Software Architecture for Immersipresence (SAI)

1. Research Team

Project Leader: Dr. Alexandre R.J. François, *IMSC and Computer Science*

Other Faculty: Prof. Roger Zimmermann, *IMSC and Computer Science*

Graduate Students: Maurya Shah, Cheng Zhu

2. Statement of Project Goals

The goal of the **Software Architecture for Immersipresence** (SAI) project is to develop a framework for the design, implementation and integration of distributed interactive, immersive, collaborative systems. It is complemented by the **Modular Flow Scheduling Middleware** (MFSM) project, an open source implementation [3].

3. Project Role in Support of IMSC Strategic Plan

SAI constitutes the software backbone of the Media Immersion Environment, IMSC's testbed for the integration, evaluation and demonstration of advanced multimedia technology.

As its implementation matures in the form of the MFSM open source project, SAI is to become the development and exploitation environment of choice for all IMSC research and development efforts.

4. Discussion of Methodology Used

SAI is a unified framework for the design, implementation and integration of distributed interactive, immersive, collaborative systems. In designing and developing such a framework, the challenges are many, and a new approach is required:

- From a computational point of view the goal is to achieve real-time analysis, synthesis, mixing and synchronization of (rich) data streams. Indeed, immersive, interactive applications typically involve a wide variety of data types and corresponding process modalities (e.g. images, sounds, graphics, user input, etc.). These are often addressed separately in research, industry and education, hence the difficulty in integrating two or more aspects into a single ad hoc system. Each field has produced its own structures and techniques for processing each data type while coping with hardware limitations. Support must now be provided for the representation and processing of these data types in data streams.
- From a Software Engineering point of view, the goal is to provide support for distributed development of integrated applications. The framework must accept and follow the realities of the research, industry and education environments. A modular architecture with well-defined data and processing models provides the flexibility required to encapsulate existing code, support easy and rapid (distributed) development of new code, and guaranty interoperability and ease of integration into complex systems.

- From a programming point of view, the goal is to provide code support to easily develop modules and applications that are efficient, scalable, extensible, reusable and that can interoperate.

The result is a generic, modular, extensible software architecture for dataflow processing of data streams (see Figure 1). A middleware layer provides an abstraction level between low-level services and applications, in the form of software components. At the core of this layer, the **Flow Scheduling Framework** (FSF) implements the foundation classes to develop such components. A generic data model allows the encapsulation of existing data formats and standards as well as that of low-level service protocols and APIs, to make them available in a system where they can interoperate. A generic distributed processing model supports control, asynchronous concurrent processing and synchronization of data streams. An application layer can host one or several data-stream processing applications built from instances of the software components, in a dataflow based programming model.

