A Taxonomy for Multimedia Service Composition

Klara Nahrstedt
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
klara@cs.uiuc.edu

Wolf-Tilo Balke
University of California at Berkeley
Berkeley, CA 94720, USA
balke@eecs.berkeley.edu

ABSTRACT
The realization of multimedia systems still heavily relies on building monolithic systems that need to be reengineered for every change in the application and little of which can be reused in subsequent developments even for similar applications. Hence, building complex large scale multimedia systems is still a difficult and challenging problem. Service-based architectures, like researched in the Web community, form a possible solution to this problem: The service-based paradigm decomposes complex tasks into smaller independent entities (e.g. Web services), and then supports a flexible service composition in a variety of ways. However, due to the characteristics of multimedia applications and rich semantic structure of multimedia data and workflows, a direct application of Web-based research results is still difficult. The reason is that Web service frameworks cannot yet cope with the complexity of multimedia applications and their metadata. In this paper, we describe a basic taxonomy for the composition of services to support complex multimedia workflows. We will investigate in detail the necessary steps and methodology for multimedia service compositions and apply our taxonomy to different service composition instances. We will illustrate all composition instances within our taxonomy with case studies and point to possible techniques for the composition problem.

Categories and Subject Descriptors
D.2.12 [**Software Engineering**]: Interoperability – *Interface definition languages.*

General Terms
Algorithms, Languages, Design.

Keywords
multimedia service composition, service-oriented architectures.

1. INTRODUCTION
Service-oriented architectures are a concept strongly discussed and researched in the Web community today. Web services promise to take over an essential part of everyday's responsibilities and their composition is expected to extend their benefits to even more complex tasks, workflows and value chains. Beside the efficient provisioning and improved reusability of components, the move from data-driven to service-driven Web architectures promises to open up a whole new field of value-adding applications. These applications can be built on top of existing components and thus reuse individual services to form new workflows. Furthermore, new innovative business models for content-, service- and network-providers can be employed and used for mutual benefit.

Today's multimedia community is also on the move from monolithic multimedia applications to more flexible solutions. The new solutions can be provided either between content providers and clients or even among peers over different types of networks. But while the multimedia community effectively handles data semantics and provides sophisticated standards for media coding accompanied with meta-data descriptions (e.g. MPEG-7, MPEG-21), useful concepts from Web services research did not make a broad impact on multimedia systems development yet. On the other hand, Web-based concepts and constructs are invariant to new data types that are being heavily explored in the multimedia community. Therefore, the benefit of bringing together novel Web-based service-oriented concepts and the sophisticated handling and processing of multimedia data and annotations will be mutual.

Let us give two examples to motivate service compositions and their respective characteristics. First, let us consider the task of a simple *communication via the Internet*. Depending on the users' demands there is a number of techniques and services that can be used to achieve basically the same goal. Communication services can be successions of simple services or they can consist of a combination of different services that grow more and more complex, e.g. *email vs. instant messaging vs. (Internet) telephony vs. video conferencing.* All four composed services basically perform the same task of facilitating communication dialogs. They are however entirely different in their characteristics:

- **Email services** provide a rather static coarse time scale successive composition of relatively simple services. A message is processed, dormant at the server, popped by a receiver, and answered in the same way.
- **Instant messaging** is a rather static tight time scale successive service composition, including services such as composition of message content and its delivery to users. Real-time constraints pose severe challenges, since they present a strong condition for each service's usefulness.
- **(Internet) telephony** represents a simple version of concurrent service compositions, since it employs voice delivery in parallel with limited session management.
- **Video and audio conferencing** needs a far more complex mix of successive and concurrent service compositions with various synchronization, session management, and transformation services. For example, video services may be enabled or disabled during the lifetime of a multimedia connection depending on bandwidth availability.

Second, let us consider the next generation of digital *interactive HDTV systems,* where each user will be able to initiate tasks such as synthesis, distribution and display of customized multimedia

MM'04, October 10–16, 2004, New York, New York, USA.

content for him-/herself. Depending on a user's demands, such as type of display device, a number of various techniques and services will need to be deployed. For example, since users may have very different receiving client devices (e.g., iPAQs, laptops, plasma displays) and may decide on very different content layouts, the streaming multimedia content will need to go through different services such as transcoding, filtering, language translation, mixing, tilting, switching, and other services.

In summary, the next generation of complex multimedia tasks in the multimedia domain is emerging and will require a strong support of service composition in order to build systems in **a scalable, easy-programmable and flexible manner.** We are aware that the overall service composition problem space is huge. In this paper, we want to start to create *bridges* between the Web service community and multimedia community with respect to service composition. Therefore, our goal is to (a) provide a short overview of service composition models from both communities, (b) present a possible approach towards a multimedia service composition by taking into account the Web community service composition model, and (c) consider carefully the extensive knowledge in the multimedia community, especially in the area of content data type specification and performance Quality of Service (QoS) support. We will approach service compositions via a *taxonomy framework*. The framework introduces Web/multimedia integrating metrics such as the *time* when a composition of Web services and multimedia services happens. Using the *time* metric, we will partition the composition space into three service composition instances

1. *successive composition*,
2. *concurrent composition*, and
3. *hybrid composition*.

