalizar a fase de Concepção e entregar os artefatos produzidos

Passo 7: Voltar ao passo 2 e agendar tarefas (fase de Elaboração do PU-Like)

Passo 8: Detalhar Casos de uso;

Passo 9: Construir Diagramas UML

Passo 10: Construir BD

Passo 11: Codificar casos de uso

Passo 12: Elaborar plano de testes

Passo 13: Finalizar a fase de elaboração e entregar os artefatos produzidos

Passo 14: Voltar ao passo 2 e executar tarefas (fase de Construção do Pu-Like)

Passo 15: Codificar MVC, Telas e Camada de Aplicação

Passo 16: Construir modelo Arquitetural decomposto (Diagrama de pacotes);

Passo 17: Depurar Código

Passo 18: Executar casos de testes de unidade

Passo 19: Executar casos de teste funcional

Passo 20: Finalizar a fase de Construção e entregar os artefatos produzidos

Passo 21: Voltar ao Passo 2 e agendar tarefas (fase de Transição)

Passo 22: Executar atividades de "reteste" e depuração

Passo 23: Executar subprocesso de reconstrução e ajustes.

Passo 24: Gerar documentação do Sistema (Relatório técnico)

Passo 24: Revisar modelos e gerenciar as mudanças

Passo 25: Preparar a entrega final (Documentação, Modelos, Códigos, BD e Testes)

Passo 26: Apresentar o projeto completo desenvolvido e funcionando.

Passo 27: Entregar todo o projeto compilado e ajustado conforme apresentado (Homologação).

***Swimlane* Experimento**

Passo 1: Agendar tarefas no Shop

Passo 2: Ministrar conteúdo correlato

Passo 3: Revisar tarefas

Passo 4: Visitar os times de desenvolvimento no shop

Passo 5: Efetuar avaliação parcial

Passo 6: Feedback expositivo (oral)

Passo 7: Enviar Feedback escrito para cada LP apontando os ajustes que devem ser realizados;

Passo 8: Voltar ao passo 1 e executar passos 1-7 para fase de Elaboração

Passo 9: Voltar ao passo 1 e executar passos 1-7 para a fase de Construção

Passo 10: Voltar ao Passo 1 e executar passos 1-2 para a fase de Transição

Passo 11: Acompanhar progresso

Passo 12: Receber Produção

Passo 13: Avaliar produção

Passo 14: Avaliar entrega Final

Passo 15: Empacotar projeto/produto e gerar relatório

Passo 16: Encerrar atividades (desenvolvimento e experimento)

I.IV Análise de Dados

O processo análise de dados tem por objetivo guiar o processo de coleta, análise e avaliação dos dados durante a execução do projeto.

Passo 1: Coletar e Armazenar dados

Passo 2: Analisar dados qualitativos em cada fase

Passo 3: Gerar estatística e gráficos

Passo 4: Realizar a análise Quantitativa

Passo 5: Responder as questões de pesquisa

Passo 6: Propor soluções

Passo 7: Documentar problemas e soluções propostas

Passo 8: Encerrar a análise de dados e documentar resultados.

I.V Identificação de Gargalos

O processo de identificação de gargalos canaliza e direciona todo o esforço deste trabalho, desde a execução experimental até a análise dos resultados processo produtivo.

Passo 1: Analisar os Processos

Passo 2: Analisar Dados Qualitativos

Passo 3: Identificar Restrições Qualitativas (mini Gargalos)

Passo 5: Tratar restrições com a Aplicação da TOC

Passo 5: Reavaliar o processo

Passo 6: Processar Dados Quantitativos

Passo 7: Analisar Survey

Passo 8: Avaliar recursos consumidos

Passo 9: Analisar qualidade dos artefatos entregue

Passo 10: Executar o subprocesso "Criar linha de base"

Passo 11: Elencar tarefas fora da linha de base

Passo 12: Responder questões

Passo 13: Analisar e apontar os gargalos encontrados no processo produtivo

Passo 14: Documentar os resultados.

Passo 15: Empacotar experimento e encerrar o processo.

A *Swimlane* Identificação de Gargalos da Figura 22 apresenta a estrutura do processo de identificação de gargalo modelado segundo a notação BPMN. A identificação de gargalo inicia-se com a atividade "analisar o processo" que compreende na análise do processo de execução do experimento em consonância com o *PU-Like*. Em seguida é aplicado o método de identificação das restrições (qualitativas), a identificação de gargalos (quantitativos) e a exibição dos resultados com a análise e respostas das questões de pesquisa.

A execução das tapas estabelecidas neste processo, comumente alinhado aos critérios de análise e medição irá apontar uma "tarefa gargalo", uma vez que identifica as tarefas com alto consumo de recursos, com baixa qualidade e uma taxa de dificuldade para desenvolvê-la muito alta.

Apêndice J: Documentos Gerados

J.I Questionário I – Avaliação do Conhecimento Técnico

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INCE – INSTITUTO TÉCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PPGI – PROGRAMA DE PÒS GRADUAÇÂO EM INFORMATICA
DCC – DEPARTAMENTO DE CIENCIA DA COMNPUTAÇÃO
DOUTORADO EM INFORMÁTICA

1

METODOLOGIA PARA REALIZAÇÃO DE ESTUDO EMPÍRICO SOBRE DETECÇÃO DE GARGALOS EM PROCESSO DE SOFTWARE COM BASE NO PRINCIPIO DA TEORIA DAS RESTRIÇÕES-TOC.

O objetivo deste roteiro é guiar a equipe de projetos para a especificação, análise, modelagem e desenvolvimento do sistema Você Aluga, além do gerenciamento do processo com foco no Processo Unificado (UP). Com relação ao processo de desenvolvimento, o foco será na identificação de gargalos durante todas as etapas do processo de desenvolvimento. A Metodologia usada na identificação dos gargalos baseia-se no princípio da TOC – Teoria das restrições, dada pelas seguintes etapas: Identificar (as restrições do sistema), Explorar (as restrições do sistema), Elevar, Subordinar (os demais recursos), Elevar (levantar a restrição), Quebrar (a inércia do sistema).
Este roteiro é composto de um questionamento que deve ser direcionado a equipe de desenvolvimento, as quais irão compor o estudo primário para a identificação e o tratamento dos gargalos encontrados. As perguntas a serem respondidas são fundamentadas nos princípios da TOC, na maturidade da equipe de desenvolvimento com relação ao ambiente de desenvolvimento, metodologia, ferramentas e ao processo unificado. Os dados fornecidos/encontrados serão utilizados em cada fase do processo para avaliação e reavaliação dos trabalhos, e serão segmentados para análise quantitativa e qualitativa.

