MMSD'96: Workshop on Multimedia Software Development

Max Mühlhäuser

University of Linz
Informatics, Telecooperation Group
A-4040 Linz

max@tk.uni-linz.ac.at

Abstract:

Multimedia imposes quite a number of specific issues and requirements on software. While system support for multimedia has been seriously investigated for several years now, the software engineering community has not yet reached a deep understanding of the impacts of multimedia on their field. MMSD'96 brought together a representative spectrum of researchers investigating this issue. Four general observations could be retained from their contributions: 1. distributed multimedia applications exploit both more long-term potential and more persistent research problems than multimedia PC applications; 2. A reference model of multimedia software is not yet in reach; the large number of models proposed, even at MMSD, still lack better rationales, theoretic foundation, and convergence. 3. the MMSD community brings together people with a "document-background" and those with a "software background"; their views and approaches are still far from being harmonized. 4. While the multimedia community struggles for a common understanding among its members, it is in desperate need of a harmonization with approaches to distributed software development. In the remainder of this article, we will review issues to be addressed in the MMSD context, discuss the contributions made at the workshop itself, and draw conclusions for the state of the art.

1 Issues in Multimedia Software Development

1.1 General Concerns

The proper coordination and handling of multiple and (in part) time-dependent media on (distributed) computer systems imposes a number of particular problems which can be gathered into five key areas. We will briefly list these key areas below and discuss their impact on software engineering.

media types: the traditional computer-centered basic data types like boolean and real are complemented by human-centered media types like audio and video. These media type are complex, diverse, and evolving, as are the operations defined on them.

time and synchronization: with the advent of time-dependent media and multimedia, the notion of time becomes important for all layers of software; concepts developed in the context of real-time systems can not be equally applied; synchronization both within and among media is the most evident time- (but also space-) related issue.

quality of service, QoS: as media types are closely related to humane perception, the unambiguous "zero-or-one" representation of data types is replaced by different representations of the same information, reflecting different quality levels. This way different quality-related parameters can be traded off for one another and for cost; service delivery can be quality-related and, most important, quality can (and must) be expressed as constraints on or statistical descriptions of parameters instead of singular values.

streams: "passing" multimedia data from one software/hardware component to another is usually different from passing messages or parameters in traditional software: the time-related nature and size of multimedia data suggest continuous connection-oriented another is usually different from passing messages or parameters in traditional software: the time-related nature and size of multimedia data suggest continuous connection-oriented transmission where individual packets (samples, frames) are forwarded without intervention of the originator of the overall media transmission; the corresponding kind of connection is usually called a stream.

semantics: the most far-reaching issue in multimedia concerns the meaning and composite structure of both media-handling software components (hereafter called media processing elements, MPE) and media. This comprises, among others, the transition from signal-level digital representation of media (based, e.g., on pixels, luminance and chrominance values for images) to "information" (identification of objects, scenes, action etc.) and the statics and dynamics of MPE. In software engineering terms, these is theses may be considered the most demanding ones: especially with respect to the semantics of the above-mentioned "information", a lot of effort is devoted to extracting these semantics a-posteriori from the digital signal representation (cf. handwriting/image/speech recognition, video indexing, etc.). However, much too little effort is devoted to *modeling and representing* this information at its origin, as will be discussed in the next section.

1.2 State of the Art

In order to relate the MMSD'96 contributions to the general context of corresponding research and development, we will briefly discuss the state of the art in multimedia software development. We will use the five categories introduced above. Since this section is only thought as a brief introduction, a comprehensive overview cannot be given. Rather, a few recent developments will be cited for each of the five areas.

Media: proper handling of media types is a prerequisite for all kinds of multimedia software development. Problems stem, e.g., from the large amounts of data involved and from the fact that new media, formats, operations etc. must be accommodated. The object-oriented approach seems to be ideal for introducing an open number of new media types, together with the operations defined. Media class hierarchies promise to encapsulate formats, operations, standards, etc. Actually building and implementing a media class hierarchy requires second thoughts, however. The discriminators for designing the class tree are not obvious; the reflection of issues like formats and conversions and the smooth integration of devices as objects must be accomplished. Class interfaces can hardly

be defined such that all future extensions can be anticipated (cf. upcoming possibilities like the extraction of a 3D model of a person from a conventional digital video).