Within each service composition instance, we refine the composition taxonomy with additional Web/multimedia integrating metrics such as *scale (number) of services, performance quality, content type and infrastructural support*. This taxonomy allows us to decompose the multimedia composition problem and better understand when to apply what Web composition concepts.

The remainder of the paper is organized as follows: In section 2 we present service composition models from the Web and multimedia community as well as assumptions that will be made in our taxonomy framework. In section 3 we will present the methodology of our framework and introduce the metrics of our taxonomy. Section 4 will present the space partitioning in representative service composition instances and an evaluation of each service composition with respect to our taxonomy metrics. Section 5 will conclude the paper with further suggestions of possible integrations between Web and multimedia-based concepts to improve the overall multimedia service composition problem space.

2. COMPOSITE SERVICE MODELS

A generic service model relies on two major models: (a) the *dependency task model* and (b) the *component model*. The task model defines the functional dependency graph among functions to perform the overall advanced service or workflow. To compose the individual tasks into an advanced task graph, we assume that each functional task will represent an atomic service and be modeled as a service component. Each service component represents an atomic unit in the application layer for composing an end-to-end service, and it is described with corresponding meta-data. Web services usually include semantic properties of the service in the meta-data, whereas the multimedia services usually include content properties in their meta-data descriptions.

2.1 Composite Web Service Model

Current research on the composition of Web services focuses on both, technical details (service interfaces, communication protocols, composition languages,…) and architectural frameworks. A **basic Web service** is seen as an object, whose methods correspond to the service's business functions**.** First implementations have focused on e-commerce services platforms like HP's *e-speak* or Microsoft's *.net*. These platforms allow providers to register services, provide descriptions (e.g. in WSDL [26]), and to advertise them (e.g. using UDDI [27]). The users on the other hand can discover registered services, and invoke suitable instances. However, though services may offer several functionalities or methods of invocation (e.g. using different parameter-lists), their usefulness is often limited to simple tasks and the service itself cannot readily answer to dynamic changes in the environment or specific needs of individual users. Creating more complex business processes from multiple Web services is often referred to as *orchestration* or *choreography*. Both refer to executable business processes that can interact with simple Web services. But whereas orchestration always represents control from one party's perspective, choreography is more collaborative and allows each involved party to describe its part in business interactions.

To form business processes, simple services can thus be combined into composite services. The first **composition models** were *simple workflow-based process schemas*, like HP's *eFlow* [10] that was modeled on top of *e-speak*. In a graphical notation, service nodes (i.e. invocations) are connected by transition arcs that can be labeled with transition predicates, which have to evaluate to true before the transition can be performed. Atomic or composite services can be executed in a certain sequence, in parallel or even within transaction environments that make the successful execution of all services contained atomic. Beside authentification issues, consistency checks, recovery and flexible exception handling, the main concern is dynamic discovery, service selection and conversation handling to stay competitive in highly dynamic business environments and to cater to a certain amount of personalization for individual customers. Later approaches refined the basic model using, for example, *communicating finite state automata* to model services and the conversations between them for composition purposes [21]. Still, the kind of data these static services and their compositions are able to process is rather limited.

To provide advanced semantics in building service compositions, the **Semantic Web** community introduced *reasoning mechanisms* to enable automatic, but nevertheless semantically meaningful compositions. Basic frameworks often use *ontologies* for the dynamic discovery and flexible selection of suitable services within an automated workflow (e.g. the DAML+OIL standard [25] or DAML-S which has specifically been adapted for the use with Web services [29]). Reasoning along these ontologies allows the dynamic discovery of individual services, whose capabilities are closest to a given description of the task needed in semantic terms. Also the flexible selection or substitution of failed services can be performed along these lines by matching services with user preferences in a cooperative fashion as shown in [22] and [23]. A complex workflow can thus be described at a high level and then automatically be broken down into atomic parts.

Complex semantic frameworks for such compositions include e.g. the Web services modeling framework (WSMF) [28]. In WSMF

so-called *pre- and post-conditions* specify on one hand what is necessary to successfully execute a specific service, and on the other hand what has changed after the execution. This helps to facilitate many of the successive compositions in that the post-conditions of an earlier executed service usually have to match or at least contribute to the pre-conditions of a later service. Moreover, simulation frameworks such as in [24] provide calculi to reason about the execution of Web services and their compositions. Here the distributed operational semantics of Petri nets is used to analyse specific compositions of services, which can then be simulated and their global behavior and output verified.

2.2 Composite Multimedia Service Model

In contrast to the relatively static data processed by Web services and their compositions, multimedia services face slightly different problems. Also here a **composite multimedia service** can be understood to consist of a set of composable service components, which are connected into a *directed acyclic service dependency graph* to express the multimedia functions dependencies. A special case of a multimedia service graph is a *service path* where all services are connected into a single successive chain from the source to the destination. For example, a voice delivery service in Internet telephony is a service that configures a service path, since the digital voice goes through a single chain of services such as the encoding service (S1), transcoding (S2), transmission services (S3) along the data path, and the decoding service at the destination (see Fig. 1).

