DESCRIBING THE SOFTWARE ARCHITECTURE OF A MULTIMEDIA DATA MANAGEMENT SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÇİĞDEM AVCI SALMA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

NOVEMBER 2013

Approval of the thesis:

DESCRIBING THE SOFTWARE ARCHITECTURE OF A MULTIMEDIA DATA MANAGEMENT SYSTEM

submitted by **ÇİĞDEM AVCI SALMA** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Halit Oğuztüzün _____
Supervisor, **Computer Engineering Dept., METU**

Prof. Dr. Adnan Yazıcı _____
Co-supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Adnan Yazıcı
Computer Engineering Dept., METU _____

Assoc. Prof. Dr. Halit Oğuztüzün
Computer Engineering Dept., METU _____

Asst. Prof. Dr. Selim Temizer
Computer Engineering Dept., METU _____

Asst. Prof. Dr. Aysu Betin Can
Information Systems Dept., METU _____

Asst. Prof. Dr. Bedir Tekinerdoğan
Computer Engineering Dept., Bilkent University _____

Date: 13.11.2013

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Çiğdem Avcı Salma
Signature :

ABSRACT

DESCRIBING THE SOFTWARE ARCHITECTURE OF A MULTIMEDIA DATA MANAGEMENT SYSTEM

Çiğdem Avcı Salma
M.Sc., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Halit Oğuztüzün
Co-supervisor: Prof. Dr. Adnan Yazıcı

November 2013, 116 pages

Multimedia Data Management Systems (MMDMS) enable storing, organizing, accessing and retrieving multimedia content effectively and efficiently. "Describing the software architecture of METU Multimedia Data Management System (METU-MMDMS), which can be a representative for contemporary MMDMS architectures, to facilitate future research on MMDMSs" is the primary objective of this study. The METU-MMDMS is placed to the focal point of the study by taking into account its unique infrastructure, non-systematic way of documentation of knowledge and experience, and the main motivation of this study is the absence of a descriptive architecture or architectural documentation for similar systems. Using the "Views and Beyond" software architecture documentation method, which was proposed by Software Engineering Institute (SEI), the documentation process is completed. Stakeholders' needs, quality attributes of the architecture and important design decisions for an effective MMDMS are recorded via the Views and Beyond (V&B). Following the process of documentation of the METU Multimedia Data Management System architecture, in order to evaluate this architecture, "Architecture Tradeoff Analysis Method" (ATAM) is carried out. Finally, the

documented architecture is discussed and V&B is evaluated from the multimedia database management system documentation perspective.

Keywords: Views and Beyond, Multimedia Database Management System, Software Architecture

ÖZ

BİR ÇOKLU ORTAM VERİ YÖNETİM SİSTEMİ YAZILIM MİMARİSİNİN BETİMLENMESİ

Çiğdem Avcı Salma
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Yar. Doc. Halit Oğuztüzün
Ortak Tez Yöneticisi: Prof. Dr. Adnan Yazıcı

Kasım 2013, 116 sayfa

Çoklu Ortam Veri Yönetim Sistemleri çoklu ortam içeriğine etkili ve verimli bir biçimde erişmeye, bu içeriği saklamaya, organize etmeye ve getirmeye olanak verir. Gerçekleştirilen çalışmanın ana hedefi güncel çoklu ortam veri yönetim sistemlerini temsilen, ODTÜ Çoklu Ortam Veri Yönetim Sistemi mimarisini Çoklu Ortam Veri Yönetim Sistemleri ile ilgili gelecekte gerçekleştirilecek olan araştırmalara olanak tanımak amacıyla tarif etmektir. Sistemin özgün bir altyapıya sahip olması ve geliştirme sürecinde edinilen bilgi birikiminin sistematik bir şekilde belgelenmemiş olması göz önünde bulundurularak örnek çoklu ortam veritabanı sistemi yapılan çalışmanın odak noktasına yerleştirilmiştir. Benzer sistemleri de kapsayan bir tanımlayıcı bir mimarinin ya da mimari belgenin bulunmaması bu çalışmanın temel motivasyonudur. "Software Engineering Institude" (SEI) tarafından ortaya çıkarılan “Views and Beyond” (V&B) metodu kullanılarak mimarinin belgelendirilme süreci tamamlanmıştır. Paydaşların ihtiyaçları, mimarinin kalite öznitelikleri ve etkili bir Çoklu Ortam Veri Yönetim Sistemi gerçekleştirimine yönelik önemli tasarım kararları V&B yardımıyla kaydedilmiştir. Örnek Çoklu Ortam Veri Yönetim Sistemi mimarisi belgelendirme süreci takiben "Architecture Tradeoff Analysis Method" (ATAM) aracılığıyla mimarinin değerlendirilmesi gerçekleştirilmiştir. Sonuç olarak,

belgelenen mimari tartışılmış, V&B yöntemi çoklu ortam veri yönetim sisteminin belgeleme süreci açısından değerlendirilmiştir.

Anahtar Kelimeler: Views and Beyond, Çoklu Ortam Veri Yönetim Sistemi, Yazılım Mimarisi

To My Family...

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to Assoc. Prof. Dr. Halit Oğuztüzün and Prof. Adnan Yazıcı for their encouragement and support throughout this study.

I would like to thank all members of Multimedia Research Group (especially Dr. Murak Koyuncu, Dr. Mustafa Sert, Elvan Gülen, Merve Aydınlılar, Saeid Sattari) for their technical guidance and support.

I would thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) for their support (109E014).

I would like to thank to Güneş Uyanıksoy for her role in architecture evaluation.

Finally, I am very thankful to my husband, sister and parents who always encouraged me to finish the thesis and their support during my graduate study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ADS	Architecture Description Specification
AMOS	Active Media Object Stores
C&C	Component and Connector
CFD	Control Flow Diagram
CORBA	Common Objects Request Broker Architecture
DFD	Data Flow Diagram
DoD	Department of Defense
EA	Enterprise Architect
GA	Genetic Algorithms
GRL	Goal-oriented Requirements Language
HMM	Hidden Markov Model
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Standards Organization
MARS	Multimedia Analysis and Retrieval System
MediaDB	Media Database
METU	Middle East Technical University
MMDMS	Multimedia Data Management System
MPEG	Moving Picture Experts Group
QG	Quality Goal
RM-ODP	ISO Reference Model of Open Distributed Processing
RUP	Rational Unified Process
SEI	Software Engineering Institute

SVM	Support Vector Machine
UCM	Use Case Map
URN	User Requirements Notation
V&B	Views and Beyond
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

Multimedia Data Management Systems (MMDMSs) have a complex nature since various state-of-art technologies and advanced techniques are brought together to construct them. Variety of modalities, communication protocols and multimedia formats are inherent in MMDMSs. There is often need to handle incomplete and uncertain information. Therefore, to understand, evaluate and compare MMDMSs, their software architecture should be presented clearly and systematically. In this study, SEI "Views and Beyond" (V&B) method [1] is used for the multimedia database management system architectural documentation process. The scope of this study is to describe a sample MMDMS architecture and the main objective to fulfill this scope is to document the software architecture with the V&B method by including what is actually implemented about the METU-MMDMS. Future work will aim to propose a reference architecture for multimedia data management systems. The present work will serve as an example.

Several high level MMDBMS architectures are briefly described in [2][3][4]. The software architecture of METU-MMDMS is examined throughout this work. METU-MMDMS processes multiple modalities inherent in a video and stores the concepts extracted from these modalities in order to support semantic querying effectively and efficiently. Furthermore, METU-MMDMS carries out information fusion both at query level [27] and score level (late) [36][41] to increase accuracy and uses a multidimensional index structure [40] that indexes content and concept for performance improvement of querying.

The organization of the software system, the selection of structural elements and interfaces, which form the system, determination of the behavior of these units while they work together, formation of an integrated system via combining the behavioral and structural elements and the collection of important decisions made throughout

this process constitute the architecture. Documenting software architecture is as important as the software design [5]. Even the most suitable software architecture designed for a system becomes unused if it cannot be understood or applied. Various methods can be followed during the documentation process, which are intended for understanding and evaluating software architectures. Different methods embody different processes and challenges. Regardless of the method used, correct documentation ensures the continuity and understandability of a software system. The improvement of the communication consistent with the architecture among the stakeholders, the opportunity of making early design decisions, the ease of evaluating the architecture, the production of the abstractions convenient for reuse can be listed as the return on the software architecture documentation [6]. In order to be followed up during this process, different methods, usually compatible with the IEEE 1471 architectural design standard [7], are proposed. In this study, SEI "Views and Beyond" (V&B) method [1] is used for the multimedia database management system documentation process. Besides the accurate and complete transfer of the knowledge, which is gained within the scope of the multi-media data management system research, after the completion of the architectural review in parallel to the documentation process, as a side effect, recommendations for the improvement of the architecture are made. In this work, SEI V&B method is summarized, the multimedia data management system is defined, the steps of documentation process of the METU-MMDMS descriptive architecture with V&B method are presented, the "Use Case Mapping" (UCM) [8] process is explained and applied, and finally the discussions are stated.

1.1 Motivation

There are many Multimedia Data Management Systems developed for industrial or research purposes. In fact, some of them are described in various reports or research papers, but generally those MMDMSs' architecture is not well documented in the published literature or a particular architectural documentation method is not explicitly used. Therefore, the main motivation of this study is to remove the lack of a well-described MMDMS architecture.

Although different stakeholders take part in the design, development and analysis activities, the documentation is carried out from the documenter's perspective. Hence, the architecture cannot be evaluated or analyzed easily since there is no suitable document at the end of the design procedure.

Architectural documentation helps making and recording the design decisions, rationales and variability points and the design process can be completed and reviewed in a smooth and systematic way, supported by documentation. Without a clear architectural design, the system will be immature. In order to record the design systematically and completely, the documentation should ideally be continued in parallel to the construction of the architectural design.

The stakeholders cannot communicate easily without a good architectural documentation. Besides seeing the architecture from others' perspective, they can be informed about the big picture via the architectural documentation. If the stakeholders cannot understand the architecture, they cannot apply it or they implement the wrong system.

Good documentation is also important for the future stakeholders, who may not be aware of early design decisions. This is critical for the evolution and maintenance of the system.

1.2 Contributions

The contributions of this thesis can be listed as follows:

- Describing a particular MMDMS architecture
- Investigating the design decisions, rationale and the variability points that are parts of the MMDMS architectural design
- Showing that the prepared MMDMS architecture documentation has enough substance to account for the required scenarios
- Extending UCM that maps scenarios to the MMDMS architecture
- Showing that V&B is capable of documenting the target MMDMS.

1.3 Organization of Thesis

The organization of the remainder of this thesis is as follows:

- Chapter 2: In this chapter, documentation methodologies, frameworks, standards and tools, which are related to this study, are explained. An overview of Multimedia Data Management Systems is given. After that, the sample MMDMS systems that are developed so far are described. Moreover, the improvements of MMDMSs' in time are addressed.
- Chapter 3: Chapter 3 describes the architecture documentation with models and architectural details. It provides the design decisions, rationale and variability points together with the architectural explanations and recommendations for the overall architecture. This chapter also covers the scenario mapping activity, which is carried out to prove the qualification of the architecture.
- Chapter 4: In Chapter 4, Architecture Tradeoff Analysis Method is summarized and METU-MMDMS architectural analysis results are discussed.
- Chapter 5: Finally, this chapter concludes the thesis with a discussion of the outcomes of the research and future research directions.

CHAPTER 2

BACKGROUND KNOWLEDGE AND RELATED WORK

In this chapter, the standards, methods, frameworks that are related with this study are presented in brief. IEEE 1471 is presented as a standard to describe the architecture of a software intensive system, and V&B, Rational Unified Process/Krutchen's 4+1 and Siemens Four Views, Data Flow and Control Flow are listed as the software architecture documentation methods. Use Case Map is the method, which is a part of URN and used for the further analysis of the METU-MMDMS architecture. C4ISR is the framework introduced by US DoD for architecture development. Finally, the last section lists the tools (UML 2.0, Enterprise Architect 7.5) used throughout this study.

2.1 Standards

2.1.1 IEEE 1471 (ISO/IEC 42010:2007)

IEEE 1471 is a standard, which recommends a software engineering practice for architectural description of software intensive systems [18]. According to IEEE 1471, the components, their relations with each other and the environment together with the method, which direct the design and the evaluation, forms the architecture of a system. The identical ISO adaptation of ANSI/IEEE 1471-2000 is ISO/IEC 42010:2007 [19]. An ISO/IEC 42010-compliant architecture can be created via V&B [1].

2.2 Methods

2.2.1 Views and Beyond (V&B)

"Views and Beyond" approach is based on the "view" concept, which consists of a group of elements and the relationships among these elements. According to V&B, different views address different quality attributes, and have different purposes and areas of use, and a nontrivial system cannot be represented completely via a single view [1]. In order to satisfy the V&B approach determining the relevant views for various stakeholders and writing the necessary information down for multiple views are required. Determination of the views according to the current stakeholders, documentation method of the interfaces of architectural elements, mapping of the elements that are present in more than one view and presentation of the behavior of the elements can be listed as the milestones of this method. V&B is not only a documentation method, but also the identification and recording process for design decisions. The facility of preparing views for different stakeholders, improvement of communication among the stakeholders, detailed documentation about the method, ability of modeling complex systems, and compatibility to the IEEE 1471 standard are the effective criteria that are considered in selection of this method.

The architecture of a system is roughly the parts of a whole and the relations and interactions among parts. The software elements, relations among them, the properties of relations and elements, together with the structures, which are necessary to reason about the system, constitute the software architecture [1]. Software architecture document aids groups of people while working together and eases the solution process for architectural problems. Documenting software architecture becomes critical when the architectural problems are subtle and quality attributes are demanding. The architectural documentation;

- helps to understand the stakeholders' needs,
- satisfies the stakeholders' needs via the recorded de sign decisions for each view,

- allows to check whether the stakeholders needs are met,
- presents the information with a format that the stakeholders can understand.

Thanks to the architectural documentation, the architects can make right decisions and developers will know how to carry out those decisions. By the help of recorded design decisions, future stakeholders can understand the rationale behind the design decisions.

It is important to capture the driving quality attributes, and how those quality attributes are satisfied by an architecture. However, while capturing quality attributes and the design decisions, the unnecessary design details should not be documented. It should not be forgotten that "Architecture is design but not all design is architecture." [1]. Documented decisions should be related to the driving quality attributes and behavioral requirements. The quality attributes can be listed together with a design approach or with the architectural elements that provide a service. Moreover, stakeholders want to know whether the quality attributes that match with their requirements are satisfied. There should be a place in the documentation where the definitions of quality attributes and how they are satisfied are presented.

