A metamodel for distributed multimedia processing systems development

Alisson Augusto Souza Sol[1]
Advisor: Arnaldo de Albuquerque Araújo[2]

DCC - Departamento de Ciência da Computação
UFMG - Universidade Federal de Minas Gerais
Caixa Postal 702 - 30.161-970, Belo Horizonte, MG, Brasil
http://www.npdi.dcc.ufmg.br
[1]sol@acm.org, [2]arnaldo@dcc.ufmg.br

1 Introduction

Simultaneous use of several media forms (text, image, voice, animation, etc.) is now very usual, with almost every page of the World-Wide Web being an example of a multimedia production [1]. The Web also demonstrates several features of a distributed file system (see Satyanarayan in [2]).

In contrast to the already universally accepted methods for identification and transmission of files in the Web, it is not yet well explored the potential of the Internet as a platform for distributed processing. This kind of expectation is being driven by increasing activity toward development of applications for Intranets, from those locally based to private networks with traffic over the Internet. Besides scalability issues [3], development of distributed multimedia processing systems is made complex by the fact that not only intrinsic problems of multimedia processing and distributed computation must be solved, but also new collateral effects that solutions in one domain generate at the other.

2 Problem scope

The main intrinsic difficulties of multimedia processing are multiplicity of formats for data representation and use of available APIs (Application Programming Interfaces) for design and development of UI (User Interface). Even for a single text file without any font styles (bold, italic, subscript, etc.) more than one code for character representation can be used (ASCII, EBCDIC, Unicode, etc.). This problem scales to large extents when the great number of data formats for several media types is considered [4] and can not be ignored in real applications.

Use of frameworks is the principal current pattern used to deal with UI implementation. Experiences with libraries of functions and classes determined that there is a great gain of time in design and development of new systems when emphasis is placed on reuse of previously developed and tested code [5,6,7]. The gain is larger if no adaptation is needed in interfaces of components. Interfaces, of classes or subsystems, are groups of functions implemented to accomplish a requested functionality. Signatures of class methods and functions in libraries specify types and semantics of input and output parameters.

The main inherent difficulties for development of systems with distributed processing are partitioning of computational load between workstations and detection of flaws on communication links or computation servers, with the due recovery. Most new distributed systems, developed to scale from small LANs to the Internet, are based on asynchronous or partially synchronous models, where the algorithms are designed as several logical segments to answer the possible messages received through links among processors (Input/Output automata) [8]. The basic issue for the distributed design phase is:

- How to distribute workload, in such way that maximum speedup is obtained in executing a distributed version of an algorithm (measured against a sequential version)?

When going to the implementation phase, programmers often raise the following practical questions:

- How to discover available servers and start execution of services in each of them?
- How to transmit information in a heterogeneous environment, where each processing server can use a different representation for numbers, strings and other data types?
- How to detect flaws, of processors or communication links, and start recovery procedures?

Difficulties of dealing with both domains

Distributed execution brings additional difficulties to multimedia processing because, besides the always present issue of portability for multimedia algorithms code, it arises the requirement of UI code portability. The most successful solution to this issue with available technologies is the development of software components implemented to run on virtual machines, like the Java Virtual Machine [9,10,11].

On the other side, multimedia data bring additional complexity to workload division in a distributed processing environment. A natural and easily implemented strategy to workload division is the equal partitioning of input data among available processors. For the Sieve of Eratosthenes, a classic prime-finding algorithm, the task of finding prime numbers below some positive integer $\boldsymbol{n}$ can be distributed among $\boldsymbol{p}$ processors by assigning to each processor the task of marking non-prime numbers in a range with size $\boldsymbol{n}/\boldsymbol{p}$. The distributed version of the algorithm will show great speedup over the sequential version, until some limits are reached for the number of processing hosts [12]. The same strategy of equal input partitioning can be used to drive distributed execution of some multimedia algorithms. However, due to the nature of the multimedia data representation, it is very usual that useful chunks of information can not be obtained at equal partitions of input data [4].

3 The metamodel contribution

There are two main categories for frameworks: *horizontal* and *vertical*. The more general *horizontal* framework will have small portions of its content used for relatively small portions of an application. However, several kinds of applications could benefit from the such framework, probably from different portions. *Vertical* frameworks have very specific targets, solving a less general problem though a set of classes and mechanisms that must be reused as a whole, representing a great portion of the final application. Of course, only a small set of applications can benefit from each vertical framework.

Several class libraries and frameworks are available to ease the work of designing and implementing algorithms, with some emphasizing multimedia processing, others dealing with distributed execution and many already approaching both technologies. However, amid the current hundreds of options, and perhaps because of that, many developers do not benefit from available frameworks. Most algorithms still debut in sequential versions and not as components but as isolated applications, conceived without use of any techniques from Software Engineering. That code will be reworked many times, for different operating environments, languages or frameworks.

An usual naive expectation is that developers would someday adopt a single framework, creating an omnipresent system for distributed multimedia processing. Such fact is very improbable and it seems there will always be several operating environments, network protocols, frameworks, versions of algorithms, programming languages, etc. Furthermore, a single choice is not needed and probably not the best answer to ease and promote reuse of requirements, design or code. In addition, both new frameworks and algorithms are needed to fill current gaps in available packages to final users or researchers, providing support for yet neglected operating systems, languages, etc.