A state of the art example is the class hierarchy as proposed in the IMA (Intl. Multimedia Association) standard [3]. This hierarchy exposes several shortcomings of current approaches as mentioned above: e.g., a class in the hierarchy (such as "AVXwindowDevice") is often a rather odd mixture of multiple inheritances and aggregations, and formats and devices are more central to programming than the actual media.

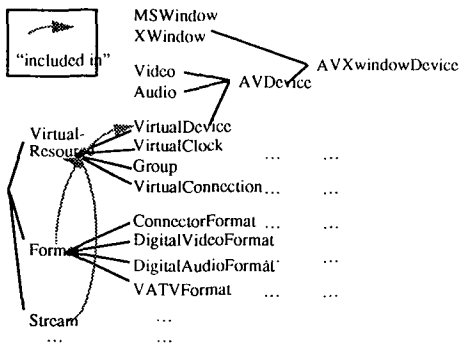

Figure 1. IMA class hierarchy, excerpt

The most pertinent shortcoming of such class hierarchies is the fact that the whole design rationale is oriented towards technology instead of semantics (cf. 1.2): if "audio" and "text" are first-class objects, how can a software engineer model a conversation among teleconference participants independent of the technology used to exchange messages?

Time and synchronization: Up to now, a satisfactory "time" model for multimedia software development has not been found. Most known approaches in this area are based on (thread-, process- or system-wide) *clocks*, similar to the approaches to traditional real-time programming. The resulting "interrupt-based" programming style is known to be tedious. As – in contrast to traditional real-time software – missing a deadline can be tolerated to a certain extend, a multimedia programmer has a great interest in details about the "missed/met deadline ratio". The clock-based programming model makes it close to impossible to receive such information. A different approach, *synchronous programming*, usually requires programs that are written as finite state machines in special 'predictable' programming language. For these reactive programs, maximum state transition times can be computed, and in consequence, the overall system response time can be computed and compared to a given mean event interarrival time (remember that for continuous media, the number of events per time unit is often known in advance). Thus, synchronous programming allows to check in advance whether or not the components of a program will meet their deadlines. At the Lancaster University Distributed Multimedia Laboratory, *reactive objects* (objects defined using the synchronous programming concepts) are investigated as a complement to active objects (=objects with threads) [8]. In the multimedia context, reactive objects might be used to guarantee timely response to events. E.g., a reactive object might control a chain of active objects (producers, transformers, consumers) and decide, based on the state of the active objects, whether they should continue to process a multimedia data sample or drop it. In addition to method interfaces, the object model for active/reactive objects must comprise states, constraints, and events. The approach proposed makes it possible to define a border line between guaranteed and non-guaranteed multimedia performance (reactive and active objects). The value of synchronous programming is not unanimously appreciated, however, in part because of the need for specialized languages and systems, in part since it does not yet provide much of a clue to the overall 'suitability' of a software configuration (e.g., as to the above-mentioned missed/met deadline ratio). Nevertheless, reactive objects currently represent probably the most advanced approach to coping with time in media.

Multimedia synchronization concerns both an inter-object aspect ("smooth delivery of samples") and an intra-object aspect (parallel and serial play-out schemes for multiple media 'tracks'). Synchronization also concerns both time (as just indicated) and space (arrangement of visual media on a common presentation device such as a monitor), yet research has concentrated a lot on the time-related problems. While the adequate delivery of synchronized multimedia presentations (i.e., smooth and coordinated playout) is considered an operating and communication system issue, the proper specification (modeling) of multimedia synchronization is a software engineering one. Several categories of such models have been proposed. Off-the-shelf multimedia authoring tools often use a common time axis as the means for specifying duration and relative ordering of media tracks. Many other approaches exist, such as scripts (powerful but difficult to handle by non-programmers), hierarchical synchronization trees and event tables (both do not scale up very well), finite coordinate systems (cf. HyTime[7]; they cover spatial synchronization, but have been criticized wrt. unpredictable duration and intuitiveness), and PetriNet-related approaches such as OPCN [4] (the formal background enables automatic inspection/verification, but threatens intuitiveness).