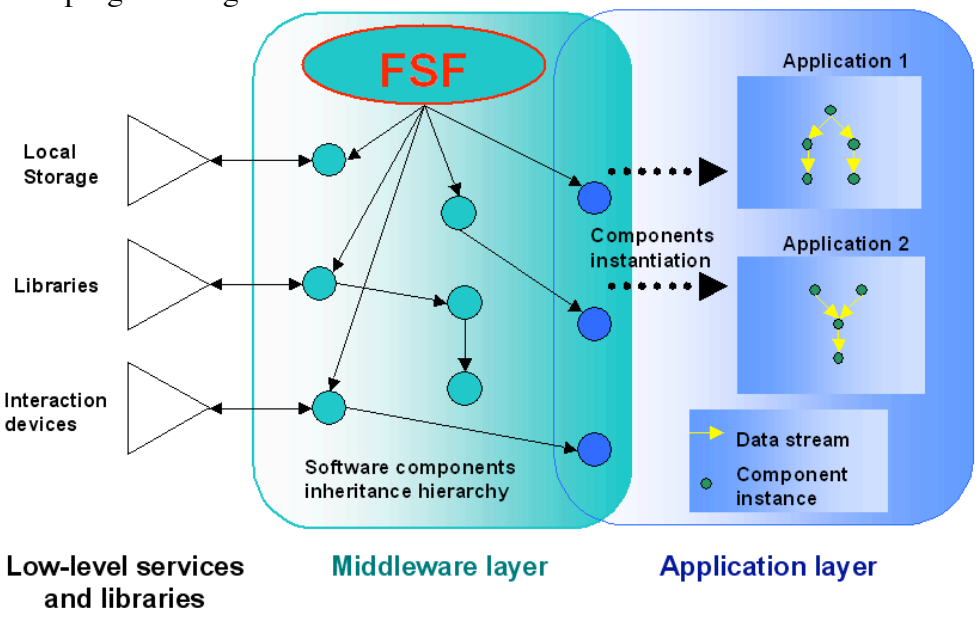

Figure 1. SAI: a modular software architecture

5. Short Description of Achievements in Previous Years

In the starting year (1999-2000), we introduced our layered software architecture. We defined the FSF [4], cornerstone of the architecture, by specifying data and processing models. We implemented a first prototype of the system with some supporting components to demonstrate the first real-time data stream processing application using this architecture, the "Blue Screen Without a Blue Screen" segmentation demonstration.

In 2000-2001, we introduced a preliminary design for an interactive application development and execution environment. Concurrently, we validated the FSF concepts and refined its implementation by integrating new functionalities in the code base (core library and supporting libraries). We demonstrated improved performance for the segmentation application [5], and introduced the "Video Painting" real-time mosaicing demonstration.

In 2001-2002, we launched the MFSM open source project, hosted on SourceForge.net [3]. It is an implementation of SAI in C++ for Microsoft Windows. The initial releases included the core library implementing FSF, some supporting modules and libraries, documentation and tutorials. Some useful existing modules were released as toolkits, one for image description and the other for interfacing with Microsoft Windows (GUI, devices, etc.). Furthermore, a toolkit implementing a scene graph structure was also developed, and used to demonstrated live video mapping onto an animated 3-D model in an interactive setting [7]. Other projects using MFSM included the "Virtual Mirror" [6].

5a. Detail of Accomplishments During the Past Year

This year we created a first of its kind course dedicated to media systems integration. The high point of the course was the realization of a distributed class project. We also started a number of focused exploration projects, with the goal of developing support for key features in MFSM. Feedback from teaching and research projects steered evolutions of the MFSM open source implementation and package design (documentation, demonstrations and tutorials), and of the FSF framework itself.

1. Integrated Media Systems course

In the fall of 2002 the USC Computer Science department offered the first course specifically dedicated to media systems integration. This graduate level seminar course, entitled "Integrated Media Systems," introduced students to the specific technological and organizational difficulties of designing and building media rich integrated systems. SAI was central to the design and implementation of the course.

Multimedia processing and application integration techniques were illustrated in a distributed class project. **Using MFSM, the 25 students developed a playable on-line (simplified) soccer game in just a couple of months**.

Small teams developed independent modules implementing key functionalities in different areas of media processing: networking (client/server), database (game recording and replay), rendering, physics/gameplay, interaction devices (including gamepad-based control and cyberglove-based gesture recognition control). These modules were then used to build the different applications forming the game (game and database servers, player and spectator clients). The game was demonstrated at IMSC's last BOC/SAB meeting.

Figure 2 is a screen shot of a player client display. The players are drawn as cylinders (different colors for different teams), the ball as a sphere. The field is marked with regulation lines. In this particular view (detached first person for the blue player), the goals are visible on the left hand side of the screen.

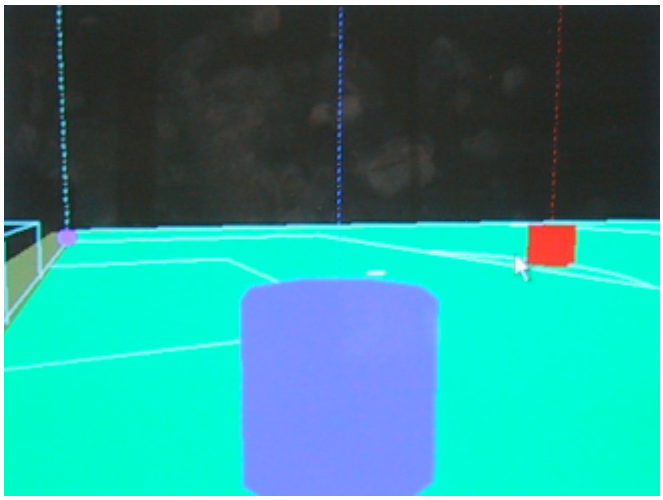

Figure 2. Screen-shot of the player client display.