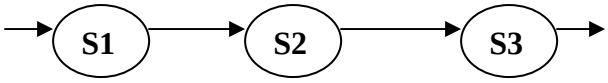

Figure 1: *Example of Voice Delivery Service*

It is, however, important to emphasize that unlike normal Web services a multimedia service usually processes **continuous data flows** such as audio and video, and if a multimedia service is composed of multiple atomic services, then the flow has to be passed from one service to another in an automated, performance- and quality-aware fashion. As mentioned above, each multimedia service component will carry meta-data descriptions to express performance quality attributes, denoted as **Quality of Service** (QoS). It means that each service component accepts input data with a certain quality level, denoted as QoS^{in}, and generates output data with a certain quality level, QoS^{out}, depending on how much resources *R* the service component gets during the processing of input data, and what quality the content arrives in (see Fig. 2).

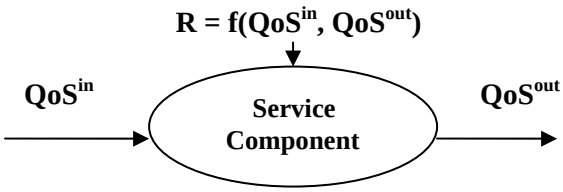

Figure 2: *QoS-aware Service Model*

The best known and researched measurements of QoS^{in} and QoS^{out} in the multimedia networking community are the available descriptive technical parameters like *bandwidth, delay, jitter or loss*. The *bandwidth* measures the network capacity that can be reserved at a certain speed for service execution. The *delay and jitter* focus on the latency and degree of synchronization of a network. The *packet loss rate* determines the faultiness of the network and the overhead by necessary resend traffic. However, it is important to stress that other meta-data attributes, besides performance measurements, are becoming standard (e.g., MPEG-7, MPEG-21), describing multimedia content in terms of the encoding format, compression, and information content.

Hence, if multimedia service components are being composed into an advanced multimedia service, then they must satisfy (a) *functional dependency correctness*, and (b) meta-data *QoS consistency correctness*. That means, if a component A is connected to a component B, then the QoS^{out} of A must match the input QoS^{in} of B. The reason is that multimedia data represent continuous flows from source(s) to destination(s) and need to flow through the various functional services in a contiguous fashion to deliver data of high temporal and spatial quality. Note that the QoS^{in} and QoS^{out} descriptions and consistency checks are similar to the Web concept of *pre-/post-conditions,* where the conditions must match in a transition from one service to another (cf. *transition predicates*).

3. METHODOLOGY AND METRICS

To support scalable, easy-programmable, and flexible ways to construct large-scale advanced multimedia services, we will aim to identify (1) what is missing in multimedia service compositions and (2) what can be imported from the Web service techniques. To achieve our goal we use a *taxonomy framework* to partition the service composition space according to integrating metrics, of interest to both service categories. In this section we will introduce the **metrics of the taxonomy**.

Service provisioning over the Internet needs even more descriptive characteristics than what we have now. Complex compositions are necessary to handle a vast variety of user-specified tasks. Therefore, beside describing performance features of the network capacity and delay, the functional complexity adds metrics like the *expected time* for instantiation of the composed service or the *number of necessary services* (possibly weighted by their individual complexity). The structural processing of different types of tasks varies, which in turn results in different utility functions applied by individual users. For instance, a service failing during a *successive service composition* will result in a suitable replacement of the service capabilities and thus only add to the delay of the service executing time. On the other hand a service failing during a concurrent service composition may (e.g. failure of voice transcoding in parallel with phone billing service) or may not (e.g. failure of a jitter filter) influence the successful execution of the intended task.

To evaluate the service composition space and its taxonomy, we need integrating metrics. For the scope of this paper, we will analyze service composition instances according to five metrics:

(a) **time** at which the service composition happens

(b) **number of services** participating in the service composition

(c) **performance quality** of service parameters

(d) **content type** that services process during the composition

(e) **infrastructural support** that the service composition needs for its instantiation and delivery

The **'time'** metric represents the instantiation time at which services need to run to deliver the composed service. This metric will partition our service composition space into *successive composition*, *concurrent composition, and hybrid composition* of services. We consider this metric at the top of our taxonomy tree

hierarchy. It means that we will divide our service composition space first according to the time metric into service instances and then within each service instance, we will evaluate the service composition according to the other four taxonomy metrics.

The **'number of services'** metric considers the number of services that are needed, invoked and sustained for composed service delivery. This metric partitions our service composition space into *centralized, distributed and hierarchical composition of services*. If we have a small or medium number of services in a composition and within a LAN environment such as smart rooms or smart buildings, then centralized composition approaches can be applied. If we have a medium number of services in a composition and the composition happens over a MAN/WAN network infrastructure (e.g. the PlanetLab infrastructure or IBM managed overlay network), then distributed composition approaches can be deployed. If a number of services with large number of participating services and over long WAN distances are deployed, then hierarchical composition approaches will be needed.

