Partition of Online Social Networks using Simulations based on Chainable Processing Modules

André F. de Angelis, Guilherme Oliani Neto, Rebecca M. Alonso, and Rebeca P. Paschoaletto[1]

University of Campinas
School of Technology, Limeira, SP, Brazil
Concrete Mathematics Laboratory

Received on October 19, 2016 / accepted on November 30, 2016

Abstract

Working with Complex Networks simulations focused on Online Social Networks (OSNs), we reached the limits of the architectural features of our software tool, because its project is not suitable to the inclusion of new resources as dynamic topological operations, on-the-fly visualizations, recognition of yet-to-be-developed algorithms, and concurrency programming support. We analyzed our current simulator and prepared a requirement list targeting to the development of a new one, privileging two features: flexibility and extensibility. We developed and applied the concept of independent processing modules that can form a processing chain according to Design Patterns and Java interface standards. These *Chainable Processing Modules* are the novelty and foundation of our architecture, that complies very well with the new simulator requirements. In this work, we present our software architecture and highlight its potential to help the development of open projects of simulators of any type of Complex Networks. Furthermore, we have implemented part of the proposed architecture and shown that new researches were fast carried out because of the reached flexibility.

Keywords: Simulation, Social Network Analyses, Software Architecture, Computational Data Analysis and Simulation in General Sciences.

1. Introduction

Computational simulation is an important tool for the study of complex networks in general [1, 2], and therefore it has a highlighted role to understand Online Social Networks (OSNs). These networks have been intensely

[1]E-mail Corresponding Author: andre@ft.unicamp.br

researched in recent years, given their importance in today's world, as one can find in the following examples: [3, 4, 5, 6].

While we were working on a special case of partition of networks [7], we developed a piece of software that simulates the operation of common OSNs, named Demortuos[8]. Its main goal was the assessment of candidate algorithms of partitioning and it has an iterative graphical user interface, as exemplified in Fig. 1.

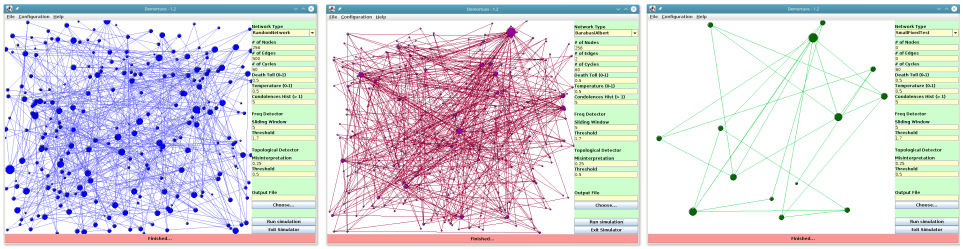

Figure 1: Three screenshots of *Dermortuos* showing, from left to rigth, a random network with 128 nodes (average degree 2), a BA network with 128 nodes (average degree 2), and a small hand made test network with 14 nodes (average degree about 3).

Briefly, the program sequentially generates one of the possible realizations of the network according to a user-selected model, simulates a number of message exchanges and operation days, and applies the algorithms under evaluation, collecting data for their posterior analyses. It had innitialy two available models: i) the Random Graph model after [9] (RN model); ii) the Scale-free model after [10] (BA model). Currently, there are additional models available. Demortuos has built in Java, following the Design Patterns [11]. We specially used Strategy and Factory to improve its flexibility and assesses three algorithms of partitioning. Because of its sequential processing, practical simulations are restricted to about 50,000 nodes per network.