Figure 3 shows students performing extensive testing of the application.

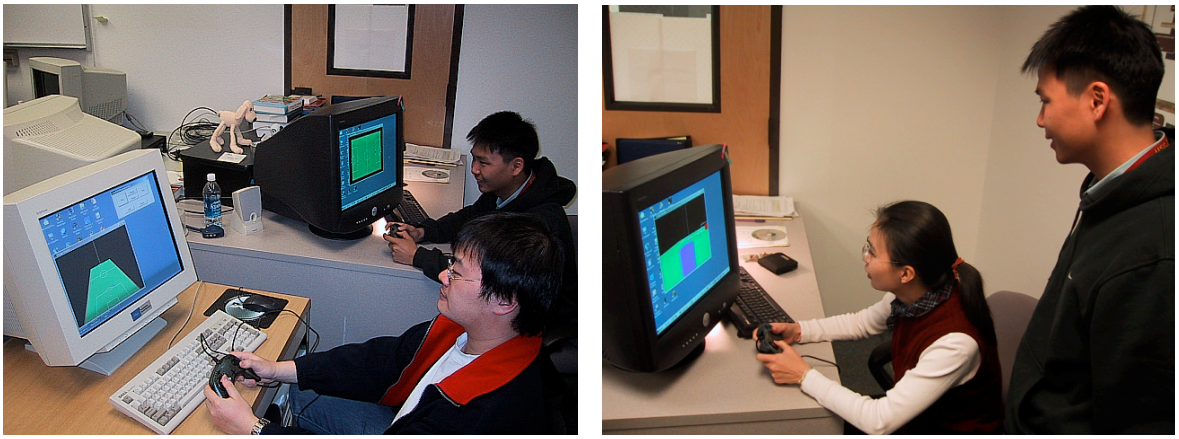

Figure 3. Students testing the project.

This course represents a major advancement in the teaching of "multimedia," by departing from the traditional “list of topics” approach. It spectacularly demonstrates the efficiency of SAI as a framework for distributed development of integrated systems. It also validates MFSM not only as an invaluable development tool for building such systems, but also as a teaching tool that the students can take from the classroom to the research or development environment.

Indeed, after taking the class, several Ph.D. students are now using the MFSM to implement their research work.

2. Experiments and Applications

Two areas were identified as critical functionalities to be developed using SAI, in order to augment the example and code bases:

- video and sound capture and synchronized rendering in SAI, and
- networking and compression for distributed SAI applications.

In both projects, the first issue is the design of SAI components and application subgraphs implementing a set of functionalities. Then comes the issue of actually coding these designs for a given platform, and using specific libraries, simultaneously conducting performance evaluations and comparing different approaches.

The first project aimed at extending the functionalities provided by an early (and now obsolete) MFSM toolkit. New data structures were designed to handle sound samples. A new live capture cell for both image and sound was developed using Microsoft DirectShow [1]. A time-stamp based audio and video sample synchronization scheme was implemented in the form of a new cell, also using Microsoft DirectShow. The second project aimed at exploring stream communication over network links. Unified data structures were designed, for transmitting generic stream data in distributed systems. Various experiments were conducted with different implementation modalities (based on sockets), various protocols (TCP/IP and UDP), and over different network configurations (10MB and 100Mb Ethernet).

The modules implemented in these projects are used in conjunction in the "AVChat" application, which demonstrates 2-way real-time audio/video communication over a network.

Both projects are continuing while the designs are being refined and several implementation modalities are being compared. We are currently testing other implementation modalities for audio and video capture and rendering, possibly more efficient and/or less platform-specific. We are also developing a generic XML data-encoding scheme, with corresponding compression techniques for reliable and uncertain data transmission.

