

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

***SMART EXECUTION OF MOLECULAR
DOCKING SIMULATIONS OF A FULLY-
FLEXIBLE RECEPTOR MODEL***

FÁBIO ANDRÉ FRANTZ

Thesis submitted to the Computer Science
Graduate Program of Pontifícia Universidade
Católica do Rio Grande do Sul – PUCRS as partial
fulfillment of the requirements for the degree of
Master in Computer Science

Advisor: Prof. Dr. Duncan Dubugras Alcoba Ruiz

Porto Alegre, May 2012.

F836s Frantz, Fábio André

Smart execution of molecular docking simulations of a fully-flexible receptor model / Fábio André Frantz. – Porto Alegre, 2012.

93 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.

Orientador: Prof. Dr. Duncan Dubugras Alcoba Ruiz.

1. Informática. 2. Banco de Dados. 3. Biologia Molecular.
4. Biologia Computacional. I. Ruiz, Duncan Dubugras Alcoba. II. Título.

CDD 005.74


**Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS**



Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "*Smart Execution of Molecular Docking Simulations of A Fully - Flexible Receptor Model*", apresentada por Fábio André Frantz como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas de Informação, aprovada em 12/04/2012 pela Comissão Examinadora:



Prof. Dr. Dencas Dubugras Alcoba Ruiz - PPGCC/PUCRS
Orientador

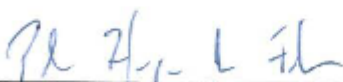


Prof. Dr. Osmar Norberto de Souza - PPGCC/PUCRS



Prof. Dr. Aad van Moorsel - *University of Newcastle*

Homologada em 08/06/2012, conforme Ata No. 012... pela Comissão Coordenadora.



Prof. Dr. Paulo Henrique Lemelle Fernandes
Coordenador.

PUCRS

Campus Central
Av. Ipiranga, 6681 - P32 - sala 507 - CEP: 90619-900
Fone: (51) 3320-3611 - Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

“Remembering that you are going to die is the best way I know to avoid the trap of thinking you have something to lose. You are already naked. There is no reason not to follow your heart.”

Steve Jobs

ACKNOWLEDGEMENTS

I would like to thank, first and foremost, my advisor, Prof. Dr. Duncan Dubugras Alcoba Ruiz for his guidance, suggestions and continuous support throughout my Master's study and completion of this thesis. I feel fortunate of being one of his research students throughout my study, which provided me with a chance of being instructed by an inspiring and insightful academic professor. Also thank you for all your support in my extracurricular life and for providing me with a great experience in my life.

Also, I would like to thank Prof. Dr. Osmar Norberto de Souza for his guidance in many stages of this work. His knowledge in Bioinformatics, especially in Molecular Docking, was responsible for a concise work. Thank you for all your useful insights.

I would also like to thank all GPIN and LABIO researchers for friendship and help every time I needed. In particular, I thank Renata de Paris, who was a great partner in the realization of this thesis, as well as a co-author in some papers.

Thanks to HP R&D Brazil that provided a scholarship for my study and helped in different times during these years of study.

Last, I am very grateful to my family for their support, in special to Maristela Trojahn, who walked side by side during this long way. Thank you for your love, encouragement and support.

SMART EXECUTION OF MOLECULAR DOCKING SIMULATIONS OF A FULLY-FLEXIBLE RECEPTOR MODEL

ABSTRACT

Molecular docking simulations of Fully-Flexible Receptor (FFR) models are coming of age. However, they demand parallelization of computing activities for their executions and generate huge amounts of data that needs to be analyzed. Many Task Computing (MTC) is an attractive paradigm routinely applied to execute intensive tasks. In this work we propose an environment to execute molecular docking simulations of FFR models to small molecules integrated with an MTC middleware. This environment is based on a new pattern called Self-adapting Multiple Instances (P-SaMI) that provide rules to reduce the number of experiments, providing a Reduced Fully-Flexible Receptor (RFFR) model. The main contribution of this research is to prove that P-SaMI rules can be used on Molecular Docking Simulations through a web environment integrated with an MTC middleware.

Keywords: Molecular Docking, Bioinformatics, P-SaMI, MTC, RDD

EXECUÇÃO INTELIGENTE DE SIMULAÇÕES DE DOCAGEM MOLECULAR DE UM MODELO DE RECEPTOR TOTALMENTE FLEXÍVEL

RESUMO

Simulações de docagem molecular com modelos de Receptores Totalmente Flexíveis (FFR) estão adquirindo maturidade. No entanto, isto demanda atividades computacionais de paralelização para geração e execução de grande volume de dados que precisam ser analisados. Computação multi-tarefa é um paradigma atrativo e que vem sendo aplicado frequentemente para executar tarefas intensivas. Neste trabalho propomos um ambiente para executar simulações de docagem molecular no modelo FFR com pequenas moléculas integradas a um componente MTC. Este ambiente é baseado no padrão Múltiplas Instâncias Autoadaptáveis (P-SaMI) que possui regras para redução do número de experimentos, provendo um modelo de Receptores Totalmente Flexíveis Reduzido (RFFR). A principal contribuição desta pesquisa está na comprovação de que as regras do P-SaMI podem ser usadas em Simulações de Docagem Molecular através de um ambiente web integrado com um componente MTC.

Palavras-Chave: Docagem Molecular, Bioinformática, P-SaMI, MTC, RDD

LIST OF FIGURES

FIGURE 1 – COMPARISON OF SCIENTIFIC WORKFLOWS AND BUSINESS WORKFLOWS [YIL09]	20
FIGURE 2 – SCIENTIFIC WORKFLOW LIFE CYCLE [LUD09].....	21
FIGURE 3 – DATA INTERACTION: TASK TO ENVIRONMENT - PUSH [RUS04].....	23
FIGURE 4 – FLOWCHART OF RATIONAL DRUG DESIGN. ADAPTED FROM [MAC11].	27
FIGURE 5 – (A) SCHEMATIC 3D REPRESENTATION OF MOLECULAR DOCKING. THE PROTEIN IS REPRESENTED IN THE FORM OF RIBBONS, IN GRAY, AND LIGAND IN (MAGENTA AND CYAN). (B) FLEXIBILITY OF INHA-NADH SYSTEM IN DIFFERENT MOMENTS DURING THE MOLECULAR DYNAMICS SIMULATION. OVERLAP OF DIFFERENT CONFORMATIONS OF INHA (CYAN, YELLOW, MAGENTA AND GREEN) GENERATED FOR MOLECULAR IN [SCH05]. PICTURE FROM [MAC07].....	28
FIGURE 6 – FINAL MODEL OF SCIENTIFIC WORKFLOW FREDoWS [MAC11A]	32
FIGURE 7 – P-SAMI: SELF-ADAPTING MULTIPLE INSTANCES PATTERN.....	34
FIGURE 8 – STATE TRANSITION DIAGRAM FOR POSSIBLE STATUS IN SNAPSHOT PROCESSING [HUB10]	36
FIGURE 9 – REPRESENTATION OF LOT GENERATION - ADAPTED FROM [HUB10].....	37
FIGURE 10 – EXECUTION OF SNAPSHOTS FROM A GROUP - ADAPTED FROM [HUB10]	38
FIGURE 11 – SEPARATION OF LOTS FROM GROUPS AND REPRESENTATION OF INDIVIDUAL RESULTS FROM EACH SNAPSHOT AND FROM ALL LOT [HUB10]	40
FIGURE 12 – ALGORITHM TO DEFINE PRIORITIES AND STATUS CHANGE.....	41
FIGURE 13 – W-FREDoW CONCEPTUAL ARCHITECTURE	44
FIGURE 14 – XML WITH GROUP OF SNAPSHOTS.....	46
FIGURE 15 – DIRECTORIES STRUCTURE OF FILES OF THE MTC LAYER WORKSPACE.....	47
FIGURE 16 – OPERATIONS FLOWS AND FUNCTIONS HAVE BEEN PERFORMED BETWEEN CREATE QUEUE AND DISPATCHER/MONITOR COMPONENTS	49
FIGURE 17 – FREDD FOR P-SAMI: DATA MODEL FOR W-FREDoW	51
FIGURE 18 – W-FREDoW UI: CLIENT LAYER	59
FIGURE 19 – W-FREDoW UI: SETUP MAIN GROUP. SOFTWARE DOCKING, CLUSTER, P-SAMI AND CONFIGURATION WINDOWS.....	60
FIGURE 20 – W-FREDoW UI: SETUP DOCKING-TEMPLATES GROUP. PREPARE FILES - GROUPS, TEMPLATE FOR PREPARE RECEPTOR, DOCKING AND GRID WINDOWS.	61
FIGURE 21 – W-FREDoW: EXECUTE EXPERIMENT.....	62
FIGURE 22 – W-FREDoW: ANALYZE.....	62
FIGURE 23 – W-FREDoW: ANALYZE. (A) DETAILS FROM EXPERIMENT 197 FOR CLUSTER AND LOT. (B) RESULTS FOR SNAPSHOTS FROM CLUSTER 0 IN EXPERIMENT 197.....	63
FIGURE 24 – W-FREDoW: 3.100 SNAPSHOTS WITH A UNIQUE CLUSTER.....	66
FIGURE 25 – W-FREDoW: SETUP OF P-SAMI FOR A FULL EXECUTION.....	67
FIGURE 26 – W-FREDoW: VALUES OF CONFIGURATION #5, USED IN TEST 1.....	67
FIGURE 27 – W-FREDoW: PARAMETERS FOR TEST 1 EXPERIMENT EXECUTION.....	68
FIGURE 28 – CRD AND PDBQT FILES USED TO CREATE SNAPSHOTS.....	69
FIGURE 29 – PARTS OF GRID AND DOCKING TEMPLATES USED IN TEST 1.	70
FIGURE 30 – STRUCTURE OF FOLDERS CREATED FOR AN EXPERIMENT.	70
FIGURE 31 – EXPERIMENT 196 WITH 3100 SNAPSHOTS FINISHED.	72
FIGURE 32 – EVENTS FOR EXPERIMENT 196.	72
FIGURE 33 – CLOUD SETUP FOR COMMUNICATION BETWEEN W-FREDoW AND FREMI.....	73
FIGURE 34 – FOUR DIFFERENT SETUPS FOR P-SAMI WITH ANALYSIS STARTING WITH 30%, 40%, 50% OR 70%.	73
FIGURE 35 – DIFFERENT CONFIGURATIONS THAT ARE USED ON EXPERIMENTS WITH ANALYSIS STARTING WITH 30%, 40%, 50% AND 70%.....	74
FIGURE 36 – ANALYSIS REALIZED IN EXPERIMENT 224. FOUR ROWS SELECTED TO SHOW THE CLUSTER 0 AND LOT 3, WHERE THE FIRST HAD ITS PRIORITY DECREASED AND AT LAST WAS DISCARDED WITH 70% BASED ON THE AVG1 (ARITHMETIC AVERAGE) AND AVG2 (ESTIMATED AVERAGE) COMPARING WITH AVG3 (P-SAMI AVERAGE).....	76
FIGURE 37 – HYDRA CONCEPTUAL ARCHITECTURE	83

LIST OF TABLES

TABLE 1 – QUANTITY OF CONFORMATIONS PER EACH GROUP, GENERATED FROM <i>K-MEANS</i> ALGORITHM FOR <i>RMS</i> AND <i>TCN_MULT2</i> SIMILARITY FUNCTIONS	35
TABLE 2 – FINAL RESULTS OF EXPERIMENT EXECUTIONS OF W-FREDO _W CONSIDERING THE <i>RMS_LISTAGEM_CLUSTER_SNAP_MEANS_6_ALL_ATOMS</i> AS SIMILARITY FUNCTION.	75
TABLE 3 – FINAL RESULTS OF EXPERIMENT EXECUTIONS OF W-FREDO _W CONSIDERING THE <i>TCN_LISTAGEM_CLUSTER_SNAP_MEANS_6_ALL_ATOMS</i> AS SIMILARITY FUNCTION.	77
TABLE 4 – FINAL RESULTS OF EXPERIMENT EXECUTIONS OF W-FREDO _W CONSIDERING THE <i>GROUP_PATRICIA_FOR_NADH_K-MEANS</i> AS SIMILARITY FUNCTION.	78
TABLE 5 – TIME SPENT TO EXECUTE MDOCK SIMULATIONS WITH W-FREDO _W USING FREMI MIDDLEWARE.	78
TABLE 6 – FINAL STATUS OF SNAPSHOTS AFTER DOCKING WITH W-FREDO _W USING FREMI MIDDLEWARE.	79

GLOSSARY

3D	Tridimensional
DBMS	Data Base Management System
FEB	Free Energy of Binding
FFR	Fully-Flexible Receptor
FReDD	Flexible Receptor Docking Database
FReDoWS	Flexible Receptor Docking Workflow System
GPIN	Business Intelligent Research Group
HPC	High Performance Computing
InhA	Enzyme 2-trans-Enoil ACP (CoA) Reductase of Mycobacterium Tuberculosis
LABio	Bioinformatics, Modeling, and Biosystems Simulation Laboratory
MDock	Molecular Docking
MD	Molecular Dynamic
MTC	Many Task Computing
NADH	Nicotinamida Adenina Dinucleotídeo, reduced form
WHO	World Health Organization
P-SaMI	Self-adapting Multiple Instances
PDB	Protein Data Bank
PUCRS	Pontifícia Universidade Católica do Rio Grande do Sul
RDD	Rational Drug Design
RFFR	Reduced Fully Flexible Receptor
RMSD	Root Mean Squared Deviation
SQL	Structured Query Language
SWfMS	Scientific Workflow Management System
VM	Virtual Machine
WfMC	Workflow Management Coalition
WfMS	Workflow Management System
W-FReDoW	Web Flexible Receptor Docking Workflow
WSGI	Web Server Gateway Interface

XML

Extensible Markup Language

TABLE OF CONTENTS

1	INTRODUCTION	15
1.1	STATEMENT OF THE PROBLEM	15
1.2	RESEARCH QUESTION	16
1.3	GOALS	16
1.3.1	<i>General Purpose</i>	16
1.3.2	<i>Specific Purposes</i>	16
1.4	RESEARCH METHODS	17
1.5	ORGANIZATION OF THE WORK	17
2	LITERATURE REVIEW	19
2.1	SCIENTIFIC WORKFLOWS	19
2.1.1	<i>Workflow patterns</i>	22
2.1.2	<i>Scientific Workflow Management System</i>	23
2.1.3	<i>Parallelism in Scientific Workflows</i>	25
2.1.4	<i>Provenance in Scientific Workflows</i>	25
2.2	BIOINFORMATICS	26
2.2.1	<i>Rational Drug Design</i>	26
2.2.2	<i>Molecular Docking</i>	27
2.2.3	<i>Flexibility of Receptor</i>	29
2.2.4	<i>Molecular Dynamics</i>	29
2.3	LAST CHAPTER CONSIDERATIONS	29
3	PATTERN P-SAMI - SELF-ADAPTING MULTIPLE INSTANCES	31
3.1	FReDOWS	31
3.2	CONCEPT OF THE PATTERN	33
3.2.1	<i>Similarity Function</i>	34
3.2.2	<i>P-SaMI Setup</i>	35
3.2.3	<i>P-SaMI: Lot Generation</i>	37
3.2.4	<i>P-SaMI: Processing</i>	38
3.2.5	<i>P-SaMI: Analyze</i>	39
3.2.6	<i>Formalism</i>	42
3.3	LAST CHAPTER CONSIDERATIONS	42
4	W-FREDOW (WEB FLEXIBLE RECEPTOR DOCKING WOKFLOW)	43
4.1	THE CONCEPTUAL ARCHITECTURE	43
4.2	CLIENT LAYER	44
4.3	SERVER CONTROLLER LAYER	45
4.4	MTC LAYER	46
4.5	W-FREDOW: DETAILS OF IMPLEMENTATION	49
4.5.1	<i>Database: FReDD for P-SaMI</i>	50
4.5.1.1	<i>Setup Tables</i>	52
4.5.1.2	<i>Execution Tables</i>	53
4.5.1.3	<i>Template Tables</i>	55
4.5.1.4	<i>Provenance Tables</i>	57
4.5.2	<i>Web Application Framework</i>	57
4.5.3	<i>UI – The Client Layer</i>	58

5	W-FREDOW: TESTS & RESULTS.....	65
5.1	W-FREDOW TEST 1: WITHOUT A SIMILARITY FUNCTION.....	66
5.2	W-FREDOW TEST2: RMS_LISTAGEM_CLUSTER_SNAP_MEANS_6_ALL_ATOMS.....	73
5.3	W-FREDOW TEST3: TCN_LISTAGEM_CLUSTER_SNAP_MEANS_6_ALL_ATOMS.....	76
5.4	W-FREDOW TEST4: GROUP_PATRICIA_FOR_NADH_K-MEANS.....	77
5.5	LAST CHAPTER CONSIDERATIONS.....	78
6	RELATED WORK.....	81
6.1	ADVANCED DATA FLOW SUPPORT FOR SCIENTIFIC GRID WORKFLOW APPLICATIONS.....	81
6.2	MOLECULAR DOCKING SIMULATION OF FLEXIBLE RECEPTOR.....	82
6.3	DATA PARALLELISM IN BIOINFORMATICS WORKFLOWS.....	82
6.4	DOVIS: AN IMPLEMENTATION FOR HIGH-THROUGHPUT VIRTUAL SCREENING USING AUTODOCK.....	84
6.5	RELAXED COMPLEX SCHEME FOR RECEPTOR FLEXIBILITY IN COMPUTER-AIDED DRUG DESIGN.....	84
6.6	FREDD: SUPPORTING MINING STRATEGIES THROUGH A FLEXIBLE-RECEPTOR DOCKING DATABASE.....	84
7	CONCLUSIONS.....	87
7.1	FUTURE WORK.....	88
	REFERENCES.....	89

1 INTRODUCTION

1.1 Statement of the Problem

The pharmaceutical industry has been seeking to reach ever-higher goals in relation to the launch of new drugs to the market, with an estimated average of 14 years to obtain the approval of an effective drug, and estimated costs of \$800 million [CAS07]. In order to achieve greater competitiveness, pharmaceutical companies are in a constant search for cost and time reduction in their process of discovering new drugs.

The rational drug design (RDD) refers in the most specific sense to the systematic exploration of the three-dimensional structures (3D) of a macromolecule of pharmacological importance, in order to design potential ligands that will bind to the target with high affinity and specificity [STO93]. During the RDD, the stage of molecular docking (MDock) is one of the most important stages. This step is responsible for providing a better guide so that a molecule will bind with another to form a stable complex [LEN96]. In general, molecular docking algorithms consider receptors such as rigid structures. However, such receptors in the cellular environment are flexible and not rigid as is the case in some algorithms. Although some algorithms already consider the explicit flexibility of the receptor, the tools responsible for implementing and analyzing docking experiments are still poor.

In order to map and execute the steps of RDD, we can cite FReDoWS (Flexible Receptor Docking Workflow System) [MAC07], a workflow-based solution in which the information flows of RDD. This workflow identifies the steps mentioned and also contributes to the automation of manual procedures, such as execution of several shell scripts, which are tedious and time-consuming procedures for manual execution. One of the needs related to FReDoWS is in the molecular docking step, which has a high computational cost. The flexible receptors, used in FReDoWS, are simulated by a set of snapshots, which represent different conformations of the receptor.

For the development of workflows, there are two well-known patterns: control flow [RUS06] and data pattern [RUS04]. In order to create a new pattern based on the data pattern, able to properly handle RDD steps, we have the Self-adapting Multiple Instances pattern (P-SaMI)

[HUB10]. P-SaMI is a pattern created to be used with clusters of snapshots, where after an execution of samples of molecular docking simulations of a cluster, it should be able to identify the most promising clusters. Thus, with promising clusters identified, it can provide priority changes in the execution, as well as discard those snapshots non-promising.

In this dissertation we create an environment able to execute the steps of RDD with focus on Molecular Docking simulations; as mentioned, a costly process. For that, rules of P-SaMI were used to achieve the benefits from this pattern. The pattern is not restricted to the RDD process. However, its process was used to guide the design of this environment.

Although P-SaMI has been defined for Scientific Workflows, its implementation occurred in a different context. The environment presented in this thesis is web-based and has communication with a Many Task Computing (MTC) middleware, responsible for MDock simulations in a High Performance Computing (HPC) environment. Thus we made the first implementation of P-SaMI, executing MDock simulations of Fully-flexible Receptor (FFR) model in order to reduce the quantity of dockings.

1.2 Research Question

Due to processing needs of high data volume in the Bioinformatics area, we have a question we attempt to address in this research: “Is it possible to use a reduction experiments pattern through the disposal of snapshots non-promising in molecular docking simulations of Fully-flexible Receptor Model while sustaining a high level of accuracy?”

1.3 Goals

1.3.1 General Purpose

The overall objective is to enable the use of P-SaMI (Pattern Self-adapting Multiple Instances) in molecular docking (MDock) simulations of a Fully-Flexible Protein Receptor (FFR) model. Thus, characteristics presented in this pattern will contribute to reduce the total execution time of molecular docking simulations through the disposal of snapshots without affecting the quality of the FFR model. Also, we want to reduce the total time through integration with a middleware responsible for running MDock simulation in Many Task Computing (MTC).

1.3.2 Specific Purposes

We want to achieve the following specific purposes:

- Enable P-SaMI in MDock simulations of Fully-Flexible Receptor (FFR) model environment.
- Build a new environment for MDock simulations of FFR model based on FReDoWS using the new version of molecular docking software;
- Create a new structure in FReDD (Flexible Receptor Docking Database) to store all steps used during the execution of this new environment, keeping the provenance of the molecular docking simulations. This structure will be the source of information for analysis of P-SaMI;
- Communicate MDock simulations web environment with MTC (Many Task Computing) middleware, sending and receiving in real time data about the process. Thus, the execution of MDock simulations, a critical step from RDD, can be executed as a multi-distributed process, providing gain of time to all RDD processes;

1.4 Research Methods

To carry out this thesis some important steps have been undertaken: a mapping to understand the steps of RDD existing in LABio (Bioinformatics, Modeling, and Biosystems Simulation Laboratory); a literature review to understand the concepts involving SWf (Scientific Workflows) and the patterns presented in P-SaMI; the creation of a new environment for MDock simulations of FFR model based on FReDoWS integrated with a MTC environment. Finally, we demonstrate the results of the execution of the environment, analyzing the use of P-SaMI and the integration of MTC middleware in RDD workflow.

1.5 Organization of the Work

This thesis is organized as follows:

- In Chapter 2 we present a literature review about scientific workflows, showing some patterns used to create a new environment. Furthermore, this chapter presents important concepts in RDD (Rational Drug Design), based on the flow presented in this thesis;
- The next chapter, Chapter 3, details P-SaMI (Pattern - Self-adapting Multiple Instances), demonstrating rules of pattern, how it works and where it can be applied;

- The new environment called W-FReDoW (Web Flexible Receptor Docking Workflow) is discussed in Chapter 4. Each part of W-FReDoW architecture is shown in details;
- Next, Chapter 5 elaborates on the results of experiments realized with W-FReDoW, with analysis of the use of P-SaMI. These experiments are real simulations of docking of FFR model. Some of them are shown in [HUB10] and can be validated;
- In Chapter 6, related works are presented, as well as a comparison of differences and improvements of this work;
- Finally, Chapter 7 provides final conclusions and future works related to this thesis.