Although the program performs very well its job, it arrived the time when some new research demands have arouse, bringing the software to its limits. For instance, currently investigations need more sophisticated models that include community structures as it is supposed to change the network dynamics. A review on this subject is found in [12]. Also, rewiring schemes need to be considered in the investigations because OSNs undergo changes in their links. Some considerations about rewiring process are found in [13, 14, 15]. Furthermore, we foresee new models for on-fly topology changes

Table 1: New architecture requirements

#	Requirement	Type
01	**Very flexible general-purpose simulator for OSNs**	mandatory
02	Automated simulation of several scenarios	mandatory
03	Compliance with Design Patterns standards	mandatory
04	Ability to deal with static and dynamic network models	mandatory
05	Acceptance of new network models	mandatory
06	Acceptance of new network operation patterns	mandatory
07	Recognition of yet-to-be-developed assessment algorithms	mandatory
08	Integrated simulation of topology evolution and network operation	mandatory
09	Entry points to pre- and post-processing algorithms	mandatory
10	Entry points to filters and general-purpose algorithms	mandatory
11	Entry points to on-the-fly data collection and storage	mandatory
12	Entry points to on-the-fly visualization	mandatory
13	Multithread support	mandatory
14	Compliance with Java 8 standards	optional
15	Compliance with Javadoc standards	optional
16	Interface with third-part programs	optional
17	Iterative and batch operation modes	optional
18	Parallel processing support	optional

that must be researched.

Therefore, we started the project of a new piece of software, for now code-named DemortuosNG, that is intended to completely replace the previous tool. We target a full OSN simulator, not only a tester for partitioning algorithms. So, the requirements undergone a significant change, as well as the software architecture. In this paper, we present the proposed software architecture as it brings a new approach to the design of this type of simulator. A point to be noticed is the chainable processing modules. These modules permit that an arbitrary number of processing steps be combined in a free and extensible fashion, pushing the flexibility of the simulator to the next level. In the following section, we present the methodology for the design of this new architecture.

2. Architecture

The objective of the new program is to produce a general-purpose OSN simulator, because we look for a flexible tool, able to process several network topology models under a diversity of operation patterns, algorithms,

and assessment procedures. Iterative and batch mode are desired features, as well the support to execution threads and concurrency[2]. Consequently, we started with a requirement analyses that leaded us towards a list of mandatory and optional features to the new simulator, as summarized in Table 1.

From the requirements, we overlooked a whole new design to the simulator. Its main feature is the use of a set of *chainable modules* inspired by the Chain of Responsibility Design Pattern[11]. These modules are intended to implement a common interface in order to be fully interchangeable. This design provides the simulator with entry points for a yet-to-be-developed set of network models, filters, and algorithms, as stated in the requirements of the system.

The proposed architecture for DemortuosNG is shown in Fig. 2. Each box indicates a module or a set of modules, whereas the arrows define the main expected information flow. The *Control* module makes the overall coordination of the system and exchanges pieces of information with virtually all other modules, but we omit its arrows to get a clean drawing. We highlight that there is a relation among the modules and actual classes or interfaces, but this is not a one-to-one map. Some modules may need several classes to their implementation, whereas two modules may be jointed in one class according to implementation decisions.

There are three modules groups: core simulation, input, and output. They are intended to organize the architecture. The first one guides the program concurrency, whereas the other two impose the production of a set of interfaces to accommodate interchangeable modules, including the adapters to communicate with other applications. In Fig. 2, the *Chainable Processing* module is only a placeholder to the actual ones. Next, we present the functionality of each module.

Input Group: all the modules of this group must implement a common interface in order to be interchangeable; they are responsible to get configuration data such as simulation scenarios from diverse sources, as follows:

- Storage: from plain text files or database systems;
- GUI: from an interactive graphical user interface;
- Batch: from command line arguments or scripts files;
- Adapter: from a third-part application.

Output Group: all the modules of this group must implement a common

[2]Here, we adopt the [16]'s approach where threads optimize the time of one single CPU, whereas the concurrency concerns to a multicore/multi-CPU computer.