What can be reasonable to expect is that developers will increase their use of frameworks and any other techniques that have been used in successful projects, given the growing competitiveness in software manufacturing, demanding rapid development, attendance of increasingly complex requisites, need of improved quality of the final product, etc. [13] As complexity grows for design and implementation of frameworks or software components, a **metamodel** can contribute to ease understanding of how parts fit in the whole scenario. Like an architectural model [14] with options for the components and connections, a metamodel can be made "active" by addition of scripting code to drive the user through the design process. For some domains, results that are more formal are available [15].

The vertical metamodel

It was designed a vertical metamodel to ease development of distributed multimedia processing systems. The basic architectural assumption is that developers of those systems look for the following services to be reused, from function libraries, class libraries or frameworks:

- **media services**: support for several media types (audio, video, text, etc.) and data formats (JPEG, MPEG, WAV, AU, etc.) [1, 5];
- **UI services**: UI basic primitives and some specific multimedia requirements (continuous media streams, real time I/O, etc.) [1];
- **communication services**: functionality to server discovery and activation, remote function invocation, language binding and network transparency for primitive data types, along with error detection and recovery mechanisms [2, 3, 11];
- **execution services**: guidance on how to add new algorithms to the system, which interfaces must be implemented or adapted, what execution strategies are available and how can the algorithm collaborate with the framework to make sure that it is not started to execute with unacceptable input or execution strategy [5, 7, 8];
- **secondary services**: implementation or use of frameworks, class or function libraries can bring new and non-trivial requisites for a developer, like security issues in data transmission, licensing policy support for software components, transactions services, etc. [11, 16] Those topics can be ignored at first, avoiding digression in initial development, but can not be averted in final versions of any modern commercial system.

4 Current state and future work

Using a CASE tool [17], several design patterns have been represented using the UML (Unified Modeling Language) notation [18, 19], along with interfaces of some frameworks for distributed and multimedia processing, like CORBA, DCOM, MET++, etc. [5, 16, 20]. Guidance for new development have been implemented using the scripting language for the chosen tool. Using the design assistants, developers become aware of and objectively evaluate more choices, while at the same time producing a model that is more complete, correct and consistent. From previously available scripts of the CASE tool, code skeleton can be generated for some languages, like BASIC, C++ or Java. Using the metamodel, distributed versions of a previously developed framework for multimedia processing are being built [21].

The main goal of future work is the use of metrics to measure the productivity obtained by use of the metamodel.

5 References

1. J. F. K. Buford [Editor], *Multimedia Systems,* Addison-Wesley, 1994
2. S. Mullender [Editor], *Distributed Systems, 2nd Ed.*, Addison-Wesley, 1993
3. M. Korkea-aho, *Scalability in Distributed Multimedia Systems,* Master's thesis, Laboratory of Information Processing Science, Helsinki University of Technology, 1995
4. T. Kientzle, *Internet File Formats,* Coriolis Group, 1995
5. P. Ackermann, *Developing Object-Oriented Multimedia Software*, dpunkt Verlag, 1996
6. *Platform Independent FAQ*, http://www.zeta.org.au/~rosko/pigui.htm
7. A. A. S. Sol and A. de A. Araújo, *PhotoPix: an Object-Oriented Framework for Digital Image Processing Systems,* Lecture Notes in Computer Science; Vol. 974 - Image Analysis and Processing, pp. 109-114, Springer-Verlag, 1995
8. N. A. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers, Inc., 1996
9. D. Lea, *Concurrent Programming in Java - Design Principles and Patterns*, Addison-Wesley Publishing Co., 1997
10. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, *PVM - Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing,* MIT Press, 1996
11. R. Orfali and D. Harkey, *Client/Server Programming with Java and CORBA*, John Wiley & Sons, 1997
12. M. J. Quinn, *Parallel Computing: Theory and Practice*, McGraw-Hill, Inc., 1994
13. S. McConnell, *Rapid Development*, Microsoft Press, 1996
14. J D. McGregor, The fifty-foot look at analysis and design models, *The Journal of Object-Oriented Programming,* Vol. 11, No. 4, pp. 10-15, 1998
15. W. R. Mallgren, *Formal Specification of Interactive Graphics Programming Languages,* The MIT Press, 1983
16. T. J. Mowbray and R. C. Malveau, *CORBA Design Patterns*, John Wiley & Sons, Inc., 1997
17. Rational Software Corporation, *Rational Rose 98 manuals*, 1998
18. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns - Elements of Reusable Object-Oriented Software,* Addison-Wesley Publishing Company, 1995
19. M. Fowler and K. Scott, *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley, 1997
20. D. Box, *Essential DCOM*, Addison-Wesley, Inc., 1998
21. A. C. R. de Almeida, A. A. S. Sol, A. de A. Araújo, *PhotoPixJ: Plataforma em Java para Implementação de Algoritmos de Processamento Digital de Imagens*, Accept for publication at the XII SBES - Brazilian Symposium of Software Engineering, 1998, In portuguese.