'Performance quality' represents the performance of each service parameterized via QoS parameters such as bandwidth, delay, jitter, information loss, and others. Performance metrics can be classified into three categories: additive, concave, and multiplicative. We will give examples of these performance metrics in each category:

Delay (d) and hop count (h) are additive metrics, it means delay *d* of a composed service path is the time required for the data to get through service path *sp*, including transmission delay and service execution delay $d(sp) = \Sigma_i d(p_i, p_{i+1})$, where p_i is the overlay proxy hosting individual services. We consider hop count (*h*) at the overlay proxy/peer granularity, and the number of proxy hops is the number of times the service path needs to switch proxies (if a proxy is visited twice for two different services, then it is counted twice). Delay and hop count performance metrics are important for the infrastructure support system, where, for example, the goal of service routing is to find a composed service path such that the aggregate delay and/or total number of hops are minimized.

Bandwidth (bw) and proxy capacity (pc) are examples of concave performance metrics. They are important for infrastructure support systems and their services (e.g., in resource and service discovery, data and service routing). In data content routing, the bandwidth requirement in a single data path is homogeneous, and bandwidth optimization is achieved by seeking the widest path [16]. In service routing, due to the heterogeneous resource requirements, selecting the widest path in absolute value is no longer appropriate. To achieve traffic balancing during service routing, the residual bandwidth and capacity need to be normalized based on the bandwidth requirement [17]. After the normalization process, the widest path selection criterion can again help to achieve better traffic balance (Methods for measuring end-to-end bandwidths can be found in [18], [19].)

Proxy volatility (v) is an example of the multiplicative performance metric. Proxy volatility is the probability of a service proxy/peer being down. The proxy volatility strongly influences the *information loss* of services residing on the proxy. This metric strongly influences the infrastructure support, especially the service routing. The objective is to find a service path (and thus composition) whose aggregate volatility is the lowest (or at least above a certain acceptable threshold), so that the transmission will most likely be successful. Let $v(p_i)$ denote the volatility of the proxy p_i. Then $v(sp) = 1 - \Pi_i(1-v(p_i))$ denotes the volatility on the service path *sp*. Note, that each proxy p_i in *sp* is counted only for *sp*'s volatility, even if p_i is visited more than once for different services.

The performance quality metrics partition the service space with respect to the achieved performance guarantee level of a service: (a) *premium/guaranteed performance service*, (b) *predictable/statistical guarantee performance service* and (c) *best effort/no guarantee performance service*.

The **'content type'** metric considers the semantic type of data information that the composed service works with. This metric will partition our composition space into *high semantic quality* (e.g. multimedia data with time and loss sensitive attributes) compositions versus *medium* (e.g. a certain acceptable loss rate) or *low semantic quality* information (e.g. best effort textual data). The basic notion of perceptual quality of service deals with different inherent levels of content quality, see e.g. the info-pyramid in [20] used to trade the fidelity (e.g. video stream resolution or frame rate) of multimedia objects against their modality (e.g. converting video streams into still images (key frames) or even descriptive text) to meet constraints in pervasive multimedia access.

Methods to describe content types are the *description schemes* and *variations* that are part of the MPEG-7 standard to describe different abstraction levels and variations of multimedia content. For example, the different abstraction levels include the extraction of key-frames or textual summaries of a video, or different variations of the multimedia data with different resource requirements. In addition, MPEG-7 contains so-called *hints* to support the transcoding, translation, summarization and adaptation of multimedia material according to the capabilities of a specific service, the client devices or network resources. For example, adaptation hints may be provided that indicate how a high resolution still image should be compressed to adapt it for hand-held computers, or how a video should be summarized to enable browsing over a low-bandwidth network.

The content type metric has to distinguish between changes of the content type that preserve information, e.g., transcoding between similar file formats, and those with information loss, e.g. decreasing a video stream's frame rate. Information preserving changes are important, but not critical, whereas a loss of information will generally decrease the individual utility of the service composition paths. Changes of the content type can also involve new atomic services, e.g. summarization services, that have to be inserted into the composition path to meet some global constraints. However, once inserted, the new content types may dynamically open up new possible paths to complete the complex workflow under the current constraints.

The **'infrastructure support'** metric (service discovery, service instantiation, level of distribution, service routing) considers the underlying resource management system and networking services and protocols that are needed to execute the composed service instantiation, service delivery, and service adaptation in case of failures such as peer unavailability, peer overload and others. The infrastructure support plays a crucial role in dynamic service level agreements for high-level tasks. The infrastructure support metric partitions our service space into two classes: service composition in *unstructured Peer-to-Peer networks (P2P)*, and service composition in structured *managed service overlay networks.* An example of a composed service in an *unstructured P2P* network is the P2P file sharing service such as Gnutella [3] or Napster [4].

n this type of network, infrastructure support will need:

- *efficient service lookup services* such as Chord [2] and CAN [1] that enable the data (e.g., MP3 files) or resources (e.g., disk storage) sharing,
- *peer selection services* to enable load balancing and appropriate server or proxy selection [5], [6],
- *service discovery* and *service routing* to assist in finding the functional service paths during initial service composition setup as well as during service composition failure and a subsequent recovery process [9], [10], [11],
- *QoS monitoring* and distribution services, especially for multimedia services [7], [9], [12], [13], and
- *service replications* to more efficiently find often needed services [14].