Profº. Eber Assis Shmitz
Professor Orientador

Profº. Antonio Juarez Alencar
Professor Co-Orientador

Sildenir Alves Ribeiro
Doutorando em Informática
Celular: (21) [redacted]
e-mail: sildenir.ribeiro@ppgi.ufrj.br

INSTRUÇÕES PARA O QUESTIONÁRIO:

1. Preencha todas as questões, sem deixar nenhuma em branco. Assim, a análise dos dados não ficará prejudicada.
2. Preencha com base apenas na sua percepção frente ao desenvolvimento do modelo e do processo os aspectos que devem ser considerados.
3. Todas as perguntas devem ser respondidas de forma sincera e de acordo com a realidade e o desenrolar do projeto e dos processos, consequentemente.

Bloco I – Perfil da Equipe **Nome:** [redacted]

1. Idade: (média de idade) 20 anos
2. Formação escolar:

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INCE – INSTITUTO TÉCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PPGI – PROGRAMA DE PÒS GRADUAÇÂO EM INFORMATICA
DCC – DEPARTAMENTO DE CIENCIA DA COMNPUTAÇÃO
DOUTORADO EM INFORMÁTICA

2

1. ( ) Doutorado 2. ( ) Mestrado 3. ( ) Especialização 4. (X) Graduação
5. ( ) Médio 6. ( ) Fundamental

4. Seu tempo na instituição: 3 anos (mesmo como aluno)

5. Qual a sua ocupação?
1. ( ) Analista de Sistemas 2. ( ) Programador 3. ( ) Estagiário
4. (X) Estudante 5. ( ) Outro: _____

6. Experiência com Análise e Modelagem de Sistemas

( ) Sim (X) Não

7. Experiência com Desenvolvimento de Sistemas

(X) Sim ( ) Não

8. Experiência com processo de desenvolvimento de software

( ) Sim (X) Não

9. Experiência com projetos de Software

( ) Sim (X) Não

10. Conhece o Processo Unificado (UP)

( ) Sim (X) Não

11. Já trabalhou com alguma ferramenta de modelagem de processos (RUP, Open UP) e outras.

( ) Sim (X) Não

12. Qual o seu nível de conhecimento / entendimento sobre a TOC?

( ) Baixo ( ) Médio ( ) Alto (X) Não sei/conheço a TOC

13. Considero trabalhar com processo de desenvolvimento de software:

		1	2	3	4	5	
13.1	Boa ideia	○	●	○	○	○	Péssima ideia
13.2	Ideia Prudente	○	●	○	○	○	Ideia insensata
13.3	Gosto da ideia	○	●	○	○	○	Detesto a ideia
13.4	Agradável	○	○	●	○	○	Desagradável
13.5	Oportuno	●	○	○	○	○	Inoportuno

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO 3
INCE – INSTITUTO TÉCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PPGI -- PROGRAMA DE PÒS GRADUAÇÂO EM INFORMATICA
DCC – DEPARTAMENTO DE CIENCIA DA COMNPUTAÇÃO
DOUTORADO EM INFORMÁTICA

Bloco II – Quadro Funções x FCS (Fatores Críticos de Sucesso)

1. Para responder as questões abaixo, marque com um "X" no quadro a alternativa que melhor representa a sua opinião com relação a cada etapa do processo de desenvolvimento de software. quanto a sua concordância ou não com cada frase a qual se refere, conforme a escala abaixo.

1. Concordo totalmente 2. Concordo parcialmente 3. Indiferente
4. Discordo parcialmente 5. Discordo totalmente

COD	Questões/Afirmações – Organização / Escopo	1	2	3	4	5
01	Sei o que é um escopo de projeto		X			
02	O Escopo não é importante					X
03	O Escopo é importante mas não necessário					X
04	O Escopo é essencial	X				
05	Sei o que é uma WBS					X
06	Sei o que é um contrato		X			
COD	**Questões/Afirmações – Planejamento**	**1**	**2**	**3**	**4**	**5**
01	O Planejamento é primordial em um projeto de software	X				
02	Conheço as etapas do planejamento				X	
03	O Planejamento é importante mas não necessário					X
04	A equipe de desenvolvimento é focada nas decisões tomadas pela direção					X
05	Conheço os processos do planejamento					X
COD	**Questões/Afirmações – Especificação**	**1**	**2**	**3**	**4**	**5**
01	Estou habilitado para analisar e especificar requisitos					X
02	Conheço UML					X
03	Conheço Processo Unificado					X
04	Conheço modelos de Maturidade de Processo de SW (CMMI e MPS-BR)					X
05	Conheço ferramenta de modelagem de processos					X
06	Sei especificar um sistema		X			
07	Não sei especificar um sistema	X				
08	Não sei o que é especificar um sistema				X	
COD	**Questões/Afirmações – Analise e Modelagem (UML)**	**1**	**2**	**3**	**4**	**5**
01	Não sei analisar					X
02	Nunca analisei um projeto de sistema	X				
03	Conheço as etapas de análise e modelagem					X
04	Já modelei um sistema					X
05	Conheço diagrama de caso de uso	X				
06	Conheço diagrama de classe	X				
07	Conheço diagrama de sequencia					X
08	Conheço diagrama de colaboração					X
09	Conheço diagrama de estados/atividades					X
COD	**Questões/Afirmações – Função Desenvolvimento**	**1**	**2**	**3**	**4**	**5**
01	Conheço as etapas do desenvolvimento		X			
02	Conheço mais de uma linguagem de programação	X				
03	Sei programar mais não gosto					X
04	Já desenvolvi sistema / software	X				
05	Já desenvolvi sistema / software em mais de uma linguagem de programação	X				

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INCE – INSTITUTO TÉCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PPGI – PROGRAMA DE PÒS GRADUAÇÂO EM INFORMATICA
DCC – DEPARTAMENTO DE CIENCIA DA COMNPUTAÇÃO
DOUTORADO EM INFORMÁTICA