While it also has been found inferior to the others in certain respects. These drawbacks have mostly been addressed in the MODE model [1]: it supports a) objects of non-predictable duration (e.g., user input); b) intra-object synchronization; c) different levels of synchrony between parallel tracks (consider, e.g., a video track with lip-synchronous audio vs. background music audio); d) user-level exceptions if synchronization deadlines are not met; and e) object integrity (for some of the other approaches, certain synchronization constellations may require the fragmentation of logically coherent media objects into 'meaningless' pieces).

Quality-of-Service (QoS): QoS concerns all software levels as an orthogonal aspect. User-level QoS, for instance, concerns aspects of human perception. Since requirements at this level are hard to quantify, recent publications have proposed a "QoS-by-example" approach [10]. Media level QoS

in contrast (which is best understood today) reflects communications issues (end-to-end delay, bit and packet error rate, etc.), issues of the overall processing chain (jitter i.e., relative distortion of continuous-media samples etc.), issues of the individual media (such as, for video, frame rate and resolution, speed ratio etc.), and others (which can in part be expressed as combinations of the above-mentioned ones). QoS modeling and handling is particularly difficult since QoS parameters a) represent *statistical values* may (mean, min./max., standard deviation, burst lengths, etc.) b) must often be *negotiated* between users, applications, and different service-providing components, and c) can usually *not be fully guaranteed* today due to a lack of system support.

QoS considerations in software engineering are in a very premature state. Some APIs have just added QoS-related parameters to the parameter lists of APIs (or to method calls of foundation classes, for object-oriented approaches); they obviously do not reflect the above-listed problems very well. The above-mentioned IMA standard [3] provides only rudimentary solutions to these problems, too: only three QoS parameters (delay, jitter, bandwidth) are considered at all; the statistical nature of these can only be reflected by indicating an accepted bandwidth (min. and max. values); the co-existence of QoS-supporting and QoS-unaware middleware can be reflected by choosing among three guarantee levels.

A number of research projects have investigated application-transparent QoS optimization and negotiation. The above-mentioned MODE system [1] provides distribution-transparent multimedia objects; this means that programmers do not have to worry about the location of media and about the distinction of local vs. remote method calls. MODE retrieves the characteristics and location of relevant media objects and presentation devices from network-wide resource files and by looking at the application specific media classes (inherited from system-defined meta-classes). Based on this information, a planning/execution process is triggered which establishes the processing chain from information sources to sinks via transformation steps (such as compression/decompression); this process comprises all necessary QoS negotiation between the components involved.

Streams: the stream concepts proposed in the literature often refer to a simple point-to-point channel between a sender and a receiver of multimedia data, thereby relieving the mainstream sender/receiver specification from details of the continuous transmission of samples (video frames, audio blocks etc.). In the object-oriented context, method calls are usually introduced for stream management (setup, disconnection, ...) and manipulation (pause, ffd, ...). We will refer to the IMA approach again as an example IMA has defined an elaborate stream model; in the model, the interconnected processing elements consist of an aggregate object with virtual devices, formats and ports; the interconnection is achieved via the object class 'virtual connection'; several other objects are involved in a stream definition. As fig. 2 illustrates, the IMA stream approach involves quite a number of predefined objects in order to obtain a single point-to-point stream — this concept appears to be rather overloaded.

Future directions in research about streams comprise a) support for multihop, multicast topologies (already investigated in several projects); b) two-tiered approaches where "system level" configurations (network connections, specific devices and location) and "application level" configurations (involving abstract resources and user-defined processing elements) can be distinguished (cf. "semantics" below); and c) a harmonization of the object-oriented with the hypermedia paradigm. This third item seems to be appealing because object-orientation is well suited for specifying processing elements whereas hypermedia is well suited for structuring multimedia information on different levels of semantics; most interesting, hypermedia emphasizes "links" between objects, a concept that may be extended to accommodate streams much better than the plain object-oriented solutions known today.