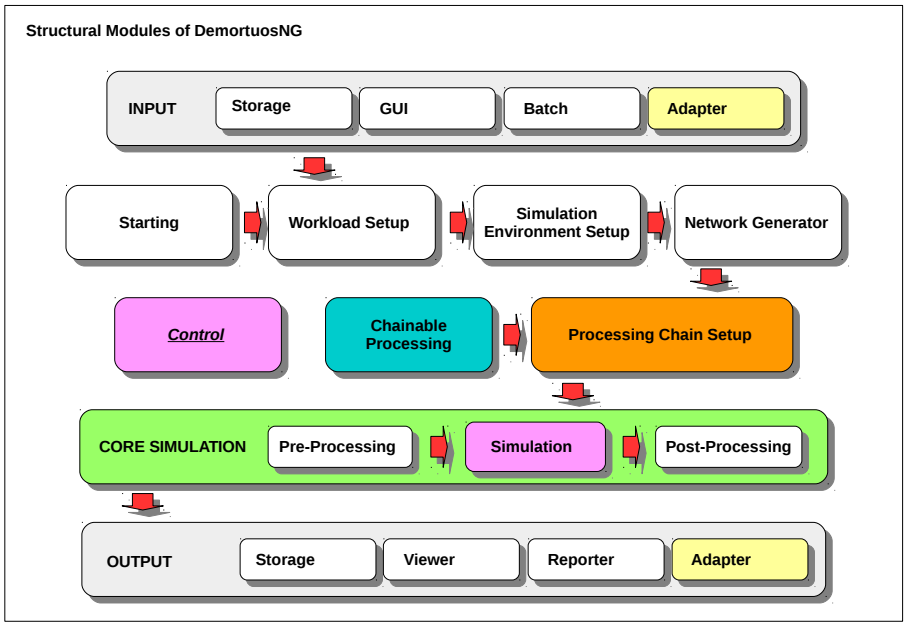

Figure 2: Overview of proposed architecture to OSNs simulators, presenting the structural modules; *Chainable Processing* is a placeholder to the actual modules; as *Contol* exchanges data with all modules, its arrows were omitted.

interface in order to be interchangeable; they are responsible to collect data from the simulation process and present or save it to a destination as follows:

- Storage: to plain text files or database systems;
- Viewer: to an interactive graphical user interface;
- Reporter: to a structured high-level document;
- Adapter: to a third-part application.

Core Simulation Group: the modules of this group execute the processing steps of the simulation.

- Pre-processing: the one-time run procedures previous to the simulation, if any;
- Simulation: the simulation processing itself, according to the sequenced chain of processing modules;

- Post-processing: the one-time run procedures posterior to the simulation, if any.

Loose modules: they have different roles as follows.

- Starting: system initialization tasks only concerned to the program;
- Workload Setup: load and configuration of the scenarios to be simulated;
- Simulation Environment Setup: setup of the run conditions, concurrency mechanism, and correlated tasks;
- Network Generator: creation of one network realization according to the chosen workload;
- Control: overall coordination of the system;
- Processing Chain Setup: creation of a sequence of chainable processing modules to be submitted to the Core Simulation;
- Chainable Processing: a placeholder for the actual modules (Fig. 3).

The Chainable Processing Modules are processing elements that may be arranged in a chain sequence in order that the same object can be inspected and processed by all of them. They are the key point of our architecture, because they can include almost all types of algorithms in the simulator in a standardized way. One network object pass through the chain in each iteration undergoing different processing routines. The foreseen modules are shown in Fig. 3 as follows.

- Filters: data selection, transformation, and conversion;
- General Algorithms: all-purpose processing;
- Internal Verification: integrity check for the simulation;
- Network: structural processing over network, as follows:
  - Characterization: topological evaluation [3];
  - Topology Dynamics: topological changes, rewiring;
- Assessment Algorithms: assessment of researches subjects;
- Operation: OSNs dynamics, as follows:
  - Network Dynamics: non-topological changes;
  - User Dynamics: user behavior or status changes;
  - Message Dynamics: message exchange simulation.

Next, we discuss how the proposed architecture complies with the requirements.

[3] See [17] for further details about characterization of complex networks.