An example of a composed service in a *managed service overlay* network is audio and video teleconferencing used by a global company such as IBM [7], [8] where many value-added and content delivery services are being offered via service level agreements. In this type of network, the infrastructure will need (1) initial efficient and cost-effective *service composition* to quickly create new services, using QoS-aware service routing to find the initial appropriate services, and (2) *dynamic service composition* to dynamically recompose the service path and quickly recover from service outages and QoS violations. An example of such a managed service overlay infrastructure is the QUEST system [7].

In both infrastructure networks there is one *critical design objective* to provide rendezvous between service providers and users, i.e., the timely delivery of data and advanced services to those in need with minimized topological and message overhead. For example, towards improving the performance of searching for a particular data item or service in overlay networks, there exist two categories of solutions. One approach, based on distributed hash tables [2], associates designed hosts with each data item or service, thus achieving search performance in the order of log *n,* with *n* being the size of the overlay network. The other approach requires overlay nodes to collaboratively replicate either the actual services in demand or their shortcuts, both of which improve search performance [14], [15].

4. SERVICE COMPOSITION INSTANCES

Service composition is one of the most important concepts to accomplish the next generation of advanced application services. In multimedia applications, time is a very important metric. Therefore, our primary taxonomy classification will be according to the *time metric.* We will consider three major service composition instances: successive, concurrent, and hybrid compositions, present user case studies, and evaluate each instance according to the above presented metrics. To take into account Web composition concepts, we will apply some of them in this section such as (a) *workflow-based process schemas*, using a graphical notation with service nodes, transition arcs and transition predicates (pre-/post-conditions); this concept maps well to a multimedia service graph with QoS^{in} and QoS^{out} service conditions, and (b) *reasoning mechanisms from Semantic Web* to enable automatic, but semantically meaningful compositions; this concept maps well to organizing multimedia meta-data into ontologies according to specific metrics in order to enable service discovery, routing and adaptation mechanisms in the infrastructure support.

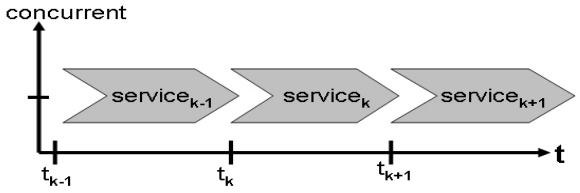

Figure 3: *Successive composition of services*

4.1 Successive Service Compositions

Successive service composition is a composition of functionally dependent services that are invoked in a successive time order (see Fig. 3). This means, if we compose *k* independent services, then the *(k-1)*-th service starts at the time of t_{k-1} and finishes before time t_k when the *k*-th service starts. It is important to note that

- the *(k-1)*-th service goes idle or serves other composed service instances when *k*-th service starts;
- the results of the *(k-1)*-th results will be used by the *k*-th service; the time scale between t_{k-1} and t_k can be minutes, hours or days;
- the response time (processing delay on the proxy) of a service component in the composition is important depending on the end-to-end delay performance quality requirement;
- each service in the composition chain adds discrete value to the overall composed service and contributes towards overall successful execution of the user's task/goal.

We demonstrate successive service composition with two case studies divided along the *performance* metric in our taxonomy:

Case study A: (Coarse time scale)
E-mail services provide successive composition of relatively simple services that do not need to pass information to each other in tight time intervals. It means that a message will be processed and delivered at the mail server, where it can wait for longer periods of time (hours/days) until the receiver invokes a mailing user agent service on a client device to pop and read email messages.

Case study B: (Tight time scale)
Instant messaging and voice streaming services represent successive service composition, where services have to pass information to each other in tight time intervals. Real-time constraints pose a severe challenge. For example, in case of instant messaging, the message content must be composed in real time as well as delivered to the users immediately. Another tight time scale service are voice streaming services that consist of voice encoding, a voice translation service (if conversation participants speak different languages), voice transmission, and a voice decoding service. These services are all invoked along the service path and each service component needs to respond within milliseconds.

Since we already used the *time metric* to partition the service composition space into successive, concurrent and hybrid composition instances, we will evaluate the successive service composition only against the remaining four taxonomy metrics:

(1) *number of services* – the successive service composition runs a single service at a time along the service path, i.e., only single service is instantiated within a time interval $[t_{k-1},t_k]$. However, in terms of service availability before the composed service starts, the successive service composition depends on the time frame. For example, in case study A, all involved services do not have to

be available at the same time, i.e., only the requested service in the service path must be available to be instantiated (i.e., number of concurrent available services is 1). On the other hand, in case study B, especially in the voice streaming service, all individual services along the successive service path must be available before the successive service composition starts to run. The reason for the simultaneous availability of all services in the service path is the very tight time scale between service switching.