4

06	Não sei programar					X
COD	**Questões/Afirmações – Mudança**	**1**	**2**	**3**	**4**	**5**
01	Sei o que é mudança		X			
02	Sei o que tenho que fazer diante de uma mudança			X		
03	Conheço gerencia de risco					X
04	Sei planejar mudanças			X		
05.	Sei analisar os impactos de mudança			X		
06.	Não sei o que significa mudança em projeto / processo de software			X		
COD	**Questões/Afirmações – Teste**	**1**	**2**	**3**	**4**	**5**
29.	Tenho experiência com teste de software					X
30.	Sei o que é system teste e equipe de system teste					X
31.	Conheço mais de uma metodologia de teste					X
32.	O teste é importante		X			
33.	O teste não é importante				X	
34.	O teste é importante, mas não necessário				X	
COD	**Questões/Afirmações – Função Entrega**	**1**	**2**	**3**	**4**	**5**
29.	Conheço essa etapa do Projeto / Processo	X				
30.	Não sei o que é entrega					X
31.	Não o que fazer nesta etapa					X
32.	A entrega depende do aceite formal do cliente		X			
33.	O produto de software / sistema só é encerrado mediante a entrega.	X				
34.	Preciso saber mais sobre a entrega.	X				

J.II Questionário II – Avaliação do Conhecimento Técnico

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO 1
INCE – INSTITUTO TÉCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PPGI – PROGRAMA DE PÒS GRADUAÇÂO EM INFORMATICA
DCC – DEPARTAMENTO DE CIENCIA DA COMNPUTAÇÃO

[redacted]

METODOLOGIA PARA REALIZAÇÃO DE ESTUDO EMPÍRICO SOBRE DETECÇÃO DE GARGALOS EM PROCESSO DE SOFTWARE COM BASE NO PRINCIPIO DA TEORIA DAS RESTRIÇÕES-TOC.

O objetivo desta avaliação é buscar o conhecimento e o entendimento sobre as dificuldades encontradas pela equipe de desenvolvimento durante a fase de Concepção do Processo Unificado, que compreende as disciplinas de Modelagem de Negócios, Especificações de Requisitos, Análise e Modelagem, Implementação, Teste e Implantação e dos fluxos de apoio, dados pelas disciplinas de Gerenciamento de Configuração e Mudanças, Gerenciamento de Projetos e Ambiente. Este roteiro é composto de um questionamento que deve ser direcionado a equipe de desenvolvimento, as quais irão compor a segunda fase do estudo primário para a identificação e o tratamento dos gargalos encontrados durante do desenvolvimento. As perguntas a serem respondidas são fundamentadas nos princípios da TOC, na maturidade da equipe de desenvolvimento com relação ao sistema a ser desenvolvimento, as metodologias empregadas e o ambiente de desenvolvimento. Os dados fornecidos/encontrados serão utilizados para mapear o conhecimento sobre o domínio do problema da equipe e serão segmentados para análise quantitativa e qualitativa.

Profº. Eber Assis Shmitz
Professor Orientador

Profº. Antonio Juarez Alencar
Professor Co-Orientador

Sildenir Alves Ribeiro
Doutorando em Informática
Celular: (21) [redacted]
e-mail: sildenir.ribeiro@ppgi.ufrj.br

INSTRUÇÕES PARA O QUESTIONÁRIO:

1. Preencha todas as questões, sem deixar nenhuma em branco. Assim, a análise dos dados não ficará prejudicada.
2. Preencha com base apenas na sua percepção frente ao desenvolvimento do modelo e do processo os aspectos que devem ser considerados.
3. Todas as perguntas devem ser respondidas de forma sincera e de acordo com a realidade e o desenrolar do projeto e dos processos, consequentemente.

Bloco I – Quadro Funções x FCS (Fatores Críticos de Sucesso) Sobre o Domínio do Problema

1. Para responder as questões abaixo, marque com um "X" no quadro a alternativa que melhor representa a sua opinião com relação a cada etapa do processo de desenvolvimento de software. quanto a sua concordância ou não com cada frase a qual se refere, conforme a escala abaixo.

1. Concordo totalmente 2. Concordo parcialmente 3. Indiferente
4. Discordo parcialmente 5. Discordo totalmente

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INCE – INSTITUTO TÉCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PPGI – PROGRAMA DE PÒS GRADUAÇÂO EM INFORMATICA
DCC – DEPARTAMENTO DE CIENCIA DA COMNPUTAÇÃO

2

COD	Questões/Afirmações – Diagramas UML	1	2	3	4	5
01	Sei quais diagramas devo utilizar para modelar um sistema	X				
02	Conheço todos os diagramas a serem modelados		X			
03	Não acho necessário usar diagramas UML para modelar o problema				X	
04	Não sei representar as funções do sistema com o Diagrama de Casos de uso				X	
05	Não entendi e não sei construir o Diagrama de Classe para o domínio apresentado.				X	
06	Não entendi e não sei representar o fluxo das atividades com o Diagrama de Atividades		X			
07	Não entendi e não sei modelar as operações com o diagrama de sequência.	X				
COD	**Questões/Afirmações – Arquitetura (MVC)**	**1**	**2**	**3**	**4**	**5**
01	Sei o que é MVC	X				
02	Não sei representar a arquitetura com o MVC			X		
03	Não consigo aplicar o MVC ao domínio do problema			X		
04	Não acho necessário modelar a arquitetura do sistema				X	
05	Não sei representar a sequencia das operações da arquitetura com o diagrama de sequencia.	X				
COD	**Questões/Afirmações – Documentação**	**1**	**2**	**3**	**4**	**5**
01	Entendi e sei desenvolver a documentação do sistema.			X		
02	Ainda não sei o que devo colocar na documentação do sistema		X			
03	Não sei como descrever os diagramas UML			X		
04	Não sei como descrever a arquitetura do sistema com base no MVC			X		
05	Não acho que a documentação seja tão necessária				X	
COD	**Questões/Afirmações – Aplicação e Banco de dados**	**1**	**2**	**3**	**4**	**5**
01	Não sei construir / configurar a camada de aplicação	X				
02	Não conheço o Tomcat ou Glass Fish (prefiro fazer na mão)	X				
03	Não sei criar o serviço do Banco de Dados		X			
04	Não sei modelar o Banco de Dados		X			
05	Sei modelar o banco de dados mas não sei quais tabelas criar.					X
COD	**Questões/Afirmações – Desenvolvimento**	**1**	**2**	**3**	**4**	**5**
01	Não conheço a LP adotada (JAVA)					X
02	Não entendo e não sei programar orientado a objetos					X
03	Conheço a LP, mas não consigo aplicar ao domínio do problema			X		
04	Já desenvolvi sistema / software				X	
05	Não tenho problema com JAVA			X		
06	Não tenho problemas com o desenvolvimento/ codificação do do domínio do problema com JAVA			X		
COD	**Questões/Afirmações – Entregáveis**	**1**	**2**	**3**	**4**	**5**
01	Sei o que devo entregar a cada rodada (fase)				X	
02	Sei o que é um artefato					X
03	Ainda não sei o que é um artefato	X				
04	Sei quais (ou qual) artefatos devo entregar a cada rodada					X
05	Preciso saber mais sobre o que entregar.		X			

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO 3
INCE – INSTITUTO TÉCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PPGI – PROGRAMA DE PÒS GRADUAÇÂO EM INFORMATICA
DCC – DEPARTAMENTO DE CIENCIA DA COMNPUTAÇÃO

Bloco II – Questões Diversas (Para este bloco questões seja objetivo e direto.)