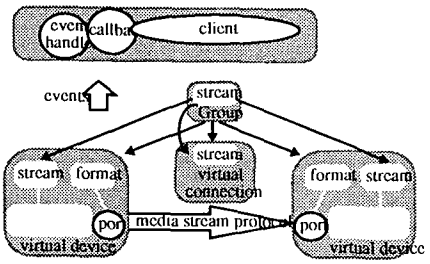

Figure 2. IMA stream example

Semantics: for this vast field of research, we can not in all cases report sample solutions for the sake of space. Rather, general comments will be given as to the areas which are emphasized or neglected in present research activities. As discussed in section 1.1, the following major aspects of semantics can be distinguished:

1. Semantics of MPE (multimedia processing elements, i.e. application programs/modules but also streams and system-defined modules, resources, and devices):
   - 1a. *MPE categories* such as "live media producer", "stored media retrieval component" etc., with the goal to support component-based programming or compiler-level checks/adaptations of the compatibility of connected MPEs, but also as a basis for appropriate runtime support. As section 2 will show, this area is well recognized as a topic of research (cf. also [9], [5], [2]).
   - 1b. *MPE configuration* i.e., means for specifying both static combination of MPEs (modules / streams) into an application and dynamic configuration changes. Many publications exist about the static aspect, whereas research on dynamic changes seems to be lagging behind, neglecting even the state reached in research about distributed applications.
   - 1c. *MPE dynamics:* as in software engineering in general, the design-time description of the functionality i.e., dynamics of MPE has been much less emphasized than their static nature.

1d. *Multi-tiered approaches* for the distinction of several abstraction layers in multimedia applications (such as: system-level, framework-level, application-level, "scenario-of-use" level) and for their coordination (e.g., with respect to QoS). Few and rather disjunct publications exist about this topic (cf., e.g., [6]), a common understanding appears to be far away however.

2. Semantics of multimedia data:

2a. *context of origin* i.e. information about the context in which media are captured (time, physical location such as "A's office", "GPS coordinates XYZ" etc.; scenario such as "conversation between X and Y"; operations on capturing devices such "pan"; etc.); in a vision where computer-controlled capturing devices (e.g., full-digital cameras) are used and media capturing becomes part of an integrated software solution, this context of origin can be in part specified at software development time and in part automatically recognized at runtime. As mentioned in section 1.1, this issue is much less emphasized in present research as the issue 2b below.

2b. *meaning* as a deliberately fuzzy term denoting several levels of semantics such as those recognized a-posteriori by today's computer-perception software (handwriting recognition, video content analysis etc.), those expressed a-priori in (maybe hypermedia-based) semantic networks, and many others. Much research is devoted to individual a-posteriori recognition and a-priori modeling issues, but there is little work on the urgently needed support for mapping high-level semantics automatically and flexibly onto different representations i.e., different media (e.g., "on-line help" mapped flexibly onto "audio", "text" etc., depending on the runtime context).

2 MMSD'96 contributions

Given the above introduction to the problems of and existing contributions to multimedia software development, the discussion of the individual workshop contributions can be rather compact. All contributions discussed below can be found in the workshop proceedings [11].

Day one of the workshop was totally devoted to architectures and building blocks for distributed multimedia applications (DMMA) with a total of seven papers.

In their invited talk, Blum and Molva [12] provided an in-depth comparison between four software platforms for DMMA development: Touring Machine by Belcore, Beteus by Eurecom, Medusa by Olivetti, and the IMA standard which was already cited. Key observations reported by Blum and Molva include the following: Touring Machine as a "pioneer" coped with hybrid analog/digital applications and was among the first to define MPE types. Its monolithic API and lack of programmer control over details of audio and video make it appear rather antiquated today, but it sure was one of the stage-setting contributions. Beteus was one of quite a number of platforms for workstation-based multimedia collaboration and stood out as to its build-in flexibility. Extensibility however was very limited. Medusa is rather unique as to its hardware: it investigates the extension of ATM into a so-called desk-area network, where devices (such as cameras, monitors, storage) are directly hooked to ATM. This dependence on new networking technology is the reason for Medusa to use a number of proprietary solutions where programmers would in the long run like to see standards (such as Corba) to be applied. Medusa is an important milestone towards a future solution. IMA finally tried to apply standards such as Corba, but still incorporates some concepts which appear to be overloaded (no. of components needed to set up a stream), out-dated (proprietary message-based communication), and power-lacking (point-to-point only streams). In the second half of their talk, Blum and Molva presented their own contribution called APMT (application pool and multimedia terminal) which excels with a high degree of extensibility, standard-compliance (Corba), support for multipoint stream-like connections ("connector box") and sophisticated runtime management.