Various architectural elements are used to construct the architecture; one of them is the module. Unlike task or process, a module is a hierarchical element, which consists of sub-modules. In order to provide enough level of hierarchy, the modules should be fine grained enough so that the system's modifiability and independent development requirements are met.

According to SEI V&B to document an architecture, the relevant views should be documented first. The information that is related to all views is documented later. The definition of the view in [1] is "A view is a representation of a set of system elements and relationships associated with them." Different views tend to contain different system elements and indicate different quality attributes.

The views are selected according to the stakeholders' concerns. The documentation of a view consists of:

- Primary presentation

- Element catalog
- Elements interfaces and behavior specification
- Variability guide
- Rationale and design information

Besides the architectural views, the documentation should contain an introduction, which guides the reader to find the desired information, describe the relationship between views and list the constraints and rationale of the overall architecture. The architectural style is related to the usage of elements and the relations with a set of constraints. To describe the architectural style and its vocabulary and rules, the architectural style guide will be used. In addition, a system can use more than one architectural style. There are three categories of styles:

- Module styles (units of implementation)
- Component and connector styles (units of execution)
- Allocation styles (relations between software and non-software resources)

Module is the unit of implementation while the component is a runtime entity. A single module can be transformed to many components or many modules to a single component. The module styles deal with assigning parts of the problem to the design and implementation units while C&C styles emphasizes the ways of interaction of the processes and the data flowing throughout the system.

The architectural style should not be confused with the architectural pattern. The architectural style specializes on the element and relations together with constraints but the architectural pattern describes a structural organization.

To create a sound software architecture document, unnecessary details and ambiguity should be avoided. The document should have a standard organization and the notation used should be clarified. The document should reflect the target group's point of view and record the related rationale, and finally the conformity of the document to the purpose should be checked. According to V&B the documentation package should include at least one module, one C&C and one allocation view.

A module style consists of specific sets of modules with some responsibilities and the combination rules of these modules. A class, layer, aspect or any implementation unit can be a module form. The component and connector style combines the component and connectors and the specific rules to bring them together. Services, processes, threads, filters, repositories, peers, clients and servers can be listed as components. Components are units of execution while connectors are units of interaction among components. Pipes, queues, request/reply protocols, and direct invocation are examples of connectors. The allocation style maps the software elements to the outer units. The outer unit can be hardware or the file system. To describe a style, style guides are constructed.

The elements together with the responsibilities, the relations of the elements (e.g. is-part-of, depends-on, is-a), the constraints and the aim of the view is presented in module views. The responsibility is the action, knowledge, decisions or the role of the module in a system. The modules can be in either aggregated or decomposed form. For example, the allowed-to-use relation is used to aggregate the modules in layered style. Module views are the blueprints of source code, necessary for requirements traceability and impact analysis. Informal notations, Unified Model Language (UML), Dependency structure matrix and Entity Relationship Diagrams can be employed in module views. Modules can be mapped to the components or non-software environments. V&B module styles, the relations that are used in this style and the objective of the style are listed in [Table 2-1].

Table 2-1 Module Styles

Style		**What's for**
Decomposition	Is-part-of	Organization of the code as modules and sub modules
Uses	Depends on	Which module uses which other modules
Generalization	Is a	Sharing and reusing
Layered	Allowed to use	Group of modules to offer a cohesive set of services to other layers

(Table 2-1 continued)

Style		What's for
Aspects	Crosscuts	Improve modifiability of the modules that deal with the business domain functionality (isolate the modules which are responsible for the crosscutting concerns)
Data model	One to one, one to many, many to many	The static information structure in terms of data entities and relationships

Table 2-2 C&C Styles

Style	What's for
Call-return	A series of services that can be invoked by other components. Connectors transfer the service requests. (e.g. client server style)
Data flow	Components are transformers, connectors transform data (e.g. pipe and filter, batch sequential styles)
Event based	Components communicate through asynchronous messages, loosely coupled federation of components (e.g. publish subscribe)
Repository	Contain one or more components, stores large collections of persistent data. (e.g. shared data style)

C&C view encapsulates elements like objects, processes, clients, servers that have runtime aspect (components) and also the elements like communication links and interaction protocols are included by the C&C view (connectors). The interaction of a component with other components is carried out through the ports. Ports are mapped with the connectors. Via C&C view, the system can be analyzed from the runtime perspective in terms of the quality attributes such as performance, reliability and availability. A component can possess its own architecture.

The structures that represent the relations and mappings of the software and non-software elements form the allocation views. Allocation styles are constructed on the allocated-to relation. The three common allocation styles are listed in [Table 2-3].

Table 2-3 Allocation Styles

Style	What's for
Deployment	Maps between software components and connectors and the hardware of the computing platform
Work assignment	Mapping between the software components and connectors and the production environment
Install	Mapping between the software components and connectors and the production environment

2.2.2 Rational Unified Process (RUP) / Krutchen 4+1

A five-view approach is presented by the RUP consistent with Krutchen's 4+1 approach [20]: Logical View (includes important design classes), Implementation View (documents architectural design decisions), Process View (contains information about tasks, processes), Deployment View (shows physical structure and configurations) and Use Case View (presents use cases and scenarios). A 4+1 architecture can also be documented by V&B [1].

2.2.3 Siemens Four Views

Siemens proposes four views to document the architecture of a system [21][22]:

- Conceptual View (maps the functionality of the system to the components and connectors)
- Module View (maps the components, connectors, ports and roles to abstract modules and interfaces)
- Code View (shows system's source and deployment organization)
- Execution View (maps system functionality to runtime elements)

Siemens Four View architecture can also be documented using V&B approach [1].

2.2.4 Data Flow and Control Flow Views

In Data Flow views system is presented via data flow diagrams (DFD). V&B C&C view types can replace DFDs. While DFD shows the system from data aspect, control flow diagrams (CFDs) models the system from control perspective. Control flow views use CFD, which can be replaced by V&B C&C view types [1].

2.2.5 Use Case Maps

Use Case Maps (UCM) aim to present the connection between structure and behavior in a prominent way [8]. UCM paths describe the causal relationship between the UCM Responsibilities, which are connected to the abstract components of the system's organizational structure. UCM paths represent the scenarios, which fill the gap between the requirements and detailed design. UCM's can be derived from the user requirements or use cases. UCMs do not analyze the system's functionalities at the level of messaging, but they are examined from the birds' eye view through the UCM paths. In this way, different architectures are encouraged.

2.2.5.1 Basic Notation

The arrangement which represents the casual relationships between a group of objects is shown in [Figure 2-1].

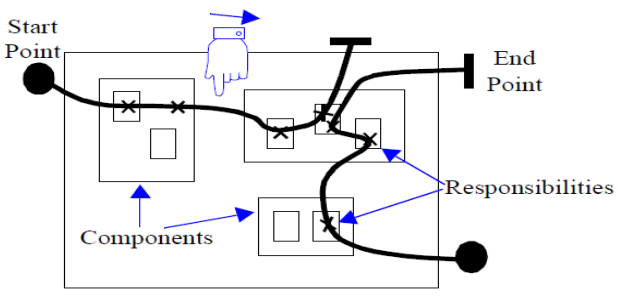


Figure 2-1 UCM Basic Notation [23]

The items of the figure can be listed as follows:

- **Start Point:** (Filled circle) Represents preconditions/triggering reasons.
- **End Point:** (Bar) Represents post conditions/effects.
- **Responsibility:** (Cross) Represents Activity, function or tasks.
- **Component:** The software parts that make up the system. The responsibilities can be connected to components.
- **Path:** Connects the start point, end point and the responsibilities.

2.2.6 User Requirements Notation

The User Requirements Notation (URN), that analyses the requirements by using the goals and scenarios, is a modeling language which is standardized by the International Telecommunication Union in 2008 [24]. URN is the first standard that handles and interconnects the goals and scenarios in a visual way. URN consists of Goal Oriented Requirements Language (GRL) and Use Case Maps (UCM). GRL models the actors and their objectives while UCM describes architectures together with the scenarios. To demonstrate that there is enough content in the architecture for the scenarios to flow, UCM notation is used in this study. Since the documentation activities are not carried out at the requirements phase, GRL is excluded from the process.

2.3 Frameworks

2.3.1 C4ISR Architecture Framework

In order to produce interoperable products and have a common architecture, The United States Department of Defense (DoD) introduced a framework for architecture development.

Main components of C4ISR Architecture Framework [25] can be listed as follows:

- Definition of common architectural views
    - Operational View (tasks and activities, operational elements, and information exchanges required to accomplish DoD missions)
    - Systems View (technical standards, implementation conventions, business rules and criteria that govern the architecture)
    - Technical View (the set of rules that governs the implementation and operation of the system)
    - All View
- Guidance for developing the architecture
- Definition of common products
- Relevant reference resources

2.4 Languages and Tools

2.4.1 UML 2.0

Unified Modeling Language (UML) [47] is a generic, standardized (ISO/IEC 19501:2005 [48]) language for software engineering which is defined by the Object Management Group (OMG). Visual models of the object-oriented software intensive systems can be created via the graphical notations included in the UML. Modeling the business process, systems engineering modeling and representing organizational structures are other capabilities of UML. UML 2.0 [49] is a revision that fixes shortcomings of the first version of UML. UML 2.0 has two general diagram sets and 13 basic diagram types [Table 2-4].

Table 2-4 UML 2.0 Diagram Sets and Types

Diagram Set	Diagram Type	Explanation
Structural Modeling Diagrams	Package	Model's logical containers and their high level interactions
	Class or Structural	Models' basic building blocks (types, classes, etc...)
	Object	Structural element instances and their run time usage
	Composite Structure	Inner details of elements' structure, relationships and construction
	Component	High level complex structures with well defined interfaces
	Deployment	Significant artifacts' physical disposition
Behavioral Modeling Diagrams	Use Case	User/System interactions
	Activity	Program flow, decision points and actions
	State Machine	To understand the run state of the model during execution
	Communication	Network/sequence of messages/communications of objects
	Sequence	Sequence of messages between objects on a vertical timeline
	Timing	Fusion of sequence and state diagrams
	Interaction Overflow	Fusion of Activity and Sequence diagrams

2.4.2 Enterprise Architect 7.5

Enterprise Architect (EA) [50] is an analysis and design tool, which is developed by Sparx Systems and based on OMG UML. Some of the uses of EA 7.5 are listed as follows:

- Design and build systems with UML,
- Model and manage complexity,
- Share models,
- Model, manage and trace requirements,
- Develop personal views and extracts of the model,

- Track and trace model structures,
- Generate documentation,
- Generate reverse engineer source code,
- Visualize, Inspect, Understand complex software.

2.5 METU Multimedia Data Management System

The architecture of METU Multimedia Data Management System (METU-MMDMS) is documented in the scope of this study. METU-MMDMS processes multiple modalities inherent in a video and stores the concepts extracted from these modalities in order to support semantic querying effectively and efficiently. As well as the basic input/output and querying functionalities, METU-MMDMS is specialized on the similarity-based operations on complex multimedia objects. The accuracy of METU-MMDMS is enhanced with the aid of information fusion, which functions on both extracted objects (data level) and query level. Furthermore, Multidimensional Index Structure is constructed to index content and concept for performance improvement. The system is composed of a Multimedia Database, Semantic Information Extractor, Coordinator and the Client Application modules. Triggered by the client queries, the information extracted through the Semantic Information Extractor component is stored together with its fuzzy properties in the multimedia database. The stored information is extracted via the index structure that is specialized for multimedia data. The Coordinator component is responsible for directing the information flow among the elements of METU-MMDMS [26][27]. The size and complexity information about METU-MMDMS is presented in [Table 2-5]

Table 2-5 Size and Complexity Information of METU-MMDMS

	Parameter	**Value**
Size	Lines of Code	395390
	Number of Modules (Packages)	65
Complexity	Highest Avg. Cyclomatic Complexity (per module)	7.0

2.6 Related Work

In this study, the METU-MMDMS architecture is documented by using the V&B method. From the MMDMS architectural perspective, the attempts of proposing more qualified multimedia database management systems led to the emergence of various software architectures, which elucidate the structure and functionality of those MMDBMS from different perspectives. Initially, the MMDBMS systems were mainly repositories.

One of the first developed MMDBMSs is the MediaDB [9], which has client/server architecture. It is an object-oriented system and supports relationships between objects. Afterwards, the systems, which provide complex object types for multimedia content and operators for these types appeared, such as the MIRROR MMDBMS [10], developed by University of Twente. The system possesses a distributed architecture and uses CORBA to allow distribution of operations. Recent MMDBMS systems utilize MPEG-7 (XML-based) and MPEG-21 (to define an open multimedia framework) standards.

Multimedia Analysis and Retrieval System (MARS) [11] integrates multimedia information retrieval and database management system. Its multimedia data model influenced MPEG-7 [12] standard. MARS involves Query Processor, Data Model, Index Structure and Multimedia Access as its main modules. Despite all these developments, only a few MMDBMS architectures are described at a very high level of abstraction. In the high-level architecture of the distributed multimedia database management system Active Media Object Stores (AMOS) [13], the connections between components are not defined and components for some key processes such as content/concept extraction and content based/concept querying are not presented. Even some of the components are not clarified for the MMDBMS architecture in [14]. And several MMDMs are present in the known literature that are not discussed from the architectural perspective [15][16][17].

METU-MMDMS architecture is described completely and precisely via the Views and Beyond method [35]. The V&B approach is based on the "view" concept, which

consists of a group of elements and the relationships among these elements. According to V&B, different views address different quality attributes, and have different purposes and areas of use, and a non-trivial system cannot be represented completely via a single view [1]. Determination of the views according to the current stakeholders, documentation method of the interfaces of architectural elements, mapping of elements that are present in more than one view and presentation of the behavior of the elements are the tenets of this method. V&B also encourages the identification and recording of design decisions. The ability of modeling complex systems and compatibility to IEEE 1471 standard [18] are the effective criteria that are considered in selection of this method. In [43] V&B method is compared with ANSI-IEEE 1471-2000. The study shows that ANSI-IEEE 1471 requirements are satisfied by the V&B approach. Besides being conceptually compatible, V&B can:

- include summary, context, glossary,
- meet stakeholders and their concerns,
- list the stakeholder, rationale selection of the view, used techniques,
- record rationale and consistencies among views.