This type of project contributes to the development of the overall SAI effort along two directions. First, the design of components implementing specific functionalities is an opportunity for validation and feedback for the framework design. Second, the modules implemented contribute to the available code base. They can be used directly in applications, or as templates for developing similar modules. Another example of such a project is a gesture recognition sub-system using Cyberglove devices and magnetic 3-D trackers, currently being developed in Prof. Shahabi's laboratory.

Targeted projects, developed in cooperation with various research groups, are the main vectors for widespread adoption of SAI inside and outside IMSC.

IMSC's Undergraduate Research Program is another opportunity for short, focused projects using MFSM. For example, this year's "Virtual Daguerreotype" project builds on the success of last year's "Virtual Mirror" project [6] to implement a Daguerreotype simulation system.

3. The Modular Flow Scheduling Middleware open source project

This year, MFSM underwent some significant evolutions. The **code was entirely re-written** to simplify and improve the efficiency by taking advantage of ANSI C++ features, including the Standard Template Library. The core library, implementing FSF, was made completely independent of external libraries, and as platform independent as possible.

Based on the feedback collected from the students attending the class and users in various projects, the whole open source package (and especially the documentation) is undergoing a major overhaul. If the implementation of the core library has now been proven quite stable, the documentation and supporting material (modules, tutorials, etc) need to be delivered in a more simple and accessible form. This re-documentation of the project should be completed by the end of Summer 2003.

4. Evolution of SAI and FSF

At the current development stage, the FSF model, core of the middleware layer of SAI, is used under the assumptions of infinite processing power and infinite bandwidth on the streams. The architecture's demonstrated scalability in terms of resource usage is currently not matched with a graceful scale-down when resources become scarce or imperfect: stream congestion is simply not handled. Perfect conditions assumptions are only valid when a single, powerful enough machine is involved. If a physical link between machines is involved, it can only be neglected if the bandwidth is sufficient for the amount of data to be transmitted, in which case, only the system input to output latency will be affected, and no congestion will occur.

Because of the modularity of the SAI processing model, an application graph can conceptually reside and be executed on a single machine, or be distributed on several machines. In fact, a collaborative distributed application can be considered as one single application graph, in which streams will be carried via network links between some of the cells in the graph. These cells represent entry and exit points in sub-graphs residing completely on separate machines. Our recent networking experiments, including the distributed collaborative game developed in the class, validate the concept of distributed SAI applications. However, the responsibility falls on the programmer/user not to overflow the system (by manually regulating the stream input or creation rate). For example, in the case of the distributed game, network congestion or processing load overflow create data accumulation. The applications must be stopped, or manually regulated, to avoid all the available memory being consumed. Ideally, in such situations, the amount of data processed should be adapted to the processing resources available (e.g., by reducing the rendering frame rate) and the amount of data transmitted should be adapted to match the constraints of the network, (e.g. by reducing the update frequency), while allowing the game to continue in the least disruptive conditions possible. Experiments have clearly shown that in order to behave robustly and gracefully under realistic general conditions, the framework must be extended to **incorporate overflow and congestion analysis, prediction, detection and intelligent management**. We have only started addressing these issues, and this line of research should continue for the coming year(s).

6. Other Relevant Work Being Conducted and How this Project is Different

Traditional interactive 3-D frameworks such as VRML [10] and more recently X3D [11], do not support consistent handling or generic processing of data streams in their framework. Similarly, data streaming frameworks, such as the MPEG21 [9] open standard, as well as proprietary solutions, focus on specific application domains (typically multimedia delivery and consumption), and thus are not natural frameworks for interactive immersion.

A large number of libraries and toolkits implementing necessary functionalities are available from the industry and from other research programs. However, they emphasize specific aspects of media processing, and do not address their integration. An example of such a heterogeneous set of libraries is Microsoft's DirectX [1]. One goal of SAI is to provide a unified framework for using these existing libraries concurrently in integrated applications.

Recent efforts have lead to the development of modular, distributed architectures for data stream processing to solve problems subsumed by that addressed in SAI. The Network Integrated Media Middleware (NMM) [8] is an open integration architecture that allows flexible usage of different networking and middleware technologies. The Distributed Media Journaling (DMJ) project [2] is developing a component-based framework for real-time media content analysis. These architecture designs result from requirements similar to some of ours. However, their application range is more restricted. The processing models designed in these frameworks are less generic than SAI's. Furthermore, they do not address mixing of 3-D data and other non-traditional immersidata types (e.g. haptics data), which are necessary for interactive immersion.