01. Quais as principais dificuldades encontradas para desenvolver o sistema.
02. Você já entendeu o domínio do problema? Existem lacunas? Quais?
03. Você teve ou tem dificuldade de entender o Processo Unificado?
04. Você sabe qual é a real necessidade de se utilizar o processo unificado no desenvolvimento de software?
05. Diante das dificuldades encontradas, o que você acha que poderia ser feito para melhorar o processo de desenvolvimento?
06. Quais as principais habilidades adquiridas com este trabalho (inclua novas habilidades e habilidades aprimoradas);

01) A Organisação no Grupo e achar informações sobre as varias partes

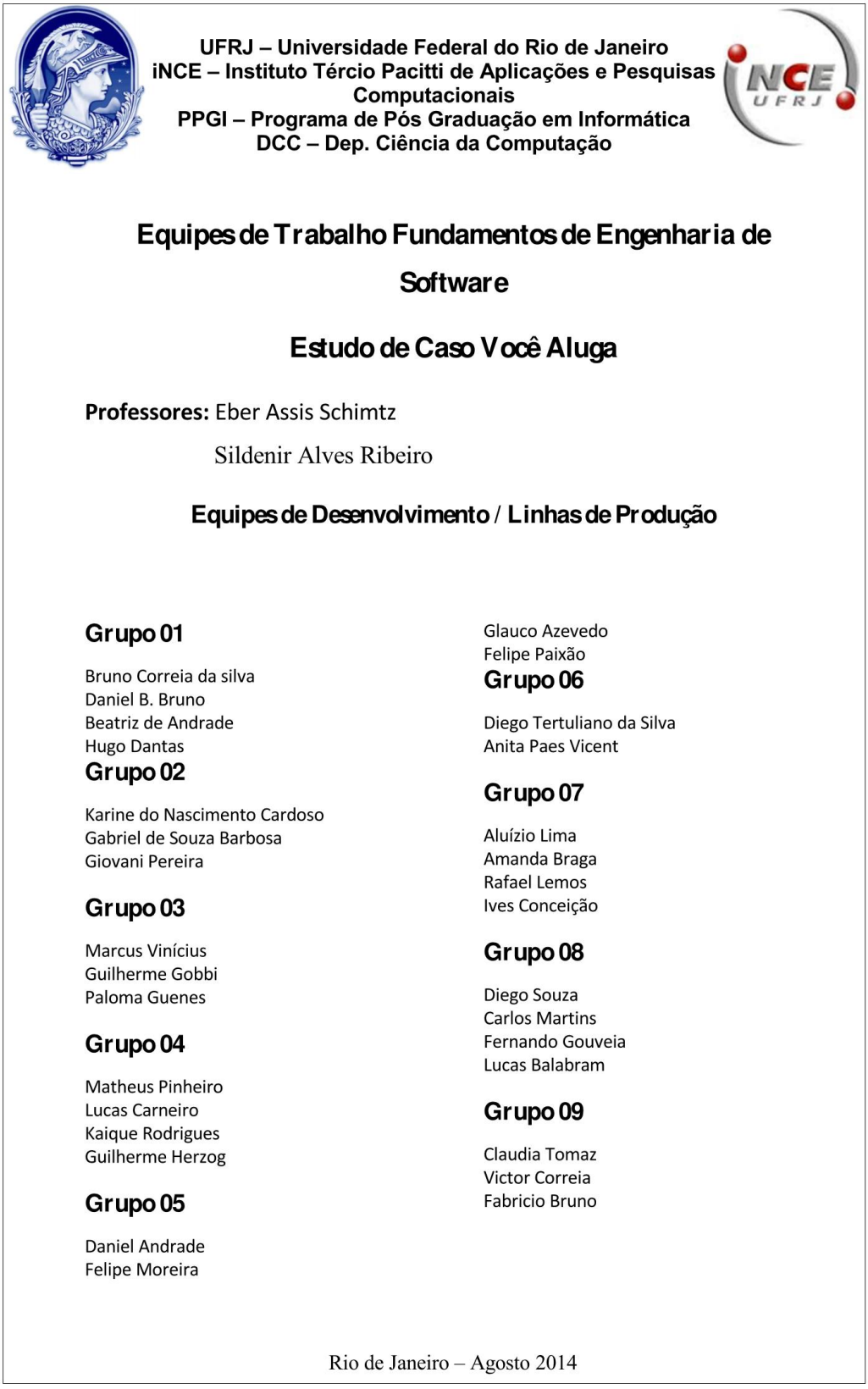
02) Ainda tenho problémas com Banco de Dados, Testes e Diagramas do sequencia

03) Acho entendi o Processo, só tem partes quais ainda não entendo

04) A real necessidade é ter mais sistema no processo e saber como vai funcionar e que precisa antes do programar. Isso tambem e importante para calcular o custo antes e dividir um projecto grande

05) Tem partes no projecto que não são parte da aula mas precisa mesm (Banco de dados, visual no Java). Dicas sobre onde procurar informacion sobre essas partes seriam grande ajuda. Tambem as coisas da aula eram as veces Difícils de achar depois dos slides. Uma lista das informações mais importantes ou algo assim seria bom.
As veces entendi os exemplos na aula mais os diagramas do projec eram, tão mais difícils, que não consegui. Podia facer exemplos mais parecido

J.III Montagem das Linhas de Produção (Equipes de Trabalhos)



UFRJ – Universidade Federal do Rio de Janeiro
iNCE – Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais
PPGI – Programa de Pós Graduação em Informática
DCC – Dep. Ciência da Computação

Equipes de Trabalho Fundamentos de Engenharia de Software

Estudo de Caso Você Aluga

Professores: Eber Assis Schimtz

Sildenir Alves Ribeiro

Equipes de Desenvolvimento / Linhas de Produção

Grupo 01

Bruno Correia da silva
Daniel B. Bruno
Beatriz de Andrade
Hugo Dantas

Grupo 02

Karine do Nascimento Cardoso
Gabriel de Souza Barbosa
Giovani Pereira

Grupo 03

Marcus Vinícius
Guilherme Gobbi
Paloma Guenes

Grupo 04

Matheus Pinheiro
Lucas Carneiro
Kaique Rodrigues
Guilherme Herzog

Grupo 05

Daniel Andrade
Felipe Moreira
Glauco Azevedo
Felipe Paixão

Grupo 06

Diego Tertuliano da Silva
Anita Paes Vicent

Grupo 07

Aluízio Lima
Amanda Braga
Rafael Lemos
Ives Conceição

Grupo 08

Diego Souza
Carlos Martins
Fernando Gouveia
Lucas Balabram

Grupo 09

Claudia Tomaz
Victor Correia
Fabricio Bruno

Rio de Janeiro – Agosto 2014

J.IV Agenda – Cronograma de Entregas (S_1, S_2 e S_3)