(2) *performance quality* – Successive service composition may have strict performance requirements on the execution of individual services in terms of *delay*, and *information loss*. In case study A, it is of importance that the mailing user agent service responds fast, once initiated, and the information is accurate, i.e., no information loss is allowed. On the other hand, there may not be a fine granularity timing requirements on the composed email service, i.e. there is no tight computing/networking end-to-end delay requirement/guarantee on delivering messages to the mail server. Due to the continuous flow characteristics of e.g. audio data, in case study B the individual services transforming the flow data along the service and data paths must perform (a) within milliseconds, if conversational services (e.g. instant messaging) are deployed, and (b) within seconds or minutes, if on-demand services (e.g. Video-on-Demand (VOD)) are deployed, and (c) with some tolerable information loss. In the case of on-demand composed services, the response time delay within which the composed service responds, very much depends on (1) cache space and other resources availability, (2) security and content protection, (3) content management, and (4) network infrastructure support.

(3) *content type metric* – All of the content types such as text, voice, video may be used by successive services independent of tight or coarse time scales. Content hints for content transcoding, translation, summarization and adaptation would be extremely useful, especially for a tight time scale successive composition, since transition time between services is very small.

(4) *infrastructure support* – The successive composition requires an appropriate set of overlay networking services as well as appropriate content (QoS) meta-data organization via ontologies that assist in service discovery, service routing, QoS and service monitoring, service publishing and subscribing, QoS-aware resource allocation to instantiated services, service scheduling, and service fault management/replication capabilities. The successive service composition performs best in managed service overlay networks, where many of the high-quality service enabling technologies may be installed or are in place [8]. But even if one would use unstructured P2P overlays, some level of management and performance must be present to maintain a service's usefulness.

4.2 Concurrent Service Compositions

Concurrent service composition in multimedia systems is needed in many applications where (several) continuous flows are manipulated such as Internet telephony. As stated above, Web service composition frameworks like *eFlow* [10] allow both, to chain services or to execute several services in parallel, however, independently of each other. In contrast, the concurrent composition instance for multimedia services often has multiple tasks in parallel with strong functional dependency between services. For example, Internet telephony will have voice streaming service in parallel with streaming control service to stop calls and forward calls. Furthermore, the instantiation time of services can greatly vary. We may have the following dependency scenarios:

- all services are *functionally and time-dependent*, and start at the same time as well as finish at the same time (we have *k* services, where all services start at time *t(0)* and finish at *t(n)*), and the output of the (*k-1*)-th service is continually the input of the *k*-th service (see Fig. 4);
- all services are *functionally independent and time-dependent* (we have *k* services, where all services start at time *t(0)* and finish at *t(n)*, but the information among the services in independent;
- all services are *begin-cascaded services*, i.e., some services start early and are joined by other services later in time (we have *k* services where $l < k$ services start at *t(0)* and (*k-1)* services start at *t(l)*, and all finish at the same time *t(n)*;
- all services are *end-cascaded services*, i.e., services start at the same time and finish at different times (we have *k* services, where *k* services start at time *t(0)*, $l < k$ finish earlier at *t(l)*, and (*k-l)* services finish at *t(n)*.

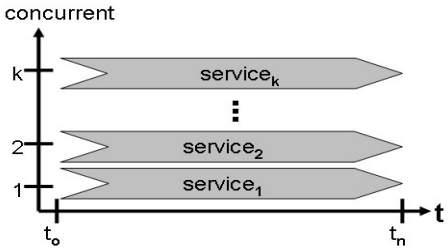

Figure 4: Concurrent composition of services

Case Study A: (Functionally and time-dependent)
A user wishes to see a customized TV-like soccer program with additional player statistics information and a commentary in each team's language. This task can be decomposed into functionally dependent and time-dependent concurrent services such as multimedia streaming service of the soccer video program, added-value services that provide the requested player information and spoken commentary information synchronized according to the video semantics. All services need to be initiated at the same time and are semantically dependent because the player information is dependent which player is shown in the video, the team statistics is dependent on the team in video, the language translation is dependent on what teams play and what the video semantics are.

Case Study B: (functionally independent and time-dependent)
But entertainment services do not always have to be functionally dependent. For instance receiver-synchronized audio and video streaming services in VOD can be decomposed into two functionally independent concurrent services that are, however, strongly time dependent, especially during the playout time.

Case Study C: (begin-cascaded)
Real time auction systems represent begin-cascaded services. While e.g. a multimedia stream showing items up for bidding, a service to deliver all related information and a service registering interested buyers have to be run concurrently during the whole time span of the bidding process, each individual user will decide whether and when to join the bidding process for a certain item. Once he/she joined by invoking a bidding service, a user has to be able constantly monitor the current bids in real time (unlike users not involved in the current bidding process, who might be fine with general updates in larger intervals) and increase his/her offer, if necessary. The services are terminated by the final acceptance

of a bid and the sale of the item, after which the next object may come up for auction and again users can join the bidding process.

Case Study D: (end-cascaded)
Video conferencing services usually represent end-cascaded services. A certain number of users start the conference at the arranged time. But whereas users' services for communication (such as A/V streaming services) start concurrently and have to be maintained during the entire duration of the conference, other basic services like services for setting up the conference or exchanging conference-related information will start distributing/exchanging all shared information (like the agenda or presentation slides, that every participant needs to see on local displays), and then seize to exist (or go dormant) once every participant can access the same information locally. Moreover, users may have to leave the conference early and thus will terminate their services.