There are also differences between the two approaches. The 1471 initially takes into account the stakeholders and their concerns. Afterwards, the viewpoints are constructed according to the stakeholders' needs. Depending on these viewpoints, the architectural views are prepared. On the other hand, V&B process begins with the selection of architectural styles that reflect the system. If the style is important for a stakeholder, it can be documented as a view.

Does V&B have the complete coverage of the software architecture domain? According to [44] no viewpoint model has the complete coverage. (May, 2005) analyses five viewpoint models (Krutchen's 4+1 View Model, V&B, ISO Reference Model of Open Distributed Processing (RM-ODP), Rational Architecture Description Specification (ADS)) to understand whether an optimal set of viewpoints from those models can be combined to obtain the widest coverage. The viewpoint models are compared to the IEEE Standard 1471-2000. As a result, the optimal set of viewpoints with the widest coverage includes;

- Requirements viewpoint, Rational ADS,
- Module view type, V&B,
- C&C view type, V&B and
- Allocation view type, V&B.

Besides the methods with wide coverage, there are methods for modeling different aspects of the architecture from a narrow perspective. (Roshandel and Medvidovic, 2003) analyse internal consistency among different models of a component. It defines a Quartet as a set of four primary aspects of component, which are interface, static behavior, dynamic behavior and the interaction protocol.

Any view includes diagrams to model the corresponding architectural perspective. V&B allows the user to select any of the informal, semiformal or formal notations to use. Each notation has its own tradeoffs. Notations that are more formal are hard to create and time consuming but more clear and detailed. Notations that are more informal are easy to create but they do not support making effective analysis as well as detailed diagrams do. Unified Modeling Language is a widely used semiformal notation. However, in [45], (Sauer, 1999) claimed that "a specialized and more advanced language is needed to describe the precise temporal ensembling of different media objects and UML lacks appropriate guidelines on how to deploy the different diagram types cooperatively to model complex multimedia applications". Due to these shortcomings, an extension of UML for multimedia applications, called OMMMA-L (Object Oriented Modeling of Multimedia Applications), is proposed in [45].

CHAPTER 3

DOCUMENTING THE ARCHITECTURE

In this chapter, the documentation process of the METU-MMDMS is described. Initially, the functional features of the system are identified. Afterwards, the architectural information in the software architecture document that is constructed in accordance with the V&B method is provided and UCM is employed to check whether the documented view includes enough content to carry out a related scenario or not. In the last section, the recommendations, especially on the use of design patterns, are offered.

3.1 Functional Features

The goal of the MMDMS study is to develop a multimedia information system, which processes multiple modalities inherent in a video and stores the concepts extracted from these modalities, in order to support semantic querying effectively and efficiently. For this purpose, the METU-MMDMS architecture is required to provide the following functional features:

[Feature-1] The system shall be able to upload a video.

[Feature-2] The system shall support annotation of an object in a video frame.

[Feature-3] The system shall support editing of an annotated object of a video frame.

[Feature-4] The system shall support annotation of an event in a video.

[Feature-5] The system shall enable the user to edit the annotated event of a video.

[Feature-6] The system shall extract concepts for an uploaded video.

[Feature-7] The system shall enable the user to edit the extracted concepts for an uploaded video.

[Feature-8] The system shall support content/concept based querying.

[Feature-9] The system shall support similarity based querying.

[Feature-10] The system shall show the query results via a video player.

[Feature-11] The system shall support unimodal/multimodal based querying.

[Feature-12] The system shall support fusion based querying.

[Feature-13] The system shall fuse the information extracted from various modalities.

[Feature-14] The system shall perform efficient indexing of content and concept of stored videos.

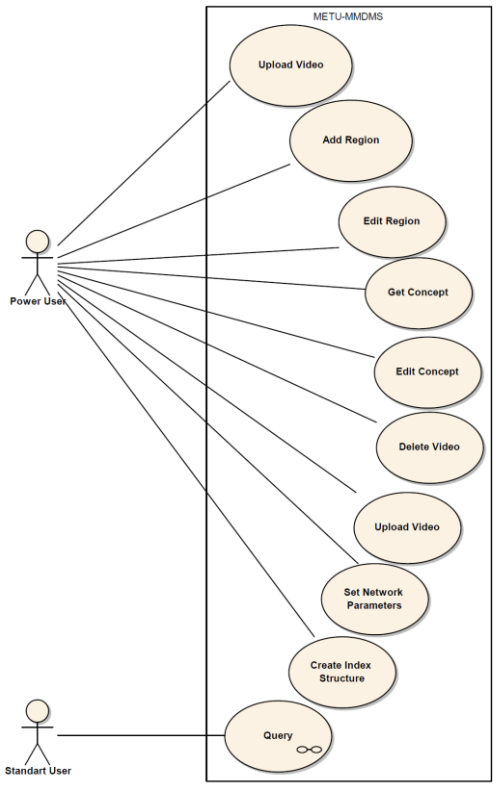


Figure 3-1 MMDMS Use Cases

In order to meet the functional features listed above, the Use Cases of the system include Upload Video, Add Region, Edit Region, Get Concept, Edit Concept, Delete Video, Upload Video, Set Network Parameters, Create Index Structure and Query as it can be seen from the [Figure 3-1] . The Query use case covers text based queries and similarity queries (e.g. Query by Content, Query by Concept, Query by Concept and Content, and Query by Example). Features traceability matrix is presented in [Appendix C].

3.2 Software Quality Goals

The desired states of some internal or external quality characteristic of a software product are described with the quality goals. The quality goals for METU-MMDMS are listed below (The accepted definitions of the quality attributes for this architecture are explained in Section 4, see [Table 4.2]):

[QG-1] Accuracy of the query results (increased by multi-modal querying capability via data fusion and query level fusion)

[QG-2] Performance of the queries (as the response time, improved by the help of the multidimensional index structure)

[QG-3] Scalability (database scalability through the multidimensional index structure)

[QG-4] Maintainability (adding new features and functionalities)

3.3 Selection of the Stakeholders

Which stakeholders will benefit from these views? We must find out the answer of that question in order to use the correct set of the views. How those stakeholders are identified? The answer will be that the stakeholders listed below are identified

according to the existing stakeholders of the METU-MMDMS. Being a research vehicle, this system has a much reduced variety of stakeholders compared to a typical enterprise information system. Thus, identifying the stakeholders and their concerns were not challenging.

One of the stakeholder groups of the METU-MMDMS, naturally, the *development team* needs documentation to understand a part or requirements of the system that they are responsible for, to see the system from bird's-eye view and to get the main ideas. Each sub-unit of multimedia database management system is a product of a separate study and in the different sub-units different developers are involved. Therefore, for the members of the development team, meeting at a common point, communication and understandability of the system highly depends on the proper documentation. The concerns of the development team can be listed as transferring the implementation and run-time information of the MMDMS.

Testing and integration team needs the black-box information of the system and its subsystems. This team is identified as a stakeholder because the METU-MMDMS consists of clearly separated modules, each of which is developed by different developers, and those modules should be tested and integrated. Different members carry out different stages of development and integration processes of the multimedia database management system. Providing the system architecture as input to the stages for this case is also important in terms of conducting the execution in a sound way. Understanding the implementation and run time behaviors of the MMDMS, and how the components are distributed among the external resources are the concerns of the integration team.

The research groups should maintain the continuity of the system. Since multimedia database management system is an R&D platform and R&D phases of the process continues, during the integration of the parts that different research groups developed, the necessity of compliance for the current parts of the system emerges. The research group is identified as a stakeholder because new features and functionality should be added to the METU-MMDMS for research purposes. The main concern of the research groups is adding new features/functionalities which

meet system's quality goals and avoiding the decrease in the performance or accuracy.

System users usually do not need to see the architecture of multimedia data management system, but, for the sake of the efficient use of the system, having a general knowledge about the architecture will be useful. The METU-MMDMS has some users therefore, they are identified as a stakeholder group. The users' concerns are gaining the capability of using the system and having fast and accurate results.

The system analysts require the documentation of the architecture to understand whether the system meets the desired quality attributes.

The other stakeholders are listed in [Table 3-2].

3.4 Choosing the Relevant Views

In METU-MMDMS architecture, the development team, integrators and maintainers are the same stakeholder. However, in [Table 3-2], it is shown as they are different stakeholders to give detail about the necessary views for them together with the necessity of the view for the corresponding stakeholder. Selected views and the reasons are explained in the following sections and excluded views are listed in [Table 3-1] together with their reasons of exclusion.

3.4.1 Module

Decomposition: The decomposition style is used to show the structure of modules and submodules [1]. The architect/Project manager/ the developer should know the modules and packages in order to advance the architecture/distribute the work order/carry on the development responsibilities. Integrators are not supposed to have detailed information on the decomposition; they just need the necessary interfaces and their locations in modules/packages. Analyst will need a brief knowledge to understand the system better. In METU-MMDMS, decomposition is used to assign desired functionaliy to units of design and implementation. Especially for the

semantic information extraction module, we can see that different sub-modules are implemented to process different modes and handle mode specific problems.

Table 3-1 Excluded Views

Style	Reason for exclusion
Generalization	Useful for extensions in the architecture, e.g. Adding, removing, and changing the children modules of a more generalized module. The possible extensions on METU-MMDMS architecture can be like adding a new mode and modes have different nature therefore they do not have a generalized way of processing.
Layered	The METU-MMDMS does not have a layered structure because the Coordinator component has two-way communication with all other components. The relations between this module and other modules are not unidirectional.
Peer to peer	The METU-MMDMS system has Client-Server style and the interaction is started from the Client. Unlike Client-Server style, each peer can start interaction in Peer-to-Peer style.
Publish-subscribe	In publish-subscribe style components are communicated via announced events. There is no publishing/subscribing structure in METU-MMDMS.
Work assignment	All modules are mapped to the researchers for the METU-MMDMS work assignment case. Therefore, there is no necessity of providing the work assignment style.
Implementation	We want to present a generic architecture and avoid too much detail.

3.4.2 C&C

Client-Server: Client-server style components interact by requesting services of other components [1]. The METU-MMDMS consists of clients (multiple for querying, single for information extraction purposes) and server. Therefore, using client-server style can be justified in terms of the original conception of the system. The client is implemented as thin-client, so that the services will be easily maintained on the server. Current and future architects should be aware of that and continue to the design accordingly. Project manager should keep track of, development team and integrators should be aware of this design decision via this view. It is not a must for the analyst to know the whole system, but he/she should have some knowledge to make logical reasoning about how the client server structure and the distribution of the components among client-server effect the performance & accuracy.

According to the concerns of the funding agency, the product will be the most important. To describe the product outer/black-box functions, it will be beneficial to provide the funding agency the overview of the client-server view, in this way, they will be aware of the usage of client as a black box, which needs server for functioning. The needs of the users in scientific community will be similar with the funding agency for client-server view.

Table 3-2 Selection of the Views

Stakeholder	Module Views			C&C Views					Allocation Views			
	Decomposition	Generalization	Uses	Layered	Pipe-and-filter	Peer-to-peer	Publish-subscribe	Shared-Data	Client-Server	Deployment	Implement.	Work assign.
Architect & Research G.	d		s		d			d	d	d		
Project manager	s		o		s			s	s	s		
Develop. team	d		d		d			d	d	d		
Integrators	o		d		s			s	d	d		
Maintainers	s		d		s			s	s	s		
Analyst	o		o		s			s	s	s		
Funding agency	o								o	o		
Users									o	o		
Key: d = detailed information, s = some details, o = overview information.												

Pipe & Filter: In a pipe-and-filter system, filters process the data input serially and send the output to the next filter through a pipe [1]. Choosing pipe and filter style can be justified on the grounds that Semantic Information Extractor component

works in a pipe-and-filter manner. The video information in Semantic Information Extractor is passed from one sub-component to the next, and information annotation can be considered as the filtering process. The architect, project manager and the development team are going to need the runtime perspective of the component in a detailed way.

Shared-Data: Repository systems where the data accessors are responsible for initiating the interaction with the repository are said to follow the shared-data style [1]. Choosing shared data style can be justified on the ground that the Multimedia Database component has a nature compliant with the shared data style. There is a single Storage component among the sub-components of the METU-MMDMS and it works as a shared resource for the queries from separate clients. During the development process, the runtime perspective of the storage component can be presented in the shared-data style.

3.4.3 Allocation

Deployment: In the deployment style, software elements native to a C&C style are allocated to the hardware of the computing platform on which the software executes [1]. Architect, manager, development team, and integrators need the deployment view to see the client server organization. The client and server machine dependencies and properties will be presented in this view. It is enough for a maintainer to know the information required for the possible maintenance activities on each selected view. The architecture document should enable maintainer to go deep in the system architecture when necessary. Therefore, the architectural information related to the predicted maintenance activities should also be documented. Deployment view of METU-MMDMS shows that which components should be deployed on client side, which are on server side. Choosing deployment style can be justified on the grounds that the system will grow to become a distributed system or a system that handles big data and the presented deployment diagram will be a basis for such a system to denote the distribution of the components on the hardware.

3.5 Architectural Views

METU-MMDMS modules are developed by multiple teams of researchers, therefore if a common modeling language and tool is used, it will be easier for the researchers to see the overall picture, communicate and contribute. Therefore, Enterprise Architect (EA) 7.5 and UML 2.0 are used to model the MMDMS architecture and the views are produced as wiki pages for stakeholders to reach them anytime/anywhere with a clear format. Classical top-down approach is carried out throughout architectural modeling activity since it simplifies the construction process of models, modules, components and even connectors. As the architecture description gets more detailed, it begins to cover implementation details and becomes harder to maintain because of the inclusion of rapidly changing properties. Modeling activity should stop at a point. To avoid too much implementation detail, the architecture is documented at package level for module views. The decomposition module views are explained in the following sections and the uses information about the views are presented in Appendix E. The classes/servlets are included in the documentation when they are strictly necessary. C&C views are constructed with components having one to one mapping with the modules in module views. Note that classes are static structures, shown in a module view, so they should not be included in C&C views. Finally, ports are numbered to easily track the connections among different models.

3.5.1 Top Level Module View

A thin client application is developed for the MMDMS. The Server module covers the Semantic Information Extractor, Multimedia Database and Coordinator modules.

Semantic Information Extractor is the module which implements object, event and concept extraction processes. DB4O database and JESS knowledge base are used in Multimedia Database as storage and fuzzy inference engine, respectively. A Multidimensional Index Structure is integrated for efficient data access. Although construction of a separate index structure is possible, since it causes a higher

performance increase, combining all the dimensions in one index structure is preferred as a design decision.