2 LITERATURE REVIEW

This chapter describes concepts related to scientific workflows, the foundation of this thesis. These concepts were important to build the environment of this research, and to understand the different workflow patterns and help achieve the main goals.

Moreover, this chapter also presents the field where the study was done, the Bioinformatics. This multidisciplinary research area is the interface between informatics and biology and has the characteristics to work with large datasets, one of the requirements of the present work. Thus, throughout this chapter, the sections explain concepts of Bioinformatics that are used in this thesis. Finally, last considerations about this chapter are presented.

2.1 Scientific Workflows

Before starting to explain scientific workflows we have to start with workflow. The Workflow Management Coalition (WfMC) [WOR99] defines workflow as “The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules”.

Workflows have been associated to business for many years. Yildiz et al. [YIL09] say business workflows aim to automate and optimize organization’s processes fulfilled by human or computer agents in an administrative context. The latter concerns agents, roles, manipulate objects (resources) and, especially, the partial order or coordination among activities. The use of business workflows is prevalent among insurance, banking, and health industries.

Yildiz et al. [YIL09] define the transition from business workflows to scientific workflows. Claiming that when workflows move from the business place to the scientific laboratory, supporting large-scale, complex, yet fault-tolerant and maintainable scientific processes, they demand another systematic approach. This approach has been named as scientific workflows. Scientific workflows, which have been around for more than ten years, aim to support scientists in designing and implementing large-scale and complex e-science processes. Figure 1 shows a comparison between scientific workflows and business workflows.

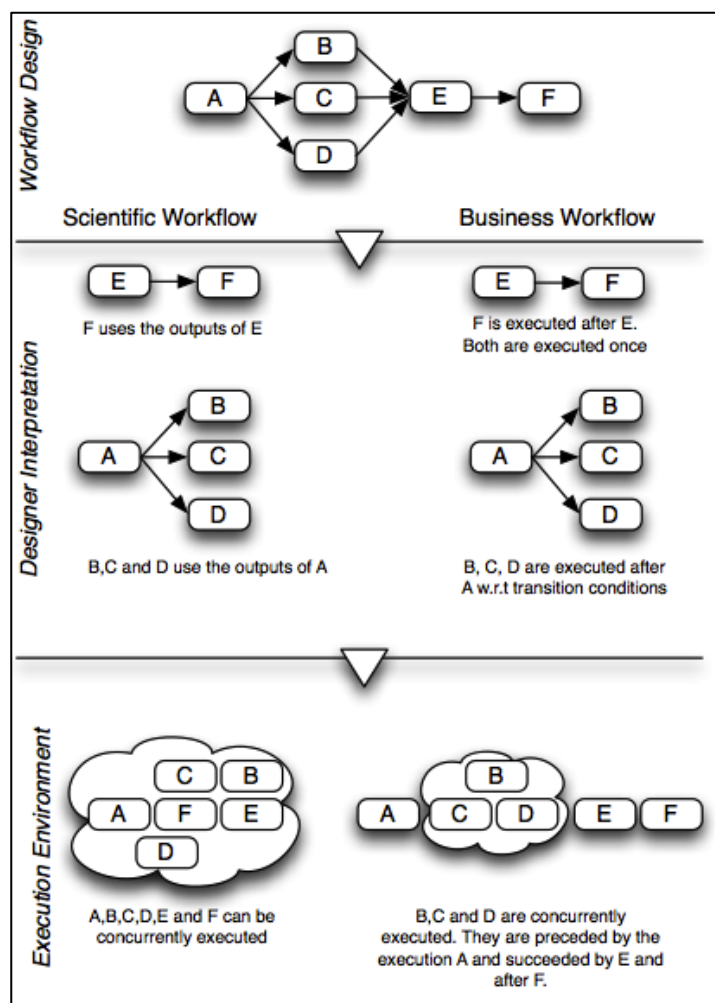


Figure 1 – Comparison of Scientific Workflows and Business Workflows [YIL09]

Barseghian et al. [BAR09] quote scientific workflows as representations of the processes involved in accomplishing a scientific analysis. They combine data and computational procedures into a configurable, structured set of steps that implement semi-automated computational solutions to a scientific question. A scientific analysis, depending on its focus, can involve a number of ad-hoc processes that a scientist may use to go from raw data to publishable results. Some areas in which we can perceive the use of scientific workflows are Bioinformatics, Geoinformatics, Cheminformatics, Biomedical Informatics and Astrophysics.

Figure 2 is a high-level representation of the scientific workflow life cycle [LUD09]. The life cycle begins occurs with one of these two goals: a scientific hypothesis to be tested; or, with some experimental goals one proposes to achieve. Then, the **Workflow Design** takes place, in which the scientists generally reuse previous workflows and templates or try to perform some adaptations. Following, there is the **Preparation** step, in which it is important to select the data sources and parameters that will be used during the execution. Also, it is important to consider where the

workflow will run, local or remote, and if it will use a high-performance computing (HPC), GRID or Cloud Computing.

Next, the **Workflow Execution** is where the data from the workflow are consumed and others are created. As we are talking about large-scale computational science simulations (executing data during hours, days or weeks at a time) the *runtime monitoring* has an important role in the life cycle. It is important that the scientist is informed through a dashboard about the status of the execution. Intermediate results and provenance are useful for any strategic decision by the scientist, such as aborting or restarting the workflow, based on preliminary bad results, for instance.

After the execution of experiments, the scientist must analyze the results. This occurs in **Post-Execution Analysis** and some questions are normally asked: “do these results make sense?”, “why did this step fail?”, and “which steps took the longest time?”. Based on these analyses, the process can start again.

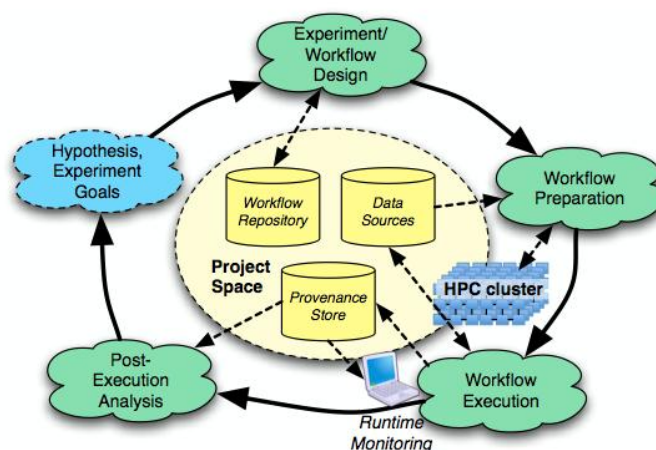


Figure 2 – Scientific Workflow Life Cycle [LUD09]

Following a study in many projects, Ludäscher et al. [LUD06] summarize a number of common requirements and desirable of scientific workflows:

- Seamless access to resources and services: web services are a simple mechanisms to deal with remote service execution and remote database access;
- Service composition & reuse and workflow design: the composition of services has to be as easy as possible for the user, even for complex processes;
- Scalability: it is important for the large volume of data and computational resources to be able to run in large parallel jobs on a cluster computer. To support

such data-intensive and compute-intensive workflows, suitable interfaces to Grid middleware components (sometimes called Compute-Grid and Data-Grid, respectively) are necessary;

- Detached Execution: the workflow engine has to be responsible for controlling the execution on background or on a remote server, allowing the scientist to do another task;
- Reliability and fault-tolerance: to make a workflow more resilient in an inherently unreliable environment, contingency actions must be specifiable, e.g., fail-over strategies with alternate web services;
- User-interaction: sometimes the workflow requires user interaction in different steps of the execution;
- “Smart” re-runs: provide the user with the opportunity to change some parameters and re-run just a sub-flow, e.g., not executing from scratch;
- “Smart” (semantic) links: the scientific workflow system must be based on semantics to facilitate the configuration by the user;
- Data provenance: the experiment executed must be reproducible and the steps, parameters and data captured have to be logged for further analysis.

2.1.1 Workflow patterns

There are two types of patterns for workflow development: flow control pattern [RUS06] and data pattern [RUS04]. The use of links & connections between tasks is the place where the patterns occur [HUB10]. In summary, we have:

- In flow control pattern, the links represent control restrictions for tasks performance. There are many control structures such as sequences, splits, joins and loops;
- In data pattern, the links between tasks are dependencies of data, where data are consumed and produced by these tasks.

Russel et al. [RUS04] classify solutions as the prospect of the data and their characteristics. Among these perspectives are:

- Data visibility: data can be accessed by different components from a workflow process;
- Data interaction: capture the various ways in which data elements can be passed between components in a workflow process and how the characteristics of the individual components can influence the manner in which the trafficking of data elements occurs;
- Data transfer: focus on the form in which the actual transfer of data elements occurs between one workflow component and another;
- Data-based routing: how data types can influence the operations and other aspects of the workflow, mainly during the flow controls.

The workflow data pattern has 39 different elements that represent all possibilities from the pattern [RUS04, HUB10]. As the goal of this thesis is not to explain every element, we exemplify one of them. Figure 3 depicts pattern 15, in which tasks from the workflow communicate with external environment.

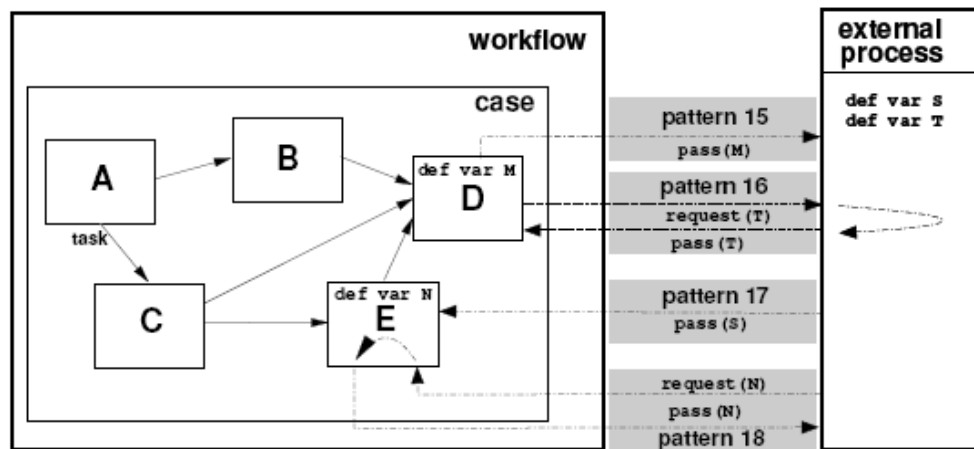


Figure 3 – Data Interaction: Task to environment - Push [RUS04]

2.1.2 Scientific Workflow Management System

Nowadays, scientists from several areas find support for their researches through the use of computational tools to get faster results, aiding in the process of analysis and discovery. These researchers can contribute during different steps of the process in real time, despite being geographically distant. A variety of tools can be used in the scientific process. With that, it is essential to use a tool to manage all different steps of the process, which sometimes can run in different places.

The use of SWfMS (Scientific Workflow Management System) in the process of computational experiment (*in-silico*) provides an environment that simplifies the design, implementation and documentation of experiments [LUD09]. Some of the advantages of using SWfMS are: automation of repetitive tasks, explanation of process documentation, monitoring of workflow execution, provenance record of results, optimization of the scientific process, as well as efficient execution and, finally, possibility of reusing known artifacts.

Mattoso et al. [MAT09] represent in some steps the experiment life cycle showed by [GOB07]. Each step is a categorization that represents parts of the experiment life cycle, as workflows concept, workflows execution and workflows analysis. They are:

Workflows concept:

- Allow scientists and researchers to work in the core of the experiment in a controlled and systematic reuse of knowledge;
- Support the modeling and record of knowledge from domain using ontologies;
- Support the protocol modeling of experiment, using ontologies and abstract and concrete workflows;
- Allow a description of all packages used in the experiment, saving semantic relationships, based to validate next executions.

Workflows execution:

- Keep the provenance, recording tasks used during the process;
- Provide support for an efficient execution of scientific workflows, allowing the execution in GRID and distributed environments;

Workflows analysis:

- Provide information during the execution of the workflow tasks about the parameters and results;
- Allow ad-hoc searches in the results of experiments, giving the provenance of data, process and knowledge.

Currently, there are some SWfMS that have different focus, such as Vistrails [CAL06], Kepler [ALT04] and Taverna [HUL06], which are strong in provenance support and concrete workflow design but lack on HPC (High-performance Computing) support. On the other hand,

others SWfMS are strong to work with GRIDs [YUJ05], such as Pegasus [9], Swift [RAI07] and Triana [TAY07], which perform well in HPC but do not provide version control to support, workflow design with provenance.

2.1.3 Parallelism in Scientific Workflows

As mentioned, scientific workflows deal with a large amount of experiments. Normally these experiments require high computational power to obtain faster results. Sequential executions presented a first and easy solution, but with a diversity of parallel environments, their use is almost mandatory.

Glatard et al. [GLA08] affirms that the first level of parallelism that can be accomplished is the intrinsic workflows parallelism. Since asynchronous calls are possible, different processors can execute different parts of the workflow. This possibility is present in most SWfMS. They also present other 3 important possibilities for parallelism: data parallelism, services parallelism and data synchronization barriers.

Data parallelism is the possibility to process any fragments of data simultaneously with a minimal performance loss. This capability involves the processing of independent data on different computing resources. The services parallelism denotes that the processing of different data sets by different services is totally independent. Finally, data synchronization barriers are a limitation to services parallelism and occur because some synchronization processors wait for all input data to be processed before being executed.

2.1.4 Provenance in Scientific Workflows

Another specific topic in scientific workflows is the provenance. The automated tracking and storage of provenance information promises to be a major advantage of scientific workflow systems [DAV07].

Davidson et al. [DAV07] claims provenance is not only used for interpreting data and providing reproducible results, but also for troubleshooting and optimizing efficiency. Furthermore, the application of a scientific workflow specification to a particular data set may involve tweaking parameter settings for the modules, and running the workflow many times during this tuning process.

Davidson et al. [DAV07] also proposes that many scientific workflow systems (e.g., myGrid/Taverna) capture provenance information implicitly in an event log. For example, these logs record events related to the start and end of particular steps in the run and corresponding

data read and write events. Using the (logical) order of events, dependencies between data objects processed or created during the run can be inferred.

Chebotko et al. [CHE08] state that while exploring a workflow run provenance, a user may be interested in data products that have been produced or consumed by only certain task runs. Therefore, an abstraction mechanism is required in order to enable a user to focus on only relevant provenance information.

2.2 Bioinformatics

This section presents a literature review on important aspects related to the Bioinformatics area, in special related to Rational Drug Design (RDD), Molecular Docking and Molecular Dynamics. All these subjects are related directly with this thesis, since they are steps that are executed during the environment created and also serve to validate the pattern used.

One of the requirements of this thesis is to work with an area that has a large volume of data. Coutinho et al. [COU10] affirms that for many years Bioinformatics scientists have manipulated a large volume of data, and that their research is based on execution of experiments through computer simulation (*in-silico* experiments) that demands a high processing capacity from computers. Mattoso et al. [MAT08] say that *in-silico* experiments use several computational tools, and when used concatenated, present characteristics of a scientific workflow.

2.2.1 Rational Drug Design

The execution of scientific experiments to discover an inhibitor, a future drug candidate, has evolved with computing assistance. The possibility using computational resources in Biology enables further accuracy and faster findings, forming a multidisciplinary area called Bioinformatics. The reference to Rational Drug Design (RDD) comes from improvements in molecular biology and the use of tools in *in-silico* simulations. There are 4 steps in RDD [KUN92]:

1. A target receptor structure (protein, DNA, RNA and others) is analyzed through its own 3D structure to identify probable ligand sites;

2. From the probable binding sites identified in the receptor, a group of potential ligand candidates is selected, allowing that interactions ligand-receptor could be tested and evaluated from simulation through the use of molecular docking software;

3. In theory, the ligands with the best interaction results with receptor are selected, for a next step that is related to synthesizing and testing in biological assays;

4. Based on finding results, an inhibitor (drug candidate) is created. Otherwise, the first step starts again.

These four steps from RDD described in [KUN92] are depicted in a flowchart in [MAC11]. Figure 4 depicts this flowchart.

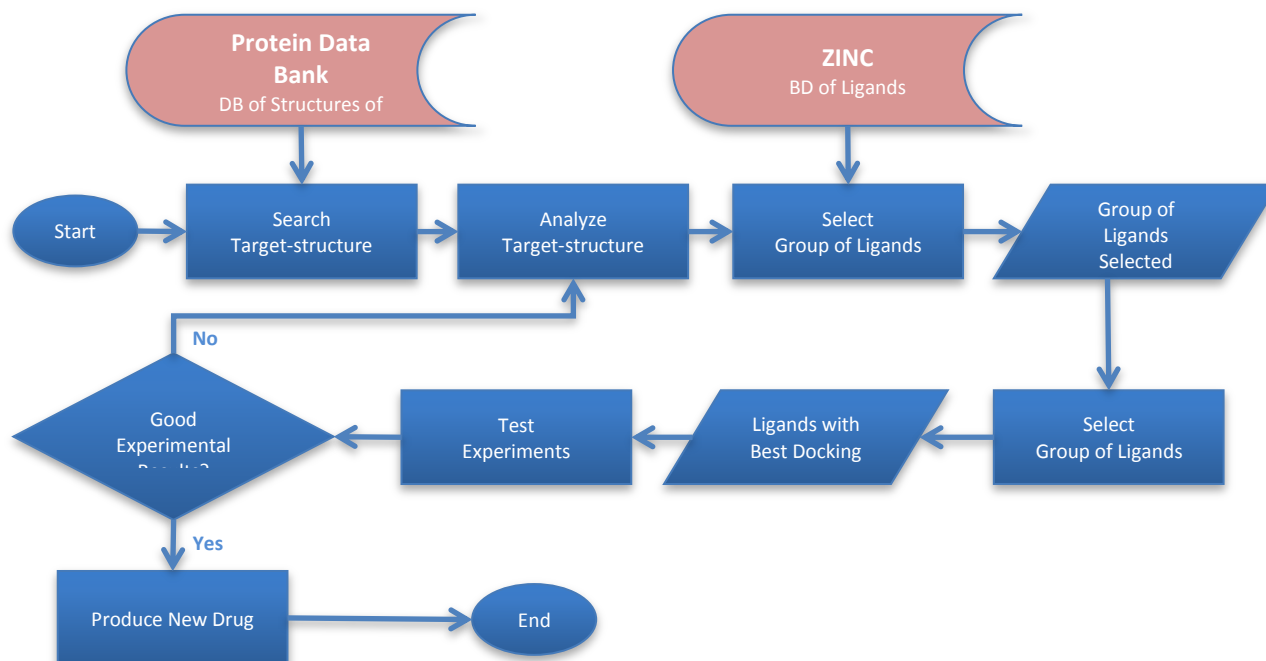


Figure 4 – Flowchart of Rational Drug Design. Adapted from [MAC11].

2.2.2 Molecular Docking

In silico molecular docking simulation is one of the main steps of RDD. During this step the main goal is to find the best possibility to bind molecules to create a stable complex [LEN96]. Through the discovery of better targeting of these molecules, one can predict the strength of association or binding affinity between them. It is during molecular docking that the best ligand fit into the receptor becomes available [4]. To assess the quality of the ligand fitness, a large number of evaluations are carried out to score and rank the best ligand conformation and orientation inside the receptor-binding pocket.

During this process it is important that docking considers the flexibility of the ligand and receptor. The ligand, usually a smaller molecule with few atoms, has ease flexibility. However, the flexibility of the receptor, due to its size and complexity, in many cases is still not considered, posing a challenge for the area of study. Figure 5 illustrates the molecular docking, including the flexibility of the receptor.

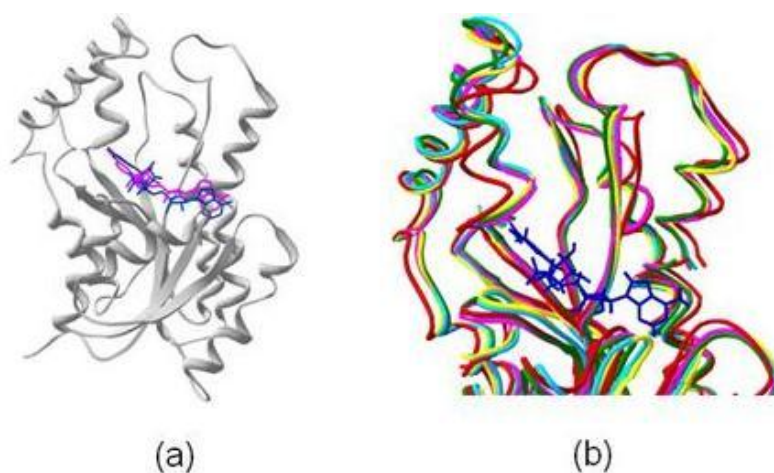


Figure 5 – (a) Schematic 3D representation of molecular docking. The protein is represented in the form of ribbons, in gray, and ligand in (magenta and cyan). (b) Flexibility of InhA-NADH system in different moments during the molecular dynamics simulation. Overlap of different conformations of InhA (cyan, yellow, magenta and green) generated for molecular in [SCH05]. Picture from [MAC07]

The molecular docking is executed with docking algorithms, which can generate a large number of ligand-receptor complexes, where they can be assessed by free energy of bindings (FEB). The more negative FEB, the better the interactions ligand-receptor. There are several docking programs, such as DOCK [EWI01], AUTODOCK [MOR10, GOO96], GOLD [VER03, JOY06], FLEXX [RAR96, KRA99], ZDOCK [CHE03], M-ZDOCK [PIE05], MS-DOCK [SAU08], Surflex [JAI03], MCDOCK [LIU99] and others. Each molecular docking program is based on a specific algorithm [DIA08] and can search for the best fit between two or more molecules, taking into account several parameters, obtained from receptor and ligand input coordinates such as: geometrical complementarity, regarding atomic Van der Waals radius and charge, receptor or ligand structure flexibility; or considering interatomic interactions, such as hydrogen bonds and hydrophobic contacts. As a result, docking applications return the predicted orientations (poses) of a ligand in the target's binding site. Usually the posing process returns several possible conformations. Scoring functions, which are able to evaluate intermolecular binding affinity or binding free energy, are employed in order to optimize and rank results, obtaining the best orientation after the docking procedure.

Machado [MAC11] says that the process of analyzing the interaction ligand-receptor is not easy. There are influences from many entropic and enthalpic factors, like the mobility from receptor and ligand, the effect of the environment on the receptor, the load balance of ligand, and other interactions from them with water that greatly complicates the description of this process.

2.2.3 Flexibility of Receptor

During the molecular docking the flexibility of ligand is an aspect that has been explored from many molecular docking tools. However, the flexibility of receptors is still a big challenge [TOT08]. Treating protein movements and active site flexibility is a double challenge, due to the high dimension of conformation space and the complexity of energy function in use.

Nevertheless, Trotoev et al. [TOT08] claim that the state-of-the-art of docking algorithms predict an incorrect binding pose for about 50-70% of all ligands when only a single fixed receptor conformation is considered. It is an interesting aspect to use the flexibility of receptor in molecular docking. Moreover, the flexibility of receptor has been one of the topics worked on by scientists [YUR10] in the last years.