Agenda – Exp1 – Shop 1				
Agosto de 2014				
1ª. Semana	**2ª. Semana**	**3ª. Semana**		
12 – 14	*19 – 21*	*26 – 28*		
Setembro de 2014				
4ª. Semana	**5ª. Semana**	**6ª. Semana**	**7ª. Semana**	**8ª. Semana**
02 – 04	*09 – 11*	*16 – 18*	*23 – 25*	*30*
Outubro de 2014				
8ª. Semana	**9ª. Semana**	**10ª. Semana**	**11ª. Semana**	**12ª. Semana**
02	*07 – 09*	*14 –16*	*21 – 23*	*28 – 30*
Novembro de 2014				
13ª. Semana	**14ª. Semana**	**15ª. Semana**	**16ª. Semana**	
04 – 06	*11 – 13*	*18 – 20*	*25 – 27*	
Dezembro de 2014				
17ª. Semana				
02 – 04				
Obs: *Datas em azul – Aulas Teóricas . Datas em Vermelho – Aulas Práticas.* *As entregas serão feitas em dias de aulas práticas – destacado em amarelo na agenda.*				

Agenda – Exp2 – Shop 2				
Março de 2015				
1ª. Semana	**2ª. Semana**	**3ª. Semana**		
17 – 19	*24 – 26*	*31*		
Abril de 2015				
3ª. Semana	**4ª. Semana**	**5ª. Semana**	**6ª. Semana**	**7ª. Semana**
02	*07 – 09*	*14 – 16*	*21 – 23*	*28 - 30*
Maio de 2015				
8ª. Semana	**9ª. Semana**	**10ª. Semana**	**11ª. Semana**	
05 - 07	*12 – 14*	*19 – 21*	*26 – 28*	
Junho de 2015				
12ª. Semana	**13ª. Semana**	**14ª. Semana**	**15ª. Semana**	**16ª. Semana**
02 - 04	*09 – 11*	*16– 18*	*23 – 25*	*30*
Julho de 2015				
16ª. Semana	**17ª. Semana**	**18ª. Semana**		
02	*07 - 09*	*14 - 16*		
Obs: *Datas em azul – Aulas Teóricas . Datas em Vermelho – Aulas Práticas.* *As entregas serão feitas em dias de aulas práticas – destacado em amarelo na agenda.*				

Agenda – Exp2 – Shop 2			
Outubro 2015			
1ª. Semana	**2ª. Semana**	**3ª. Semana**	
13 – 15	***20 – 22***	***27 – 29***	
Novembro de 2015			
4ª. Semana	**5ª. Semana**	**6ª. Semana**	**7ª. Semana**
03 – 05	***Jornada***	***17 – 19***	***24 – 26***
Dezembro de 2015			
8ª. Semana	**9ª. Semana**	**10ª. Semana**	
01 - 03	***08 – 10***	***15 –17***	
Janeiro de 2016			
11ª. Semana	**12ª. Semana**	**13ª. Semana**	**14ª. Semana**
05 – 07	***12 – 14***	***19 – 21***	***26 – 28***
Fevereiro de 2016			
15ª. Semana	**16ª. Semana**	**17ª. Semana**	**18ª. Semana**
02 – 04	***Carnaval***	***16 – 18***	***23 – 25***
Março 2016			
19ª. Semana	**20ª. Semana**	**21ª. Semana**	
01 - 03	***08 – 10***	***15 – 17***	
Obs: *Datas em azul – Aulas Teóricas . Datas em Vermelho – Aulas Práticas. As entregas serão feitas em dias de aulas práticas – destacado em amarelo na agenda.*			

Legenda: **Azul** = Aula Eber (Teórica) – **Vermelho** = Aula Sildenir (Prática)