From the variability perspective, the Client Application can be re-designed in order to meet new types of queries. Furthermore, the fields of queries can be rearranged and the classification algorithms can be replaced with other suitable algorithms. As a design rationale, in order to meet the maintainability quality attribute, the system is structured in a modular way.

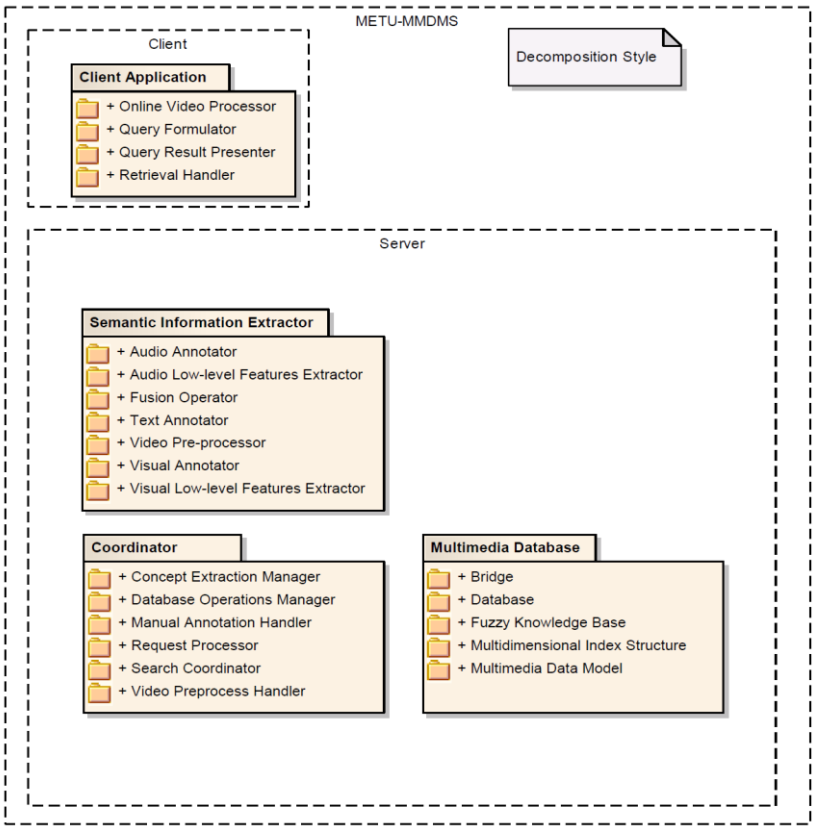


Figure 3-2 Top Level Module View – Decomposition Style

3.5.1.1 Module View of Client Application

Client Application consists of four components, namely, Retrieval Handler, Query Result Presenter, Query Formulator and Online Video Processor [27]. Query

formulator helps creating textual and similarity based queries and object-event annotation function calls.

Retrieval Handler is responsible for passing the constructed query to the server and receiving the results from Coordinator component. Query results are displayed visually and textually at the Query Result Presenter. Online Video Processor is utilized for updating, deleting, uploading, annotating and manipulating videos.

As a design rationale, the heavy loaded functions are moved to the server components to avoid a potential performance loss at the Client Application.

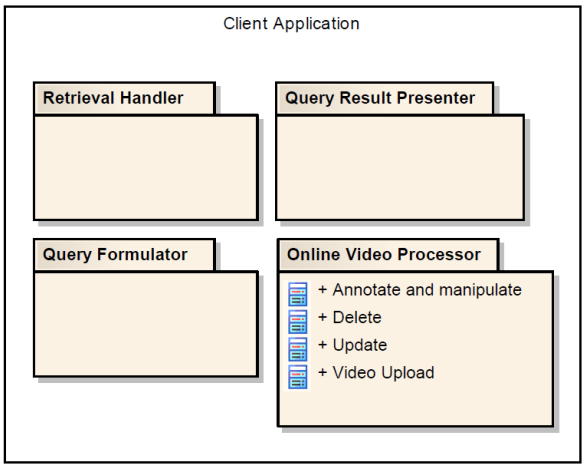


Figure 3-3 Module View of Client Application – Decomposition Style

3.5.1.2 Module View of Semantic Information Extractor

Before extraction, annotation and fusion operations are performed; the video input should be pre-processed to prepare suitable inputs for extraction, annotation and fusion modules. Video transcoding, automatic shot boundary detection, important frame extraction and key frame segmentation take place at this module. Video transcoding module carries out direct digital-to-digital data conversion of one encoding to another. As a preliminary step of annotation, the shout boundaries, which are shot-to-shot transitions, are detected by Automatic Shot Boundary Detector module [26].

The important frames of the video are extracted by the IBM MPEG Annotation Tool at IFrame Extractor module. Key frame segmentation unit performs the segmentation of the extracted important frames. Audio/Visual Low-level Features Extractor modules extract the low-level features required for audio and visual annotation modules to operate. The extracted features are listed at [

Figure 3-6] [Figure 3-7].

Visual event annotator is developed to annotate visual objects and events. The image segments are inputs for the object extraction phase. Depending on a domain ontology, events are detected and extracted via the evaluation of the spatial and temporal relations between the extracted objects. Audio annotator includes three modules: Acoustic Classifier, Semantic Classifier and Audio Learner. Hidden Markov Models (HMM) and Support Vector Machines (SVM) are used for Acoustic classification. Acoustic classification results are inputs for the semantic classification, which distinguish the higher-level classes (e.g. outdoor). Text annotator does not go into a low-level feature extraction process. Named entity recognizer detects named entities in the video text that can be classified into predefined categories (name of person, organization, location, etc.).

To enrich its resources, the named entity recognizer is enhanced with a rote learner module. In the fusion module, the information obtained from different modalities enters a late fusion process in order to increase the detection accuracy of the objects/concepts/events.

Module View of Visual Annotator

Visual annotator employs Visual Object Annotator and Visual Event Annotator modules for extracting and annotating objects and events. The object annotator performs learning phase with the best representative feature set obtained from the feature comparator component. Finally, GA-based classifier determines possible objects. After the objects are forwarded to the Visual event annotator, spatial

relations and movements are calculated to find the spatial changes at Spatial Changes Extractor module.

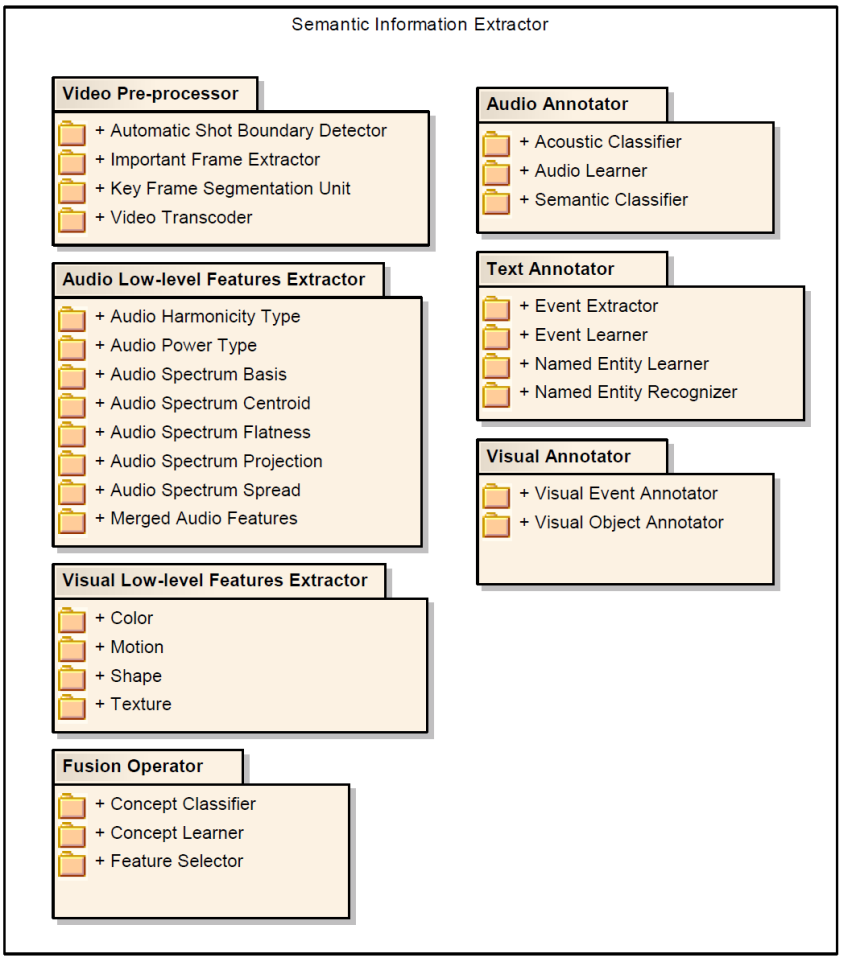


Figure 3-4 Module View of Semantic Information Extractor – Decomposition Style

Temporal spatial changes extractor module detects temporal spatial changes via the relations of temporal and spatial changes by using domain ontology. The outputs of spatial changes extractor and temporal spatial changes extractor help event extractor determine events by checking similarity (similarity relations extractor) with the defined events in domain ontology [38].

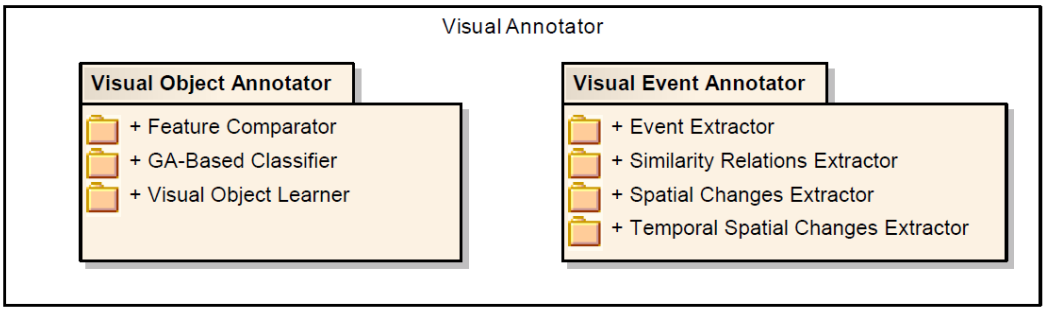


Figure 3-5 Module View of Visual Annotator – Decomposition Style

Module View of Visual Low Level Features Extractor

The MPEG-7 features that are extracted for annotation processes are presented in [

Figure 3-6]. Visual low-level feature extractor completes the extraction process of the color, shape, motion and texture features.

Module View of Audio Low Level Features Extractor

The MPEG-7 low-level descriptors that are extracted for annotation processes are presented in [Figure 3-7]. Audio low-level feature extractor completes the extraction process of those audio harmonicity type, audio spectrum basis, audio power type, audio spectrum centroid, audio spectrum flatness, audio spectrum projection, audio spectrum spread and merged audio features.

3.5.1.3 Module View of Coordinator

Search coordinator is able to do multimodal querying, querying on fused data, content/concept based, query by example type searches and their combinations. The

query requests are initially transferred to the request processor and query analyzer determines the query type and analyses query content. Query level fusion aims to improve query results by fusing multimodal data at query level. Concept extraction manager controls the concept extraction process.

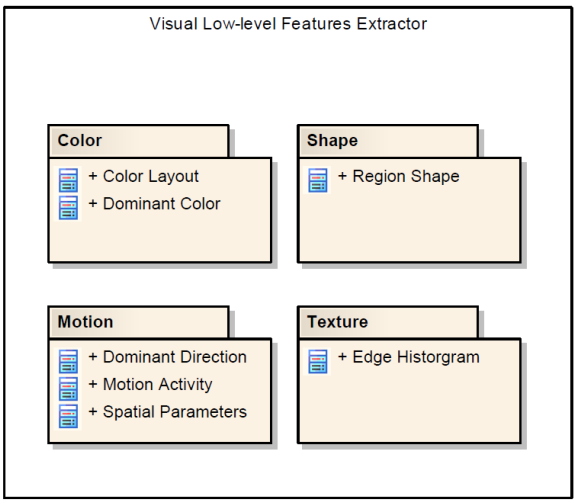


Figure 3-6 Module View of Visual Low Level Features Extractor – Decomposition Style

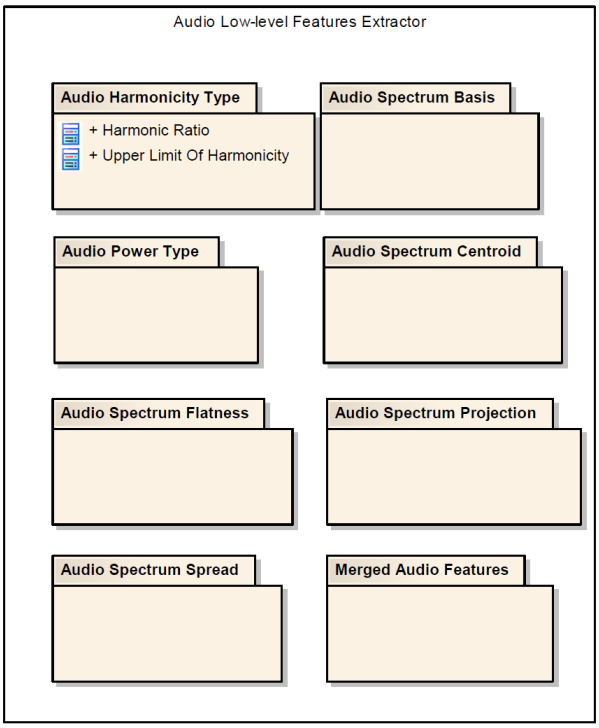


Figure 3-7 Module View of Audio Low Level Features Extractor – Decomposition Style

Video preprocess handler triggers video preprocessor module of semantic concept extractor module before extraction and annotation phases begin. In order to correct the automatic annotation results and improve information retrieval precision, manual annotation option is also provided to the user. Manual Annotation Handler module is developed for text/audio/visual manual annotation purposes. Database operations such as video upload, update, delete are coordinated by the Database Operations Manager module [27].