2.2.4 Molecular Dynamics

Molecular Dynamics (MD) has been used to identify receptor flexibility. It was mentioned before that protein flexibility is a subject that is investigated by many scientists. MD is one of the computational techniques used for that.

In MD, atoms and molecules are allowed to interact over time at a given temperature following the laws of classical mechanics and which provides a detailed description of atomic motion [DOD08]. Cozzini et al. [COZ08] affirm MD simulations are an easier technique to execute and has a reasonable cost to generate conformations of a receptor, the snapshots. MD simulations, when combined with information from other techniques, open the path for the understanding of genetic, thermodynamic and functional behavior of biological processes [DOD08].

2.3 Last Chapter Considerations

This chapter presented the literature review on important concepts used in this thesis. The subjects presented here are used during the other chapters and are a base for the environment created to execute MDock simulations of FFR model.

First, the subject presented was scientific workflows. This topic has discussed the characteristics of scientific workflows and also differences in regards to business workflows. Also, the patterns were addressed. Patterns are the base of the study that is presented in the next chapter.

The other subjects were bioinformatics, RDD, molecular docking and DM. This study field was used in many steps of developing the environment that will be presented, such as validation

of the pattern, result analysis, setup, control and execution. Thus, we reproduce in this thesis a workflow for RDD.

3 PATTERN P-SAMI - SELF-ADAPTING MULTIPLE INSTANCES

This chapter presents concepts on P-SaMI (Pattern - Self-adapting Multiple Instances) [HUB10], a pattern able to manage a large volume of data, to optimize the process execution and to identify promising data groups. Bioinformatics is the reference area used to create the pattern, but it is not restricted to it, since the requirements to handle a large volume of data and the possibility to group them are present in other fields.

Besides presenting the pattern, we also present FReDoWS (Flexible Receptor Docking Workflow System) [MAC11a], a workflow system that motivates the creation of P-SaMI. Finally, we state some last considerations about the chapter.

3.1 FReDoWS

The P-SaMI was created based on FReDoWS (Flexible Receptor Docking Workflow System) [MAC11a]. FReDoWS is a scientific workflow developed in LABIO (Bioinformatics, Modeling, and Biosystems Simulation Laboratory) and is used to execute molecular docking simulations in a FFR model.

Figure 6 depicts activities by FReDoWS model. Each activity has a type that is represented by different colors. Dark green are executed from the user, and activities in purple are subflows with other activities inside. Light green activities are automated without user intervention and can invoke one or more external applications. Pink activities are just used to synchronize the model activities.

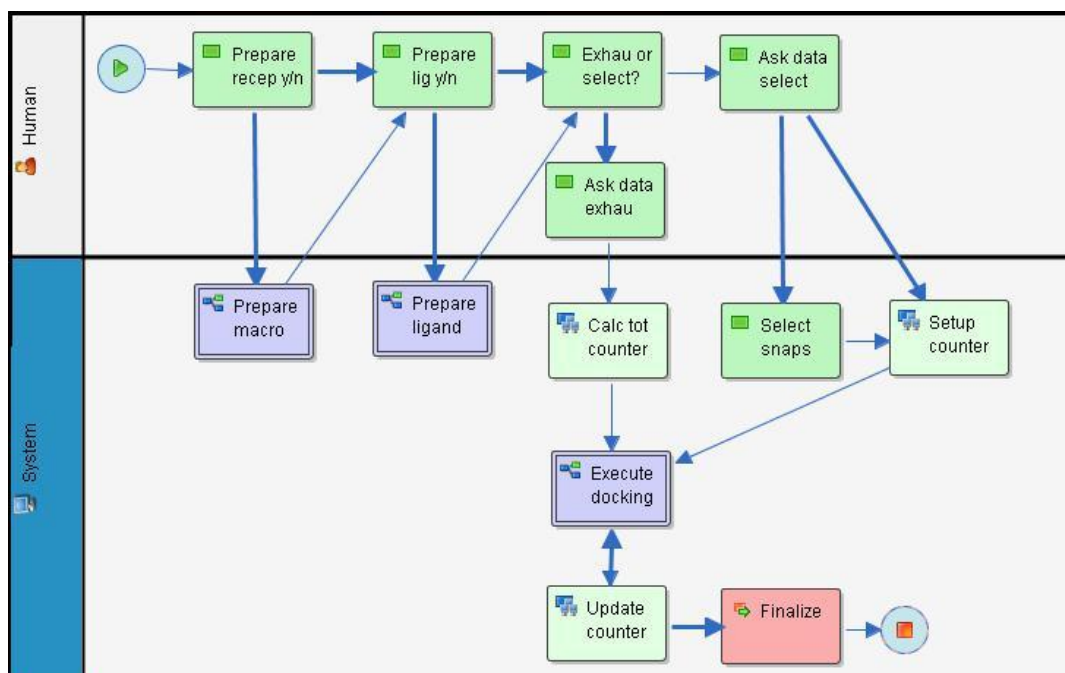


Figure 6 – Final Model of Scientific Workflow FReDoWS [MAC11a]

The main steps from FReDoWS execution are presented below [MAC07, MAC11]:

- Prepare macromolecule files: after the execution of MD, the files are prepared to be used in molecular docking. Ptraj, a module from AMBER, is used, creating PDB extension files based on DM;
- Prepare ligand: the ligand is placed in its first orientation for molecular docking simulations;
- Choose type of process: the user chooses between Exhaustive and Selective; In Exhaustive option all conformations from FFR model are used. In Selective, the user has the option to pick the conformations up based on RMSD value informed and a ranking of FEB results;
- Execute docking: notwithstanding the process type chosen, exhaustive or selective, Autodock3.0.5 executes the molecular docking based on ligand and receptor.

The Enhydra JaWE [ENH11] and Enhydra Shark [ENH11a] software tools were used to model and execute the workflow, respectively. Thus Machado et al. [MAC07] describe three study cases using 3.100 conformations from InhA receptor, generated from Schroeder et al. [SCH05], and the ligands NADH [DES95], PIF [OLI94] and TCL [KUO03]. The study case gave an average elapsed time of around 15 minutes per each molecular docking, with 775 hours as a total

sequence execution (exhaustive mode). These experiments for flexible receptor docking experiments were executed on Pentium III PCs of 1GHz and 256 MB RAM.

Some improvements have been presented to reduce the number of conformations [MAC11, MAC11a] added to FReDoWS. Through a Select module the scientist can filter the number of conformations. Machado [MAC11] says the selection is based on the following thought: “if a docking of a conformation resulted in value of FEB or RMSD, it is possible that this same conformation, when interacting with a similar ligand, also get good values of FEB and RMSD”. The steps to select the conformations are [MAC11]:

- User informs the total of conformations that he wants to pick up, a maximum RMSD value and the exhaustive table for FFR model;
- The table is sorted by FEB;
- This table is shared in two new tables based on maximum RMSD value informed;
- If the total value of conformations on table within maximum RMSD is the same as user conformations informed, the list of conformation to be used in selected docking is ready, otherwise, conformations are added with docking values bigger than the maximum RMSD informed.

3.2 Concept of the pattern

The pattern Self-adapting Multiple Instances (P-SaMI) came up from analyses related to FReDoWS. It was observed that the execution of the workflow takes a long time. Based on FReDOWS and in its long time execution, Hübler created P-SaMI, a pattern focused on scientific workflows, since it uses basically data-flow elements.

Bioinformatics is not the only area where the pattern can be used. Although based on it, the pattern can be used in any area. For that, Hübler [HUB10] mentions some features:

- Handling large volume of data;
- Data can be clustered from any similarity criteria;
- The result of each process data is a number;
- There is good value to be compared with process results.

Hübler [HUB10] quotes the pattern can manipulate large volume of data and optimize the processing time, identifying promising groups, as a component of a scientific workflow. Figure 7 depicts P-SaMI in all stages of its use:

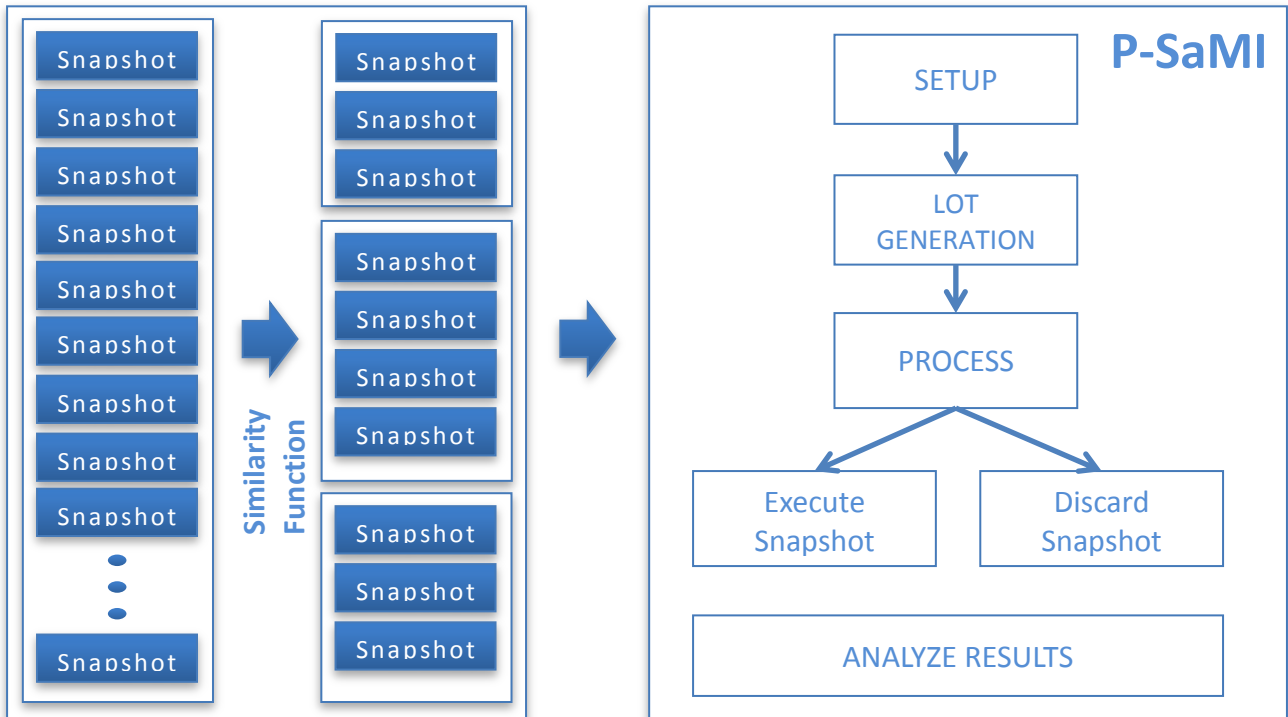


Figure 7 – P-SaMI: Self-adapting Multiple Instances pattern

There are some stages in which to use P-SaMI:

1. Create the data: this step occurs before the execution of P-SaMI; In this case we represent these data through snapshots that were created from an MD;
2. Apply Similarity Function: next step is to cluster the data. Based on similar characteristics, the snapshots are grouped;
3. Execute P-SaMI: with data clustered, the pattern can be used, controlling the clusters in order to manage the most promising ones.

3.2.1 Similarity Function

This step is not defined in P-SaMI but its understating is fundamental to obtain good results from the pattern. Before starting P-SaMI, the snapshots have already been created from MD. After that, it is time to execute a function to group the snapshots. Any form of grouping snapshots can be used. However, for this thesis, we are employing the one introduced by Machado [MAC11].

In order to validate P-SaMI [HUB10], Machado [MAC11] prepared the data to group all the snapshots. For that, a *TCN_Mult2* function is used with THT+NADH as entrance for *K-means* algorithm, *ALL* for a set of atoms and 6 as a number of groups. Also, the same parameters were used for *RMS* function. The ligand PIF was selected. Table 1 shows the groups and the quantity of snapshots in each group for *RMS* and *TCN_Mult2* similarity functions.

Table 1 – Quantity of conformations per each group, generated from *K-means* algorithm for *RMS* and *TCN_Mult2* similarity functions

Groups	Quantity of <i>RMS</i> conformations	Quantity of <i>TCN_Mult2</i> conformations
0	291	293
1	474	379
2	801	1
3	507	1.011
4	522	807
5	505	609

The execution of the similarity function is not part of this work, but it has great value to achieve the results of P-SaMI. The pattern design and tests were performed with the results presented in table 1 and are presented in chapter 5. It is important to emphasize that P-SaMI does not restrict the area, neither the way that the data is grouped. It means that P-SaMI can be used not just in RDD process, but also in other processes with similar characteristics.

One question proposed by Hübler in her thesis was regarding the quality of how the data are grouped. The question “The similarity function applied is directly related with the success of the pattern execution?” was one of the questions that Hübler was attempting to answer. Then, after analyzing the results through some test tables, the author concluded that similarity function has great importance in the use of this pattern.

3.2.2 P-SaMI Setup

The pattern does not define a setup task. However, for a better understanding, we created this task to facilitate the description about some rules that need to be defined before an execution of P-SaMI. Before explaining it, some important concepts need to be defined:

- *Snapshot*: identify the snapshot number;
- *Group*: identify the group of snapshots generated from a similarity function;
- *Lot*: created from P-SaMI, identify the lot of snapshots contained in a group;

- *Status*: Identify the snapshot status during the flow; the possible values are: A (Active and waiting for execution), F (Finalized/Executed), D (Discarded based on bad result from the Group) and P (Priority changed).

A state transition diagram in figure 8 depicts all the possibilities defined for P-SaMI during the execution of the workflow.

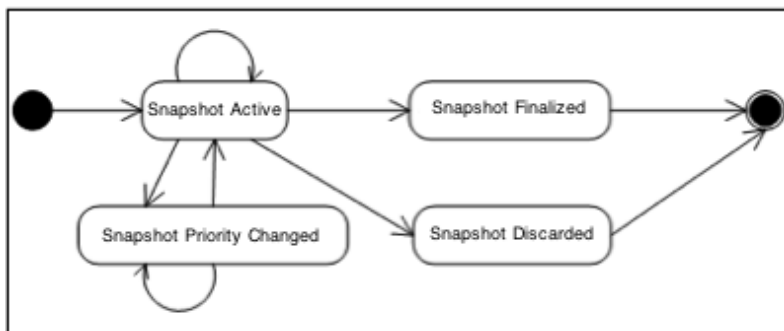


Figure 8 – State transition diagram for possible status in snapshot processing [HUB10]

After some names of P-SaMI are defined, it is possible to start the setup. Some requirements have to be defined before the workflow execution. There are two parameters:

- *Minimum amount*: this represents the minimum amount to create a lot with snapshots. If not defined, 50 is the default number. The default value was based on tests.
- *Sampling*: a percentage to create a lot with snapshots. If not defined, 30% is the default number. The default value was based on tests.

Besides that, other parameters have to be defined: *best value* and *worst value*. During the execution of P-SaMI, these criteria correspond to a range to analyze the results of each execution. During the execution of molecular docking, one result that can be used is FEB (Free Energy of Binding). In this case, lower FEB is a better result. The priorities and status of execution are based on results and these two parameters.

To set up the values for *best value* and *worst value* in P-SaMI, there is a way other than the user definition. Both can be set up automatically for the workflow through a random test with some snapshots, obtaining the best result and the worst result. Thus, there are two possibilities to set these values.

3.2.3 P-SaMI: Lot Generation

The Lot Generation is a step that runs in the beginning, before starting the execution of the workflow. A lot is a subgroup of group with snapshots. Thus, when the P-SaMI parameters are defined, the lot generation is ready to start.

Hübler [HUB10] affirms that *lots* are an important characteristic of P-SaMI, since their use occurs in preliminary and intermediate analysis. The number of lots per group can vary in quantity and is defined in execution time, based on parameters *minimum_amount* and *sampling*. Hübler [HUB10] justifies the use of lots based on some tests that proved better results could be obtained from sets with fewer snapshots. Moreover, the use of both parameters gives more flexibility to the pattern.

Figure 9 represents three test tables for lot generation. All of them have 100 snapshots and have different parameters defined for *minimum_amount* and *sampling*.

test	execution	sampling	minimum_amount	total_snapshots_group	lot
1	1	40%	30	100	40
1	2	40%	30	60	30
1	3	40%	30	30	30

test	execution	sampling	minimum_amount	total_snapshots_group	lot
2	1	70%	30	100	70
2	2	70%	30	30	30

test	execution	sampling	minimum_amount	total_snapshots_group	lot
3	1	40%	40	100	40
3	2	40%	40	60	60

Figure 9 – Representation of lot generation - Adapted from [HUB10]

In figure 9, *test* represents the number of test tables; *execution* represents each lot created; and *sampling* is a percent value used to create lots; *minimum_amount* is the minimum of quantity to create lots; *total_snapshots_group* is total of snapshots from group and; *lot* is the quantity of snapshots in a lot.

In the first test, after three executions, three lots are created. For the first execution the criteria used is *sampling* (40%) and 40 snapshots are grouped in the first lot. The second execution used *minimum_amount* (30), because 40% of 60 snapshots are 24, less than 30. The third execution uses the same criteria and creates a lot with last 30 snapshots.

In the second test the value for *minimum amount* is 30 and *sampling* is 70%. In the first execution, 70% of 100 snapshots are 70, greater than 30. In the second execution, 70% of 30 snapshots are 21, but the minimum is 30, then this lot will contain 30 snapshots.

At last, the third test has *minimum amount* with 40% and *sampling* with 40. The first execution used 40%. The second execution should have 40 snapshots, but then the next execution will have 20 snapshots, which is not allowed because there must be the minimum of 40. Then the second execution has 60 (40+20).

3.2.4 P-SaMI: Processing

After distributing the snapshots in lots, the pattern is ready to execute the experiments. Each execution can be treated as an instance of the process, and each snapshot can be submitted to the scientific workflow. Figure 10 depicts the execution of snapshots.

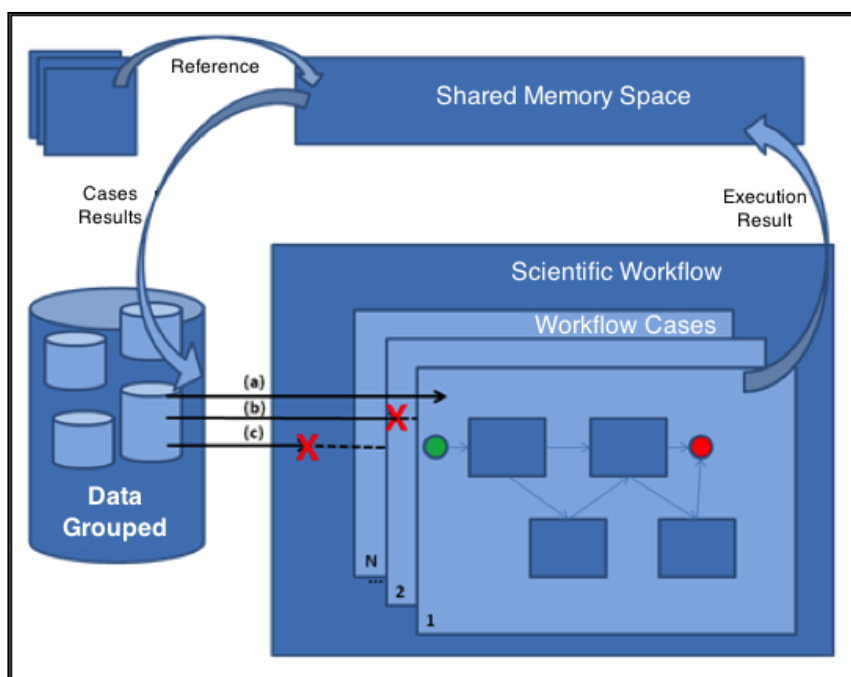


Figure 10 – Execution of snapshots from a group - Adapted from [HUB10]

Hübler [HUB10] explains figure 10 as follows: (a), (b) and (c) are snapshots from a same group submitted to a scientific workflow. The result value is “Execution Result“, which is stored in a “Shared Memory Space“, a place that is not in the workflow, e. g., a database. Then this value is analyzed based on *best_value* and *worst_value*, parameters of P-SaMI. Thus, the value result closed by *worst_value* has better chances for docking.

The execution and analysis of the snapshots can happen in sequence or parallel. Although the test tables from Hübler [HUB10] show a sequence test, the author also says that a parallel approach can be used. Thus, in a parallel process more than one group can be executed at the

same time and the results have to be stored for analysis, a step that gives a great flexibility and gain of processing to the pattern.

3.2.5 P-SaMI: Analyze

The analysis is a stage that occurs after the execution of snapshots. The pattern aims to execute the maximum of snapshots with results close to the parameter *best_value*. As the snapshots are being processed, P-SaMI starts some analyses to identify the most promising groups and lots. This identification can occur in different moments of the execution, but always taking into account the percentage executed for group per lot.

Figure 11 depicts the results and analysis that occur in lots and groups of P-SaMI. Hübler [HUB10] explains it as follows:

- R1', R1'', R1''', R2', R2'', R3', R4' and R4'' correspond to final result from each snapshot;
- R11, R21, R31, R41 correspond to the average result of all snapshots of a lot of a group.

The letters "A" and "B" correspond to a possibility of analysis in two ways:

- *Horizontal (B)*: Analysis of each result of snapshots from each lot and the possibility of continuing executing other lots from the same group comparing with the average of results of the same lot;
- *Vertical (A)*: Analysis comparing the average results from a lot of a group with lots from other groups.

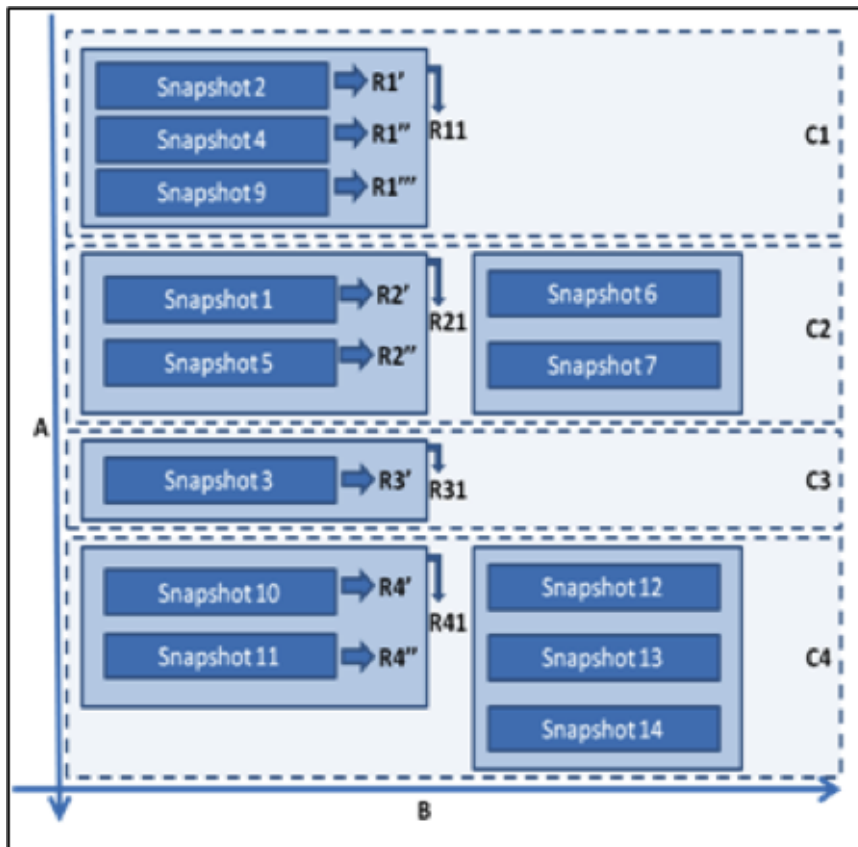


Figure 11 – Separation of lots from groups and representation of individual results from each snapshot and from all lot [HUB10]

The analysis uses average of execution to change the status or priority of the execution of lots and groups. Those averages are calculated in real time, as soon as each docking is executed. Three different averages are used:

- *average_setup*: arithmetic average from *best_value* and *worst_value* set up before the execution;
- *average_execution*: arithmetic average from all snapshots executed in a lot;
- *average_estimated*: this is an average estimated from a lot. The base are the snapshots that are not yet executed. The standard deviation from that one executed is used to calculate this average.