Entregas

1ª. Entrega = 24 de Novembro

2ª. Entrega = 05 de Janeiro

3ª. Entrega = 16 de Fevereiro

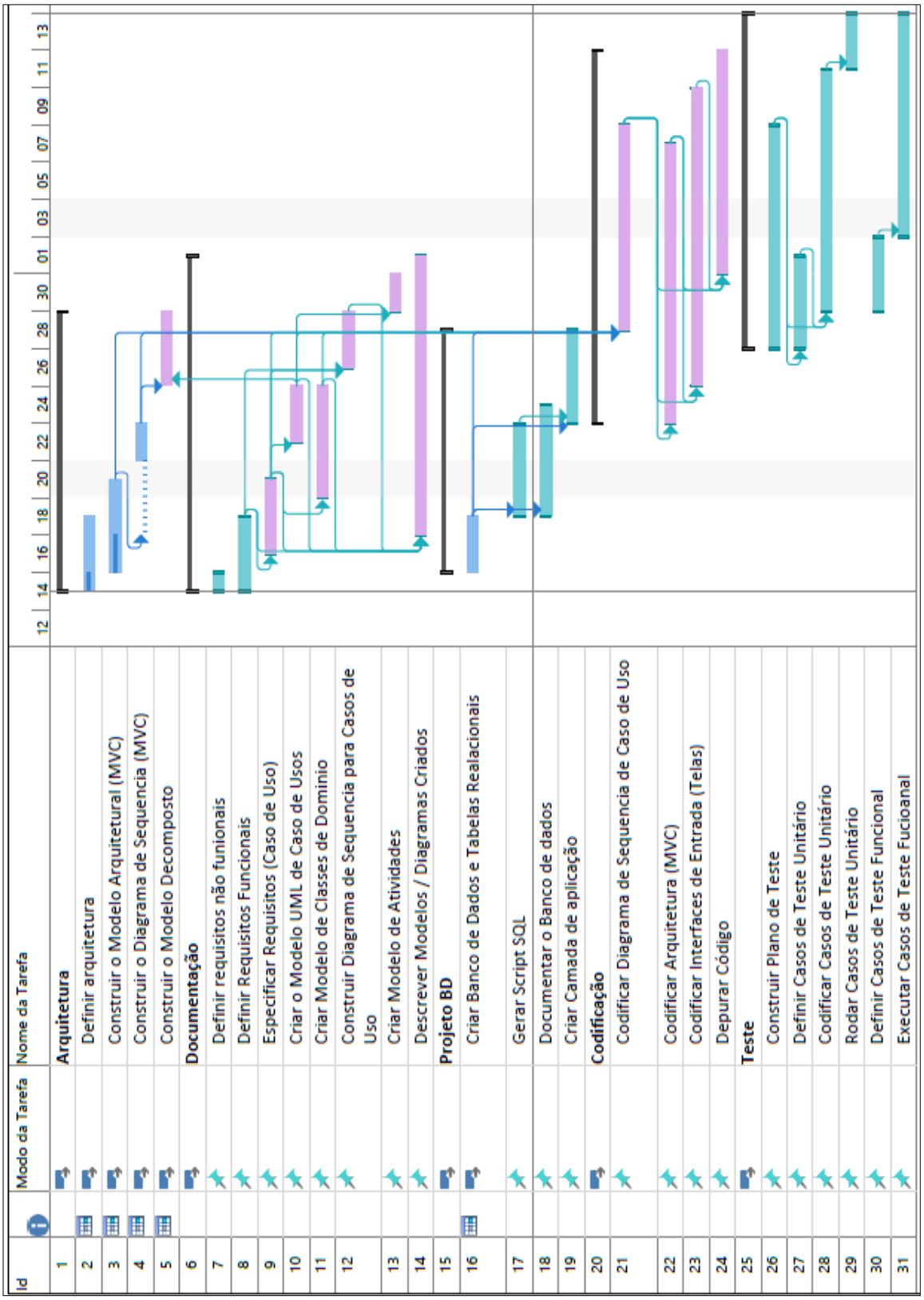
4ª. Entrega = 17 de Março

Prova: 02 de Março de 201

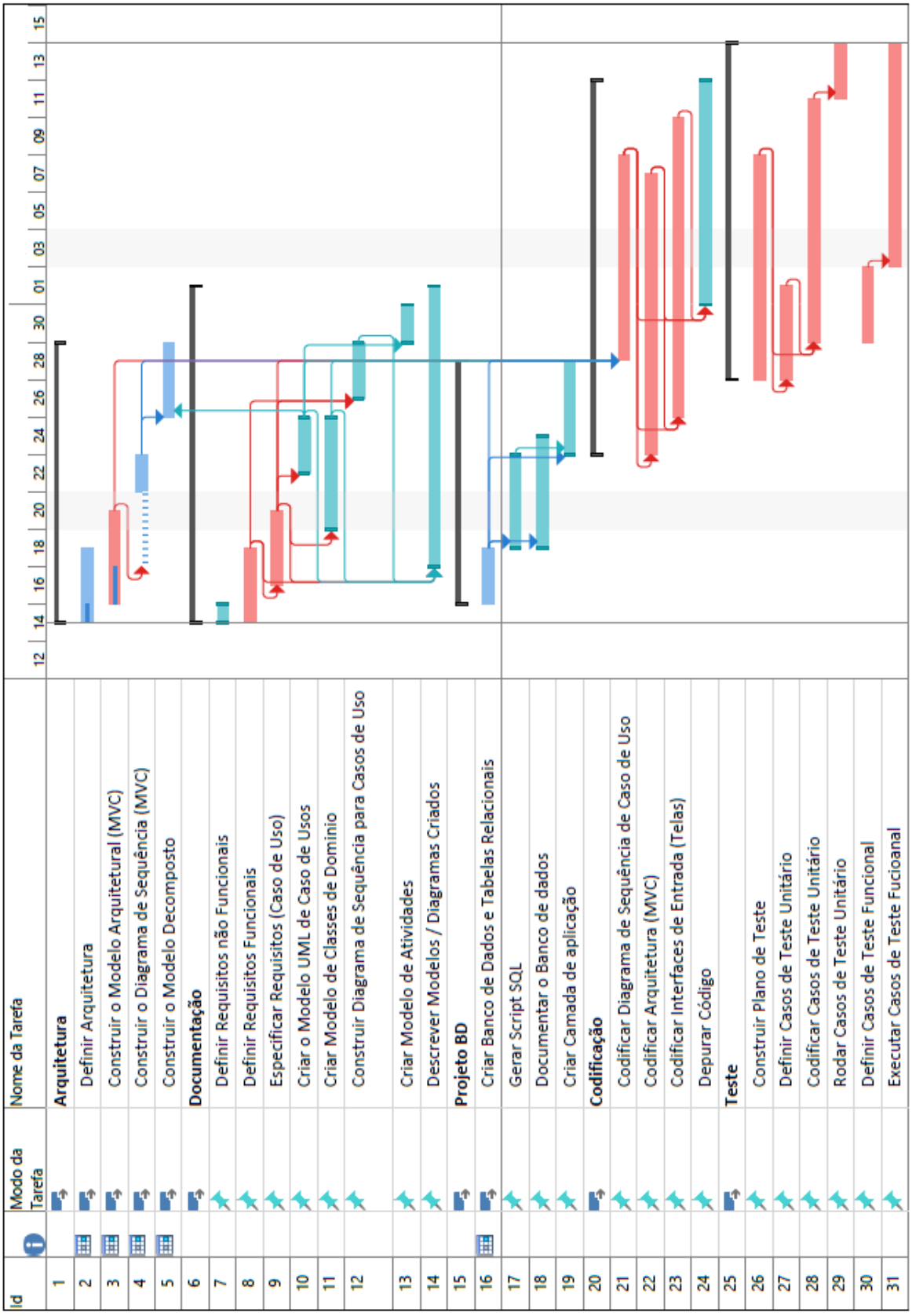
J.V Lista de Tarefas a Serem Inseridas no Shop