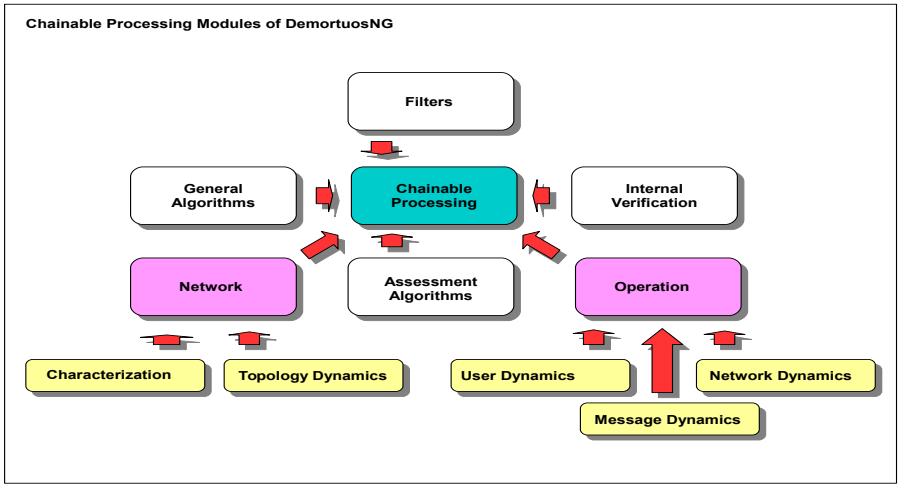

Figure 3: Overview of the *Chainable Modules* of the proposed architecture.

- Very flexible general-purpose simulator for OSNs: the architecture was designed to perform all-purpose simulations of OSNs by means of the chainable modules that provide the desired level of flexibility; the input module permits a series of different sources for the configuration of the simulator, whereas the output modules are intended to be extensible classes, able to provide several forms to show the processing results, even for the intermediate ones.
- Automated simulation of several scenarios: the architecture was planned to support multiple instances of the core simulation module; it is intended to automatically run several scenarios without user intervention.
- Compliance with *Design Patterns*, Java 8, and Javadoc standards: although the compliance is deeply related to the detailed project and codification of Java classes, the proposed architecture fully supports these standards.
- Ability to deal with static and dynamic network models: the independence among the network generator modules and the simulation core allows the system works with the both types of networks.
- Acceptance of new network models and operation patterns: the use of independent extensible modules for generation, evolution, and simula-

tion of networks provides this compliance.

- Recognition of yet-to-be-developed filters, general-purpose and assessment algorithms: the chainable modules may be extended to implement a new functionality; thus, it is possible to include new code in a standardized way as soon it becomes available.
- Integrated simulation of topology evolution and network operation: the architecture does not impose the separation between these two tasks as in the previous software tool, so it is possible to integrate them according to the user needs.
- Entry points to pre- and post-processing algorithms: they are located in the Core Simulation group, as specific processing objects.
- Entry points to on-the-fly data collection, visualization, and storage: the chainable extensible output modules provide support for these required operations.
- Multithread and parallel processing support: the core simulation module may be parallelized to simultaneously run different scenarios and/or repetitions related to a configuration set of parameters; the actual implementation will choose the most suitable technique.
- Interface with third-part programs: there is a prevision for Adapters in both input and output modules, then it is possible to write interface code to other programs.
- Iterative and batch operation modes: this feature is supported by the independence between the input modules and the processing ones.

3. Results

We have started the development of the new tool using the proposed architecture. As we want the benefits of this approach as soon as possible, we are still using extensive portions of the old software code to support the inclusion of the new features, towards a complete replacement process.

Thus, we redesigned some classes from scratch according with the proposed architecture and we built a set of new classes that implement additional network models. The first one was that presented by [18], henceforth SZW model. The SZW model generates scale-free networks with community structure, starting with an arbitrary number of initial high-connected small networks. Because the originating configuration of these initial networks is not specified by the model, it was possible to implement several

Table 2: OSNSim Standard - Part I - 2016

Parameter	Scenario 01	Scenario 02	Scenario 03	**Scenario 04**	Scenario 05	Scenario 06
Model	RN; SF	RN; SF	RN; SF	RN; **SF**	RN; SF	RN; SF
Order	5K	5K	5K	**20K**	20K	50K
Average node degree	10; 100	10; 100	500	**100**	500	500
Cycles	30; 90; 360	30; 90; 360	30; 90; 360	30; 90; **360**	30; 90; 360	30; 90; 360
Mortality rate	0.050; 0.500	0.005; 0.050	0.005; 0.050	**0.005; 0.050**	0.005; 0.050	0.005; 0.050
Inactivity rate	0.50	0.05	0.05; 0.20	0.05; 0.20	0.05; 0.20	0.05; 0.20
Temperature	0.0; 0.5; 1.0	0.0; 0.5; 1.0	0.0; 0.5; 1.0	**0.0; 0.5; 1.0**	0.0; 0.5; 1.0	0.0; 0.5; 1.0
Combinations	72	72	72	72	72	72

variations in the subject. For instance, the initial networks can be random, scale-free, hand-made, etc. In the Fig. 4 we show one realization of this model, compared to an BA network with equal dimensions.