The rule for *average estimated* was adapted from a rule of thumb [LAR09]. Hübler [HUB10] quotes the reason for this adaptation in overestimating the probability of good results. Equation 1 represents the rule of thumb used for Hübler:

$$\sum xi = \frac{\frac{\sum_{i=1}^n xi + (((68\%f * (m - (s * 2)) + (68\%f * (m + s))) + ((95\%f - 68\%f) * (m - (s * 2))))}{2} + \frac{(((95\%f - 68\%f) * (m - (s * 3))) + ((95\%f - 68\%f) * (m + (s * 2))))}{2} + \frac{(((99,7\%f - 95\%f) * (m - (s * 4))) + ((99,7\%f - 95\%f) * (m - (s * 3))))}{2}}{t}$$

For the understanding of equation 1:

- f = quantity of snapshots not yet processed;
- m = arithmetic average from snapshots processed;
- s = sampling standard deviation;
- t = total quantity of snapshots;
- n = quantity of snapshots processed;

During the execution of snapshots the analysis can increase or decrease a priority of a lot or change the status. Figure 12 presents the algorithm to change the priorities and status of snapshots. The methods *increase_priority_group* and *decrease_priority_group* increase and decrease, respectively, the priority of execution of group. Method *discard_group* changes the status of the snapshots not yet processed to “D”, discarded.

```

1  if (( average_executed and average_estimated ) <= average_setup ) then
2      |increase_priority_group()
3  else if (( average_executed and average_estimated ) > average_setup ) then
4      |discard_group()
5  else
6      |decrease_priority_group()

```

Figure 12 – Algorithm to define priorities and status change

Hübler [HUB10] says that the analysis can be done in lots or groups. The difference occurs in process or no more lots from the same group. If the analysis is taken by lot, then:

- the other snapshots from the lot will be discarded: go to the next lot and the snapshots from this lot will not be processed;
- the other snapshots from the lot will have the priority decreased: go to the next lot but, before that, the snapshots not processed have to be processed.

On the other hand, if the analysis is taken by group, then:

- the other lots from the group will be discarded: go to the next group and the snapshots from all lots of this group will not be processed;
- the other lots from the group will have the priority decreased: go to the next group, but the snapshots from this group need to be processed later.

One step that occurs when a snapshot is processed is to set it as “F”. Thus, this snapshot can be considered in the average calculations and will never be executed again.

3.2.6 Formalism

Hübler [HUB11] says P-SaMI, formally, is a tuple $P_SaMI = \{C, L, s, P.S, MP.S, PP.S\}$, where:

- $C = \{C_1, C_2, \dots, C_m\}$ finite set of groups of snapshots;
- $L = \{L_1, L_2, \dots, L_m\}$ finite set of lots created from an individual group of snapshots;
- s is the snapshot contained in a lot from a group
 - $s \subseteq L \mid L \subseteq C$
- $P.S = \{P.S_1, P.S_2, \dots, P.S_m\}$ finite set with results of each snapshot processing;
- $MP.S$ is the best value from a sample of snapshots processed:
 - $MP.S \subseteq P.S$
- $PP.S$ is the worst value from a sample of snapshots processed:
 - $PP.S \subseteq P.S$
- $MP.S \neq PP.S$

3.3 Last Chapter Considerations

This chapter presented P-SaMI, a new pattern to execute experiments in scientific workflows. This pattern aims to reduce the execution time by discarding non-promising snapshots. The rules of the pattern were addressed here.

The pattern was validated through table tests [HUB10] where the efficiency could be proved. Unfortunately, the use of manual tests did not show all possible benefits from the pattern, such as parallel executions, and the communication with other environments, like communication with an MTC. Thus, building an environment where the rules of P-SaMI can be validated is one of the challenges of this thesis.

4 W-FREDOW (WEB FLEXIBLE RECEPTOR DOCKING WOKFLOW)

In this chapter we introduce W-FReDoW (Web Flexible Receptor Docking Workflow), an environment built to execute MDock simulations of FFR model through the web integrated with MTC environment, and using P-SaMI to achieve an FFR reduced model. The environment consists of several components, such as a database, web server, MTC middleware, P-SaMI, web front-end and Autodock4.2. All these aspects are presented in this chapter.

4.1 The Conceptual Architecture

This section describes the conceptual architecture of the environment designed for this thesis. We call it W-FReDoW, an acronym to Web Flexible Receptor Docking Workflow. W-FReDoW is an environment designed to execute MDock simulations of FFR model in order to achieve a Reduced FFR (RFFR) model. Thus, in order to achieve a RFFR model, W-FReDoW was guided by P-SaMI.

W-FReDoW architecture is based on the interactions between three layers: Client, Server Controller and MTC. It also has the support of a database. The interactions involve all layers, but always with the Server Controller in the middle. Client layer is a web interface for setup, execution and analysis of the workflow. Server Controller has a Web Server that controls the pages and also has P-SaMI algorithms that manipulate the execution of docking by MTC. MTC layer deals with distribution and parallelization of dockings. FReDD for P-SaMI is the database that keeps data to provenance and support P-SaMI algorithms. Figure 13 illustrates the layers of the conceptual model.

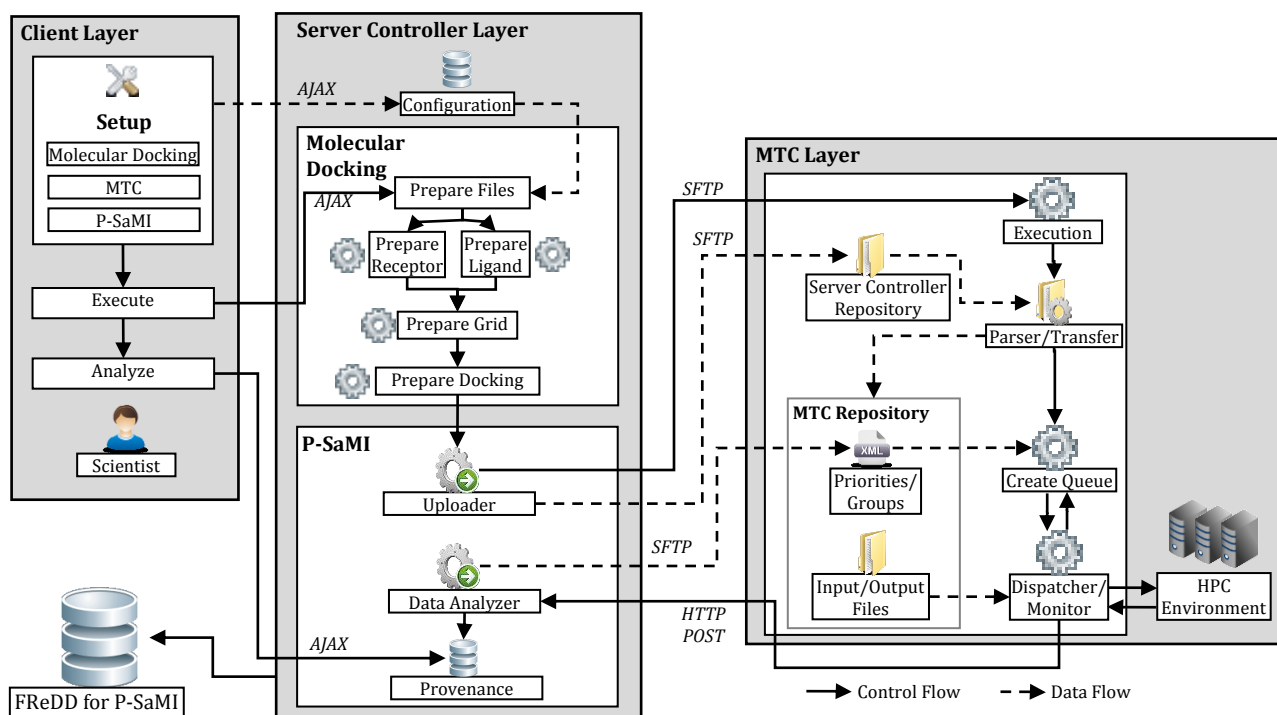


Figure 13 – W-FReDoW Conceptual Architecture

The interactions between layers take place in different ways. Client layer interacts with Server Controller through AJAX requests. The interaction between Server Controller and MTC occurs in two ways: SFTP and HTTP POST. The SFTP protocol is used from Server Controller and sends the files to MTC, while HTTP POST is the way MTC sends the results from each docking to Server Controller. Throughout all the workflow, Server Controller records data on database.

4.2 Client Layer

The Client Layer is a web interface in which the domain specialist has the possibility to configure, start and get information on the MDock simulations. During these steps, every communication is done with the Server Controller Layer through Ajax. This layer has 3 main groups: Setup, Execute and Analyze.

The *Setup* group is divided in three steps that configure the environment. The *Molecular Docking* is responsible for the configuration of the parameters that Autodock4.2 will use during the execution process, as defined in [MOR09]. *MTC* sets the information related with the credentials used in the remote server. *P-SaMI* is responsible for the configuration of the pattern, a paramount process to achieve the RFFR model.

Once the Setup has been done, it is time to *Execute*. The *Execute* component starts the process. This component communicates with the Server Controller Layer letting it know about the configuration chosen for the whole process.

The last component in this layer is *Analyze*. Here the scientist can get information about the execution of the process, accessing the provenance task in the Server Controller Layer that provides, in real time, data about the status of executions.

4.3 Server Controller Layer

This layer is the core of the environment realizing the communication with the other two layers. The components are responsible for many parts of the workflow to MDock simulations, executing important tasks during the whole process. There are three groups that are part of this layer: *Configuration*, *Molecular Docking* and *P-SaMI*.

Configuration is a unique component that is responsible for storing data from the Setup group in the Client Layer. Basically, it works as a middle area, employed to save the data that other steps will make use of during their activities.

Molecular Docking is a group responsible for the preparation of the MDock simulation. The first component, Prepare Files, is responsible for organizing the snapshots in groups [MAC11], through similar characteristics of snapshots. After this task is finished, two parallel components are executed: Prepare Receptor and Prepare Ligand. Both components are part of Autodock4.2, just like the next two components. Prepare Grid and Prepare Docking are two important tasks for MDock simulation, because the results are parameter files that will be used during the MTC Layer. First, Prepare Grid prepares the grid parameter files (extension GPF), and then Prepare Docking creates the docking parameter files (extension DPF). Both steps are essential in the process of MDock simulations [MOR10].

Once the Prepare Docking execution is completed, it is time to start the group *P-SaMI* that is related with the standard to achieve RFFR model. The first task, Uploader, generates lots for each group as mentioned in [HUB10]. After that, an XML file is stored in Web Server Repository of MTC Layer, with the structure used for an initial setup of the MTC. Figure 14 shows the structure of the XML, where there are groups of snapshots generated for the component Prepare Files. Besides, there are two other elements in the XML file that correspond to the status and priority of the group and snapshots. The last task from Uploader is to start the execution of MDock simulations in the MTC Layer by calling the component Execution.

```

1 | <experiment id=1>
2 |   <group>
3 |     <idGroup>1</idGroup>
4 |     <stat>A</stat>
5 |     <priority>2</priority>
6 |     <snapshot>
7 |       <idSnap>000001</idSnap>
8 |       <status>A</status>
9 |     </snapshot>
10 |    <snapshot>
11 |      <idSnap>000045</idSnap>
12 |      <status>A</status>
13 |    </snapshot>
14 |  </group>
15 |  ...
16 | </experiment>

```

Figure 14 – XML with group of snapshots

4.4 MTC Layer

The MTC Layer handles a large amount of data and controls the distribution of tasks for parallel execution in HPC environment. The scope of this layer is part of Renata de Paris' thesis, my partner in achieving the results of W-FReDoW. The MTC uses two main functions, one heuristic function for creating queue of tasks, and another function that shares out the parallel execution of the tasks on a cluster. The layer has eight components: Execution, Parser/Transfer, Create Queue, Dispatcher/Monitor, Server Controller Repository, Priorities/Groups, Input/Output Files, and HPC Environment. The components Execution, Parser/Transfer, Create Queue and Dispatcher/Monitor have been developed as a library of the programming language C, and the others just store the files on directories.

Server Controller Repository component stores all files uploaded by Web Server Layer. Web Server Layer prepares these files. They are input files to execute on the Autogrid4 and Autodock4. The XML file is on Priorities/Groups component and it is updated and uploaded by the Data Analyzer component during the execution time. This file contains information on the groups of the snapshots, such as priority and classification, and it identifies the group that belongs to each snapshot. Therefore, the Create Queue component uses an XML file to create the balanced queue of tasks. The Input/Output Files component has files, which are used by MTC Layer functions. Firstly, the Web Server Layer uploads these files on the Web Server Repository component. Then, they are managed by the Parser/Transfer component. Finally they are sent to the workspace of the MTC Layer. Priorities/Groups and Input/Output files components are within MTC Repository because both have files, which are organized on MTC workspace (see Figure 15).

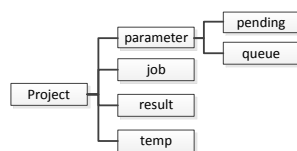


Figure 15 – Directories structure of files of the MTC Layer workspace

The Execute Component creates the workspace for execution of the MTC Layer and organizes the files used during the execution. It performs two different functions: (i) start the execution of MTC Layer and; (ii) create the directory structure of the project. The directory structure (see Figure 15) of the workspace has four main directories to store files. Directories job and parameter store input files for execution of the autogrid4 and autodock4, and the directories result and temp store output files from parallel execution of the HPC environment. The directory job stores the snapshot files from FFR model (pdbqt format). The directory pending stores snapshot files, which are waiting for insertion in the queue of tasks. The directory queue stores snapshot files that have already been processed by HPC environment. The directory parameter stores the ligand (pdbqt format), and the input files for the execution of autogrid4 (gpf format) and autodock4 (dpf format). The directory result stores output files of the autodock4 (dlg format). At last, the directory temp stores temporary files, which are used just in execution time by HPC environment and, after the execution, they are deleted.

The Parser/Transfer is the component responsible for handling and organizing the files received by Web Service Layer for the workspace of the MTC Layer. Then, the files prepared in Web Service Layer are sent to the Data Repository Component, and the Parser/Transfer component transfers these files to the directories structure of files (see Figure 15).

The Create Queue component is responsible for applying the heuristic function. It employs the priorities of the different groups of snapshots to create balanced task queues. This component performs three main functions: (i) read XML files; (ii) insert snapshots pending in the task queue and; (iii) create the queue of tasks to be executed in the HPC environment.

For the HPC environment a task is the execution of the Autogrid4 and Autodock4 in a single snapshot that makes up an FFR model. Then, a queue of tasks is made of a vector of snapshots created in the Create Queue component, and this vector is sent to the Dispatcher/Monitor component for handing out the tasks on nodes and cores of the HPC environment. The amount of snapshots inserted in queue depends on the size of the queue that is calculated based on nodes and cores allocated on HPC. The queue of tasks determines the amount

of snapshots, which are inserted in queue. For that, the heuristic function counts the number of nodes and cores allocated in HPC environment. Thus, for a homogeneous cluster, the quantity of the queue is calculated as follows:

$$Q_{queue} = N_{cluster} \times C_{internal} \quad (1)$$

A balanced task queue keeps one or more snapshots from each activated group. In order to do that, we use P-SaMI to set the amount of snapshots that each active group inserts in queue. Then, with the priorities of the P-SaMI, it is possible to determine the percentage of snapshots that each group inserts in a queue of tasks. Thus, if a group holds high priority, then it has a greater amount of snapshots in the queue; on the other hand, if a group holds low priority, then it has fewer snapshots in the queue. The amount that each group of snapshots inserts in a queue of tasks is calculated according to the equation:

$$S_{group} = Q_{queue} \times \left(\frac{P_{group}}{\sum_{i=1}^n P_{group}} \right) \quad (2)$$

S_{group} is the quantity of snapshots of each group, Q_{queue} is the result of the equation (1), P_{group} is the priority of the group, and the sum of the priorities of every group on the denominator of the equation. Hence, from the sum of the outcome of equation (2) the queue of tasks is created with the following function:

$$Q_{tasks} = \sum_{i=1}^n S_{group} \quad (3)$$

Due to intrinsic characteristics of the P-SaMI, which perform changes constantly on status and priority attributes of the different groups of snapshots during the execution time, it is necessary to create task queues balanced and updated according to the attributes of each group of snapshots. Then, the analysis of these attributes is performed to result in files processed by the HPC environment in the Web Server Layer by P-SaMI. Furthermore, Server Controller Layer updates the result analysis, as priorities and status, in XML file, and the MTC Layer reads it at the moment of to creating a new queue of tasks. A rule determines that the maximum size of each queue of tasks is equal to the maximum number of tasks which the cluster can execute in parallel. A new queue is created whenever a node is idle and there are no tasks in the queue. Figure 16 shows the operation flow between a Create Queue and Dispatcher/Monitor component. The first uses heuristic function and calls *get_property()* and *get_snapshots()* functions to create the balanced queues of tasks; while the other component controls the tasks to execute in parallel by HPC environment.

The Dispatcher/Monitor component is responsible for invoking an external scheduling to execute the queue of tasks in HPC environment. The hierarchical hardware design of the HPC environment has shared memory nodes with several multi-core CPUs, which are connected via a network infrastructure. For this reason, we employ a hybrid-programming model, which uses OpenMP for parallelization inside the node and MPI for message transmission between nodes. Furthermore, the use of master-slave paradigm is applied to setup the tasks used for each cluster node. Figure 16 shows the operation flow of the Dispatcher/Monitor component. The code involves a main function, called the function master on 1 node and the slave functions on all other nodes selected by the user. The master node performs three main functions: (1) manage the task distribution among the slave nodes; (2) send request for creating new queues of tasks in the Create Queue component when the queue is empty and; (3) execute tasks of the queue of tasks. The slaves just execute the tasks of the queue of tasks.

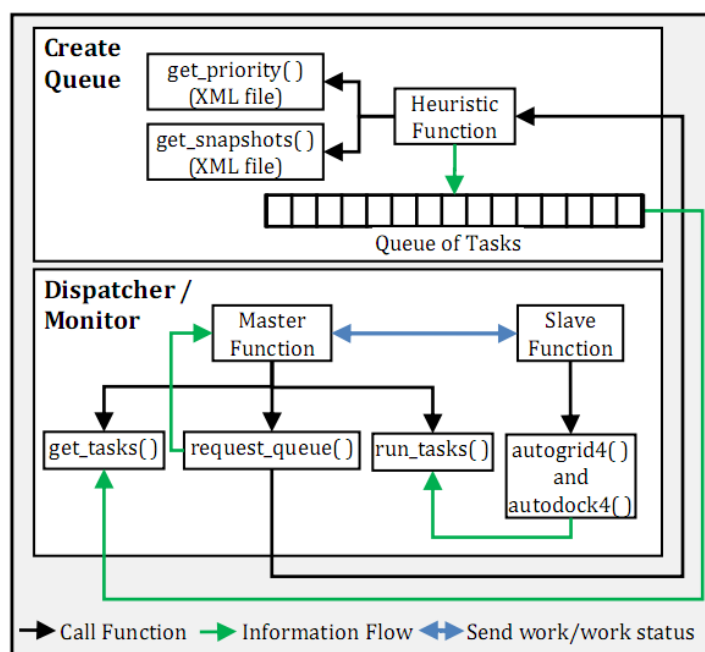


Figure 16 – Operations flows and functions have been performed between Create Queue and Dispatcher/Monitor components

4.5 W-FReDoW: Details of implementation

There are many important aspects that are under the construction of the environment and have been highlighted during this chapter. The database, the web server, the program language and the communication with MTC are important features in the environment and must to be explored.

4.5.1 Database: FReDD for P-SaMI

We defined the architecture in three layers as shown in figure 13. An important component is the database used. This is called FReDD for P-SaMI, an extension of FReDD (Flexible Receptor Docking Database) [WIN10, MAC11] following the rules in P-SaMI pattern. The focus of the original FReDD is to keep the results of MDock simulations of FFR model to be used for data mining. Therefore, with this thesis, we extend the FReDD data model, creating a new structure that stores all steps used during the execution of the experiment. Also, with the use of P-SaMI, a database to store the data is a requirement since the calculation in real time needs to be done to different methods and keeping the results of experiments in memory is not a good idea.

FReDD for P-SaMI is a database responsible for storing all steps during the process of running MDock simulations in an FFR model to achieve an RFFR model, and capturing the provenance of the process. The communication with the database happens always with the *Server Controller Layer* (see Figure 13). During the execution some tables store the results individually, for example for each snapshot, as well as grouped, for example for cluster of snapshots, to be used for P-SaMI methods.

Figure 17 shows the data model for FReDD for P-SaMI (design with *Microsoft Visio*). The tables are separated in four main areas: Setup, Execution, Template and Provenance tables.