Lista de Artefatos / Atividades			
Ordem	Grupo de Artefatos	Artefato	Atividade
1	Documentação	Requisitos Funcionais (RF)	1.1-Definir os Requisitos Funcionais
		Requisitos Não Funcionais (RnF)	1.2-Definir os requisitos Não Funcionais
		Especificação de Requisitos (ER)	1.3-Especificar requisitos (Casos de Uso)
		Modelo de Casos de Uso (MCU)	1.4-Criar modelo UML de casos de uso
		Modelo de Classes de Domínio (MCD)	1.5-Criar modelo de classes de domínio
		Modelo de Sequência (MS)	1.6-Construir modelo de sequência para caso de uso
		Modelo de Atividades (A)	1.7-Criar o modelo de Atividades
		Descrição dos modelos (DM)	1.8-Descrever os modelos / diagramas criados
2	Arquitetura	Esboço Arquitetural (EA)	2.1-Definir arquitetura
		Modelo MVC (M-MVC)	2.2-Construir o Modelo Arquitetural (MVC)
		Modelo Decomposto (MD)	2.3-Construir o modelo decomposto (pacotes de classes)
		Diagrama de sequência do MVC (DS-MVC)	2.4-Construir diagrama de sequência
3	Projeto BD	Banco de Dados (BD)	3.1-Criar o banco de dados e as tabelas relacionais
		Scripts SQL (S-SQL)	3.2-Gerar script SQL
		Documentação BD (Doc-BD)	3.3-Documentar o Banco de dados
		Codificação BD (C-BD)	3.4-Criar a Camada de Aplicação (Interação entre Sistema e BD)
4	Codificação	Código Função de Software (CFS)	4.1-Codificar diagrama de sequência de caso de uso
		Telas (T)	4.2-Codificar Interface de Entradas (Telas)
		Código MVC (C-MVC)	4.3-Codificar arquitetura MVC
		Código Depurado (CD)	4.4-Depurar código
5	Teste	Plano de Teste (PT)	5.1-Construir Plano de Teste
		Casos de Teste Unitário (CTU)	5.2-Definir casos de teste unitário
		Classes de Teste (CT)	5.3-Codificar casos de teste (unitário e Funcional)
		Teste Unitário (TU)	5.4-Rodar casos de teste unitário
		Casos de Teste Funcional (CTF)	5.5-Definir casos de teste funcional
		Teste Funcional (TF)	5.6-Executar casos de teste funcional

J.VI Caminho Crítico



J.VII Corrente Crítica



J.VIII *Survey* Executados – Exp1, Exp2 e Exp2.

Survey Sobre a entrega #1 – VC Aluga (Fase 1 – PU-Like)

Nome: [redacted] Grupo: 9

Ordem	Categoria	Tarefa	Grau
1	Documentação	1.1-Definir os Requisitos Funcionais	3
		1.2-Definir os requisitos Não Funcionais	1
		1.3-Especificar requisitos (Casos de Uso)	3
		1.4-Criar modelo UML de casos de uso	2
		1.5-Criar modelo de classes de domínio	0 *
		1.6-Construir diagrama de sequencia para caso de uso	0 *
		1.7-Criar o modelo de Atividades	1
		1.8-Descrever os modelos / diagramas criados	2
2	Arquitetura	2.1-Definir arquitetura	
		2.2-Construir o Modelo Arquitetural (MVC)	
		2.3-Construir o modelo decomposto (pacotes de classes)	
		2.4-Construir diagrama de sequencia	
3	Projeto BD	3.1-Criar o banco de dados e as tabelas relacionais	2
		3.2-Gerar script SQL	2
		3.3-Documentar o Banco de dados	1
		3.4-Criar a Camada de Aplicação (Interação entre Sistema e BD)	0
4	Codificação	4.1-Codificar diagrama de sequência de caso de uso	0
		4.2-Codificar Interface de Entradas (Telas)	0
		4.3-Codificar arquitetura MVC	0
		4.4-Depurar código	0
5	Teste	5.1-Construir Plano de Teste	
		5.2-Definir casos de teste unitário	
		5.3-Codificar casos de teste unitário	
		5.4-Rodar casos de teste unitário	
		5.5-Definir casos de teste funcional	
		5.6-Executar casos de teste funcional	
6	Outras	6.1-<nome outras>	

1. Escreva na coluna Grau, o nível de dificuldade em uma escala de 1 a 3, onde 1 = pouca dificuldade, 2 = dificuldade moderada e 3 = muita dificuldade. *(ex. Atividade 1.5 – Representação do domínio/Modelagem (3), UML (2), identificar as classes do sistema (1).* Deixe em branco caso vc. não tenha trabalhado nela.

OBS: Não fizemos nada exposto da Arquitetura pois não sabíamos como externalizar e.g. Diferença entre diagrama de sequência de caso de uso e diagrama de sequência (comum?).

Survey Sobre a entrega #2 - SCOA (Fase 2 – PU-Like)

FES 2015/1 - Data: 26/05/2015

Nome: [redacted] **Grupo:** 2

Ordem	Categoria	Tarefa	Grau
1	**Documentação**	1.1-Definir os Requisitos Funcionais	—
		1.2-Definir os requisitos Não Funcionais	—
		1.3-Especificar requisitos (Casos de Uso)	1
		1.4-Criar modelo UML de casos de uso	1
		1.5-Criar modelo de classes de domínio	1
		1.6-Construir diagrama de sequencia para caso de uso	2
		1.7-Criar o modelo de Atividades	3
		1.8-Descrever os modelos / diagramas criados	1
2	**Arquitetura**	2.1-Definir arquitetura	1
		2.2-Construir o Modelo Arquitetural (MVC)	—
		2.3-Construir o modelo decomposto (pacotes de classes)	—
		2.4-Construir diagrama de sequencia	—
3	**Projeto BD**	3.1-Criar o banco de dados e as tabelas relacionais	2
		3.2-Gerar script SQL	1
		3.3-Documentar o Banco de dados	2
		3.4-Criar a Camada de Aplicação (Interação entre Sistema e BD)	2
4	**Codificação**	4.1-Codificar diagrama de sequência de caso de uso	2
		4.2-Codificar Interface de Entradas (Telas)	3
		4.3-Codificar arquitetura MVC	2
		4.4-Depurar código	1
5	**Teste**	5.1-Construir Plano de Teste	1
		5.2-Definir casos de teste unitário	1
		5.3-Codificar casos de teste unitário	—
		5.4-Rodar casos de teste unitário	—
		5.5-Definir casos de teste funcional	—
		5.6-Executar casos de teste funcional	—
6	**Outras**	6.1-<nome outras>	—

1. Escreva na coluna Grau, o nível de dificuldade em uma escala de 1 a 3, onde: 1 = pouca dificuldade, 2 = dificuldade moderada e 3 = muita dificuldade. *(ex. Atividade 1.5 – Representação do domínio/Modelagem (3), UML (2), identificar as classes do sistema (1).*

Senti muita dificuldade em alinhar corretamente os diagramas com o padrão desejado, pois, a princípio, não ficou claro para mim como deveriam ser os diagramas.

Além disso, me senti confusa sobre o que deveria ser feito e se, de fato, meu grupo estava fazendo certo, de modo que tivemos muito "retrabalho".