Beyond being a useful tool for easily integrating existing functionalities, SAI proposes a new approach to application design and development, allowing envisioning more ambitious integrated systems. This new approach raises new questions, with a matching set of research topics. In fact, the field of real-time distributed generic processing is mostly unexplored, and SAI is a perfect candidate platform for conducting the necessary research and developing and deploying the corresponding technology.

7. Plan for the Next Year

Our plan for next year is to build on this year's successes and continue our groundbreaking research in media systems integration:

- The IMS course will be offered for the second time with an improved format, including a more ambitious term project.
- Existing experiment and demonstration projects will be continued as needed, and new ones will be initiated. One area of particular interest for new explorations is the design and development of an MFSM-based graphical development and exploitation environment, to facilitate the design of MFSM-based applications.
- The open source project will be constantly updated to reflect the feedback collected in the class and from other MFSM users. Particular emphasis will be put on improving the documentation and augmenting the number of example modules and applications, and tutorials.
- Basic research will continue for the software architecture. In particular, we envision FSF to undergo a fundamental evolution from its current state. Analysis and control tools will be developed to predict and handle overflow and congestion in (possibly distributed) application graphs. FSF will be extended to model processing and transmission resources, cost and uncertainty. This will allow implementing and incorporating robust compile-time or application design-time congestion control strategies. Later on, we intend to build on these analysis tools to design, implement and incorporate mechanisms for dynamic adaptation and congestion control.

Our goal is to develop SAI into a comprehensive framework for strictly modular, adaptive distributed computing for interactive, immersive, collaborative applications.

8. Expected Milestones and Deliverables

The MFSM package will be maintained and updated. The new documentation, generic modules, demonstration applications and tutorials will be released gradually over the year.

Simultaneously, module development will continue as MFSM is used in various research projects. These modules will be used in various integrated demonstrations and experiments.

9. Member Company Benefits

SAI provides a uniform framework for the design of integrated media systems inside IMSC or in cooperation with industrial partners. The open source MFSM project provides a free common code base for independent (but concerted) development of processing modules and their integration into complex applications and systems, while respecting and accommodating all degrees of intellectual property protection policies.

10. References

[1] DirectX. http://www.microsoft.com/directx/

[2] Eide V.S.W., Eliassen F., Granmo O.-C. and Lysne O. "Scalable Independent Multi-level Distribution in Multimedia Content Analysis," Proceedings of the Joint International Workshop on Interactive Distributed Multimedia Systems / Protocols for Multimedia Systems (IDMS/PROMS 2002), Coimbra, Portugal, 2002.

[3] François A. Modular Flow Scheduling Middleware. http://mfsm.SourceForge.net

[4] François A. and Medioni G., "A Modular Middleware Flow Scheduling Framework." Proc. ACM Multimedia 2000, pp. 371-374, Los Angeles, CA, November 2000.

[5] François A. and Medioni G., "A Modular Software Architecture for Real-Time Video Processing." Proc. International Workshop on Computer Vision Systems, Vancouver, Canada, July 2001.

[6] François A., Kang E. and Malesci U. "A Handheld Virtual Mirror," SIGGRAPH Conference Abstracts and Applications proceedings, p. 140, San Antonio, TX, July 2002.

[7] François A. "Components for Immersion," Proceedings of the IEEE International Conference on Multimedia and Expo, Lausanne, Switzerland, August 2002.

[8] Lohse M., Repplinger M. and Slusallek P. "An Open Middleware Architecture for Network-Integrated Multimedia," Proceedings of the Joint International Workshop on Interactive Distributed Multimedia Systems / Protocols for Multimedia Systems (IDMS/PROMS 2002), Coimbra, Portugal, 2002.

[9] MPEG21 "Multimedia Framework". http://mpeg.telecomitalialab.com/

[10] The Virtual Reality Modeling Language. International Standard ISO/IEC 14772-1:1997. http://www.vrml.org

[11] X3D. http://www.web3d.org/x3d.html