Architectural and Design Patterns in Multimedia Streaming Software

Article · October 2009

CITATION
1

READS
1,840

2 authors:

Yanja Dajsuren
Centrum Wiskunde & Informatica
45 PUBLICATIONS **237** CITATIONS
SEE PROFILE

M. G. J. van den Brand
Eindhoven University of Technology
232 PUBLICATIONS **3,905** CITATIONS
SEE PROFILE

Architectural and Design Patterns in Multimedia Streaming Software

Yanja Dajsuren
Dept. of SoC Architectures and Infrastructure
NXP Semiconductors
Eindhoven, the Netherlands
yanja.dajsuren@nxp.com

Mark van den Brand
Dept. of Mathematics and Computer Science
Eindhoven University of Technology
Eindhoven, the Netherlands
m.g.j.v.d.brand@tue.nl

***Abstract*—Software developers typically build streaming applications using multimedia streaming frameworks like DirectShow, GStreamer, and Symbian MMF. Although a significant amount of work has been done on architectural and design patterns in software engineering, there is a limited notion of patterns in the development of multimedia streaming software. This article explores architectural and design patterns in the field of multimedia streaming software to facilitate the understanding process of multimedia frameworks and development of streaming applications.**

Keywords-streaming pattern; multimedia framework; streaming application;

I. INTRODUCTION

Since it is costly and time-consuming to build multimedia streaming software from scratch, multimedia frameworks enable streaming applications to be assembled by integrating pluggable media components. In addition, multimedia frameworks isolate applications from a variety of complex tasks such as handling of the complex multimedia acceleration hardware, data transport, and synchronization between various tasks.

Multimedia frameworks are available on different operating systems e.g. Windows supports DirectShow [1], Linux provides GStreamer [2] and Symbian enables Multi-Media Framework (MMF) [3]. Since frameworks are concrete realizations of groups of patterns that enable reuse of code [4] and there is a limited notion of architectural and design patterns in the multimedia streaming software, it is essential to identify patterns used in the multimedia streaming software.

In this article, we present design patterns that are based on the GStreamer, DirectShow, and Symbian MMF multimedia frameworks. We use class diagram and streaming notations of the UML 2.0 [5] to illustrate the patterns.

II. MULTIMEDIA STREAMING SOFTWARE

One of the broadly recognized approaches in the development of the multimedia streaming software is to structure the streaming software as Pipes and Filters. The Pipes and Filters architectural pattern [6] divides a complex functionality into several sequential processing sub-functionalities forming a streaming graph as shown in Fig 1. The nodes of the graph are the media components that process the data. The output of one media component can be used as input for another media component. The edges of the graph are (mostly) data buffers that establish connections between the media components.

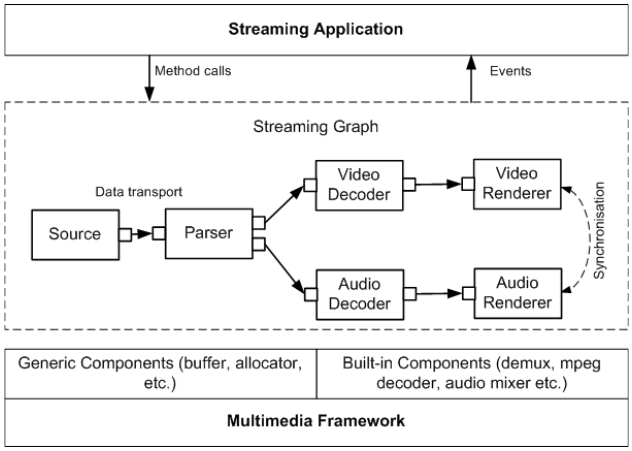

Figure 1. Overview of multimedia streaming software

We summarize below a list of main tasks that are explored from the GStreamer, DirectShow, and Symbian MMF multimedia frameworks:

- **Building a streaming graph.** Every streaming application starts by building a streaming graph mostly by instantiating media components and connecting them using the functions provided by a multimedia framework.
- **Streaming data in the graph.** Primitives for moving media data through the streaming graph are usually provided by the framework designers.
- **Responding to events.** Besides facilitating mechanisms for an interaction between application and streaming graph such as seeking to a position in a media file, there is also an event handling needed between media components such as End of Stream.

Multimedia frameworks enable developers to build custom media components by providing specific APIs or a set of base classes that provide the developer with a default implementation for certain tasks.

III. DESIGN PATTERNS

We present three composite patterns as depicted in the directed acyclic graph of Fig 2.