Survey Sobre a entrega #3 - SCOA (Fase 3 – PU-Like)

FES 2015/1 - Data: 16/06/2015

Nome: [redacted] **Grupo:** 9

Ordem	Categoria	Tarefa	Grau
1	**Documentação**	1.1-Definir os Requisitos Funcionais	1
		1.2-Definir os requisitos Não Funcionais	1
		1.3-Especificar requisitos (Casos de Uso)	1
		1.4-Criar modelo UML de casos de uso	1
		1.5-Criar modelo de classes de domínio	1
		1.6-Construir diagrama de sequencia para caso de uso	1
		1.7-Criar o modelo de Atividades	1
		1.8-Descrever os modelos / diagramas criados	1
2	**Arquitetura**	2.1-Definir arquitetura	2
		2.2-Construir o Modelo Arquitetural (MVC)	2
		2.3-Construir o modelo decomposto (pacotes de classes)	2
		2.4-Construir diagrama de sequencia	1
3	**Projeto BD**	3.1-Criar o banco de dados e as tabelas relacionais	1
		3.2-Gerar script SQL	1
		3.3-Documentar o Banco de dados	1
		3.4-Criar a Camada de Aplicação (Interação entre Sistema e BD)	2
4	**Codificação**	4.1-Codificar diagrama de sequência de caso de uso	2
		4.2-Codificar Interface de Entradas (Telas)	2
		4.3-Codificar arquitetura MVC	2
		4.4-Depurar código	1
5	**Teste**	5.1-Construir Plano de Teste	3
		5.2-Definir casos de teste unitário	3
		5.3-Codificar casos de teste unitário	3
		5.4-Rodar casos de teste unitário	3
		5.5-Definir casos de teste funcional	3
		5.6-Executar casos de teste funcional	3
6	**Outras**	6.1-<nome outras>	

1. Escreva na coluna Grau, o nível de dificuldade em uma escala de 1 a 3, onde: 1 = pouca dificuldade, 2 = dificuldade moderada e 3 = muita dificuldade. *(ex. Atividade 1.5 – Representação do domínio/Modelagem (3), UML (2), identificar as classes do sistema (1).*

Survey Sobre a entrega #4 - VVV (Fase 4 – PU-Like)

FES 2015/2 - Data: 15/03/2016

Nome: [redacted] Grupo: 5

Ordem	Categoria	Tarefa	Crau
1	Documentação	1.1-Definir os Requisitos Funcionais	1
		1.2-Definir os requisitos Não Funcionais	1
		1.3-Especificar requisitos (Casos de Uso)	1
		1.4-Criar modelo UML de casos de uso	1
		1.5-Criar modelo de classes de domínio	3
		1.6-Construir diagrama de sequencia para caso de uso	2
		1.7-Criar o modelo de Atividades	1
		1.8-Descrever os modelos / diagramas criados	1
2	Arquitetura	2.1-Definir arquitetura	1
		2.2-Construir o Modelo Arquitetural (MVC)	1
		2.3-Construir o modelo decomposto (pacotes de classes)	1
		2.4-Construir diagrama de sequencia	3
3	Projeto BD	3.1-Criar o banco de dados e as tabelas relacionais	1
		3.2-Gerar script SQL	1
		3.3-Documentar o Banco de dados	1
		3.4-Criar a Camada de Aplicação (Interação entre Sistema e BD)	1
4	Codificação	4.1-Codificar diagrama de sequência de caso de uso	3
		4.2-Codificar Interface de Entradas (Telas)	1
		4.3-Codificar arquitetura MVC	1
		4.4-Depurar código	1
5	Teste	5.1-Construir Plano de Teste	1
		5.2-Definir casos de teste unitário	1
		5.3-Codificar casos de teste unitário	2
		5.4-Rodar casos de teste unitário	2
		5.5-Definir casos de teste funcional	3
		5.6-Executar casos de teste funcional	3
6	Outras	6.1-<nome outras>	

1. Escreva na coluna Grau, o nível de dificuldade em uma escala de 1 a 3, onde: 1 = pouca dificuldade, 2 = dificuldade moderada e 3 = muita dificuldade. *(ex. Atividade 1.5 – Representação do domínio/Modelagem (3), UML (2), identificar as classes do sistema (1).*
2. *Zero (0) ou branco para atividades não realizadas ou que não participaram.*

J.IX Comunicação Pessoal (E-mail do Mr. Grady Booch)

Sildenir Alves Ribeiro <sildenir.ribeiro@gmail.com>

Unified Process: Methodology, Method or Simply Process

3 mensagens

Sildenir Alves Ribeiro <sildenir.ribeiro@gmail.com> 20 de fevereiro de 2016 16:44
Para: egrady@booch.com

Dear Mr Booch;

I'm a Doctoral Student and I and my advisor have discussed a lot about the unified process.

After all, the Unified Process is a methodology, a method or a process?

My thesis is about identification and treatment of restrictive elements or bottleneck in the software development process.

It would be very important to your response to our goals.

Sincerely

--
Sildenir Alves Ribeiro
Professor de Informática e Ciência da Computação
Centro Federal de Educação Tecnológica - CEFET/RJ
(Federal Centre of Technological Education)

Grady Booch <egrady@booch.com> 20 de fevereiro de 2016 21:21
Para: Sildenir Alves Ribeiro <sildenir.ribeiro@gmail.com>

Thank you for your kind email

It really is all three

As a methodology the unified process defines a strategy

As a method the unified process defines the modes whereby that strategy is to be carried out

As a process that unified process defines a series of steps which one may follow through achieve those modes

On Feb 20, 2016, at 09:44, Sildenir Alves Ribeiro <sildenir.ribeiro@gmail.com> wrote:

> Dear Mr Booch;
>
> I'm a Doctoral Student and I and my advisor have discussed a lot about the unified process.
>
> After all, the Unified Process is a methodology, a method or a process?
>
> My thesis is about identification and treatment of restrictive elements or bottleneck in the software development process.
>
> It would be very important to your response to our goals.
>
> Sincerely
>
> --
> **Sildenir Alves Ribeiro**
> *Professor de Informática e Ciência da Computação*
> **Centro Federal de Educação Tecnológica - CEFET/RJ**
> ***(Federal Centre of Technological Education)***

Sildenir Alves Ribeiro <sildenir.ribeiro@gmail.com> 21 de fevereiro de 2016 00:43
Para: Grady Booch <egrady@booch.com>

Dear Mr. G. Booch

Thank you for your precise and fast response.

I got really happy, because your answer validated my / our understanding about the UP.

I'm a great admirer of his work;

Thanks for all;

Sildenir Ribeiro
Federal University of Rio de Janeiro
Federal Center of Technological Education of Rio de Janeiro