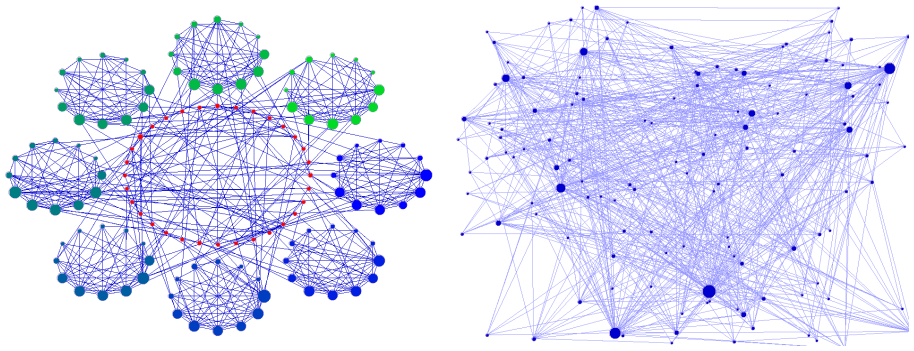

Figure 4: Comparison between two networks of the same order and size. The one at left is an SZW uncorrelated scale-free with communities, based in the proposal of [18] – the picture emphasizes the initial communities that are BA networks; the one at right was a BA newtwork generated acording to the [10]'s model.

As supposed, the presence of communities in the SF network changed the simulation results. Preliminary data indicates that the efficiency of the partition algorithms has an improvement, as exemplified by the simulation of a subset of the Scenario 4 of OSNSim standard [19], highlighted in Table 2.

In Table 3 it is possible to notice the improvement of efficiency, ranging from 0.067 to 0.268. The higher Temperature, the higher the volume of messages in the network. So, the algorithm performs better when the Temperature parameter is high, independently of the network model.

Table 3: Efficiency of a partition algorithm for OSNs based in message frequency over BA (without communities) and SZW (with communities) network realizations. The both networks have 20,000 nodes and average degree 100, according to Scenario 4 of OSNSim standard.

Mortality Rate	**Temperature**	**BA Model**	**SZW Model**	**Abs Diff**
0.005	0.0	0.062	0.129	0.067
0.005	0.5	0.151	0.282	0.131
0.005	1.0	0.290	0.558	0.268
0.050	0.0	0.092	0.147	0.055
0.050	0.5	0.173	0.302	0.129
0.050	1.0	0.285	0.540	0.255

4. Conclusion

We proposed a software architecture to OSNs simulators, based in our previous experience with this type of program. As our goal is to produce a general-purpose OSN simulator, we prepared a list of ambitious requirements for the software development, which are as general as possible, but they presuppose an actual object oriented implementation. Although we are going to make our code using Java, the architecture is suitable for C++ as well.

We presented a new, modular, flexible, and extensible software architecture that has chainable processing modules as its strongest novelty. These modules impose a particular model of operation to the simulator, that supports the inclusion of a yet-to-be-developed functionality set. Thus, our architecture may be used to create open OSNs simulators and to inspire the design of other simulators.

Currently, we are validating the proposed architecture, starting the project of the actual Java classes, and implementing the network generating classes. Initial simulations has used extensive portions of the previous version of Demortuos to allow early tests.

As we already have implemented additional models, we showed that the community structure within a scale-free network changes the behavior of algorithms in that network. We exemplified this change presenting the efficiency increase of partitioning algorithms designed to OSNs. Our actual goal is to finish the new software according to our proposed architecture to improve largely our OSNs analysis capacity.

References

[1] NEWMAN, M.E.J., The structure and function of complex networks. 2003. SIAM Rev., 45(2), 167256. 90p. DOI:10.1137/S003614450342480.