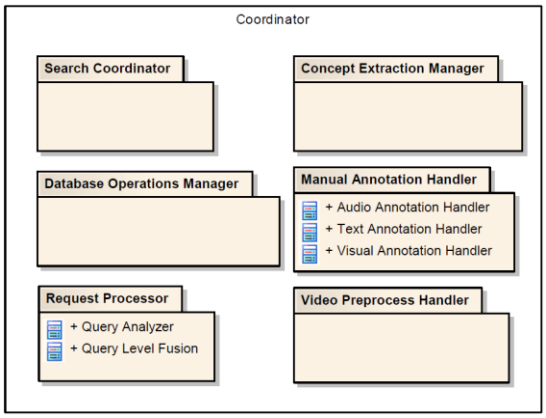


Figure 3-8 Module View of Coordinator– Decomposition Style

3.5.1.4 Module View of Multimedia Database

Multimedia Database module view stores multimodal information and includes a fuzzy knowledge base to apply fuzzy inference rules. The user requests are initially transferred to the Bridge module as the abstraction layer of the Multimedia Database. The Bridge module manages the coordination and communication between the Database and Fuzzy Knowledge Base modules. The object oriented Database module covers the Audio, Text, and Visual Databases. For efficient data access, a Multidimensional Index Structure is developed [27].

3.5.1.5 Multimedia Data Model

The data model is the blueprint for the implementation of the data entities [42].

In order to represent complex multimedia objects, a conceptual multimedia data model is constructed. The classes and their definitions are listed in [Table 3-3]. In [Figure 3-10], entities, relationships and the hierarchical structure of the data model are shown.

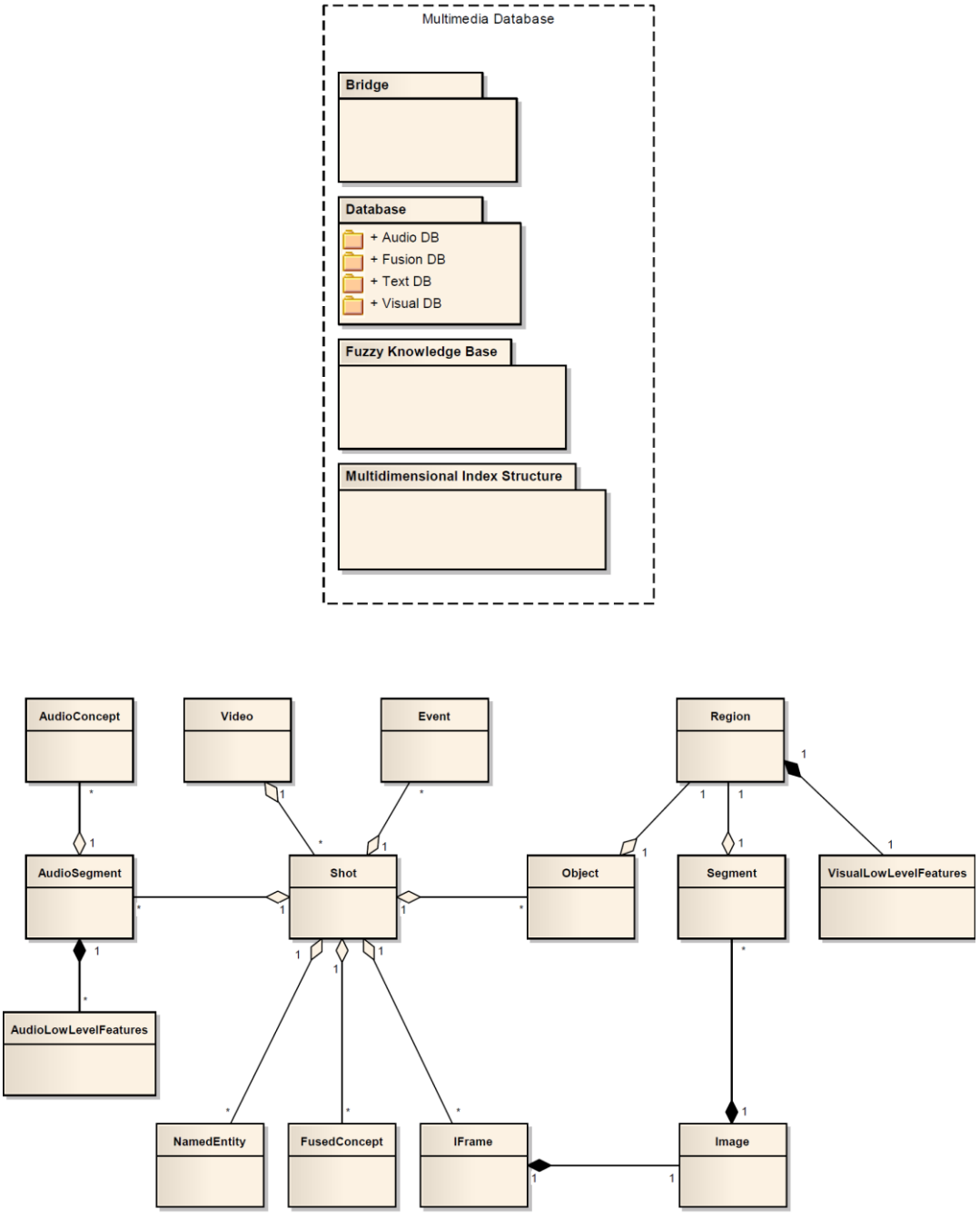


Figure 3-10 Multimedia Data Model

Table 3-3 Classes and Definitions

NAME	DESCRIPTION
Audio Concept	Audio concept is the audio information that is obtained by analyzing the relationship between feature, object and other concepts.
Video	Video objects are composed of sequences (shot-scene-sequence-video hierarchy).
Event	Events are interactions between concepts, objects, and spatial/temporal relations.
Region	The selected portion of an image is region.
Audio Segment	Audio segments are used for detecting object and event boundaries.
Shot	The smallest temporal segments are shots. They are defined as the minimal group of adjacent frames, stating a continuous action and having images from the same area, therefore contain some common low-level features.
Object	Object is the independent information that is gathered by analyzing the video features and will take the actor role in an event.
Segment	Sets of pixels. (In computer vision, image segmentation is the process of partitioning a digital image into multiple segments. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.)
Visual Low Level Features	MPEG-7 visual low level features.
Audio Low Level Features	MPEG-7 audio low level features.
Named Entity	The named entities in a text will be listed as "person", "location", "organization", "time", "date" and "money".
Fused Concept	Audio, visual and text concepts are combined to form a fused concept.
IFrame	Frame is the smallest temporal granularity for video objects, which can also be represented with image. IFrame defines starting and ending points of any smooth transition.
Image	Image is an array, or a matrix of square pixels (picture elements) arranged as columns and rows.

3.5.2 Top Level C&C View

From the runtime perspective, METU-MMDMS has a Client-Server C&C view style. Textual and similarity based queries are built using the interfaces of Client component. In addition, object and event annotation functions are invoked from the relevant interfaces in this part. Sending the constructed query to the server and receiving the results coming from Coordinator are also the Client's responsibilities. Coordinator component has an interface role among other components and runs as a facade. Semantic Information Extractor component provides information to be stored in Multimedia Database via Coordinator component. Multimedia Database is an intelligent information management system, which offers an efficient interaction between database and knowledge base.

In METU-MMDMS architecture, as a requirement of thin client technology, all the operational functions are recommended to be executed in server side components and the supplementary functions, which mostly require user interactions, are suggested to be performed on Client Application. In order to obtain a scalable system, an extensible and flexible Coordinator structure should be developed. In this way, new high level components can be added to the system with minimum effort and the additional data load can be managed by the Coordinator component.

3.5.2.1 C&C View of Semantic Information Extractor

Semantic Information Extractor works in a pipe-and-filter manner. Video Preprocessor component is responsible for video transcoding, automatic shot boundary detection, important frame (IFrame) extraction and key frame segmentation.

The preprocessed video is sent to Audio/Visual Low-level Feature Extractor components before the audio/visual annotation phase. Text annotation phase starts immediately after the video-preprocessing activity. Video preprocess and annotation

phases are triggered by the Coordinator component. Duration of annotation differs for each modality.

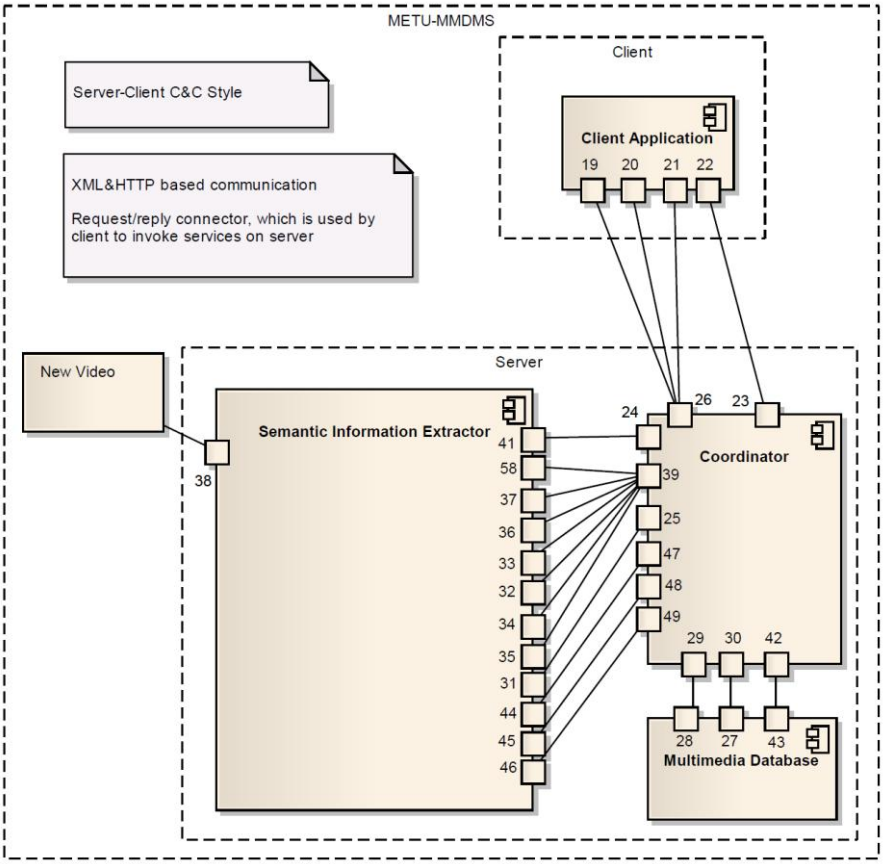


Figure 3-11 Top Level C&C View – Client-Server Style

Besides, excessive memory consumption occurs at the time of annotation. Therefore, annotation results are first stored into the Multimedia Database through the Coordinator structure.

Afterwards, in the fusion step, the stored annotation results are extracted from Multimedia Database in order to generate fused information extraction results via Fusion component. Data level fusion outperforms single modality approach in information extraction. The classification methods in Audio/Visual/Text Annotator are interchangeable with others. The goal for Semantic Information Extractor component is high accuracy and performance.

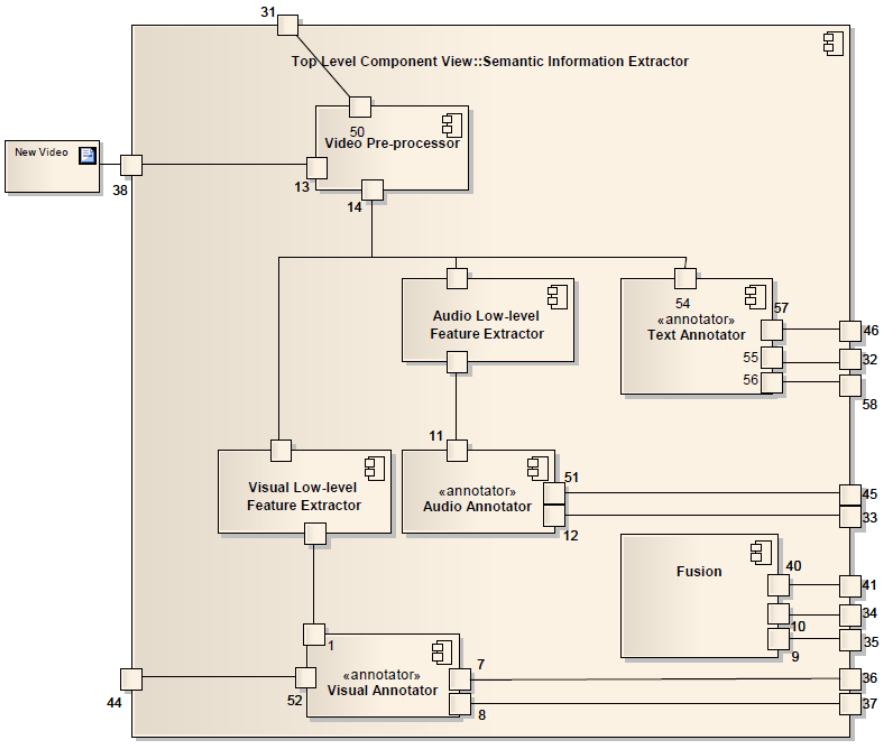


Figure 3-12 C&C View of Semantic Information Extractor – Pipe and Filter Style

The flow of semantic concept extraction activities are strictly controlled by the Coordinator component. Various Semantic Concept Extractor module functionalities can be triggered by Coordinator's components (see port numbers). The reason of that design decision is that the duration of the activities differs among modalities. For example, the annotation process for the audio, visual and text dimensions are different and they are annotated separately and stored in the database afterwards. In addition, for the fusion process, the required annotations are extracted from the multimedia database. To indicate that the annotator components can have some common/similar properties, as a modeling decision, their types are stated as "annotator" with the angle brackets in [Figure 3-12].

C&C View of Video Pre-processor

Video preprocessor component, triggered by the Coordinator-Video Preprocess Handler, prepares the suitable data for the annotation and extraction components. First, the Video Transcoder component converts the video into a suitable format for the successor processes.