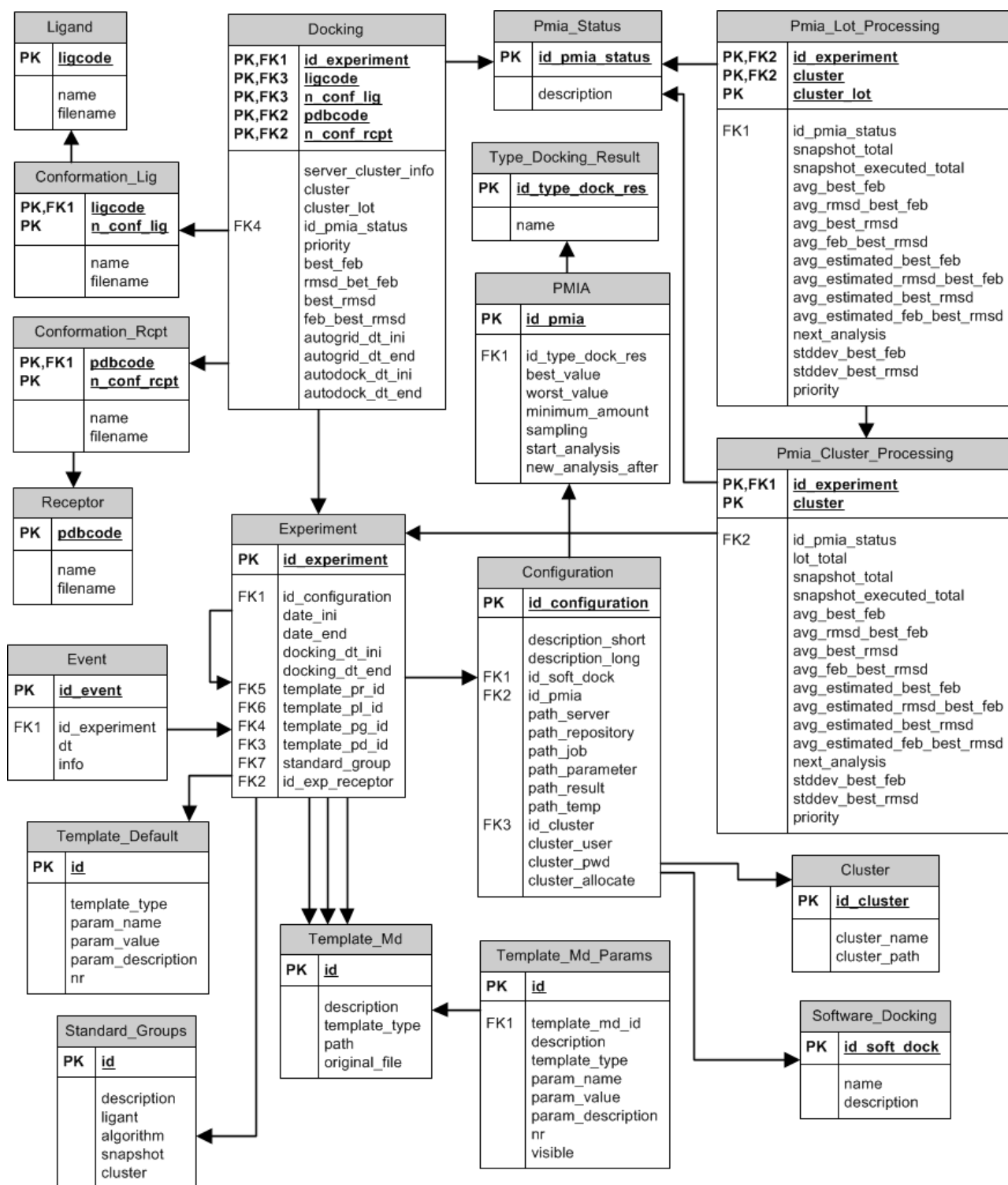


Figure 17 – FReDD for P-SaMI: Data Model for W-FReDoW

The FReDD for P-SaMI was created in PostgreSQL. PostgreSQL is an object-relational database management system (ORDBMS) based on POSTGRES, version 4.2, which was developed at the University of California at Berkeley Computer Science Department [POS11]. PostgreSQL supports a large SQL (Structured Query Language) and features complex queries, foreign keys, triggers, transactional integrity, etc.

There are two reasons for FReDD for P-SaMI to use PostgreSQL. The first is because its license can be distributed for any purpose, be it private, commercial or academic. The second reason is to keep the same database used in FReDD, and to store the data in a unique database.

This database presents 19 tables. The data inserted comes from UI Interface or is manually uploaded, in case of static data. The next subchapters explain about tables, attributes and the goals related with W-FReDoW.

4.5.1.1 Setup Tables

These tables have the characteristics of storing data that hardly changes. The data stored are fundamental for the execution of the environment, i. e., P-SaMI parameters and cluster path. Part of this group of tables is: *Software_Docking*, *Cluster*, *Type_Docking_Result*, *PMIA*, *Configuration* and *Pmia_Status*.

The W-FReDoW was thought of with the idea of running different kinds of experiments, since it contains the requirements of P-SaMI. For MDock simulations it would be possible to use different software for docking, like Autodock 3.0.5, Autodock 4.2, Vina, etc. Thus, *Software_Docking* table stores the type of MDock simulations that can be chosen during the configuration of experiment execution.

The execution of MDock simulations occurs in another environment and, in order to setup the parameters for the communication with MTC, the *Cluster* table stores the communication data necessary for such. Therefore, this parameter can be set up before starting the workflow.

As presented in chapter 3, P-SaMI needs some configuration parameters to start its execution and result analysis. The *Type_Docking_Result* table stores the type of result that will be analyzed, i. e., “FEB” and “RMSD”. W-FReDoW does not restrict only for this two result types, but for any type; even a formula can be considered. Following the characteristics of P-SaMI, the *PMIA_STATUS* table stores the different status of each snapshot (“A” for Active; “F” for Finalized, “D” for Discarded and “P” for Priority change). Another table related with P-SaMI is *PMIA*. *PMIA* stores P-SaMI rules, essential for execution of W-FReDoW, and has the following fields:

- *id_pmia* and *id_type_dock_res*: *id_pmia* is the primary key and *id_type_docking_res* is the foreign key to *Type_Docking_Result* table;
- *best_value* and *worst_value*: refer to the range for the pattern during the execution. Both attributes are criteria used to analyze the group results;

- *minimum_amount* and *sampling*: the lots are created based on *minimum_amount*, which is related with the minimum value for lot, and *sampling* correspond to this minimum value, but in percent value;
- *start_analysis* and *new_analysis_after*: during the execution of the process, the analysis occurs during some moments of execution. *start_analysis* field is this first moment and *new_analysis_after* is the others moments, in percent value, in which the analysis has to occur. In order to illustrate this, we can start the analysis with 30% (*start_analysis*) and after every additional 10% (*new_analysis_after*), meaning 40%, 50%, 60% until 90%, and the analysis of P-SaMI is done.

The last setup table is *Configuration*. This table contains data on the other setup tables and is used to start an execution of W-FReDoW. There are data related to cluster, P-SaMI and path that will be used during the execution of experiments. The following fields can be found:

- *id_configuration*, *description_short* and *description_long*: fields used for identification of configuration;
- *id_soft_dock* and *id_pmia*: fields that correspond to foreign keys for type of docking and P-SaMI setup used;
- *path_server*, *path_repository*, *path_job*, *path_parameter*, *path_result* and *path_temp*: these fields are parameters used to create a local structure to store data during the execution;
- *id_cluster*, *cluster_user*, *cluster_pwd* and *cluster_allocate*: data about the parameters used to access the cluster to execute the experiments;

4.5.1.2 Execution Tables

The W-FReDoW has some tables where records are created when a new execution starts. The first table to be created is *Experiment*, which stores data related to a new execution. After that, the snapshots that will be used in the execution are stored in the *Docking* table. This table stores the relationship between ligand and receptor that are based on four tables: *Ligand*, *Conformation_Lig*, *Receptor* and *Conformation_Rcpt*. During the execution, when a snapshot changes its status, this table is the first to be updated. Also, the *Pmia_Cluster_Processing* and *Pmia_Lot_Processing* are tables that are updated whenever a snapshot is docked and have the goal of helping the analyses of P-SaMI, since they keep the total average during the process.

As mentioned before, the first table that has data inserted in is *Experiment*. Each row in the *Experiment* table corresponds to a unique execution of W-FReDoW, storing all parameters for that. They are:

- *id_experiment*: created when the execution of the process starts;
- *id_configuration*: foreign key for the configuration used in experiment execution;
- *date_ini* and *date_end*: start and end date/time of execution;
- *docking_dt_ini* and *docking_dt_end*: start and end date/time of cluster execution;
- *template_pr_id*, *template_pl_id*, *template_pg_id* and *template_pd_id*: foreign keys for receptor, ligand, grid and docking templates;
- *standard_group*: foreign key to similarity function used in the process;
- *id_exp_receptor*: foreign key to another experiment. It indicates that the experiment created was based in another existent experiment.

Some tables store data on ligand and receptor: *Ligand* and *Receptor* tables, respectively. Also, other two tables store the flexibility of each one that is *Conformation_Lig* and *Conformation_Rcpt*. Today we are testing with 3.100 receptors from InhA protein, but there is no limitation for the number of conformations.

The *Docking* table stores the relationship between ligand and receptor for MDock simulations. When the workflow is running, some attributes are updated; i. e., status and priorities of snapshots can change. This table has the following attributes:

- *id_experiment*: the rows in this table are linked to one experiment;
- *ligcode* and *n_conf_lig*: ligand code and number of its conformation;
- *pdrcode* and *n_conf_rcpt*: receptor name and conformation number;
- *server_cluster_info*: field used to keep information about the specific place where the snapshot was docked. This information is provided from MTC (FReMI) and can store, for example, virtual machine, cluster or node used;
- *cluster* and *cluster_lot*: *cluster* and *lot* refer to one group and lot for each snapshot. The cluster is generated for similarity function [MAC11] and lot for P-SaMI [HUB10], based on *minimum_amount* and *sampling* criteria;

- *id_pmia_status* and *priority*: status and priority of each snapshot during the process. Status is related with the *PMIA_STATUS* table and priority is a number between 1 and 3, which mean low, medium and high priority;
- *best_feb*, *rmsd_best_feb*, *best_rmsd* and *feb_best_rmsd*: these fields are captured in the results files of autodock execution and correspond to best run;
- *autogrid_dt_ini* and *autogrid_dt_end*: start and end date/time of autogrid execution;
- *autodock_dt_ini* and *autodock_dt_end*: start and end date/time of autodock execution;
- *snapshot*: reference to the number of the snapshot;

The next two tables, *Pmia_Cluster_Processing* and *Pmia_Lot_Processing* have almost the same attributes and the same reason: totalize the snapshots for cluster and lot for P-SaMI rules execution and real time status. Whenever a snapshot is docked, after updating the Docking table, these two tables are also updated, totalizing the results per group and lot.

During the execution of P-SaMI, the criteria to change the priority or status of a group during its execution are based on standard deviation, average or estimated average. *Pmia_Cluster_Processing* and *Pmia_Lot_Processing* store these data to facilitate, in real time, the action taken from W-FReDoW.

4.5.1.3 Template Tables

To execute the MDock simulations for Autodock4.2, some tables were created to store data related with templates used during the process. These template tables can store either a path to a template file or the data inside each template, important to create the snapshots or parameter files to Autogrid and Autodock executions. The following tables belong to this group: *Template_Default*, *Template_Md*, *Template_Md_Params* and *Standard_Groups*.

The *Template_Md* table stores a reference for different templates used in W-FReDoW. It keeps linked data like the template used for a Ptraj file, in case of PR (Preparation of Receptor), for example. This table has the following fields:

- *id* and *description*: fields to identify the template

- *template_type*: identify the type of this template. Today there are some possibilities: PG (Preparation of GRID), PD (Preparation of docking) or PR (Preparation of Receptor).
- *path*: path to template file that is the reference for other files.
- *original_file*: This is used for PR, where an Amber file is the file that contains the trajectories used to create the files.

The *Template_Md_Params* stores all parameters necessary to create a .DPF or a .GPF file. This table contains the details about *Template_Md*, in case this table does not have a physical file as a template. Thus all data in the parameter files (DPF and GPF extensions) that will be created are stored in *Template_Md_Params*, which contains the attributes:

- *id* and *description*: these are the identification of the template and can be visualized when the user chooses the template in W-FReDoW;
- *template_type*: identify where this template can be used. Today there are two possibilities: PD (Preparation of GRID) and PG (Preparation of docking);
- *param_name*, *param_value* and *param_description*: data stored inside the files generated. These three fields correspond to one row in the parameter file and specify the name, value and description (comment) of each parameter;
- *nr* and *visible*: *nr* identify the row where the parameter will be written in the file and *visible*, whether it is written or not;
- *template_md_id*: this is a foreign key to the *Template_Md* table. This identifies each template the parameters are related to.

The *Template_Default* stores the default parameters for creation of .DPF and .GPD. It has the same fields that *Template_Md_Params* table and it is used to show in UI interface the parameters that the user can choose to execute with W-FReDoW. After chosen, data are saved in *Template_Md_Params* table.

The *Standard_Groups* table contains snapshots, the reference for their groups and the method used to cluster them in these groups. This data are generated from a similarity function [MAC11]. This table is used in the beginning of W-FReDoW execution, updating the snapshots with the groups that are saved in the *Standard_Groups* table.

4.5.1.4 Provenance Tables

The last group consists of only one table called *Event*. This table is responsible for keeping the provenance of the execution and saving the information of actions executed during the W-FReDoW. As a scientific workflow, W-FReDoW provides information about the steps executed, helping the scientist analyze all the process chronologically.

The *Event* table has 4 attributes. The first is *id_event*, which is a primary key of this table. Second is *id_experiment*, which keeps the integrity of the experiment. The last two attributes are *dt*, which saves datetime of the action, and *info*, which saves a description of each step executed.

4.5.2 Web Application Framework

The initial idea was to build the environment under a scientific workflow. But the need of portability and communication between the Server Controller and MTC led to the development of W-FReDoW with a web approach, where the scientists can run their experiments from anywhere. Another reason was the portability to run in different Operating Systems.

Then, with a web approach chosen, some features were identified as requirements for W-FReDoW: (1) The use of dynamic websites, which allow a better manipulation of the front-end and a fast development; (2) Web server, that keeps the application on the web; and (3) Web service, which allows the communication between the server and MTC to get the results of MDock simulations. Based on these features, we decided to use a specific web application framework.

As there are several web application frameworks available, we chose the one built under Python, a widely used program language in Bioinformatics. Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and for rapid development in many areas on most platforms [PYT11].

Then, for W-FReDoW, Flask was chosen. Flask is a Python web framework also called a "microframework" [FLK11] whose idea is to keep the core simple but extensible. There is no database abstraction layer or an object relational mapper, nor a form validation because Flask is not a full stack framework. However, there are several extensions for Flask, like those mentioned.

Flask depends on two external libraries, Werkzeug [WER11] and Jinja2 [JIN11]. Werkzeug is a toolkit for WSGI [JAM09], the standard Python interface between web applications and a variety of servers for both development and deployment. Jinja2 renders templates.

As Flask does not have a native database adapter, we decided to use Psycopg2 [PSY11]. Psycopg is a PostgreSQL database adapter for the Python programming language. A relevant characteristic for this thesis is that Psycopg is thread safe (threads can share connections) and was designed for heavily multi-threaded applications that create and destroy lots of cursors and make a large number of concurrent inserts or updates in database. This feature is strongly used when MTC sends back to W-FReDoW the results of MDock simulations, since these results are different POSTs processed for a parallel execution in MTC.

The Python methods created in Flask for W-FReDoW handle jointly with the algorithm for P-SaMI, analyzing HTTP POST sent from MTC. They also handle the communication with database, keeping the provenance during the workflow execution. Moreover, Flask renders several templates used in Client Layer, the front-end interface for the scientist.

4.5.3 UI – The Client Layer

The User Interface (UI) is the front-end for the scientist. In W-FReDoW it is called Client Layer. The UI is based on different templates that Flask, through Jinja2, renders to the browser. All steps executed in the workflow are started from the UI, and the scientist can handle them.

Flask implements RESTful through Routing and W-FReDoW uses the path requested by the browser to identify the correspondent method in the framework. After processing the method, through JINJA2, the results are passed to be rendered in template, which is so-called context of the template.

From browser, a request can be sent as a HTTP GET or HTTP POST to web server. The request from HTTP POST arrives from an http form method to the server. HTTP GET requests are sent with Ajax, from JQuery (using version 1.6). All front-end interfaces are developed using HTML and JQuery in JINJA2 templates that are always rendered from Flask.

The Client Layer is shared in three groups: Setup, Execute and Analyze. These groups are placed in the menu on the left side of the site. In Setup, there is a Main Setup, composed of Software Docking, Cluster, P-SaMI Rules and Configuration links. Also, there is a Setup for Docking and Templates, where it is possible to Prepare Files, Receptor, Ligand, Reference Ligand, Grid and Docking. In Execute there appears the Experiment link, for the execution of new experiments. Finally, the last group, Analyze, has a link with the same name. Figure 18 presents the UI of W-FReDoW and it is possible to see the menu options on the left side.

w-fredow - beta
Executing Molecular Docking Simulation of a Fully-Flexible Receptor Model

Homepage About Us

W-FReDoW
Web for Flexible Receptor Docking Workflow

P-SaMI
(Pattern) Self-adapting Multiple Instances

The diagram illustrates the conceptual architecture of W-FReDoW, divided into three main layers:

- Client Layer:** Contains a 'Setup' box with sub-sections for 'Molecular Docking', 'MTC', and 'P-SaMI'. Below this is an 'Execute' button and an 'Analyse' button. A 'Scientist' icon is shown at the bottom.
- Server Controller Layer:** Contains a 'Molecular Docking' section with sub-sections for 'Prepare Files', 'Prepare Receptor', 'Prepare Ligand', 'Prepare Grid', and 'Prepare Docking'. Below this is a 'P-SaMI' section with sub-sections for 'Scheduler', 'Data Analyzer', and 'Preprocessor'. A 'FileDD for P-SaMI' icon is shown at the bottom left.
- MTC Layer:** Contains a 'Server Controller Repository', 'MTC Repository' (with sub-sections for 'Properties/Groups' and 'Input/Output Files'), 'Execution', 'Director/Transfer', 'Execute/Queue', 'Repository Manager', and 'MPC Environment'.

Flow arrows indicate 'General Flow' (solid arrows) and 'Data Flow' (dashed arrows). Connections include 'AJAX' between Client and Server Controller, 'SFTP' between Server Controller and MTC, and 'HTTP POST' from Client to MTC. A 'Database' icon is also present at the bottom left.

W-FReDoW: Conceptual Architecture

Home

SETUP Main

- [Software Docking](#)
- [Cluster](#)
- [P-SaMI Rules](#)
- [Configuration](#)

SETUP Docking-Templates

- [Prepare Files](#)
- [Prepare Receptor](#)
- [Prepare Ligand](#)
- [Prepare Reference Ligand](#)
- [Prepare Grid](#)
- [Prepare Docking](#)

EXECUTE

- [Experiment](#)

ANALYZE

- [Analyze](#)

Thesis

Using Self-adaptive Multiple Instances (P-SaMI) in a Scientific Workflow from Molecular Docking Simulations of a Fully-Flexible Receptor Model

Figure 18 – W-FReDoW UI: Client Layer

Setup Main is the general setup of the W-FReDoW. The first link, *Software Docking*, refers to the software used for MDock simulations. Today we are using Autodock4.2 for our environment, but others can be used. The next link is *Cluster*, where the scientist has the possibility of setting up the path for the place where MDock simulations will run. *P-SaMI Rules* refers to pattern rules, required when the environment starts the execution. In this link, the scientist chooses which result will be used for analysis, informs the best and worst value for P-SaMI analysis, fills minimum amount and sampling that is responsible for creating several snapshots and, at last, informs the starting analysis and next analysis, steps that serve as a reference for the workflow to control when the analysis has to occur.

Finally, the last link in *Setup Main* is *Configuration*. First, the scientist chooses the docking software for preparation of files during the workflow execution. After that, s/he has to choose the P-SaMI rules configured before. Also, s/he has to setup the paths (server, repository, job, parameter, result and temp), structures of folders that are created to store files used in workflow. Cluster information is the last step and the scientist has to inform the place and credentials for remote execution.

Figure 19 shows windows for the four links inside Setup Main explained before.

The screenshot displays four configuration windows in a grid layout:

- Cluster:** Cluster Name: Cloud Amazon NC; Cluster Path: ec2-204-236-215-213.compute-1.amazonaws.com; Save Cluster button.
- P-SaMI:** Type Docking Result: FEB; Best Value: -7.2000; Worst Value: -4.8000; Minimum Amount: 50; Sampling: 30; Start Analysis: 30; Next Analysis: 10; Save PMIA button.
- Software Docking:** Name: Autodock 4.2; Description: Autodock 4.2; Save Software Docking button.
- Configuration:** Description Short: Config FEB Start 30%; Description Long: Config FEB Start 30%; Software Docking: Autodock 4.2; P-SaMI: FEB - 6; Cluster Path: fredows/files; Path Server: repository; Path Repository: job; Path Job: parameter; Path Parameter: result; Path Result: temp; Path Temp: temp; Cluster Connection: Cloud Amazon NC; Cluster: user; Cluster Password: *****; Cluster Allocate: [empty]; Save Configuration button.

Figure 19 – W-FReDoW UI: Setup Main group. Software Docking, Cluster, P-SaMI and Configuration windows.

Another group is *Setup Docking-Templates*. This group is responsible for keeping the templates used for docking or to cluster the snapshots for P-SaMI. All links here start with “Prepare” because they are responsible for the preparation of files in the workflow. They are *Files*, *Receptor*, *Ligand*, *Reference Ligand*, *Grid* and *Docking*. Figure 20 shows these windows.

Prepare Files - Groups		
Description	Clusters	Snapshots
Group_Patricia_for_NADH_K-means	6	3100
Group_Patricia_for_PIF_K-means	6	3049
RMS_listagem_cluster_snap_means_6_ALL_ATOMS	6	3100
TCN_listagem_cluster_snap_means_6_ALL_ATOMS	6	3100

Template - Prepare Receptor		
Description	Path	Original file
1ENY_NADH_ChargesGasteiger	template_1ENY_NADH_Charges_Gasteiger.pdbqt	m MCP_00_3100ps_tr_inhA_NADH_fit.crd
1ENY_NADH_ChargesKollman	template_1ENY_NADH_Charges_Kollman.pdbqt	m MCP_00_3100ps_tr_inhA_NADH_fit.crd

Template - Prepare Docking		
Template Name:	TemplateTestDocking	
----- TEMPLATE VALUES -----		
outlev:	1	# diagnostic output level
intelec:		# calculate internal electrostatics
seed:	71277 142554	# seeds for random generator
ligand_types:	A HD OA CI	# atoms types in ligand
fld:	snap%\$.maps.fld	# grid_data_file
map:	snap%\$.A.map	# atom-specific affinity map
map:	snap%\$.HD.map	# atom-specific affinity map
map:	snap%\$.OA.map	# atom-specific affinity map
map:	snap%\$.CI.map	# atom-specific affinity map
elecmap:	snap%\$.e.map	# electrostatics map

Template - Prepare Grid		
Template Name:	TemplateOfficial	
----- TEMPLATE VALUES -----		
gridfld:	snap%\$.maps.fld	# grid_data_file
spacing:	0.375	# spacing(A)
receptor_types:	A C H HD N NA OA P SA	# receptor atom types
ligand_types:	A HD OA CI	# ligand atom types
receptor:	snap%\$.pdbqt	# macromolecule
gridcenter:	-12.281 0.921 2.439	# xyz-coordinates or auto
smooth:	0.5	# store minimum energy w/in rad(A)
map:	snap%\$.A.map	# atom-specific affinity map
map:	snap%\$.HD.map	# atom-specific affinity map
map:	snap%\$.OA.map	# atom-specific affinity map
map:	snap%\$.CI.map	# atom-specific affinity map

Figure 20 – W-FReDoW UI: Setup Docking-Templates group. Prepare Files - Groups, Template for Prepare Receptor, Docking and Grid windows.