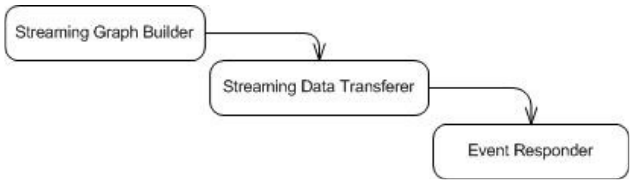

Figure 2. Graph of streaming design patterns

A. *Streaming Graph Builder*

Complex streaming software is time consuming and cumbersome to be developed from scratch or procedural way. The Streaming Graph Builder pattern is similar to the Builder Pattern. Its intention is to abstract steps of construction of a specific streaming graph so that different implementations of these steps can construct different type of streaming graphs (e.g. video capturing graph). The Streaming Graph Builder pattern is depicted in Fig 3.

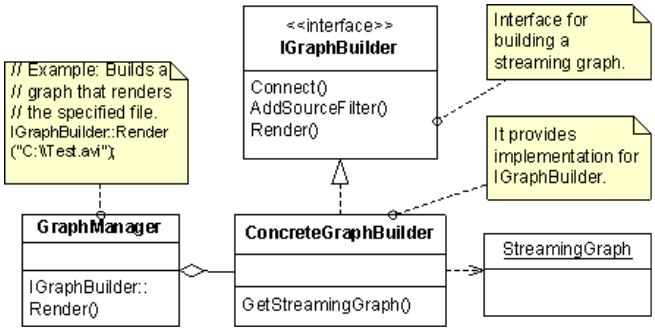

Figure 3. Streaming Graph Builder pattern

As a result, it can cope with many variations in building a streaming graph and enable clients to treat media processing components constituting the streaming graph uniformly.

B. *Streaming Data Transferer Pattern*

The Streaming Data Transferer pattern consists of Transport Data, State Transition, and A/V Sync patterns. The Transport Data pattern is elaborated in Fig 4. It enables moving media data through the streaming graph using common data transfer protocol and mechanisms like Media Data pool.

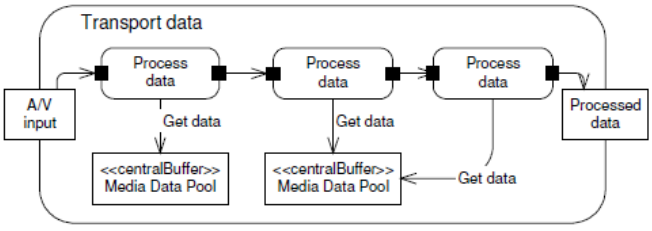

Figure 4. Transport Data pattern

A streaming application, which uses general mechanisms to transfer data through the graph independently of the media format. State changes are handled consistently between application and media component or streaming graph. A/V synchronization is handled overall.

C. *Event Responder Pattern*

Event handling is needed between media components as well as application. The Event Responder pattern consists of Vertical Event Handling and Horizontal Event Handling patterns. The Vertical Event Handling pattern is elaborated in Fig 5.

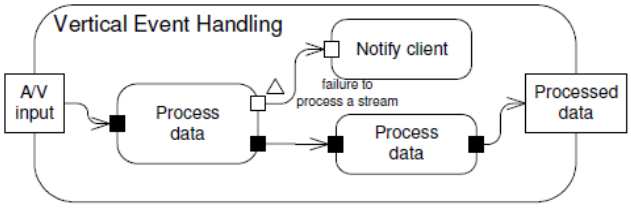

Figure 5. Vertical Event Handling pattern

Besides facilitating mechanisms for an interaction between application and streaming graph such as seeking to a position in a media file, an event handling mechanism between associate media components is provided. This uniform solution improves the quality of streaming software development.

IV. Concluding Remarks

Multimedia frameworks ship with a large collection of media processing components by default, making the fast development of a large variety of streaming applications possible. The key capabilities of existing multimedia frameworks include building of streaming graphs, streaming data through the graph, and responding to events invoked in the graph and the application. However, these tasks are realized differently in the multimedia frameworks.

Therefore, developers need to understand the concepts and mechanisms of multimedia frameworks to build quality-streaming applications cost effectively, particularly when it comes to the development for different platforms. We have presented sample design patterns based on the GStreamer, DirectShow, and Symbian MMF frameworks to facilitate the development of multimedia streaming software.

Future work will be focused on a case study illustrating how we apply design patterns to enhance the understandability and extensibility of multimedia frameworks and evaluate how much the application and framework developer's effort is facilitated by the streaming design patterns.

References

[1] Microsoft, "Microsoft DirectShow 9.0" http://msdn2.microsoft.com/en-us/library/ms783323.aspx

[2] GStreamer open source multimedia framework, http://gstreamer.freedesktop.org/

[3] Symbian, "Multimedia Framework" http://www.symbian.com/

[4] D. C. Schmidt, F. Buschmann, "Patterns, Frameworks, and Middleware: Their Synergistic Relationships", IEEE, 2003.

[5] D. Pilone, N. Pitman, "UML 2.0 in a Nutshell", O'Reilly Media Inc., 2005

[6] F. Buschmann et al., "Pattern-Oriented Software Architecture", John Wiley and Sons, pp. 53-70, 1996