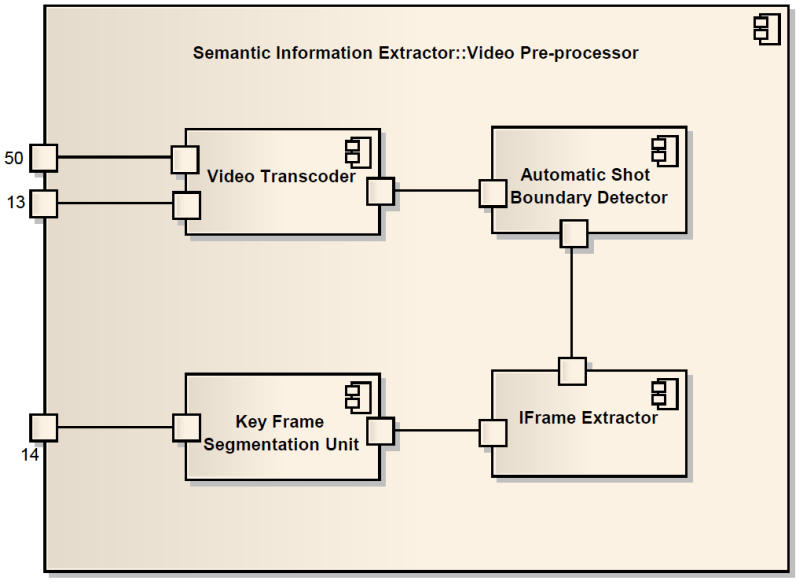


Figure 3-13 C&C View of Video Pre-processor – Pipe and Filter Style

C&C View of Text Annotator

The text is extracted from the video segments. The name entity recognizer processes this text to extract the named entities. The text is also input to the Event Extractor component. By using Non-disjoint Event Text Groups, Event Learner component

finds out the Common Event Keywords. These frequent keywords take part in the event extraction process.

Named entity recognition results are enhanced with the help of the Named Entity Learner component. The resources of text annotation are extended since models for the new resources can be created as the output of the learning activities [39].

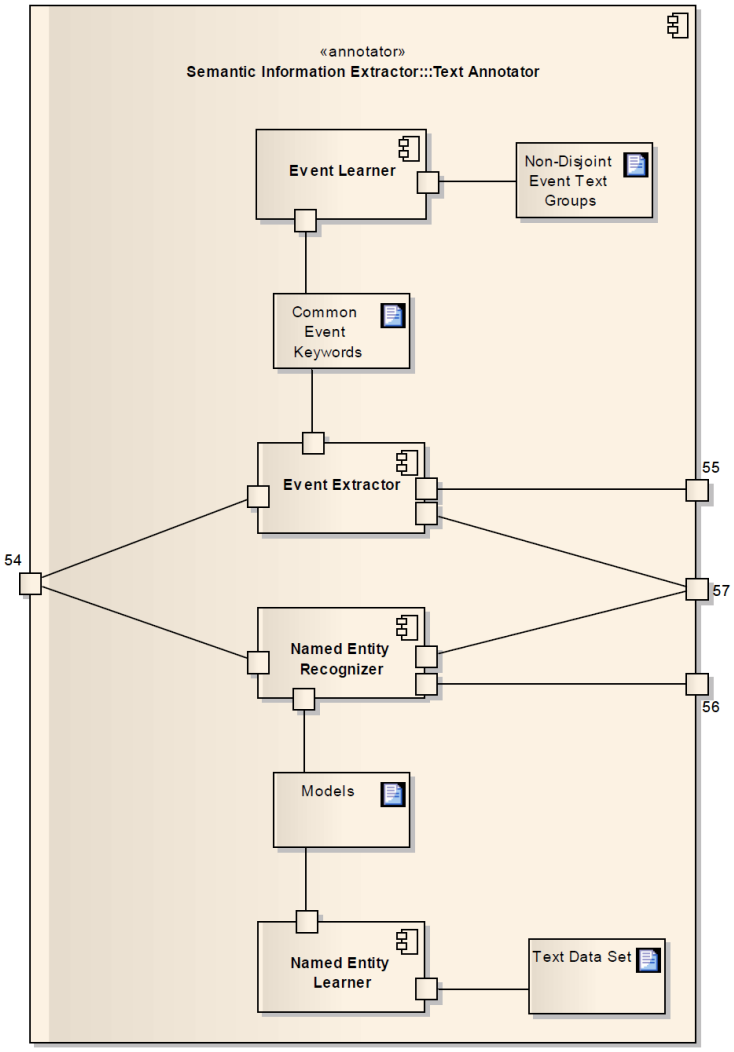


Figure 3-14 C&C View of Text Annotator– Pipe and Filter Style

C&C View of Audio Annotator

Audio annotation activity is carried out in this component. The separated audio is divided into silence and non-silence segments. Non-silence segments are classified

by the Acoustic classifier component. The acoustically classified segments are transferred to the Semantic Classifier component and enter a smoothing process. Afterwards, those segments are classified into semantic (outdoor, nature, violence, meeting,etc.) classes. Learned Concept Models plays role in Acoustic Classification process. The Concept Models are outputs of the Support Vector Machine (SVM) and Hidden Markov Models (HMM) learning techniques. MPEG-7 audio features, Mel Frequency Cepstral Coefficients (MFCC) feature and Zero Crossing Rate (ZCR) feature are entered to the learning procedure [37].

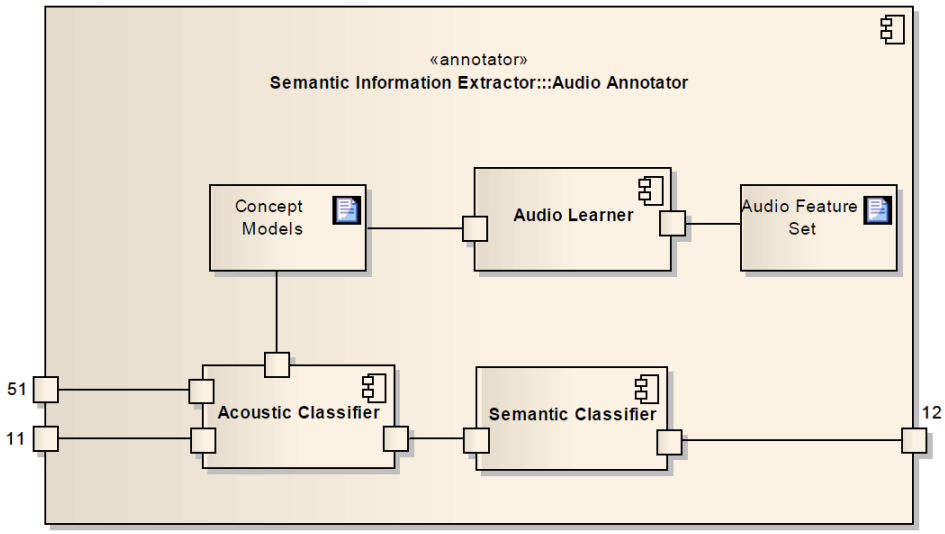


Figure 3-15 C&C View of Audio Annotator – Pipe and Filter Style

C&C View of Visual Annotator

Visual annotation is achieved through two components: Visual Object Annotator and Visual Event Annotator. The Visual Low Level Feature Extractor forwards low level features to the Visual Annotator component. Depending on the extracted features, visual objects are annotated. The annotated objects and spatio-temporal relationships between them are passed to the Visual Event Annotator component to complete the annotation process. Visual events and objects are transferred to the coordinator separately to be stored in multimedia database.

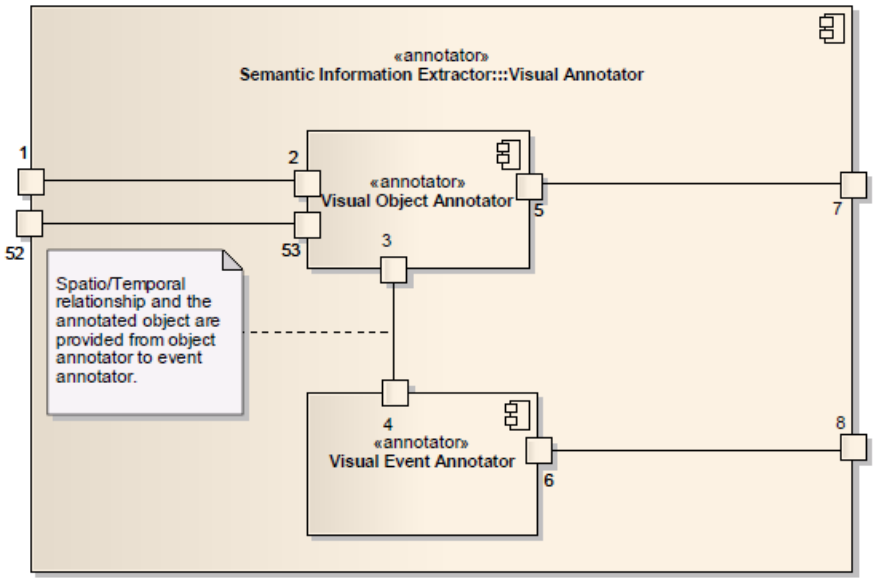


Figure 3-16 C&C View of Visual Annotator – Pipe and Filter Style

C&C View of Visual Event Annotator

Visual Event Annotator takes Visual Objects and spatio-temporal relations from the visual object annotator. By analyzing the spatial relations, the spatial changes extractor finds out the spatial changes of objects. Temporal Spatial Changes Extractor adds temporal relations defined in the domain ontology onto the spatial changes.

The spatial relations, spatial changes and temporal spatial changes are evaluated according to the event definitions in the domain ontology. If they satisfy corresponding event rule definitions, event extraction will be successful [38].

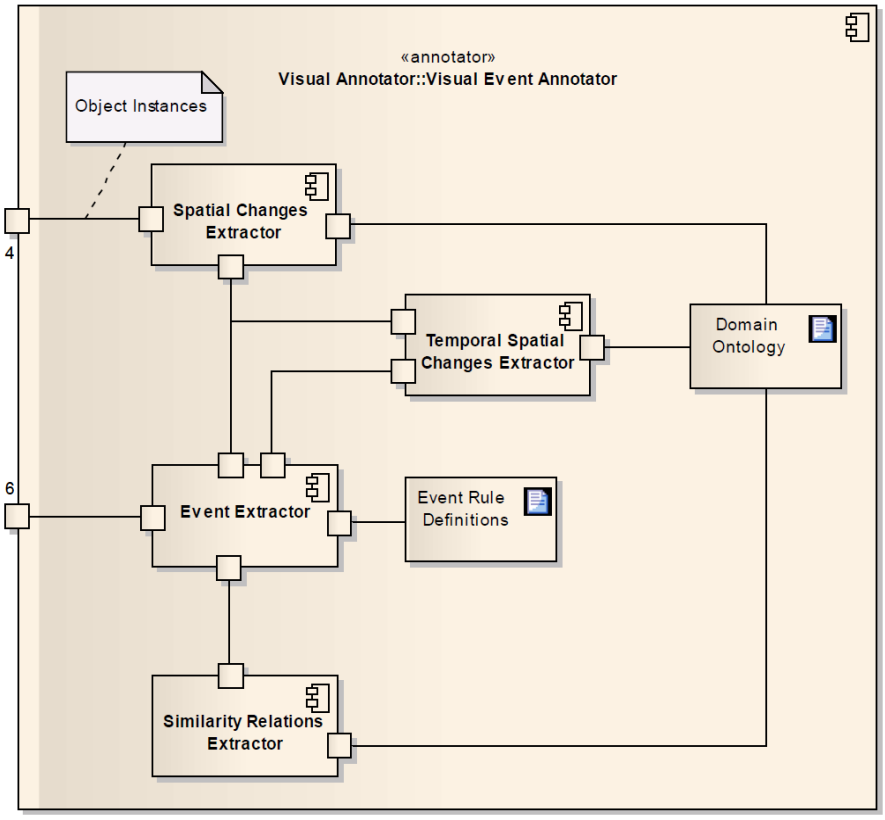


Figure 3-17 C&C View of Visual Event Annotator – Pipe and Filter Style

C&C View of Visual Object Annotator

Object Classifier component conducts the visual object annotation together with the Feature Comparator and Visual Object Learner components. MPEG-7 features are employed during classification and learning processes. Object models belonging to different object classes are produced as an output of the Visual Object Learner component that performs learning activity depending on the objects' features. The objects are distinguished according to the object models. Furthermore, Feature Comparator can compare the features of separate objects and if there is an acceptable similarity amount between those features, the objects can be combined as the parts of the same object [38].

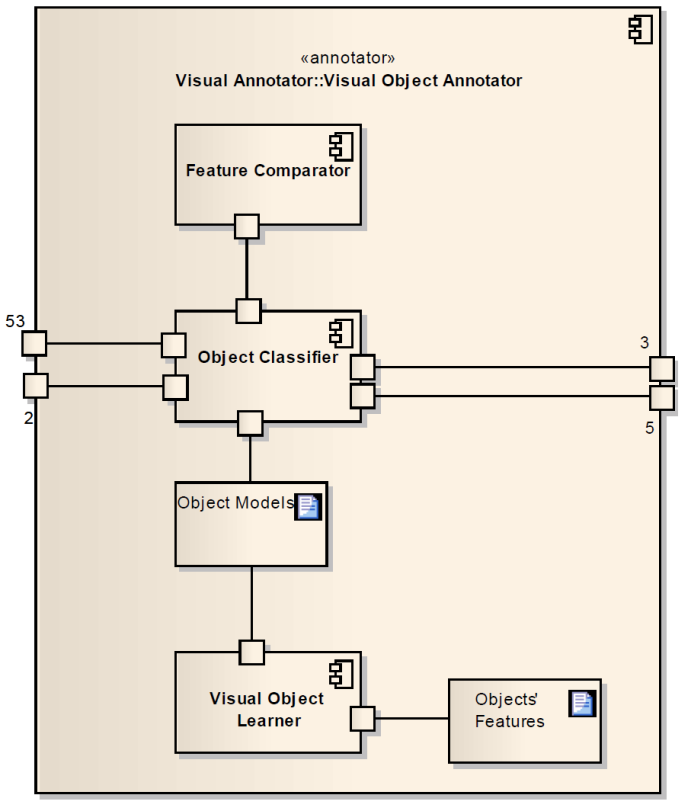


Figure 3-18 C&C View of Visual Object Annotator– Pipe and Filter Style

C&C View of Fusion

Fusion module combines the information obtained in all three modalities to form fused concepts appropriate for the Concept Models. Feature selection module calculates the weights of all features according to the training data and eliminates the features having the weight values below the threshold.

After the training data is transferred to the SVM format, it passes into the concept learner. Then the concept learner constructs the Concept Models after some series of processes.

In the testing phase, the test data is formed according to the scores obtained from several modalities and then given to the concept classifier to create a new concept or generate a new integrated concept score [36] [41].