[2] BOCCALETTI, S.; LATORA, V.; MORENO, Y.; CHAVEZ, M.; HWANG, D.U, Complex Networks: Structure and Dynamics. Physics Reports. 424(4-5), 175-308. 2006. http://dx.doi.org/10.1016/j.physrep.2005.10.009.

[3] Chen, L.; Gable, G.G.; Hu, H. Communication and organizational social networks: A simulation model. Comput Math Organ Theory (2013) 19: 460. doi:10.1007/s10588-012-9131-0.

[4] DUNBAR, R.I.M.; ARNABOLDI, V.; CONTI, M.; PASSARELLA, A. The structure of online social networks mirrors those in the offline world. Social Networks. 43. 3947. 2015. http://dx.doi.org/10.1016/j.socnet.2015.04.005.

[5] VISWANATH, B.; MISLOVE, A.; CHA, M.; GUMMADI, K.P., On the evolution of user interaction in Facebook. Proceedings of the 2Nd ACM Workshop on Online Social Networks - WOSN '09. 37-42p. 2009. http://doi.acm.org/10.1145/1592665.1592675.

[6] McCALLIG, D., Facebook after death: an evolving policy in a social network. Int J Law Info Tech (2014) 22 (2): 107-140. DOI:https://doi.org/10.1093/ijlit/eat012.

[7] LIBARDI, P.L.O. Detecção Computacional de Falecidos em Redes Sociais Online. (Dissertação de Mestrado) Faculdade de Tecnologia. Unicamp. Jan. 2015. 75p.

[8] Revista da Propriedade Industrial 2355. 02.07.15 BR 51 2015 000664 9 DEMORTUOS André Franceschi de Angelis e Paula Luciene Oliveira Libardi

[9] ERDÖS, P.; RÉNYI, A., On random graphs. Publ. Math. Debrecen. 6 1959. 7p.

[10] BARABÁSI, A.L., ALBERT, R., Emergence of Scaling in Random Networks. Science. 1999. 4p. DOI: 10.1126/science.286.5439.509.

[11] ERICH GAMMA, RICHARD HELM, RALPH JOHNSON, JOHN VLISSIDES. Design Patterns: Elements of Reusable Object-Oriented Software. 1st Edition. Addison-Wesley Professional. 1994.

[12] SALLABERRY, A.; ZAIDI, F.; MELANCON, G. Model for generating artificial social networks having community structures with small-world and scale-free properties. Soc. Netw. Anal. Min. (2013) 3: 597. doi:10.1007/s13278-013-0105-0.

[13] QU, J.; WANG, S.J.; JUSUP, M.; WANG, Z. Effects of random rewiring on the degree correlation of scale-free networks. Scientific Reports.(2015) 5:15450.

[14] SUN, H.J.; ZHANG, H.; WU, J.J. Correlated scale-free network with community: modeling and transportation dynamics. J. Nonlinear Dyn (2012) 69: 2097. doi:10.1007/s11071-012-0411-5.

[15] MANNA, S. S.; KABAKCIOĞLU, A. Scale-free network on Euclidean space optimized by rewiring of links. Journal of Physics A: Mathematical and General 36.19 (2003): L279.

[16] SCHILDT, H. Java The Complete Reference. 9th ed. Oracle Press. 2014.

[17] COSTA, L.F.; RODRIGUES, F.A.; TRAVIESO, G.; VILLAS-BOAS, P.R., Characterization of Complex Networks: A Survey of measurements. Advances in physics 56.1 (2007): 167-242.

[18] Sun, H.; Zhang, H.; Wu, J. Nonlinear Dyn (2012) 69: 2097. doi:10.1007/s11071-012-0411-5.

[19] ALONSO, R.M.; LIBARDI, P. L.; ANGELIS, A.F. The OSNSim Standardization Proposal for Input Parameters in Online Social Networks Simulations. In: 4th Conference of Computational Interdisciplinary Science (CCIS 2016), 2016, So Jos dos Campos/SP. CCIS 2016 Proceedings. So Jos dos Campos/SP: epacis, 2016. v. 1. p. 1-8.