Let us again evaluate the service composition against our remaining four taxonomy metrics:

(1) *number of services* – The concurrent service composition definitely needs to deal with the scale of number of services. Moreover, also the relative distance between services used within a composition is important to satisfy QoS requirements across several services that could each fulfill the requirement, but cannot quickly enough exchange data to fulfill the overall QoS required.

(2) *performance quality* – Like indicated by the first two metrics, the biggest attention in concurrent compositions lies on performance. Concurrent services require strict functional and/or timely synchronization which requires careful service monitoring of delays such as jitter and end-to-end delay. Especially concurrent video services will pose strong demands on available bandwidth and therefore need effective bandwidth and resource management.

(3) *content type* – Concurrent service compositions will need to support different types of content, ranging from voice, compressed video similar to successive service composition. However, it will also have to control information (e.g. synchronization control), when multiple services have to process data together (e.g. synthesizing A/V streams with subtitles). Synchronization information can usually be gained from meta-data. Similarly, it provides helpful decision criteria (e.g. for summarization), if a certain information loss has to be suffered to meet QoS goals.

(4) *infrastructural support* – The performance aspects also have to be reflected by the supporting infrastructure. For instance concurrent voice services usually demand very low proxy volatility, and network reliability. This can be achieved via efficient peer selection service, data resolution/modality support, efficient service lookup service, QoS monitoring and service replications, in addition to service discovery, routing and adaptation.

4.3 Hybrid Service Compositions

Hybrid service compositions involve both, successive and concurrent service executions showing functional dependencies and time synchronizations (cf. Fig. 5). Hybrid composition schemes can thus be decomposed into areas with concurrent composition, whose results are handed on to other services forming a successive composition chain.

Case Study:
A good example of hybrid service composition is the interactive HDTV system, where we have due to customized content concurrently HD audio, HDTV video streaming services in parallel with synchronization and session control services. The multimedia content streams may go through successive services such as transcoding, translation, or watermarking at different times.

This service composition instance requires similar metric considerations as discussed in the successive and concurrent compositions. Note that the complexity in satisfying performance guarantees and in provisioning the needed infrastructure support increases significantly. Moreover, in hybrid composition schemes also exchanging failing services against services with similar capabilities is harder to facilitate, because failing services in concurrent areas may add to the delay of successive compositions that in turn may also be part of a concurrent execution of services. Thus tackling the problem of discovery/selection for replacement services and the hybrid monitoring of QoS parameters is essential.

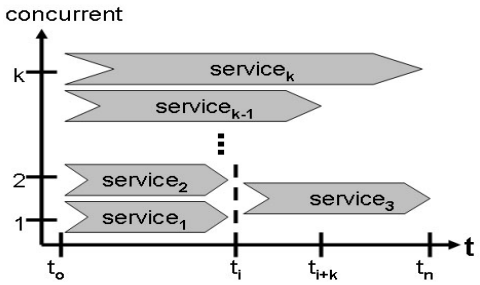

Figure 5: Hybrid composition of services

5. SUMMARY AND OUTLOOK

The necessity of building complex multimedia applications and large scale systems has forced the design methodology from designing and prototyping huge and expensive monolithic systems to embedding applications into flexible service-oriented architectures with a maximum reusability of individual components. Complex multimedia applications are instantiated by composing these relatively simple services to form the desired workflows. In this paper we have presented a taxonomy of metrics to effectively partition the service composition space for the multimedia domain. We have shown that the number of services, the instantiation time for composition, the content type, the performance quality, and the infrastructural support form helpful metrics to divide and conquer the complex composition space. These metrics lead us to organized multimedia composition spaces, and multimedia-specific ontologies, which again assist us in discovering semantically correct and quality-aware service compositions.

Since our aim is to map Web service concepts into the multimedia composition space, we present short *recommendations of concepts* and the useful *lessons learned* in the mapping:

- Basic techniques like *graphical flow composition* as given in e.g. eFlow [10] will definitely be of specific value.
- Some of the main problems in Web services provisioning like privacy, transactional support and semantic problems in describing service capabilities, currently are no major concerns for multimedia applications. The problems in multimedia applications are on a deeper semantic level.
- Concerning the mere *discovery of services* simple yellow page style *lookups* using UDDI/WSDL [27], [26] can very well prove to be an efficient enough technique, because the basic functionalities/capabilities of multimedia components can usually be clearly described.

- When it comes to evaluating semantic properties of content throughout the composed workflow like quality of service, information loss, etc., today's Web services techniques will not yet prove sufficient.
- Technologies from the Semantic Web like *service ontologies* to describe general to more specific capabilities or characteristics of individual services and preference-based negotiations of session parameters (e.g. [30]) may prove especially useful, but still need extensive research.

Given that the multimedia community uses the abundance of sophisticated methodologies for service compositions in a suitable way, we expect that a more efficient prototyping even for complex and large scale systems will be enabled. Moreover, service-oriented architectures will not only ease development, but also pave the way to new business models for content, service and network providers in multimedia applications.