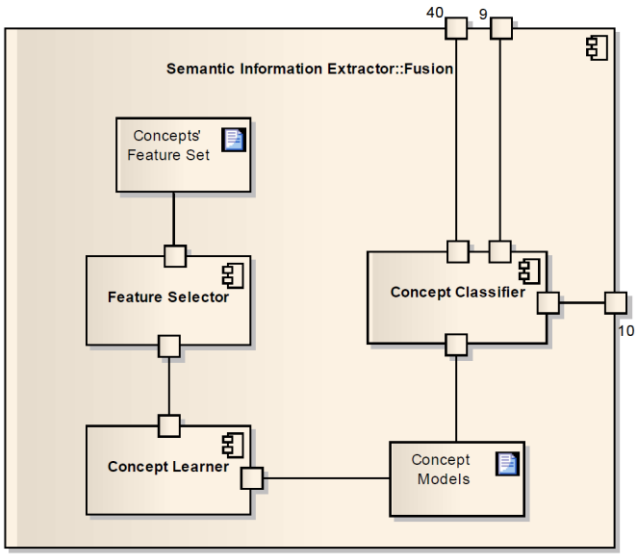


Figure 3-19 C&C View of Fusion – Pipe and Filter Style

3.5.2.2 C&C View of Multimedia Database

Multimedia Database component consists of the Bridge, Multidimensional Index structure, Knowledge Base (Jess), Object-Oriented Database Management System (DB4O) and the Multimedia Database components. Bridge is utilized for managing communication and interaction between the DB4O system and Knowledge Base. It is used as an abstraction layer and entry point for user requests. Multimedia Database component stores visual/audio/text object/concept/event information. In order to deal with uncertain information, the system uses a fuzzy knowledge base in conjunction with an object-oriented database. Scalability is the major quality factor for the Multimedia Database, especially for Multidimensional Index Structure.

Although DB4O has its own B+ tree index, a Multidimensional Index Structure has been developed to perform query by concept and content together (first search conceptually and find candidates, then do similarity matching on resulting candidates). Multidimensional Index Structure also increases efficiency and accuracy of semantic querying.

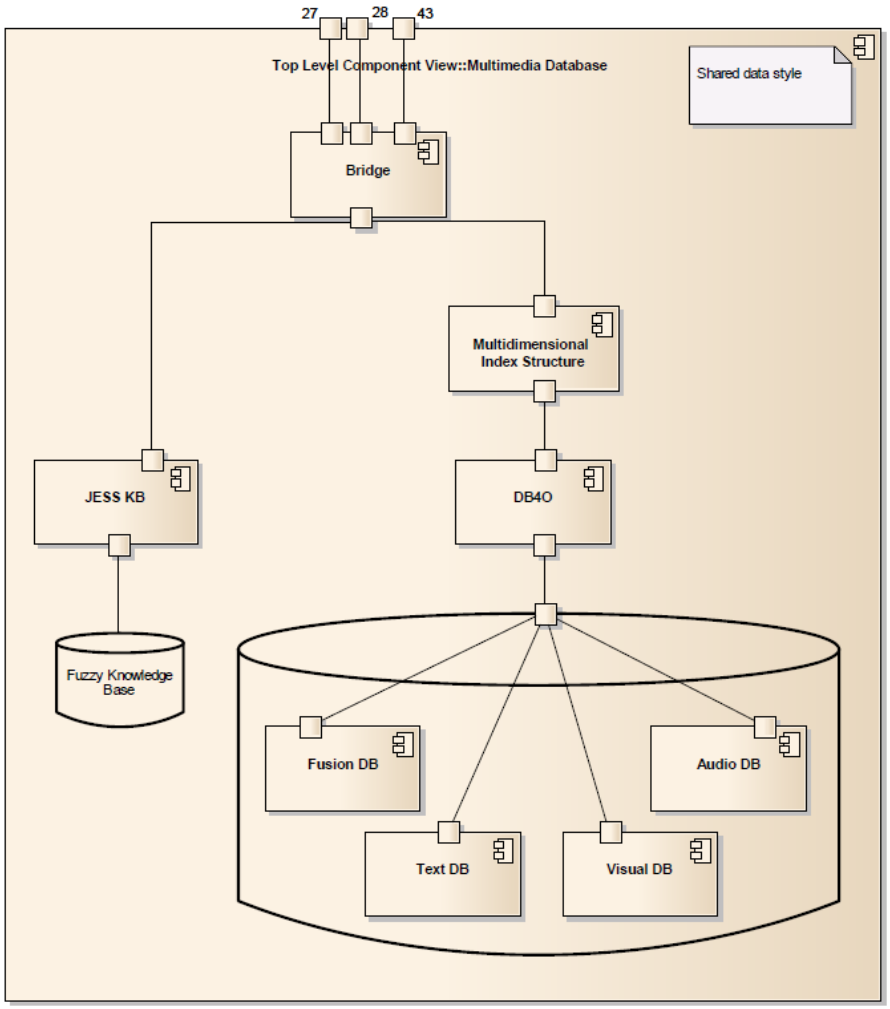


Figure 3-20 C&C View of Multimedia Database – Shared Data Style

Increasing the dimension of the Multidimensional Index Structure can be considered as a variability point. However, as the dimension of Multidimensional Index Structure increases, its performance decreases drastically. This is known as "the curse of dimensionality". Since an object-oriented database is used, the size of the objects also affect the insertion and retrieval time (performance).

3.5.2.3 C&C View of Coordinator

Request Processor component gets the request from Client. If the request is a query, it is passed to the Query Request Processor. By the help of the Query Analyzer, the

query is forwarded to Search Coordinator or Database Operations Manager depending on the query content. Query results are transferred to the Query Level Fusion component via the Query Analyzer. In addition to data level fusion, query level fusion can increase retrieval precision.

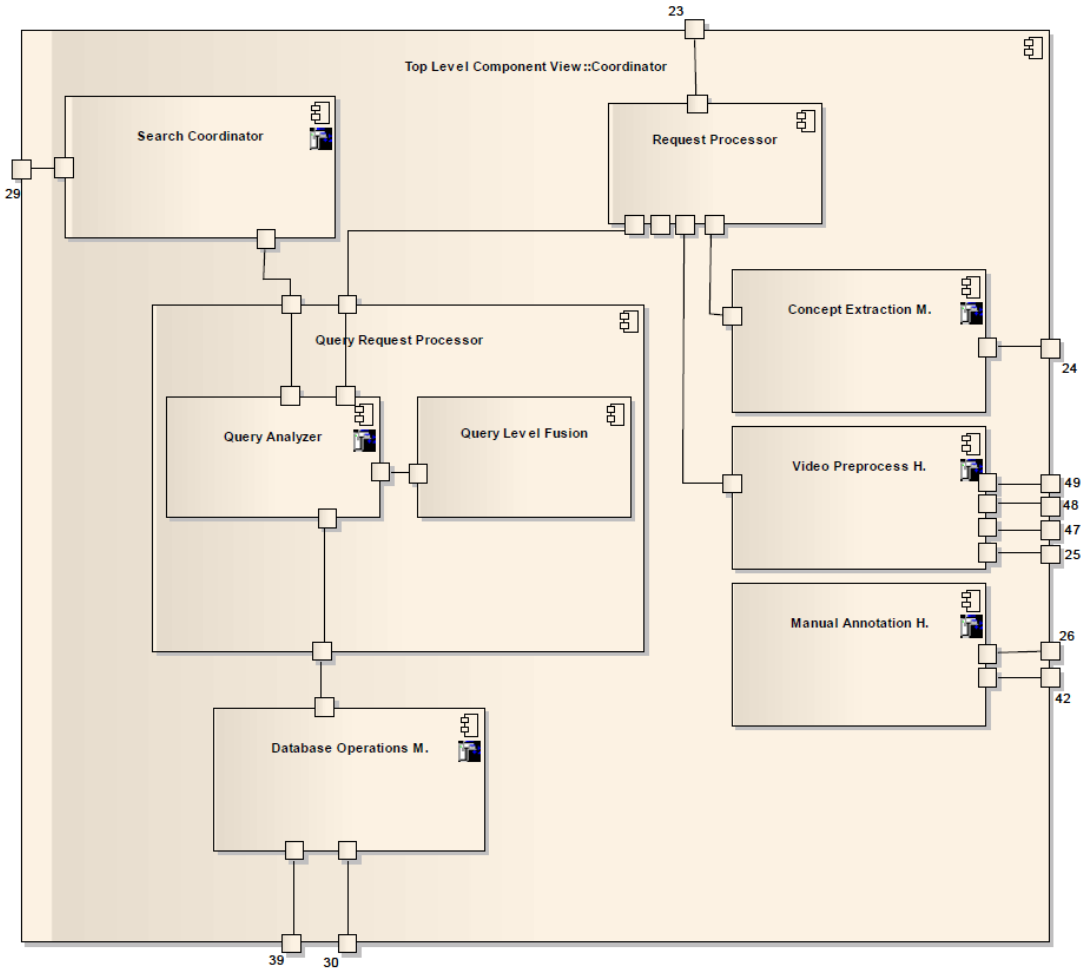


Figure 3-21 C&C View of Coordinator

The fused results are presented to the user. Other requests such as concept extraction, video preprocessing and manual annotation are directed to the related components directly from the request processor [27].

To support different types of components, developed in different environments and platforms, the Coordinator structure should be flexible and extendible to make METU-MMDMS architecture more flexible.

Therefore, Coordinator is developed in a web-based manner, using servlet instances, since servlets are fast and easy to use. Servlet templates can also be provided for search, database operations, concept extraction and manual annotation servlets. The realized structure of Coordinator component is expected to be easily manageable and maintainable, as it needs only minor modifications during the integration of new components.

3.5.2.4 C&C View of Client Application

A multithreaded client application has been developed for querying, online video processing and query result presentation purposes. The main functionality of the Client is controlled from Query Formulator where textual and similarity based queries are built and object-event annotation functions are invoked. Retrieval Handler sends the constructed query to the server and receives the results from Coordinator component. Query Result Presenter displays query results in multimode and supports result correction via manual annotation. Online Video Processor is used to update, delete upload, annotate and manipulate videos. Thin client is suitable for performance purposes, with time-consuming functionalities moved to the server components [27].

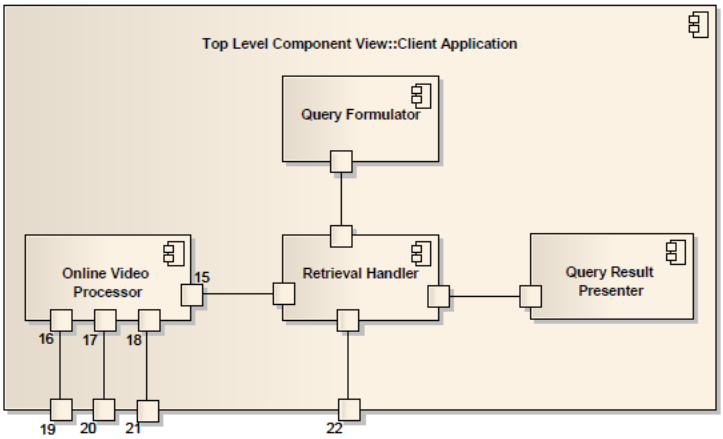


Figure 3-22 C&C View of Client

Experts, who have experience with the system and have the ability to construct effective queries accordingly, construct the queries in METU-MMDMS. To increase usability of the system and increase retrieval effectiveness, query recommendation functionality should be added to the client

3.5.3 Top Level Allocation View

Client is the node on which the client component runs. Server node provides execution environment for Semantic Concepts Extraction, Coordinator, Multimedia Database components. Centralized server eases the system maintenance. Interaction between server and client instances is established using both XML and HTTP messaging standards. Therefore, any other client application can easily connect to the server framework and use the implemented sample multimedia database framework so long as they satisfy the requirements of messaging protocol.

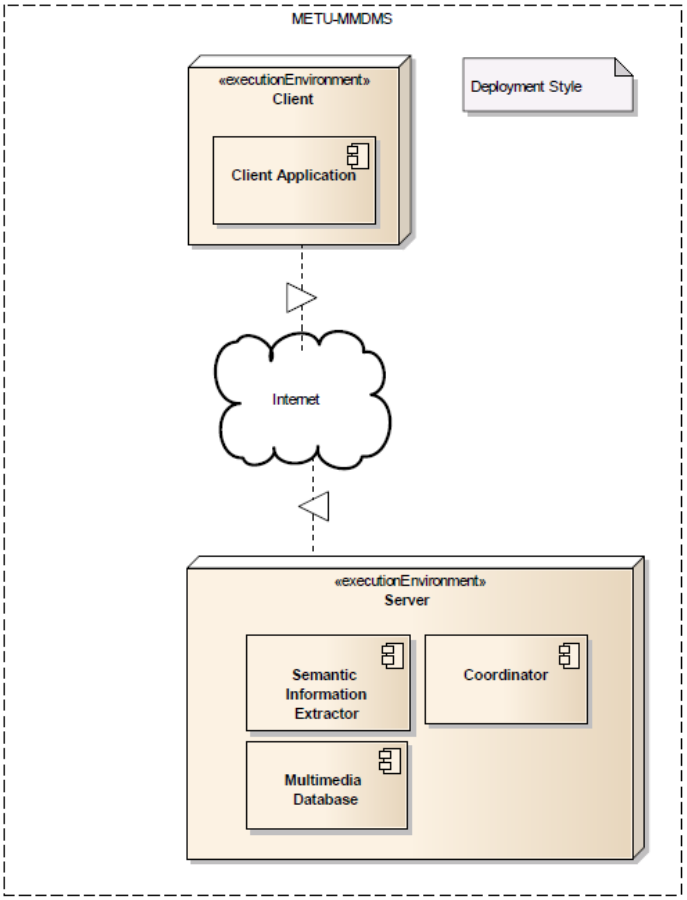


Figure 3-23 Top Level Allocation View - Deployment Style

The client is implemented as thin-client, so that the services will easily be maintained on the server. The workload is on the server to be able to complete client operations efficiently on the client side.

3.6 Interface Documentation

Software components communicate with each other through the interfaces and the system cannot be analyzed appropriately without the required information about the software interfaces. Therefore, for a complete architectural documentation the necessary details of the interfaces should also be included in the document.

According to [1] , "An interface document is a specification of what an architect chooses to make publicly known about an element in order for other entities to interact or communicate with it.". The principles about software interfaces can be listed as follows:

- Each element has at least one interface,
- The interface of an element should be separated from the implementation,
- Interfaces are provided and required by the elements,
- An element can interact with more than one actor via an interface,
- Generalization can be used to extend the interfaces.