Prepare Files comes from results of similarity functions [MAC11]. This data are updated through Python scripts. *Prepare Receptor* involved two files. The first contains a template for snapshots file. The second one is the CRD file, generated by PTRAJ, a tool from AMBER, which sets the trajectories in different positions. In our test case, there are 3.100 different trajectories. The combination between template and CRD will result in 3.100 snapshot files with extension PDBQT; default extension in Autodock4.2. *Prepare Grid* has the parameters for a GPF (Grid Parameter File) extension for each snapshot and is divided in three columns: parameter name, parameter value and parameter description. The result of these columns will create the file. Like *Prepare Grid*, *Prepare Docking* has three columns that are part of DPF (Docking Parameter File) extension files. The column “template name” helps identify both grid and docking templates.

The Execute is the main activity that the scientist can perform. Once in the Execute Experiment screen, s/he can choose the parameters for a new experiment execution. Figure 21 shows the Execute Experiment screen, where the scientists, at first, must choose the configuration. Configuration has all parameters for P-SaMI, Cluster and paths needed for W-FReDoW. Next, a method for preparing files (based on similarity function) has to be chosen to group the snapshots. In Templates, one must choose a template or other experiment that is a reference for this new experiment that is being created. Thus, 3.100 snapshots, or those they are defined, are created. Ligand indicates the ligand used for docking. Templates for GRID and Docking also have to be chosen and are important to create the GPF and DPF extension files. At the end,

when the user clicks on Execute docking Experiment, the workflow starts, the files are created, the database are updated and FReMI is called.

Figure 21 – W-FReDoW: Execute Experiment

In Analyze, the scientist has the possibility to see the status from any experiment that has been executed, even the ones in execution. Figure 22 shows the Analyze screen with experiments, dates when the execution started and finished (initial and end columns), molecular docking execution started and finished in FReMI and summary snapshots per status.

Analyze								
ID	Initial	End	Docking Ini	Docking End	A	D	F	
197	22/12/2011 14:57:15	None	22/12 14:58:46	22/12 20:02:22	531	0	2569	detail
196	21/12/2011 13:24:48	21/12 19:28:39	21/12 13:25:56	21/12 19:28:39	0	0	3100	detail
195	21/12/2011 13:05:29	None	21/12 13:06:37	None	3100	0	0	detail
194	21/12/2011 12:54:20	None	21/12 12:55:35	None	3100	0	0	detail
193	21/12/2011 12:47:30	None	None	None	3100	0	0	detail
192	18/12/2011 18:15:10	None	18/12 18:20:15	None	6	0	3094	detail
191	17/12/2011 13:59:46	None	17/12 14:00:48	None	1719	0	1381	detail

Figure 22 – W-FReDoW: Analyze

If the scientist decides to see more summaries about the snapshots s/he can always click on the detail button. Figure 23 (a) shows the results of snapshots processed per cluster and lot, whereas figure 23 (b) shows the results of each snapshot execution.

Experiment 197							
		All Snapshot		All Events/Logs			
CLUSTERS							
Cluster	Status	Priority	Lots	Snaphots Total	Snaphots Exec	%	
0	A	3	3	291	290	99	detail
1	A	3	3	474	408	86	detail
2	A	3	4	801	516	64	detail
3	A	3	4	507	466	91	detail
4	A	3	4	522	467	89	detail
5	A	3	4	505	422	83	detail
LOTS							
Cluster	Lot	Status	Priority	Snaphots Total	Snaphots Exec	%	
0	1	A	3	87	87	100	detail
0	2	A	3	87	86	98	detail
0	3	A	3	117	117	100	detail
1	1	A	3	142	139	97	detail

Experiment: 197 - Cluster: 0											
SNAPSHOTS											
Snap	C	L	S	P	Autogrid	Autodock	B.FEB	Rmsd B.FEB	B.RMSD	FEB B.RMSD	
000001	0	1	F	1	0:00:57	0:01:10	-6.5300	4.6300	3.1100	-5.5800	detail
000002	0	1	F	1	0:01:00	0:01:07	-6.4600	4.6000	3.0400	-6.4300	detail
000003	0	1	F	1	0:00:58	0:01:11	-6.5400	4.5200	2.2400	-5.3000	detail
000004	0	1	F	1	0:00:57	0:01:06	-6.9000	4.9000	3.1800	-6.1500	detail
000005	0	1	F	1	0:00:56	0:01:12	-7.0400	4.7500	2.6400	-5.9500	detail
000006	0	1	F	1	0:00:54	0:01:11	-7.3300	4.9200	3.0000	-5.9800	detail
000007	0	1	F	1	0:00:57	0:01:12	-6.8000	4.7900	3.0200	-5.7200	detail

Figure 23 – W-FReDoW: Analyze. (a) Details from Experiment 197 for Cluster and Lot. (b) Results for Snapshots from Cluster 0 in Experiment 197.

As mentioned before, the client layer of W-FReDoW is the UI, where the scientist can setup paths about the cluster, P-SaMI rules and create new templates to execute experiments for MDock simulations of an FFR model. Thus, with the rules from P-SaMI setup and during the execution, the scientist can see what it is happening with the experiment.

5 W-FREDOW: TESTS & RESULTS

After presenting W-FReDoW, an environment for smart execution of MDock simulations of FFR model, this chapter focuses on tests and results realized with W-FReDoW. The main goal of this chapter is to prove that the environment can present good results and, for that, the samples presented here show, step by step, the W-FReDoW in use.

With P-SaMI in use, the environment becomes a smart environment, because it applies rules that will reduce the amount of snapshots executed, getting better result time to finish all experiments. The result of tests is directly related with similarity criteria used by Karina Machado [MAC11], who organizes groups of snapshots with the same characteristics based on the FFR model.

All tests were executed using the same machine configuration for W-FReDoW and for middleware FReMI to ensure the same conditions to make comparisons between them. Thus, W-FReDoW was setup on a MacBook Pro, running Mac OS X, with 2.7 GHz Intel Core i7 and 8 GB 1333 MHz DDR3. The Web Server from Server Controller Layer of W-FReDoW was set up on this computer. The middleware FReMI, responsible for docking executions, was setup to run on HPC Cloud on Amazon. The cloud was set up by Renata de Paris, sponsor of FReMI. The cloud has 5 VMs (Virtual Machines) that receive 32 jobs per VM. Thus, 160 jobs can run simultaneously. Each virtual machine has the same configuration, runs under an Ubuntu 10.04, with 8 virtual cores per node and with about 2.13 Ghz/core.

To validate W-FReDoW with P-SaMI, four different tests were done. For all tests, 3.100 ps of InhA receptor were considered [MAC11]. Also, we always used Triclosan (TCL400) [KUU03] as the ligand for docking. The ligand was initially positioned in the region close to its protein binding pocket and contains two rotatable bonds. In the first test, no similarity function was used. Thus, all 3.100 snapshots are grouped in only one group and a full execution was realized. The goal for this test was to measure the execution of all snapshots without any validation of P-SaMI and to get the total time. Also, in this test we detail each step to execute the experiment.

For the three other tests, a full configuration of P-SaMI was employed. The same snapshots were grouped in 6 clusters based on similarity functions generated in [MAC11] and also

presented in [HUB10]. Each test used a specific similarity function, which means that a specific snapshot could be grouped in different clusters, and it was executed different times to validate the use of P-SaMI. Thus, the analyses were setup to start with 30%, 40%, 50% and 70% when a lot of snapshots was finished. Following, we present considerations about this chapter.

5.1 W-FReDoW test 1: Without a similarity function

One important criterion from P-SaMI is the similarity function used to group the snapshots. This function groups similarity snapshots that, during W-FReDoW, are executed in parallel and can be analyzed in real time. During the execution, groups can have their priority changed to further the most promising groups to finish before others. Also, snapshots with the worst results can perform that and their respective groups could be discarded.

In this first test, we are setting up the environment not to use the analysis that P-SaMI is responsible for doing. Our goal is to execute all snapshots in a unique group, without considering rules to analyze the results during the MDock simulations. Thus, W-FReDoW was configured for this purpose.

Before starting the execution, we created the configuration that this first test needs. First, we grouped 3.100 snapshots in only one cluster. Figure 24 shows the group *OneGroup* that was created for this test.

Prepare Files - Groups				
Description	Ligand	Algorithm	Clusters	Snapshots
Group_Patricia_for_NADH_K-means	NADH	k-means	6	3100
Group_Patricia_for_PIF_K-means	PIF	k-means	6	3049
OneGroup	NADH	None	1	3100
RMS_listagem_cluster_snap_means_6_ALL_ATOMS	NADH	means	6	3100
TCN_listagem_cluster_snap_means_6_ALL_ATOMS	NADH	means	6	3100

Figure 24 – W-FReDoW: 3.100 Snapshots with a Unique Cluster

After creating and grouping snapshots, P-SaMI was configured for a full execution without any analysis during the MDock simulation. Figure 25, shows the setup created for this purpose, with a FEB analysis, considering a “-99” and “+99” as best and worst values. These parameters are required, and we are using large numbers to be sure that there are no results out of this range. The *Lot* was configured to have all snapshots, with 3.100 snapshots, and the analysis starts only when all snapshots are finished, meaning that no analysis will be executed. Figure 25 shows a row that corresponds with the values mentioned.

P-SaMI							
id	Type Dock	Best Value	Worst Value	Minimum Value	Sampling	Start Analysis	Next Analysis
1	FEB	-10.8000	-9.2000	50	30	30	10
2	FEB	-6.0000	1.0000	40	30	30	20
3	FEB	-7.0000	0.0000	50	20	20	10
6	FEB	-7.2000	-4.8000	50	30	30	10
7	FEB	-99.0000	99.0000	3100	100	3100	0

Figure 25 – W-FReDoW: Setup of P-SaMI for a full execution.

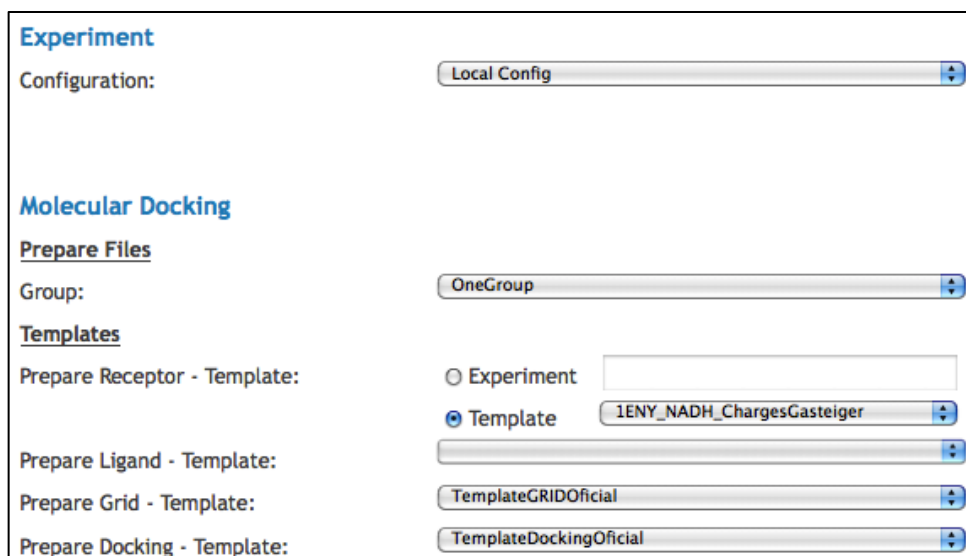
To finalize W-FReDoW Setup, the last step is to create a new configuration. The configuration uses Autodock 4.2, and it has the reference to P-SaMI id 7 (illustrated in Figure 25) and two groups of parameters. The first group is related to path and corresponds to folders that are created to store the files for FReMI execution. This structure of folders is created locally, to store parameter files and XML by group, status and priorities of snapshots, and remotely, for execution of MDock simulations. The other group is related to connection with Cluster, the remote server where the MDock simulation is executed. In our test, we are using a remote server configured on Amazon. For our part of the thesis, the important issue is to have access to the server: thus, FReMI is activated to execute the experiments as soon W-FReDoW finishes its initial preparation. Figure 26 shows the configuration for test 1.

Configuration 5	
Description Short:	Local Config
Description Long:	Local Config
Software Docking:	Autodock 4.2
P-SaMI:	FEB - 7
----- Cluster Path-----	
Path Server:	fredows/files
Path Repository:	repository
Path Job:	job
Path Parameter:	parameter
Path Result:	result
Path Temp:	temp
----- Cluster Connection -----	
Cluster:	Cloud Amazon NC
Cluster User:	user
Cluster Password:	*****
Cluster Allocate:	

Figure 26 – W-FReDoW: Values of Configuration #5, used in test 1.

Once finished the configuration, it is time to create the experiment to start the execution. Figure 27 shows the values created for this test. First, we associated the configuration and group

that were created before and presented above in figures 24, 25 and 26. Moreover, the receptor, ligand, grid and docking templates are informed.



Experiment
 Configuration: Local Config

Molecular Docking
Prepare Files
 Group: OneGroup

Templates
 Prepare Receptor - Template: Experiment Template 1ENY_NADH_ChargesGasteiger
 Prepare Ligand - Template:
 Prepare Grid - Template: TemplateGRIDOfficial
 Prepare Docking - Template: TemplateDockingOfficial

Figure 27 – W-FReDoW: Parameters for test 1 experiment execution.

The Receptor template used is called “1ENY_NADH_ChargesGasteiger”. This template is a combination of a PDBQT extension file and a CRD extension file. The first one, PDBQT file, is a template created based on InhA protein receptor, used as a model to create all 3.100 snapshots. The other value, CRD file, is a result file generated by Ptraj, a module from Amber, where the coordinates of the InhA protein trajectories are distributed inside the file. Figure 28 shows parts of both files. On the top is the CRD file with X, Y and Z coordinates for each atom. Down is the PDBQT file with 4.008 atoms for a snapshot. It is also presented the place the coordinates will be positioned.

trajectory generated by ptraj										
15.849	-20.274	8.069	15.942	-19.966	9.026	14.873	-20.259	7.808	16.183	
-21.223	7.184	16.631	-19.400	7.185	17.641	-19.377	7.596	16.687	-19.963	
5.765	15.686	-20.122	5.365	17.244	-19.285	5.119	17.220	-20.914	5.782	
16.146	-17.963	7.228	16.630	-17.177	8.039	15.185	-17.594	6.381	14.872	
-18.270	5.699	14.588	-16.247	6.288	13.712	-16.193	6.934	15.306	-15.505	
6.639	14.193	-15.866	4.862	15.065	-15.656	4.017	12.897	-15.721	4.566	
12.232	-16.030	5.260	12.373	-15.483	3.210	12.653	-16.337	2.593	10.824	
-15.414	3.266	10.429	-16.161	3.955	10.539	-14.435	3.651	10.154	-15.616	
1.890	10.740	-15.175	1.085	10.046	-17.104	1.576	9.373	-17.595	2.279	
9.671	-17.247	0.563	11.029	17.566	1.661	8.783	-14.961	1.814	8.868	
-13.883	1.953	8.342	-15.152	0.836	8.139	-15.366	2.596	12.953	-14.239	

REMARK	4	XXXX	COMPLIES	WITH	FORMAT	V	0			
ATOM	1	N	ALA	1	0.000	0.000	0.000	0.00	0.00	-0.067 N
ATOM	2	H1	ALA	1	0.000	0.000	0.000	0.00	0.00	0.275 HD
ATOM	3	H2	ALA	1	0.000	0.000	0.000	0.00	0.00	0.275 HD
ATOM	4	H3	ALA	1	0.000	0.000	0.000	0.00	0.00	0.275 HD
ATOM	5	CA	ALA	1	0.000	0.000	0.000	0.00	0.00	0.172 C
ATOM	6	HA	ALA	1	0.000	0.000	0.000	0.00	0.00	0.098 H
ATOM	7	CB	ALA	1	0.000	0.000	0.000	0.00	0.00	-0.042 C
ATOM	8	HB1	ALA	1	0.000	0.000	0.000	0.00	0.00	0.030 H
ATOM	9	HB2	ALA	1	0.000	0.000	0.000	0.00	0.00	0.030 H
ATOM	10	HB3	ALA	1	0.000	0.000	0.000	0.00	0.00	0.030 H
ATOM	11	C	ALA	1	0.000	0.000	0.000	0.00	0.00	0.248 C
ATOM	12	O	ALA	1	0.000	0.000	0.000	0.00	0.00	-0.271 OA

Figure 28 – CRD and PDBQT files used to create snapshots.

Some attributes from grid and docking templates used for this experiment are shown in figure 29. These templates are stored in the database, are created in W-FReDoW and are based on ADT Tools. The template “Prepare Grid” refers to a “GPF” file extension used on Autodock4.2 process while “Prepare Docking” template refers to “DPF” file extension. For an experiment it is necessary to do the changes on these templates and save them. Thus, during the execution, W-FReDoW creates 3.100 “GPF” and “DPF” based on both definitions. The parameters that reference snapshots must be dynamic, because they are modified during W-FReDoW execution. For example, the “map” parameter that appears on both templates has “%s” concatenated to its value. When the file “snap000001.dpf” is created, tags with “%s” are modified for “000001” to keep a same file reference. In this case, the value for “map” parameter on “Prepare Grid” will be set to “snap000001.A.map”.

Template - Prepare Grid			Template - Prepare Docking		
Template Name:	TemplateOficial		Template Name:	TemplateTestDocking	
----- TEMPLATE VALUES -----			----- TEMPLATE VALUES -----		
gridfld:	snap%.maps.fld	# grid_data_file	outlev:	1	# diagnostic output level
spacing:	0.375	# spacing(A)	intelec:		# calculate internal electrostatics
receptor_types:	A C H HD N NA OA P SA	# receptor atom types	seed:	71277 142554	# seeds for random generator
ligand_types:	A HD OA CI	# ligand atom types	ligand_types:	A HD OA CI	# atoms types in ligand
receptor:	snap%.pdbqt	# macromolecule	fld:	snap%.maps.fld	# grid_data_file
gridcenter:	-12.281 0.921 2.439	# xyz-coordinates or auto	map:	snap%.A.map	# atom-specific affinity map
smooth:	0.5	# store minimum energy w/in rad(A)	map:	snap%.HD.map	# atom-specific affinity map
map:	snap%.A.map	# atom-specific affinity map	map:	snap%.OA.map	# atom-specific affinity map

Figure 29 – Parts of grid and docking templates used in test 1.

Before starting the execution, MTC has to be setup. For this test, FReMI was setup to run on HPC Cloud on Amazon. The cloud was set up by Renata de Paris, sponsor of FReMI. The cloud has 5 VMs (Virtual Machines) that receive 32 jobs per VM. Thus, 160 jobs can run simultaneously. Each virtual machine has the same configuration, runs under an Ubuntu 10.04, with 8 virtual cores per node with about 2.13 Ghz/core.

After the configuration of W-FReDoW and MTC environment (FReMI) for a new experiment, the scientist can start the execution. Next step occurs completely in the Server Controller layer of W-FReDoW. W-FReDoW first creates a new experiment and the structure of folders under this new experiment. Every experiment is kept physically in the computer and also stored in the database. This step is responsible for a preparation of files for MDock simulations. Also, it stores in FReDD for P-SaMI database all information of this new experiment that was created. The experiment 196, created for this test, and its structure of folders and files are shown in figure 30.

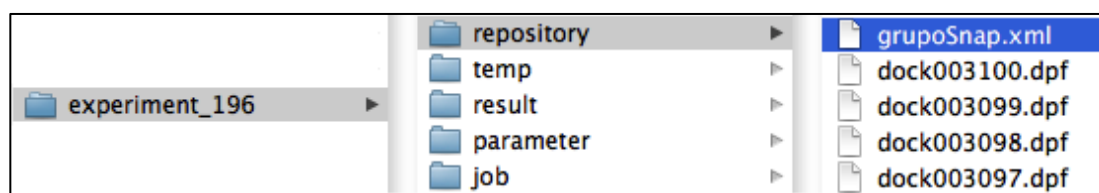


Figure 30 – Structure of folders created for an experiment.

In Molecular Docking preparation, the files of experiments are created. For this test, Server Controller creates 3.100 PDBQTs (snapshots), 3.100 GPF (parameters for Autogrid) and 3.100 DPF (parameters for Autodock) files. These files are created based on templates setup. So, 9300 files are created and saved in “/experiment/repository/” folder.

Other step of preparation is related with P-SaMI. With all data saved in the database, and grouped, the lot generation starts. For this test, just one group and 1 lot were created, and both have all 3.100 snapshots inside. Moreover, with 9.100 files created, the last one, called "grupoSnap.xml", is created. This file contains all snapshots and their respective groups. This file is also saved under the repository folder with other files of Server Controller (see figure 30).

After the creation of files in Server Controller, the Uploader sends "grupoSnap.xml" file to FReMI. This occurs through an SSH connection with the remote server. The file is sent by SFTP (Secure File Transfer Protocol). Thus, FReMI starts its execution, based on this setup inside this file.

The Preparation and Uploader steps of W-FReDoW are very fast. For these 9.101 files the time is 1:08 minutes. The web server of W-FReDoW is setup in MacBook Pro, running Mac OS X, with 2.7 GHz Intel Core i7 and 8 GB 1333 MHz DDR3.

After sending files to the remote server, the Data Analyzer, component from Server Controller, waits for requests. As presented, Data Analyzer is a method that waits for GETs or POSTs requests. In default configuration, the requests are sent as an HTTP POST. For the experiment 196, snapshot 1, this is a request sample with a result of MDock simulation, sent from FReMI to Server Controller:

```
"http://127.0.0.1/update_docking_status?id_experiment=196&snapshot="snap000001"&  
best_feb=6.5300&rmsd_best_feb=4.6300&best_rmsd=3.1100&feb_best_rmsd=-5.5800&  
autogrid_dt_ini="2011-12-21 15:30:29"&autogrid_dt_end="2011-12-21 15:31:26"&  
autodock_dt_ini="2011-12-21 15:31:26"&autodock_dt_end="2011-12-21 15:32:37"
```

As soon as a new request comes, Data Analyzer, first, validates the parameters. For example, a snapshot that was already processed cannot have its status changed again. Start date of Autogrid cannot be less than finish date.

As mentioned, experiment 196 was setup to execute all snapshots without any analysis for P-SaMI. Thus, whenever an HTTP POST from FReMI comes to Analyzer, its status of snapshots is updated in the database. Figure 31 shows, in Analyze, the final results of experiment 196.

Analyze									
ID	Initial	End	Docking Ini	Docking End	A	D	F		
196	21/12/2011 13:24:48	21/12 19:28:39	21/12 13:25:56	21/12 19:28:39	0	0	3100		detail
195	21/12/2011 13:05:29	None	21/12 13:06:37	None	3100	0	0		detail
194	21/12/2011 12:54:20	None	21/12 12:55:35	None	3100	0	0		detail
193	21/12/2011 12:47:30	None	None	None	3100	0	0		detail

Figure 31 – Experiment 196 with 3100 snapshots finished.

The Analyze helps the scientist observe, in real time, a status from execution. For experiment 196, the execution has already finished (see figure 31). The Preparation step was executed in 1:08 minute. This time is the difference between “Initial” and “Docking Ini” columns. “Docking Ini” corresponds to the beginning of execution from FReMI, while “Docking End” is the last snapshot processed on MTC and analyzed from Data Analyzer. Therefore, the difference between “Docking End” and “Docking Ini” is the total time that the 3.100 experiments took to finish; for this experiment, it took 6:02 hours. The total time of execution is the difference between “End” and “Initial” and here it was of 6:03 hours. This was the total time for this execution.