Roy Campbell, one of the authors, presented the paper by Qian et al. [13]. He argued very much in favor of harmonizing the vast amount of efforts in DMMA engineering with the upcoming standards for distributed application development, Corba in particular. is proposed TOMA architecture complements Corba with special support for multimedia; thereby, customizable interfaces to relevant network and operating system services are provided and the (in this context) inappropriate Corba event service is complemented by a multimedia-proof TOMA-specific solution.

In contrast to the above, Bochmann et al. [14] presented a solution which is still very non-standard. Their talk revealed that the non-conformance to standards is the penalty paid for an outstanding level of sophistication, including support for dynamic QoS negotiation, flexible mapping of high-level to low-level semantics (cf. item 2b in section 1.2), and distribution transparency. Their proposed solution focuses on News-on-Demand applications.

While the MPE types introduced in the talks of Fritzsche [15] and Barth [18] seemed to suffer a bit from the "yet another" syndrome, they both advanced the state of the art in MPE specification and configuration support. Fritzsche emphasized formal specification support with custom specification languages and use of ASN.1, Barth made a convincing demonstration of his graphical run-time configuration tool based on the MPE model called CINEMA.

Frick's [17] MPE components are not of the "yet another" kind, but complement the long list of MPE contributions cited (in section 1.2, in the invited talk, in the "day one" contributions) in that they belong to a higher level of abstraction. Frick's MPE can be used in order to describe the scenario of

use, including participants and documents used in a collaborative setting, and including workflow-like combinations of different such scenarios. An early version of a formal specification aid for the (high-level) cooperation protocol was also presented, to be developed further.

Another rather complementary contribution to the "day one" topic was made by Eliassen & Nicol [16]. They brought up the importance of type checking in the stream context: when connecting MPEs via streams, such support can assure compliance between the components connected. The convincing presentation of the difficulties associated with the issue was followed by the introduction of a solution developed by the authors. The solution was related to the Corba and ODP standards for distributed applications.

In summary, day one contributed to QoS and stream issues and in particular to item 1 listed under "semantics" in section 1.2. In line with the observations about the state of the art mentioned there, the MMSD contributions emphasized sub-items 1a and 1b, but they also contributed to 1c (Bochmann, Frick). The good news were considerable advancements in many fields related to DMMA development, the bad news were that a common reference model and architecture for MPE is not yet in reach and that compliance with standards for distributed application development makes it hard to cope with advanced multimedia-specific issues such as dynamic QoS negotiation (which is not surprising).

Day two started with a much-acclaimed keynote that brought a refreshing insight into an attractive application domain: the movie industry. Peter Krieg's [19] vision of "digital Hollywood" as a world-spanning digital multimedia virtual enterprise projected much increased demands on the software side. The must urgent requirement concluded for the software engineering community was *lifecycle-support*, a topic which had been pretty much left out on day one.

At this point, another open question was emphasized: the historical gap between software development and document creation (authoring). Multimedia tends to make the boundaries between software and documents become blurred (cf., e.g., the difference between a captured video on disk and a computer-generated video created on-the-fly), but the communities that come together (and their concepts, jargon, etc.) stay as yet rather disjunct.