6. REFERENCES

[1] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker: A scalable content-addressable network, ACM SIGCOMM 2001, San Diego, USA, 2001.

[2] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishan: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, ACM SIGCOMM 2001, San Diego, USA, 2001.

[3] Gnutella, http://gnutella.wego.com/

[4] Napster, http://www.napster.com/

[5] D. Xu, K. Nahrstedt: Finding Service Paths in a Media Service Proxy Network, SPIE/ACM Multimedia Computing and Networking Confrence (MMCN'02), San Jose, USA, 2002.

[6] R. Carter, M. Crovella: Server Selection using Dynamic Path Characterization in Wide-Area Networks, ACM Multimedia (Multimedia Middleware Workshop), Ottawa, Canada, 2001.

[7] X. Gu, K. Nahrstedt, R. Chang, Ch. Ward: QoS-assured Service Composition in Managed Service Overlay Networks, IEEE ICDCS 2003, Providence, USA, 2003.

[8] X. Gu, K. Nahrstedt, R. Chang, Z. Shae: An Overlay Based QoS-Aware Voice-Over-IP Conferencing System, IEEE Intern. Conf. on Multimedia and Expo (ICME2004), Taipei, Taiwan, 2004.

[9] J. Jin, K. Nahrstedt: Large-Scale Service Overlay Networking with Distance-Based Clustering, ACM/IFIP/USENIX Middleware 2003, Rio de Janeiro, Brazil, 2003.

[10] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, M. Shan: Adaptive and Dynamic Service Composition in eFlow, Intern. Conf. on Advanced Information Systems Engineering CAiSE'00, LNCS 1789, Stockholm, Sweden, 2000.

[11] X. Fu, W. Shi, A. Akkerman, V. Karamcheti: CANS: Composable, Adaptive Network Services Infrastructure, USENIX Symp. on Internet Technologies and Systems, San Francisco, USA, 2001.

[12] L. Subramanian, I. Stoica, H. Balakrishnan, R. Katz: OverQoS: Offering QoS using Overlays", Workshop on Hot Topics in Networks (HotNets-I), Princeton, USA, 2002.

[13] X. Gu, K. Nahrstedt: A Scalable QoS-aware Service Aggregation Model for P2P Computing Grids, IEEE HPDC 2002, Edinburgh, UK, 2002.

[14] E. Cohen, S. Shenker: Replication Strategies in Unstructured Peer-to-Peer Networks, ACM SIGCOMM, Pittsburgh, USA 2002.

[15] J. Guo, B. Li: Distributed Algorithms in Service overlay Networks: A Game Theoretic Perspective, IEEE ICC 2004, Paris, France, 2004.

[16] Z. Wang, J. Crowcroft: QoS Routing for Supporting Resource Reservation, IEEE Journal on Selected Areas in Communications (JSAC), vol. 14(7), 1996.

[17] J. Jin, K. Nahrstedt: Source-based QoS Service Routing in Distributed Service Networks, IEEE ICC 2004, Paris, France, 2004.

[18] B. Melander, M. Bjorkman, P. Gunningberg: A new End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks, Global Internet Symposium, San Francisco, USA, 2000.

[19] M. Jain, C. Dovrolis: End-to-End Available Bandwidth Measurement Methodology, Dynamics and Relation with TCP Throughput, ACM SIGCOMM, Pittsburgh, USA, 2002.

[20] J. Smith, R. Mohan, C. Li: Scalable Multimedia Delivery for Pervasive Computing. ACM Multimedia, Orlando, USA, 1999.

[21] T. Bultan, X. Fu, R. Hull, J. Su: Conversation Specification: A New Approach to Design and Analysis of E-Service Composition. WWW'03, Budapest, Hungary, 2003.

[22] W. Balke, M. Wagner: Cooperative Discovery for User-centered Web Service Provisioning. ICWS'03, Las Vegas, USA, 2003.

[23] W. Balke, M. Wagner: Towards Personalized Selection of Web Services. WWW'03, Budapest, Hungary, 2003.

[24] S. Narayanan, S. McIlraith: Simulation, Verification and Automated Composition of Web Services. WWW'02, Honolulu, Hawaii, 2002.

[25] D. Connolly et al.: DAML+OIL Reference Description. W3C Note, 2001.

[26] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana: Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/2001/NOTE-wsdl-20010315, 2001.

[27] UDDI. The UDDI Technical White Paper. http://www.uddi.org.

[28] D. Fensel, C. Bussler: The Web Service Modeling Framework WSMF. Journal of Electronic Commerce Research and Applications. Elsevier, 2002.

[29] M. Paolucci, N. Srinivasan, K. Sycara, T. Nishimura: Towards a Semantic Choreography of Web Services: From WSDL to DAML-S. ICWS'03, Las Vegas, USA, 2003.

[30] W. Kellerer, M. Wagner, W. Balke, H. Schulzrinne. Preference-based Session Management for IP-based Mobile Multimedia Signaling. Europ. Trans. on Telecommunications, Vol. 15(4), Wiley, 2004.