A last consideration about this experiment is related to events. Each event is information about some step executed on the W-FReDoW flow. Figure 32 depicts what happened in the begging of experiment 196.

Experiment: 196	
EVENTS	
Date Created	Info
2011-12-21 13:32:05.484438	Updated snap002769
2011-12-21 13:32:05.481064	snap002769 done!
2011-12-21 13:32:05.463038	Updating snap002769
2011-12-21 13:31:58.712956	Updated snap002578
2011-12-21 13:31:58.709422	snap002578 done!
2011-12-21 13:31:58.688737	Updating snap002578
2011-12-21 13:25:56.082106	Starting docking into cluster
2011-12-21 13:25:49.856330	Creating XML Files
2011-12-21 13:25:42.005486	Creating PMIA - LOTS
2011-12-21 13:25:38.403277	Preparing .DPF files
2011-12-21 13:25:36.835671	Preparing .GPF files
2011-12-21 13:24:57.895948	Creating Snapshots
2011-12-21 13:24:48.089103	Creating experiment 196

Figure 32 – Events for Experiment 196.

5.2 W-FReDoW test2: RMS_listagem_cluster_snap_means_6_ALL_ATOMS

For this second test, we intend to depict the use of P-SaMI rules during the execution of steps of W-FReDoW. Thus, we can prove that P-SaMI can be used in an environment for MDock simulations for an FFR model considering the prioritization and discard of snapshots.

The first step was to setup the Cluster. This corresponds to the place where the MDock simulations will be running during the process. In our tests, we used, as mentioned, a Cloud on Amazon. Figure 33 shows the setup used during the execution of W-FReDoW to send data to FReMI. The *Hostname* parameter is the server location, *Path Key* is the key to access Amazon Cloud and open a connection, and *Path Repository* is the folder path where the files will be added for FReMI execution.

Cluster				
id	Name	Hostname	Path Key	Path Repository
6	Cloud Amazon LABIO	ec2-204-236-213-81.compute-1.amazonaws.com	fredows/key/esc_key_LABIO	/mnt/data/Repository/

Figure 33 – Cloud setup for communication between W-FReDoW and FReMI.

Next step is to setup all P-SaMI rules used in this test. We are using four different setups for each experiment executed as shown in Figure 34. Also, this figure shows other parameters: *Best Value* and *Worst Value*, that represent the best and worst result values to get the average between them; in this case the average is “-6.0”; *Minimum Value* and *Sampling*, that are parameters used on lot creation; *Start Analysis* and *Next Analysis*, which are the only parameter values that changed between each P-SaMI id. In id “6”, *Start Analysis* is “30%” and *Next Analysis* is “10%”. It means that for an execution using this id, the analysis starts with 30% of snapshots finished per lot and always 10% after the first analysis (40%, 50%, ... 90%). For id “9”, for example, the analysis starts with 50% and more 4 analyses for the same lot are executed (60%, 70%, 80% and 90%).

P-SaMI							
id	Type Dock	Best Value	Worst Value	Minimum Value	Sampling	Start Analysis	Next Analysis
6	FEB	-7.2000	-4.8000	50	30	30	10
8	FEB	-7.2000	-4.8000	50	30	40	10
9	FEB	-7.2000	-4.8000	50	30	50	10
10	FEB	-7.2000	-4.8000	50	30	70	10

Figure 34 – Four different setups for P-SaMI with analysis starting with 30%, 40%, 50% or 70%.

Following the steps for a new execution on the environment, it is necessary to create different configurations for each execution. In figure 35 the ids 6, 7, 8 and 9 represent four different setups to be used during the execution of experiments. The only difference between them is the use of different “Id P-SaMI”, “Desc Short” and “Desc Long” for each configuration. The column “Id P-SaMI” is a link to P-SaMI rules used and presented in figure 34.

Configuration								
id	Desc Short	Desc Long	Soft Dock	Id P-SaMI	Path Server	CI	SF	
1	Configuration 1	Configuration long 1	1	1	fredows/files	1	True	detail
2	Default	Default Configuration	2	1	fredows/files	1	True	detail
5	Local Config	Local Config	2	7	fredows/files	5	False	detail
4	Config FEB Start 30%	Config FEB Start 30%	2	6	fredows/files	5	True	detail
6	Config FEB 30% Cloud LABIO	Config FEB 30% Cloud LABIO	2	6	fredows/files	6	True	detail
7	Config FEB 40% Cloud LABIO	Config FEB 40% Cloud LABIO	2	8	fredows/files	6	True	detail
8	Config FEB 50% Cloud LABIO	Config FEB 50% Cloud LABIO	2	9	fredows/files	6	True	detail
9	Config FEB 70% Cloud LABIO	Config FEB 70% Cloud LABIO	2	10	fredows/files	6	True	detail

Figure 35 – Different configurations that are used on experiments with analysis starting with 30%, 40%, 50% and 70%.

The last step before starting the execution is to setup an experiment and start the process execution. This step is based on Experiment window, mentioned on Figure 27, which allows to the scientist the possibility of setting up the data. For this test we are using the four configurations created and the similarity function `RMS_listagem_cluster_snap_means_6_ALL_ATOMS`, which was presented by Machado [MAC11].

Following all setup, this experiment was executed 6 times with different setups. Table 2 presents the results of each experiment executed. The ids 220, 221 and 222 were executed with the same configuration and have very similar results. They started to analyze the results with 30%. The other ids started the analysis in different moments.

Table 2 – Final results of experiment executions of W-FReDoW considering the RMS_listagem_cluster_snap_means_6_ALL_ATOMS as similarity function.

No of experiment	Initial Analysis (%)	Preparation Time	FReMI Time	Amount Finished	Amount Discarded
220	30%	00:01:02	04:02:17	2249	851
221	30%	00:01:03	04:02:01	2249	851
222	30%	00:01:01	04:00:19	2249	851
224	40%	00:01:28	04:16:42	2407	693
225	50%	00:01:04	04:25:25	2495	605
226	70%	00:01:03	05:01:35	2819	281

Also, table 2 presents other information about each experiment executed. First, the “No of experiment” is the experiment executed with an individual setup. “Initial Analysis” represents in percent value the beginning of analysis. “Preparation Time” is the total time to create the 9.300 files and send the XML configuration file to the cloud. “FReMI time” is the total time to finish all 3.100 MDock simulations. “Amount Finished” is the amount of snapshots finished on the experiment. “Amount Discarded” is the amount of snapshots that were discarded based on the rules of P-SaMI setup for the experiment. These snapshots were not executed on FReMI.

The “Preparation time” is a very fast execution step and is not a bottleneck process of the whole. This step does not depend on the configuration of the experiment. In other words, “FReMI time” is totally related with the configuration used. When the “Initial Analysis” starts earlier, the FReMI finishes its execution earlier too. This happens because of the number of snapshots discarded. When more snapshots were discarded, FReMI spent less time to finish the experiment. When the analysis starts with 70%, i.e., less snapshots were discarded and more time was spent to finish the whole experiment.

During the process, W-FReDoW records all moments when a status or priority of a lot changes. Figure 36 shows some results from P-SaMI analysis about experiment 224. A red line indicates the cluster 0 and lot 3. In the first row, when “snap000167” is processed, 40% of this cluster/lot was already executed and the analysis starts. Thus, based on the averages (-5.9760, -6.1091 and -6.0000) the priority is changed from 2 to 1; it means that this cluster/lot had its priority decreased on FReMI. The AVG1 (-5.9760) corresponds to the arithmetic average from results of snapshots of the cluster/lot already executed. The AVG2 (-6.1091) corresponds to the estimated average for the rest of snapshots from this cluster/lot. The AVG3 is the arithmetic average from *Best Value* (-7.2) and *Worst Value* (-4.8) setup on P-SaMI parameters. Then, as the AVG1 is lower than AVG3 and AVG2 is higher than AVG3, the priority changes to “decrease”. It is

possible to observe on the other rows selected that the analysis keeps trying to decrease this lot, until the analysis with 70% is executed and the cluster 0 and lot 3 are discarded, because AVG1 and AVG2 are lower than AVG3.

Experiment: 224

Analysis												
C	L	Snapshot	%	Avg1	Avg2	Avg3	SO	PO	SN	PN	Descr	Date Created
0	3	snap000167	40.00	-5.9760	-6.1091	-6.0000		2		1	decrease	2012-02-02 14:26:03
0	3	snap000173	50.00	-5.9560	-6.0681	-6.0000		1		1	decrease	2012-02-02 14:57:05
0	3	snap000177	60.00	-5.9170	-6.0072	-6.0000		1		1	decrease	2012-02-02 15:30:28
0	3	snap000183	70.00	-5.8866	-5.9539	-6.0000		0	D	0	discard	2012-02-02 15:58:55
1	1	snap002677	40.00	-5.7591	-5.9454	-6.0000		0	D	0	discard	2012-02-02 16:05:24
1	2	snap002810	40.00	-5.5823	-5.6582	-6.0000		0	D	0	discard	2012-02-02 15:29:59
1	3	snap002897	40.00	-5.6661	-5.7520	-6.0000		0	D	0	discard	2012-02-02 14:56:39
1	4	snap002954	40.00	-5.7325	-5.8292	-6.0000		0	D	0	discard	2012-02-02 14:25:25
1	5	snap003003	40.00	-5.6850	-5.7686	-6.0000		0	D	0	discard	2012-02-02 14:25:21
1	6	snap003063	40.00	-5.8715	-5.9598	-6.0000		0	D	0	discard	2012-02-02 14:52:29
2	4	snap001358	40.00	-5.9579	-6.0067	-6.0000		2		1	decrease	2012-02-02 15:11:04

Figure 36 – Analysis realized in experiment 224. Four rows selected to show the cluster 0 and lot 3, where the first had its priority decreased and at last was discarded with 70% based on the Avg1 (Arithmetic Average) and Avg2 (Estimated Average) comparing with Avg3 (P-SaMI Average).

5.3 W-FReDoW test3: TCN_listagem_cluster_snap_means_6_ALL_ATOMS

For this third test, we depict the results obtained by using a different similarity function from the test before. The setup presented on sub-chapter 5.2 was the same. The only change is the similarity function used to group the snapshots for this test. Then, for this third test, the similarity function used was TCN_listagem_cluster_snap_means_6_ALL_ATOMS, presented by Machado [MAC11].

The process to setup the environment was created for test2 and can be reused on this experiment. Thus, just the Experiment page (Figure 27) was setup to use a different way to prepare files. The preparation of the files is related to how the files (snapshots) were grouped. This test was based on TCN_listagem_cluster_snap_means_6_ALL_ATOMS. After this configuration, it is time to execute the MDock simulations.

As presented on test 2, here on this test different experiments were executed. Table 3 shows four different experiments, where the rules of P-SaMI have started with 30% on experiment 227, 40% on experiment 228, 50% on experiment 229 and 70% on experiment 230. The

preparation time between them was around 01:09 minute. For experiment 227, FReMI Time was 03:59:02 hours, 2210 snapshots were docked and 890 snapshots were discarded. For experiment 228, FReMI Time was 04:18:28 hours, 2423 snapshots were docked and 677 snapshots were discarded. For experiment 229, FReMI Time was 04:30:04 hours, 2514 snapshots were docked and 586 snapshots were discarded. On the last experiment using this similarity function, experiment 230, FReMI Time was 05:06:59 hours, 2868 snapshots were docked and 232 snapshots were discarded.

Table 3 – Final results of experiment executions of W-FReDoW considering the TCN_listagem_cluster_snap_means_6_ALL_ATOMS as similarity function.

No of experiment	Initial Analysis (%)	Preparation Time	FReMI Time	Amount Finished	Amount Discarded
227	30%	00:01:10	03:59:02	2210	890
228	40%	00:01:09	04:18:28	2423	677
229	50%	00:01:05	04:30:04	2514	586
230	70%	00:01:13	05:06:59	2868	232

5.4 W-FReDoW test4: Group_Patricia_for_NADH_K-means

After presenting two tests using similarity functions from Machado [MAC11], this test presents experiments that were used by Hubler [HUBLER10]. The grouping used by Hubler is referred to in this thesis as Group_Patricia_for_NADH_K-means. The steps to setup were the same as in the two tests before. The only difference was to inform this similarity function on Preparation Files field on Experiment page (Figure 27).

Again, we executed four experiments based on this similarity. Table 4 shows four different experiments where the rules of P-SaMI were started with 30% on experiment 231, 40% on experiment 232, 50% on experiment 233 and 70% on experiment 234. The preparation time between them was around 01:24 minute. For experiment 231, FReMI Time was 04:00:33 hours, 2264 snapshots were docked and 836 snapshots were discarded. For experiment 232, FReMI Time was 04:16:49 hours, 2377 snapshots were docked and 723 snapshots were discarded. For experiment 233, FReMI Time was 04:42:37 hours, 2537 snapshots were docked and 563 snapshots were discarded. On the last experiment using this similarity function, experiment 234, FReMI Time was 05:03:02 hours, 2818 snapshots were docked and 282 snapshots were discarded.

Table 4 – Final results of experiment executions of W-FReDoW considering the Group_Patricia_for_NADH_K-means as similarity function.

No of experiment	Initial Analysis (%)	Preparation Time	FReMI Time	Amount Finished	Amount Discarded
231	30%	00:01:14	04:00:33	2264	836
232	40%	00:01:28	04:16:49	2377	723
233	50%	00:01:26	04:42:37	2537	563
234	70%	00:01:30	05:03:02	2818	282

5.5 Last Chapter Considerations

This chapter presented four tests performed with W-FReDoW. The first test showed a complete setup of the environment to run the experiments. For the other tests, only the results were shown. For all tests, a real environment was used. First, the parameters are set up on W-FReDoW, the preparation files are executed, the communication with the middleware FReMI (cloud on Amazon) and P-SaMI is performed, and the dockings are analyzed since the moment they arrived on W-FReDoW (web server).

Thus, based on the results presented on each test, table 5 shows the total time for each test executed for each experiment. The lines show the tests. The columns show the percentage used for W-FReDoW to start the analysis of results of snapshots through P-SaMI rules. The cells present the total time spent to dock the snapshots by FReMI.

Table 5 – Time spent to execute MDock simulations with W-FReDoW using FReMI middleware.

Test	30%	40%	50%	70%	100%
Test 1	-	-	-	-	06:02:00
Test 2	04:02:17	04:16:42	04:25:25	05:01:35	-
Test 3	03:59:02	04:18:28	04:30:04	05:06:59	-
Test 4	04:00:33	04:16:49	04:42:37	05:03:02	-

The first test, presented on table 5, has the total time of 06:02 hours to execute all experiments. Tests 2, 3 and 4 presented very similar results for each analysis of P-SaMI performed. For example, starting the analysis of P-SaMI with 30% of finalized snapshots from lot, test 2 finished in 04:02:17 hours, test 3 finished in 03:59:02 hours and test 4 finished in 04:00:33 hours. Thus, we can observe that the time difference is very low, with a maximum of around three minutes of difference between the best and worst times.

Following this analysis, it is possible to say that the three similarity functions (RMS_listagem_cluster_snap_means_6_ALL_ATOMS, TCN_listagem_cluster_snap_means_6_ALL_ATOMS

and Group_Patricia_for_NADH_K-means) have very similar results when used with P-SaMI rules. Also, the other analysis has similar times to finish the MDock simulations.

On the other hand, we can observe a gain of time when comparing test 1 with other tests. Test 1 spent 6:02 hours, while the other tests spent less time because snapshots were discarded. Also, it is possible to say that the early the analysis starts, the faster the experiment is finished. Always for the same test, when analysis started earlier, for example with 30%, the time to finish was of around 4 hours, while with 50% around 4:30 hours.

Another comparison is made and shown in table 6. The lines represent the tests and the columns the experiments with analysis of snapshots starting in different moments. Also for each analysis column “F” represents Snapshots Finished and “D” Snapshots Discarded. For example, in test 1 3.100 snapshots were finished without one discarded. As presented on test 1, the purpose of this test was to execute dockings of all snapshots to obtain the total time. Test 2 presented 2.249 snapshots finished (docked) and 851 snapshots discarded (not executed) when analysis started with 30%. When starting with 40%, 2.407 snapshots were finished and 693 were discarded.

Table 6 – Final status of snapshots after docking with W-FReDoW using FReMI middleware.

Test	30%		40%		50%		70%		100%	
	F	D	F	D	F	D	F	D	F	D
Test 1	-	-	-	-	-	-			3100	-
Test 2	2249	851	2407	693	2495	605	2819	281	-	-
Test 3	2210	890	2423	677	2514	586	2868	232	-	-
Test 4	2264	836	2377	723	2537	563	2818	282	-	-

Another consideration involves the time to execute each snapshot by FReMI. For the experiments mentioned on this chapter, FReMI spent, on average, 50 seconds for Autogrid and 1:16 minute for Autodock for each snapshot docked. Therefore, we have an average of 2:06 minutes for each MDock simulation on FReMI using Autodock 4.2. The use of W-FReDoW does not interfere on the time executed by FReMI. If FReMI executes the docking without communication with W-FReDoW, the time to execute each docking is the same. Some time is spent by FReMI to handle nodes and allocating them, whenever necessary. This loss of time is shown in Renata de Paris’ thesis.

At last, as the Server Controller is based on a Web Server and FReMI is a middleware that runs in another server, it will be possible to parallelize different experiments. We could have an experiment based on an MD with 3.100 snapshots running on FReMI set up to Amazon Cloud, and

another MD based on 6.200 snapshots running in another cloud, or in another machine that must have to be Internet access to communicate with the Server Controller. For this thesis we did not test experiment parallelization because we only had access to one cloud on Amazon.

6 RELATED WORK

This chapter surveys previous work in some subjects related with this thesis. Some common subjects in several parts of this work are:

- Use of a data pattern in Bioinformatics;
- Tools used for execution of MDock simulations;
- Integration with MTC in Scientific Workflows;
- Database to store real time data for MDock simulations of FFR model;

6.1 Advanced Data Flow Support for Scientific Grid Workflow Applications

Qin and Fahringer [QIN07] affirm there is a lack to flexible dataset-oriented data flow mechanisms to meet the complex requirements of scientific Grid workflow applications. This complexity comes through scientists and engineers who need to manage and process large datasets on distributed Grid resources. Then, they present a sophisticated approach to address this problem by introducing a data collection concept and the corresponding collection distribution constructs, applied to Grid workflow applications.

The authors also claim that a Grid workflow application can be seen as a collection of computational tasks that are processed in a well-defined order to accomplish a specific goal. Many control flow constructs have been identified and developed in Grid workflow systems to enable users to define the exact execution order of tasks. There are four categories: sequential, parallel, conditional and iterative constructs. Data flows in scientific Grid workflow applications are commonly complex because datasets are involved.

Similarity to this thesis, Qin and Fahringer also work with large volume of data and groups of data. Another common approach occurs in execute experiments in parallel. But the way datasets and the corresponding data elements can be specified in data flow links, especially how datasets can be distributed onto the parallel loop iterations, is a problem not sufficiently

addressed by most Grid workflow languages. The authors also work with a reduction of experiments.

Although this thesis does not work with Grid, we addressed the parallelization through the integration with an MTC environment that manages this question.

6.2 Molecular docking simulation of flexible receptor

FreDoWS [MAC07, MAC11a] has already been presented in this thesis as an important workflow system for MDock simulations of FFR model. First, the WS aimed at the execution of all conformations [MAC07], but the possibility to reduce the number of conformations for MDock was already addressed [MAC07, MAC11a].

The criteria added to FReDoWS to pick up conformations based on the idea that, if a result docking with a conformation has a good FEB and RMSD, it is possible that this same conformation, in an interaction with a similar ligand, can also present good FEB and RMSD. Trying to check the efficiency of this idea, Machado [MAC11, MAC11a] concluded through some tests that for ligands from the same class it is not necessary to use all conformations of the FFR model. The way the conformations are picked up in FReDoWS presented good results.

FReDoWS has a strong connection with this thesis. The workflow defined in [MAC07] was used to create W-FReDoW. However, the use of the dynamism through P-SaMI in W-FReDoW proves that both theses have different approaches to work with fewer conformations. While FReDoWS works with a mechanism to pick up the conformations manually, W-FReDoW works with a pattern to discard conformations based on the analysis of groups with similar characteristics.

Another difference between both is the environment where the MDock simulations on FFR model occur. FReDoWS is a scientific workflow modeled through the JaWe editor, and is executed by Enhydra Shark. It uses AutoDock 3.05 for the docking simulations. W-FReDoW uses a web server to communicate between web client and the MTC environment and uses Autodock 4.2 for docking simulations.

At last, W-FReDoW executes parallel docking simulations. The environment presented in this thesis communicates with an MTC environment that manages those snapshots to be executed, e.g., in a HPC or Cloud. FReDoW does not implement parallel execution.

6.3 Data parallelism in Bioinformatics workflows

Coutinho et al. [COU10] say that Bioinformatics activities within a workflow often involve large-scale data processing and several data conversions using shimming activities. Each one of the

workflow activities may receive huge amounts of data to be processed. This large scale processing may be unfeasible if scientists do not run their experiments in high performance environments and using parallelism techniques such as data fragmentation.

Based on these concepts, Coutinho et al. [COU10] developed Hydra. Hydra is a middleware that provides a set of components to be included in the workflow specification of any SWfMS to control parallelism of activities following the MTC paradigm. Hydra is based on a homogeneous cluster environment and relies on a centralized scheduler (such as Torque). Using Hydra, the MTC parallelism strategy can be registered, reused, and provenance may be uniformly gathered during the execution of workflows. Hydra still lacks on data parallelism mechanisms coupled to provenance facilities. Figure 37 shows Hydra conceptual architecture.