The first session of day one addressed lifecycle issues and brought this gap to evidence. Fedchak et al. [20] looked at electronic publishing on the web. Their presentation was governed by a lot of experience with WWW and led to an interesting taxonomy of web documents which were related to lifecycle issues. Interesting parallels to historic landmarks of software engineering were drawn. Lux [21] brought up an effect of using "realistic" multimedia interfaces (based, e.g., on virtual reality): such interfaces put particular requirements on the trustworthiness of the applications behind. Lux' major contribution was a design methodology that helps to increase this trustworthiness, based on particular object-oriented patterns. Morris and Finkelstein [22] started with a "document" background, but related their lifecycle considerations to a more general view where documents and software become blurred as described above. Their "discourse-driven" lifecycle-model seems to be much more adequate for multimedia than traditional "requirements-driven" models known in software engineering.

Morris and Finkelstein's observation that the essence of multimedia engineering is the *combination* of artifacts rather than their creation represented an excellent bridge to the second session. There, three papers concentrated on the synchronization of multiple media into multimedia, providing approaches which complement known synchronization models. Vuong et al. [23] extended and improved the well-known, PetriNet-based OPCN notation, Vazirgiannis et al. [24] covered synchronization in time *and* space, an aspect which is as yet underrepresented in the literature, and Shish et al. [25] adopted the Z notation and thus started a new 'line' of synchronization models which promises more far-reaching support for formal checks.

Two papers emphasized hypermedia standards. Rösch & Bäntsch [26] reviewed the MHEG and HTML standards and came to a surprising conclusion: while MHEG as an elaborate standard offers a large number of concepts for compiling multimedia presentation its functionality can be prototyped in Java with a considerably lower effort. The consequences may be far-reaching as to how standards and their use will evolve. Leidig & Rösch [27] presented a sophisticated graphical editor for MHEG-based multimedia authoring, relieving at least some of the MHEG burden about which Bäntsch had complained. Their authoring tool is based on a tool-building-tool that supports easy adoption to other standards and base platforms.

The final session brought up two disjunct issues that emphasized issues underrepresented in the other contributions: Koumpis et al [28] treated user modeling for multi-modal user interfaces, a predominant problem that comes up as multimedia is about to dramatically increase the diversity of user interfaces. Li et al. [29] set the final point with a quite astonishing experience gained with a security-enhanced MPEG video player: their tool turned out to be only marginally slower than known video players! The reason obviously lies in the fact that much of the run-time overhead of decryption is bound to disk-to-memory operations. The clever implementation presented is interleaving MPEG decompression and decryption in a way that minimizes the decryption overhead

In summary, day two brought the importance of dedicated lifecycle support and the gap between the software and the document "worlds" to evidence; important contributions were made in this respect, but many issues still deserve further research. The state of the art in synchronization models was advanced and MHEG was put into perspective.

In addition, quite a number of focused contributions were made, not only by the last two papers presented

3 Acknowledgments

First of all, the author would like to thank the MMSD co-chair Wolfgang Effelsberg for his contribution; it was a pleasure to work with him. The excellent coverage of the relevant areas of MMSD deserves special thanks to all authors. In particular, however , the co-chairs are indebted to the program committee for their support. We are thankful that a representative selection of world leading experts in MMSD have devoted their time and energy to the success of the first International Workshop on Multimedia Software Development. Last but not least, the organizers of ISEW and the editor-in-chief of IEEE CS press, Bob Werner, shall be assured of our particular gratitude.

4 References

1 B. Blakowski, J. Hübel, U. Langrehr, and M. Mühlhäuser, "Tool Support for the Synchronization and Presentation of Distributed Multimedia," Butterworth Jl. on Computer Communications, Vol. 15, No. 10, 1992, pp. 611-618.

2 S.J. Gibbs and D.C. Tsichritzis, Multimedia Programming - Objects, Environments& Frameworks, Addison-Wesley 1994.

3 Interactive Multimedia Association, "Multimedia Systems Services," Draft Rec. Practice, 1995.

4 T.D. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects" IEEE Jrnl. Selected Areas in Communications, Vol. 8, No. 3, April 1990.

5 R. Mines, J. Friesen, and C. Yang, "DAVE: A Plug and Play Model for Distributed Multimedia Application Development," Proc. 2nd ACM Intl. Conf. Multimedia, Oct. 15-20, 1994, San Francisco, ACM New York, pp. 59-66.