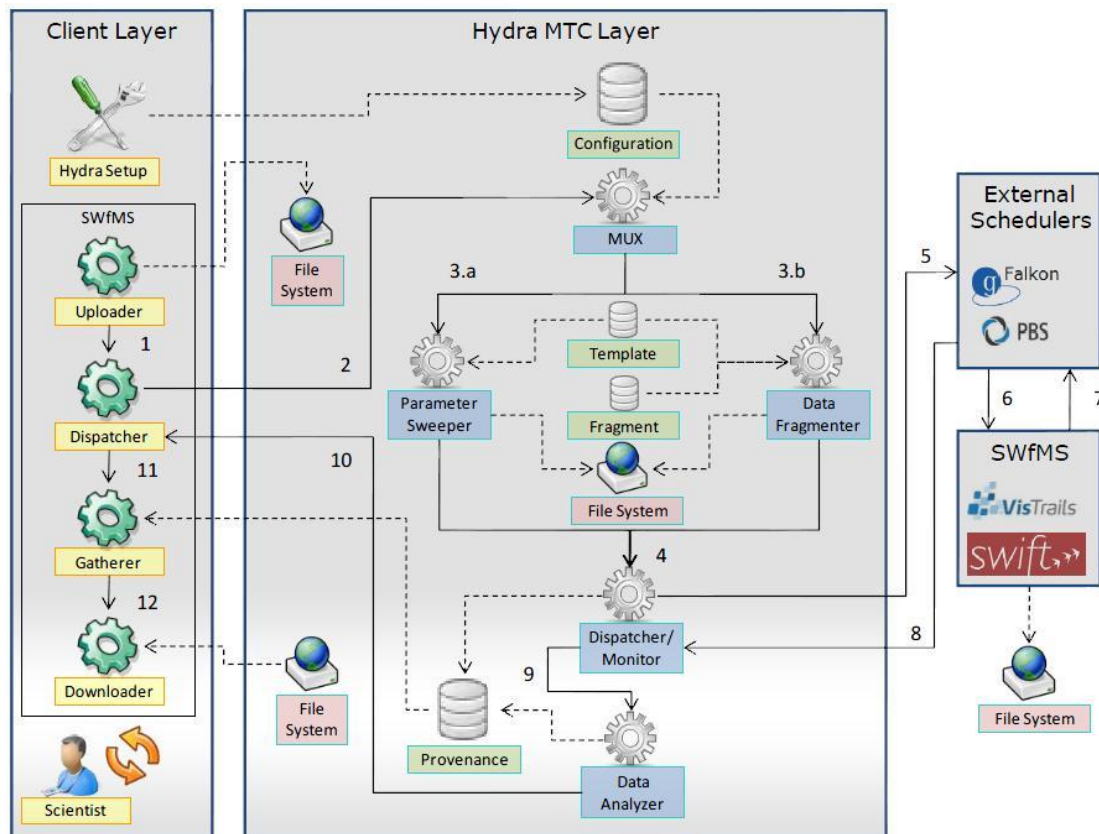


Figure 37 – Hydra Conceptual Architecture

Hydra has great relevance for this thesis. It presents common characteristics such as exploring Bioinformatics area and large data processing. The environment is separated on layers that can be coupled, like Hydra middleware. Another similar feature, and one of the challenges, is the parallelism through MTC paradigm. It also allows keeping the provenance of the execution of

workflow. Hydra goes deep into data fragmentation, something that W-FReDoW does not support yet. We did not identify the use of a pattern that can reduce the number of experiments, one of the main approaches of this thesis.

6.4 DOVIS: an implementation for high-throughput virtual screening using AutoDock

Zhang et al. [ZHA08] worked on techniques to use HPC platforms for execution of high data processing from molecular docking experiments. The authors say the development of molecular docking tools with HPC system is not trivial. Thus, an application called DOVIS was developed to use AutoDock (version 3), running in parallel on a Linux cluster. The authors claim DOVIS can efficiently dock large numbers (millions) of small molecules (ligands) to a receptor, screening 500 to 1,000 compounds per processor per day.

W-FReDoW has a broader scope than DOVIS, since it works with an MTC environment as a module integrated with it. Neither the approach to execute MDock simulations of FFR model nor to achieve an RFFR model is implemented in DOVIS.

6.5 Relaxed Complex Scheme for Receptor Flexibility in Computer-aided Drug Design

Amaro et al. [AM08] affirm that while ligand flexibility is well accounted for in computational drug design, the effective inclusion of receptor flexibility remains an important challenge. The relaxed complex scheme (RCS) is a promising computational methodology that combines the advantages of docking algorithms with dynamic structural information provided by molecular dynamics (MD) simulations, therefore explicitly accounting for the flexibility of both the receptor and the docked ligands.

The authors discuss new extensions and improvements of RCS methodology that include virtual screening, and methods to improve its computational efficiency by reducing the receptor ensemble to a representative set of configurations.

Therefore the methods used for Amaro et al. propose the same aim as the present thesis: to reduce the number of receptors. Amaro used some methods to reach this reduction. On the other hand, this thesis presents a pattern that helps in this activity.

6.6 FReDD: Supporting Mining Strategies through a Flexible-Receptor Docking Database

FReDD is an acronym to Flexible-Receptor Docking Database. FReDD repository stores all features of the receptor snapshots, the ligands, and the docking simulation of MDock simulations of the FFR model [WIN09]. This repository allows easy retrieval of its information to produce comprehensive data to be mined. Wink et al. [WIN09] say the use of this database helps

concentrate efforts on data mining to explore the docking results in order to accelerate the identification of promising ligands against the InhA target.

The database presented in this thesis, FReDD for P-SaMI, is an extension of FReDD. Although both store data from MDock simulations of FFR model, they have different focuses. While FReDD stores data for mining process, FReDD for P-SaMI maintains the provenance of W-FReDoW. In FReDD, data are stored after the execution of workflow. FReDD for P-SaMI stores data in real time, during the setup, execution and analysis of W-FReDoW.

7 CONCLUSIONS

This study presented a new environment for MDock simulations of the FFR model. First, literature reviews about Bioinformatics and Scientific Workflows (SW) are presented. The first one proved important for the understanding of the workflow for RDD. The other, SW, was paramount to understand the flow control and data pattern used in this environment. Workflow concepts helped understanding the use of P-SaMI, a data pattern for Scientific Workflows.

One of the artifacts of this research is a new environment built using rules of P-SaMI called W-FReDoW (Web Flexible Receptor Docking Workflow). With W-FReDoW the domain specialist can run docking experiments using a web front-end interface. So the docking files are prepared and submitted to an MTC environment. After docking, the results from each group of conformations are analyzed based on P-SaMI rules. Finally, non-promising groups are discarded.

Another deliverable from this thesis is FReDD for P-SaMI. This database is designed to support MDock simulations of an FFR model based on P-SaMI. The tables and attributes store all data during the workflow of the process and are an important support of analysis of P-SaMI. Also, FReDD for P-SaMI provides all the provenance of the execution to the domain specialist.

The scalability of the environment is another topic achieved with the different components from W-FReDoW. First, W-FReDoW can scale simply by increasing the number of Virtual Machines or Clusters used by FReMI. Thus, it is possible to parallelize more dockings. Furthermore, the environment is not limited to execute just one experiment at a time. W-FReDoW was built to support parallel executions of different experiments. Each experiment has a unique ID with a corresponding set up, allowing a parallel execution with different treatments per experiment.

The communication between Server Controller and MTC is an important achievement. Besides providing a fast communication, here executed by SFTP and HTTP POST, the MTC should understand rules from P-SaMI, like discard, increase or decrease a lot of snapshots. Then an algorithm was developed from Renata de Paris thesis to use these rules. The heuristic is presented in her thesis, demonstrating how FReMI deals with allocation of snapshots on MTC environment.

We understand the environment created has a great contribution to RDD. Considering the flexibility of receptor on molecular docking is a subject from researches, the computational time an obstacle for scientists, we developed a smart molecular docking, through the use of a MTC and validation of promising groups of conformations.

We can say that the research question “Is it possible to use a reduction experiments pattern through the disposal of snapshots non-promising in molecular docking simulations of Fully-flexible Receptor Model keeping a high level of accuracy?” is answered positively in this thesis. W-FReDoW was designed under P-SaMI concepts reducing the number of conformations for docking through promising groups.

Based on the results of this thesis, the use of W-FReDoW really targets the reduction of the number of conformations. This achievement only happens with the use of P-SaMI, because without it, all snapshots will be docked. Then, it is possible to say that a RFFR model was achieved by W-FReDoW.

7.1 Future Work

An important contribution is the use of W-FReDoW in a different workflow area. For that, it is important that the new workflow presents same characteristics P-SaMI uses. So W-FReDoW can be validated as a multi-area environment. Also, another contribution is to create modules in W-FReDoW for those different workflows. Thus, scientists can setup each type of experiment that will be executed.

The grouping of snapshots process could be automatized. Today, W-FReDoW loads results generated from Karina Machado’s [MAC11] thesis to create groups of snapshots with similar characteristics. The Data Mining used for her thesis results in entrance data to W-FReDoW. This process is manual and could be automatized.

The most important improvements for the system will naturally be related with molecular modeling steps. Thus, the use of different molecular docking tools (today we are docking with Autodock4.2), can be a differential in the flexibility of the environment.

To finish the list of future works, an interesting area that can be explored is KDD (Knowledge Discovery and Data Mining) using FReDD for P-SaMI database. The decision to execute a new experiment could be defined from a KDD process. Today this process is manual, through the parameter changes to a new experiment.

REFERENCES

- [ALT04] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. Kepler: an extensible system for design and execution of scientific workflows, In: 16th SSDBM, p. 423- 424, Santorini, Greece, 2004.
- [AMA08] R. E. Amaro, R. Baron, and J. A. McCammon. An improved relaxed complex scheme for receptor flexibility in computer-aided drug design. *J. Comput. Aided Mol. Des.*, 22:693–705, 2008.
- [BAR09] D. Barseghian, I. Altintas, M. B. Jones, D. Crawl, N. Potter, J. Gallagher, P. Cornillon, M. Schildhauer, E. T. Borer, E. W. Seabloom, and P. R. Hosseini. Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis, *Ecol. Inform.* (2009), doi:10.1016/j.ecoinf.2009.08.008, 2010, p. 42-50.
- [CAL06] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. VisTrails: visualization meets data management, In: Proceedings of the 2006 ACM SIGMOD, p. 745-747, Chicago, IL, USA, 2006.
- [CAS07] C. T. Caskey. The Drug Development Crisis: Efficiency and Safety. *Annual Review of Medicine* 2007, 58:1-16.
- [CHE03] R. Chen, L. Li, and Z. Weng. *Proteins*, 52(1), 80-87, 2003.
- [CHE08] A. Chebotko, S. Chang, S. Lu, F. Fotouhi, and P. Yang. 2008. Scientific Workflow Provenance Querying with Security Views. In Proceedings of the 2008 The Ninth International Conference on Web-Age Information Management (WAIM '08). IEEE Computer Society, Washington, DC, USA, 349-356.
- [COU10] F. Coutinho, E. Ogasawara, D. de Oliveira, V. Braganholo, A. A. B. Lima, A. M. R. Dávila, M. Mattoso. "Data parallelism in bioinformatics workflows using Hydra". In 19th ACM international Symposium on High Performance Distributed Computing, 2010, pp. 507-515.
- [DAV07] S. Davidson, S. C. Boulakia, A. Eyal, B. Ludäscher, T. McPhillips, S. Bowers, M. K. Anand, and J. Freire. Provenance in scientific workflow systems. *IEEE Data Eng. Bull.*, 30(4):44–50, 2007.
- [DEE07] E. Deelman, G. Mehta, G. Singh, M. Su, and K. Vahi, 2007, "Pegasus: Mapping Large-Scale Workflows to Distributed Resources", *Workflows for e-Science*, Springer, p. 376-394.
- [DES95] A. Dessen, A. Quemard, J. S. Blanchard, W. R. Jacobs, J. R. Sacchettini, and J. C. Sacchettini. "Crystal structure and function of the isoniazid target of Mycobacterium tuberculosis". *Science*, vol. 267-5204, 1995, pp. 1638-1641.

- [DIA08] R. Dias and W. F. de Azevedo Jr. "Molecular Docking Algorithms", Porto Alegre, Rio Grande do Sul, Brazil, 2008.
- [DOD08] G. G. Dodson, D. P. Lane, and C. S. Verma. Molecular simulations of protein dynamics: new windows on mechanisms in biology. *EMBO Reports*, 9(2):144–150, 2008.
- [ENH11] Enhydra JaWE (Java Workflow Editor). Captured in: <http://sourceforge.net/projects/jawe/>, December of 2011.
- [ENH11a] Enhydra Shark. Captured in: <http://sourceforge.net/projects/sharkwf/>, December of 2011.
- [EWI01] T. J. Ewing, S. Makino, A. G. Skillman, and I. D. Kuntz. DOCK 4.0: search strategies for automated molecular docking of flexible molecule databases. *J Comput Aided Mol Des* 2001, 15 (5):411–428.
- [FLK11] A. Ronacher. Flask Documentation – Flask: web development, one drop at a time. Release 0.8, 2011. Captured in <http://flask.pocoo.org/docs/flask-docs.pdf>. December of 2011.
- [GLA08] T. Glatard, J. Montafnat, D. Lingrand, X. Pennec. "Flexible and efficient workflow deployment of data-intensive applications on grids with MOTEUR". *International Journal of High Performance Computing and Applications*, vol. 22-3, 2008, pp.347-360.
- [GOB07] C. Goble, D. de Roure. "myExperiment: social networking for workflow-using e-scientists". In: *WORKS*, Monterey, California, USA. <http://eprints.ecs.soton.ac.uk/15095/>, 2007.
- [GOO96] D. S. Goodsell, G. M. Morris, and A. J. Olson. Automated docking of flexible ligands: applications of AutoDock. *J. Mol. Recognit.*, 9(1), 1-5, 1996.
- [HUB10] P. N. Hübler. P-SaMI: self-adapting multiple instances – a data pattern to scientific workflows (in portuguese: P-MIA: padrão de múltiplas instâncias autoadaptáveis – um padrão de dados para workflows científicos). PhD Thesis. PPGCC – PUCRS. Porto Alegre, RS, Brasil (2010).
- [HUL06] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, 2006, Taverna: a tool for building and running workflows of services, *Nucleic Acids Research*, v. 34, n. Web Server issue, p. 729-732.
- [JAI03] A. N. Jain. Surfex: Fully Automatic Flexible Molecular Docking Using a Molecular Similarity-Based Search Engine. *J. Med. Chem.*, 46(4), 499-511, 2003.
- [JAM09] G. James. The Web Server Gateway Interface (WSGI). In: *The Definitive Guide to Pylons*. Apress, 2009, p. 369-388.
- [JIN11] A. Ronacher. Jinja2 Documentation. Release 2.7, 2011. Captured in <http://jinja.pocoo.org/docs/jinja-docs.pdf>. December of 2011.
- [JOY06] S. Joy, P. S. Nair, R. Hariharan, and M. R. Pillai. Detailed comparison of the protein-ligand docking efficiencies of GOLD, a commercial package and ArgusLab, a licensable freeware. *In Silico Biology*. 2006. 6, 601-605.
- [KRA99] B. Kramer, M. Rarey, and T. Lengauer. Evaluation of the FlexX Incremental Construction Algorithm for Protein-Ligand Docking. *PROTEINS: Structure, Function and Genetics* 37 (1999) 228-241.

- [KUN92] I. D. Kuntz. "Structure-based Strategies for Drug Design and Discovery." *Science*, vol. 257, 1992, pp.1078–1082.
- [KUU03] M. R. Kuo, H. R. Morbidoni, D. Alland, S. F. Sneddon, B. B. Gourlie, M. M. Staveski, M. Leonard, J. S. Gregory, A. D. Janjigian, C. Yee, J. M. Musser, B. Kreiswirth, H. Iwamoto, R. Perozzo, W. R. Jacobs Jr., J. C. Sacchettini, and D. A. Fidock. "Targeting tuberculosis and malaria through inhibition of Enoyl reductase: compound activity and structural data". *J. Biol. Chem.*, vol. 278, 2003, pp20851–20859.
- [LAR09] R. Larson, B. Farber. "Estatística Aplicada". São Paulo: Editora Pearson, 4ª. Edição, 2009, 476p.
- [LEN96] T. Lengauer, M. Rarey. Computational methods for biomolecular docking. *Curr. Opin. Struct. Biol.*, 6 (1996), 402-406.
- [LIU99] M. Liu and S. Wang. MCDOCK: a Monte Carlo simulation approach to the molecular docking problem. *J Comput Aided Mol Des* 1999, 13(5): 435–451.
- [LUD06] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, Y. Zhao. "Scientific Workflow Management and the Kepler System". *Concurrency and Computation: Practice & Experience*, vol.18-10, 2006, pp.1039-1065.
- [LUD09] B. Ludäscher, M. Weske, T. McPhillips, S. Bowers. "Scientific Workflows: Business as Usual?". *Lecture Notes in Computer Science*, vol. 5701, 2009, pp.31-47.
- [MAC07] K. S. Machado, E. K. Schroeder, D. D. Ruiz, and O. N. de Souza. "Automating molecular docking with explicit receptor flexibility using scientific workflows". In: *Brazilian Symposium On Bioinformatics, 2.*, 2007, Angra dos Reis. Proceedings... Heidelberg: Lecture Notes in Computer Science (4643), Set. 2007, p.1-11.
- [MAC07a] K. S. Machado. "Um Workflow Científico para a Modelagem do Processo de Desenvolvimento de Fármacos Assistido por Computador Utilizando Receptor Flexível." Dissertação (Mestrado em Ciência da Computação) – Faculdade de Informática – PUCRS, Porto Alegre, 2007, 75p.
- [MAC11] K. S. Machado. Efficient Selection of Conformations of Flexible Receptor in Molecular Docking Simulations. PhD Thesis. PPGCC – PUCRS. Porto Alegre, RS, Brazil (2011).
- [MAC11a] K. S. Machado, E. K. Schroeder, D. D. Ruiz, E. M. L. Cohen, and O. N. de Souza. Fredows: a method to automate molecular docking simulations with explicit receptor flexibility and snapshots selection. *BMC Genomics*, vol. 12, 2011, pp. 2-13.
- [MAT09] M. Mattoso, C. Werner, G. Travassos, V. Braganholo, L. Murta, E. Ogasawara, F. Oliveira, W. Martinho. "Desafios no Apoio à Composição de Experimentos Científicos em Larga Escala". In: *SEMISH - CSBC*, Bento Gonçalves, RS – Brasil, 2009, 15p.
- [MOR09] G. M. Morris, R. Huey, W. Lindstrom, M. F. Sanner, R. K. Belew, D. S. Goodsell, A. J. Olson. AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *J. Comput. Chem.* 30 (2009), 2785-2791.
- [MOR10] G. M. Morris, D. S. Goodsell, M. E. Pique, W. Lindstrom, R. Huey, S. Forli, W. E. Hart, S. Halliday, R. Belew, and A. J. Olson. AutoDock Version 4.2 User's Guide - AutoDock: Automated Docking of Flexible Ligands to Flexible Receptors. Version 4.2., 2010.
- [OLI04] J. S. Oliveira, E. H. S. Sousa, L. A. Basso, M. Palaci, R. Dietze, D. S. Santos, I. S. Moreira. "An inorganic iron complex that inhibits wild-type and an isoniazid-resistant mutant

- 2-transenoyl-ACP (CoA) reductase from *Mycobacterium tuberculosis*". *Chem. Commun.*, vol. 3, 2004, pp.312–313.
- [PIE05] B. Pierce, W. Tong, and Z. Weng. M-ZDOCK: a grid-based approach for Cn symmetric multimer docking. *Bioinformatics* 21, 1472–1478, 2005.
- [POS11] The PostgreSQL Global Development Group. PostgreSQL 9.0.6 Documentation, 2011. Captured in: <http://www.postgresql.org/docs/9.0/interactive/>, November 2011.
- [PSY11] F. Di Gregorio. Psycopg: PostgreSQL database adapter for Python. Release 2.4.4. Captured in <http://initd.org/psycopg/docs>. December of 2011.
- [PYT11] Python Software Foundation. The Python Tutorial, 2011. Captured in: <http://docs.python.org/tutorial/index.html>, December of 2011.
- [QIN07] J. Qin and T. Fahringer. "Advanced data flow support for scientific grid workflow applications". In: Proceedings of the ACM/IEEE conference on Supercomputing (SC), ACM, 2007, pp.1–12.
- [RAI07] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde. Falcon: a Fast and Lightweight task execution framework, In: 2007 ACM/IEEE conference on Supercomputing, p. 1-12, Reno, Nevada, 2007.
- [RAR96] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe. A Fast Flexible Docking Method Using an Incremental Construction Algorithm. *J. Mol. Biol.* 261, 470-489, 1996.
- [RUS04] N. Russel, A. ter Hofstede, D. Edmond, W. van der Aalst. "Workflow data patterns." Technical Report FIT-TR-2004-01, Queensland Univ. of Techn., 2004, 75p.
- [RUS06] N. Russel, A. H. M. ter Hofstede, W. M. P. van der Aalst, N. Mulyar. "Workflow Control-Flow Patterns - A Revised View". s.l. : BPM Center Report, BPM-06-22, 2006, pp.6-22.
- [SAU08] N. Sauton, D. Lagorce, B. O. Villoutreix, and M. A. Miteva. *BMC Bioinformat.*, 9, 184-196, 2008.
- [SCH05] E. K. Schroeder, L. A. Basso, D. S. Santos, and O. N. Souza. "Molecular dynamics simulation studies of the Wild-Type, I21V, and I16T Mutants of Isoniazid-Resistant *Mycobacterium tuberculosis* Enoyl Reductase (InhA) in Complex with NADH." *Toward the Understanding of NADH-InhA Different Affinities. Biophys. J.*, vol.89, 2005, pp.876-884 (2005).
- [STO93] B. L. Stoddard and D. E. Koshland Jr. Molecular Recognition Analyzed by Docking Simulations: The Aspartate Receptor and Isocitrate Dehydrogenase from *Escherichia coli*. *Proc. Natl. Acad. Sci.*, 90 (1993), 1146-115.
- [TAY07] I. Taylor, M. Shields, I. Wang, and A. Harrison, 2007, "The Triana Workflow Environment: Architecture and Applications", *Workflows for e-Science*, Springer, p. 320-339.
- [TOT08] M. Totrov and R. Abagyan. Flexible ligand docking to multiple receptor conformations: a practical alternative. *Current Opin. Struct. Biol.*, 18(2):178–184, 2008.
- [VER03] M. L. Verdonk, J. C. Cole, M. J. Hartshorn, C. W. Murray, and R. D. Taylor. Improved protein-ligand docking using GOLD. *Proteins*, 52(4), 609-623, 2003.
- [WER11] A. Ronacher. Werkzeug: The Python WSGI Utility Library. Release 0.9, 2011. Captured in <http://werkzeug.pocoo.org/docs>. December of 2011.

- [WIN09] A. T. Winck, K. Machado, O. N. Souza, and D. D. Ruiz. "FReDD: supporting mining strategies through a flexible-receptor docking database". In: IV Brazilian Symposium on Bioinformatics, 2009, Porto Alegre. *Advances in Bioinformatics and Computational Biology*. Berlin : Springer, 2009, pp.143-146.
- [WIN10] A. T. Winck, K. S. Machado, O. N. de Souza, and D. D. Ruiz. "Supporting Intermolecular Interaction Analyses of Flexible-Receptor Docking Simulations". In: IADIS International Conference Applied Computing, 2010, Timisoara. *Applied Computing*, 2010, p. 183-190.
- [WOR99] WORKFLOW Management Coalition. "The Workflow Management Coalition Specification – Terminology & Glossary". Document Number WFMC-TC-1011, Feb 1999, 65 p.
- [YIL09] U. Yildiz, A. Guabtini, and A. H. H. Ngu. "Business versus scientific workflow: A comparative study," University of California, Tech. Rep. 3, 2009.
- [YUJ05] J. Yu and R. Buyya. A Taxonomy of Workflow Management Systems for Grid Computing, *Journal of Grid Computing*, v. 34, n. 3-4, p. 171-200, 2005.
- [YUR10] E. Yuriev, M. Agostino, and P. A. Ramsland. Challenges and advances in computational docking: 2009 in review. *J. Mol. Recognit.*, 2011, 24(2), 149-164.
- [ZHA08] S. Zhang, K. Kumar, X. Jiang, A. Wallqvist, J. Reifman. DOVIS: an implementation for high-throughput virtual screening using Autodock. *BMC Bioinformatics* 9 (2008), 126.