6 M. Mühlhäuser, "Modeling and Design of Complex Cooperative Software," Proc. 1st IEEE Conf. Engineering of Complex Systems, Ft. Lauderdale, FL, Nov. 6-10, 1995.

7 S. Newcomb, N. Kipp, and V. Newcomb, "the HyTime Hypermedia/Time-based Document Structuring Language," CACM, Vol. 34, No. 11, Nov. 1991, pp. 67-83.

8 M. Papathomas, G.S. Blair, G. Coulson, P. Robin, Addressing the Real-Time Synchronization Requirements of Multimedia in an Object-Oriented framework, IS&T/SPIE Proc. on Multimedia Computing and Networking, Vol. 2417, 1995.

9 R. Steinmetz and J. Fritzsche, "Abstractions for Continuous-Media Programming," Computer Communications, Vol. 15, No. 6, July 1992, pp. 396-402.

10 A. Vogel, B. Kerhervé, et al. "Quality of Service Management", IEEE Jl. Multimedia Sys., Vol. 2, No. 2, Summer 1996.

Proceedings:

11 M. Mühlhäuser and W. Effelsberg: MMSD '96, Proc. Intl. Workshop on Multimedia Software Development, March 25-26, 1996, Berlin. IEEE Press, Los Alamitos, CA, 1996.

Individual Contributions:

12 Ch. Blum and R. Molva, "A SW Platform for Distributed Multimedia Applications". in [11], pp. 10-21

13 T.Qian,S.M. Tan, and R. Campbell, "An Integrated Architecture for Open Distributed MM Computing". in [11], pp. 24-30

14 G. v. Bochmann and B. Kerherv , "Architectural Design of Adaptive Distributed Multimedia Systems". in [11], pp. 31-40

15 J. C. Fritzsche, "Multimedia Building Blocks for Distributed Applications". in [11], pp. 41-50

16 F. Eliassen and J. R. Nicol, "A Flexible Type Checking Model for Stream Interface Binding". in [11], pp. 52-60

17 O. Frick, "Multimedia Conferencing Systems as Building Blocks for Complex Cooperative Applications". in [11], pp. 61-68

18 I. Barth, "Configuring Distributed Multimedia Applications using CINEMA". in [11], pp. 69-76

19 P. Krieg, "Digital Hollywood — The Turbulent Marriage of Computer, Telecom, and Media Industry". in [11], p. 78

20 E. Fedchak and L. Duvall, "An Engineering Approach to Electronic Publishing". in [11], pp. 80-88

21 G. Lux, " A Design Methodology to Maintain Consistency between Functional Behaviour and Multimedia Presentation". in [11], pp. 89-97

22 S. J. Morris and A. C. W. Finkelstein, "Integrating design and development in the production of multimedia documents". in [11], pp. 98-108

23 S. Vuong, K. Cooper, and M. Ito, "Specification of Synchronization Requirements for Distributed Multimedia Systems". in [11], pp. 110-119

24 M. Vazirgiannis, Y. Theodoridis, and T. Sellis, "Spatio - Temporal Composition in Multimedia Applications". in: [11], pp. 120-127

25 T. Shish, D.A. Chiang, and H.Ch. Keh, "Formal Specification of Multimedia Authoring". in: [11], pp. 128-138

26 P. Rösch and M. Bäntsch, "Reviewing two Multimedia Presentation (quasi-) Standards". in: 11], pp. 140-149

27 T. Leidig and P. Rösch, " Authoring MHEG Presentations with GLASS-Studio". in: [11], pp. 150-158

28 A. Koumpis, Ch. Karagiannidis, and C. Stephanidis, "Adaptations of the Text Media Type: Addressing the User's Perspective". in: [11], pp. 160-168

29 Y. Li, Z. Chen, S.M. Tan, and R. H. Campbell, "Security Enhanced MPEG Player". in: [11], pp. 169-175

Editor's Filler

Heard any good quotes lately? If so, send them in to SEN.

I will use them as fillers like this one :-)