While documenting the interfaces, the externally visible interactions with the other elements should be considered. The unnecessary information or the information that the user is not supposed to know should not be documented. The interface which is used in multiple views should be documented only once at the point where it is more beneficial. Moreover, the interface documentation should be specific and precise. An interface document should be organized as the following sections:

- Interface identity
- Resources (syntax, semantics, error handling)
- Data types and constants
- Error Handling

- Variability
- Quality attribute characteristics
- Rationale and design issues
- Usage guide

The Coordinator component has interface with each high level component. Online Video Processor and Query Formulator are the interfaces that have the highest importance among coordinator interfaces. Each interface provides multiple *resources*, which are described in the next sections.

3.6.1 Online Video Processor Interface

Online Video Processor interface provides resources for video uploading, updating, deletion, annotation and manipulation. The most important resources that it uses are "XML ExtractShotBoundariesandIFrame", "getMBR" and "getFrameRegion Annotation". The details of the interface are listed in [Table 3-4].

Table 3-4 Online Video Processor Interface Details

Resource	**Pre-condition**	**Post-condition**	**Error Handling**
bool XMLExtractShotBoundariesAndIFrame (string videoPath) The function that extracts shot boundaries and IFrames and saves temporal information about shots and IFrames in XML files and database.	Initialization must be completed (set path for all video/image files in db, set database parameters, connect to database)	Shot boundaries and IFrames should successfully be created. XML files should be formed. Information about shots and IFrames should be stored in database.	FileNotFoundException: No video in given path/created XML is not found or IFrames are not found. SAXException: Any problem about XML parsing.

(Table 3-4 continued)

Resource	Pre-condition	Post-condition	Error Handling
bool getMBR (string IFramePath) For each key frame and in a key frame for each region, this function finds the maximum bounding rectangle (MBR). Returns true if and only if it is successfully completed.	Shot boundaries and IFrames are extracted. The selection of segmentation algorithm is done and appropriate parameters are selected.	Segmented image file are formed. Map file that shows which pixel belongs to which region is constructed. For each IFrame some region is extracted as bounding box and be stored in the database.	FileNotFoundException: No video in given path/created XML is found or IFrames are not found.
bool getFrameRegionAnnotation (int currentSessionId) For each MBR, this function finds the class that this region corresponds to. Returns true if it is successfully completed.	MBRs are extracted. The classifier is trained and initialized.	The function should output the class that MBR maps and its degree of mapping.	ObjectAnnotationExcepti on: No result is returned from annotation process - internal error in annotation component.

3.6.2 Query Formulator Interface

Query Formulator Interface supports the textual and similarity based queries that are built by using the Client Application GUI and transferred via this interface. The most important resources that it uses are “queryAnalyzerServlet” and "getShot". The details of the interface are listed in [Table 3-5].

Table 3-5 Query Formulator Interface Details

Resource	Pre-condition	Post-condition	Error Handling
bool queryAnalyzerServlet (string queryString)	Connection to database is established.	Ranked list of shots are extracted.	IOException: The necessary files for indexing are not found. DatabaseConnectionException QueryStringParseException
bool getShot (string queryString) Among the list of shots, this function retrieves the selected shot in a video and plays in query result presenter.	Intended shot is selected.	A byte stream which represents the requested video is produced.	JMFInternalException: JMF player throws the exception. IOException:Requested video is not found.

3.7 Mapping

One or more elements in a view may correspond to one or more elements in another view. The mapping elements between the architectural views of METU-MMDMS are presented in the form of tables. After a small analysis process, the views for which we should provide explicit mapping are selected.

Table 3-6 Module-Component Mapping Example

Top Level Module View (Decomposition)	Top Level C&C View (Client-Server)
Client Application	Client Application
Semantic Concept Extractor	Semantic Concept Extractor
Coordinator	Coordinator
Storage	Storage

Table 3-7 Hierarchical Mapping Example

Top Level Module View Components Decomposition	Level-1 Decomposition	Level-2 Decomposition	Level-3 Decomposition
Client Application	Retrieval Operator		
	Query Result Presenter		
	Query Formulator		
	Online Video Processor	Annotate and Manipulate	
		Delete	
		Update	
		Video Upload	
Semantic Information Extractor	Video Pre-processor	Automatic Shot Boundary Detector	
		Important Frame Extractor	
		Key Frame Segmentation Unit	
		Video Transcoder	
	Audio Annotator	Acoustic Classifier	
		Semantic Classifier	
		Audio Offline Training	
	Text Annotator	Event Extractor	
		Event Offline Training	
		Named Entity Offline Training	
		Named Entity Recognizer	

Table 3-7 (continued)

	Visual Annotator	Visual Object Annotator	Feature Comparator
			GA-Based Classifier
			Visual Object Offline Training
		Visual Event Annotator	Similarity Relations Extractor
			Temporal Spatial Changes Extractor
			Spatial Changes Extractor
			Event Extractor
	Audio Low Level Features Extractor		
	Visual Low Level Features Extractor		
	Fusion Operator	Concept Constructor	
		Concept Learner	
		Concept Classifier	
		Feature Selector	
Coordinator	Request Processor	Query Analyzer	
		Query Level Fusion	
	Annotator		
	Concept Extractor		
	Database Operator		
	Search Operator		
Storage	Bridge		
	Multimedia Database	Audio DB	
		Fusion DB	
		Text DB	
		Visual DB	
	Fuzzy Knowledge Base		
	Multidimensional Index Structure		

The three rules of thumb are followed during mapping process:

- Provide a mapping between the module decomposition view and every C&C view.
- Ensure at least one mapping between a module view and a component-and-connector view.
- If your system uses more than one module view, map them to each other.[1]

3.8 Behavior

We answered three main questions before beginning the documentation process [1]:

1. What kind of questions should the documentation answer?
2. What behavioral information is available/can be stated to the developers?
3. Which notation should be chosen?

For METU-MMDMS architecture, the behavior is documented to give more insight to the developers about the behavioral details of the system such as the ordering of interactions and patterns of interaction among the elements. The main functionalities of the system are extraction and querying and the ordering of the communication is important for these functionalities, so the documentation should answer the question "What is the order of communication for extraction and querying functionalities of METU-MMDMS?".

The behavioral information about the system which can be stated/is available to the developers are which elements exchange data, whether the system is synchronious or asynchronious (METU-MMDMS is a synchronous system) and whether the system elements are local or remote (the client is a remote element).

We use UML notation and UML provides multiple options to model behavior (e.g. Use Cases, Sequence Diagrams, Communication Diagrams, Activity Diagrams. We employed Use Cases to capture functional requirements. If the concurrency was in

focus, the Activity diagrams should have been preffered over others, but it isn't. Sequential diagrams puts more emphasis on the order of communication than the communication diagram while communication diagrams' emphasis is on explaining which element interacts with which others. Therefore, the Sequential diagrams are used for modeling the behavior.

The basic use cases of METU-MMDMS are Video Upload/Update/Delete/Annotate and Query. To elucidate typical querying behavior, “Visual Query by Content” case of Query use case is presented in [Figure 3-24].

The scenario begins with selection of a video. Then the Client Application requests the selected video’s shot information from Multimedia Database through Coordinator component. By using the requested shot information, the shot frame and object region, whose content is the input of content based querying, are selected by the standard user. The Client forwards the shot frame and object region IDs to Coordinator and Coordinator fetches the corresponding low-level features from Multimedia Database. Afterwards, Coordinator uses the fetched low-level features to find top similar objects by the help of Multidimensional Index Structure. When the ID’s of the similar objects are obtained, Coordinator fetches the shots of those objects from Multimedia Database and passes them to the Client. Finally, the Client shows the results of the Visual Query by Content [27].

Similar to the Visual Query by Content, the Audio Query by Content scenario begins with selection of a video. Then the Client application requests the selected video’s shot info from Multimedia Database through Coordinator component. By using the requested shot info, the shot frame and audio concept, whose content is the input of content based querying, are selected by the standard user. The client forwards the shot and audio concept IDs to Coordinator and Coordinator fetches the corresponding low-level features from Multimedia Database. Afterwards, Coordinator uses the fetched low-level features to find top similar object by the help of multidimensional index structure. When the ID’s of the similar objects are obtained, Coordinator fetches the shots of those objects from Multimedia Database and passes them to the Client. Finally, the Client is able to show the results of the Audio Query by Content [27].

The Coordinator component sets the video to be preprocessed by the Video Pre-processor component. After the video is transcoded, Automatic Shot Boundary Detection starts to operate. It first calculates Edge Change Ratio for all frames and determines edge boundaries with these values. Then, Automatic Shot Detector sends frame numbers of start and end frames of each shot to the Coordinator Component.

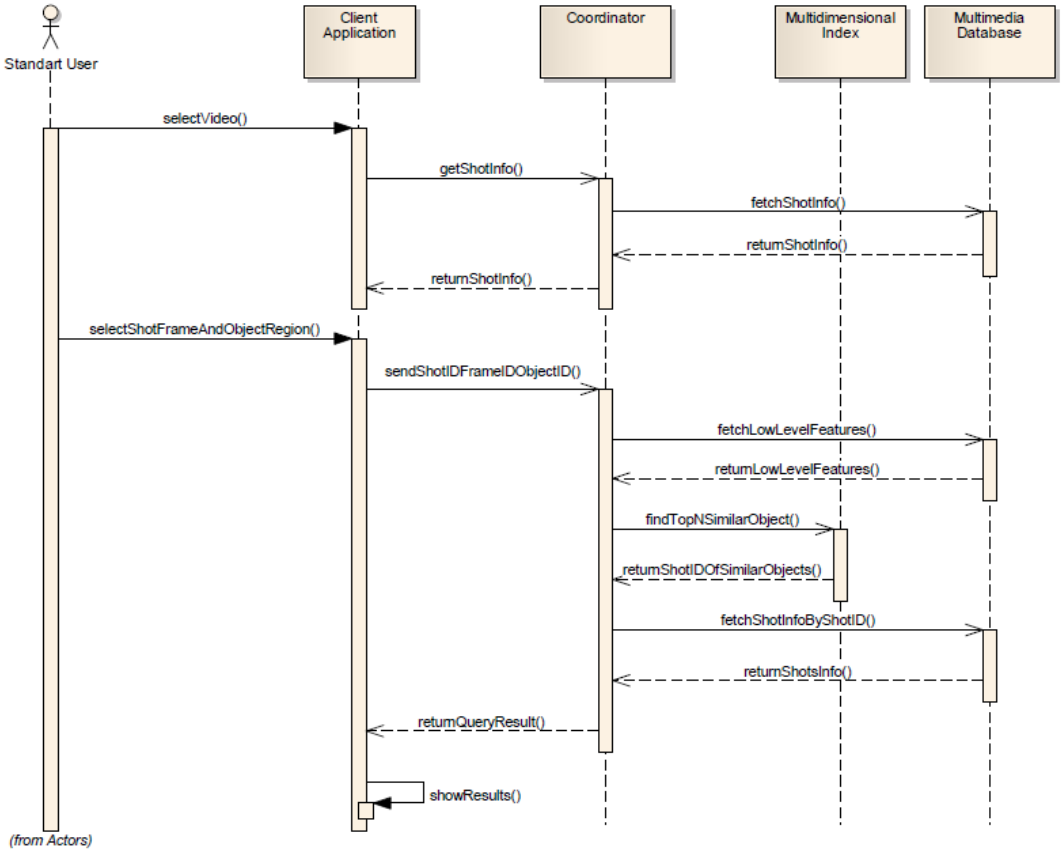


Figure 3-24 Visual Query by Content [27]

There may be gaps between shots because of gradual changes; so last frame of shot S_N and first frame of S_{N+1} may not be consecutive. Coordinator component gets these frame numbers and sends them to IFrame Extractor component. IFrame Extractor saves shot start and end frames and I-frames between these frames. These frames are

considered as key frames of a shot. Each shot's start frame number, end frame number and key frame numbers are saved to a text file. For each video file, Coordinator component checks this text file and loads shot boundary information if exists [27].

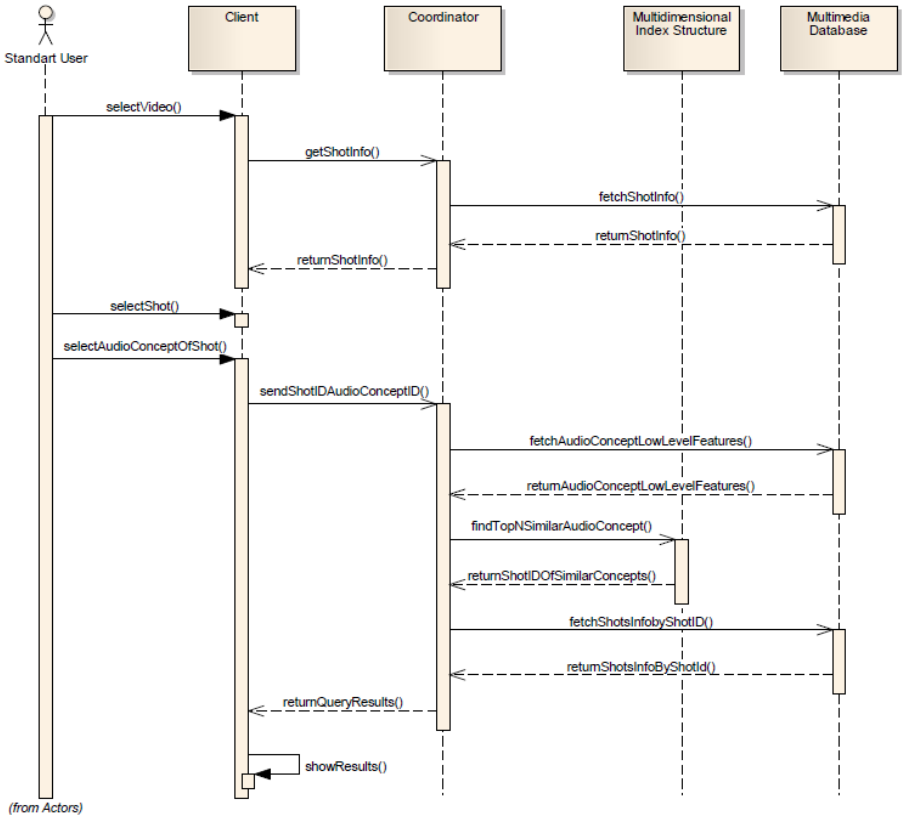


Figure 3-25 Audio Query by Content[13]

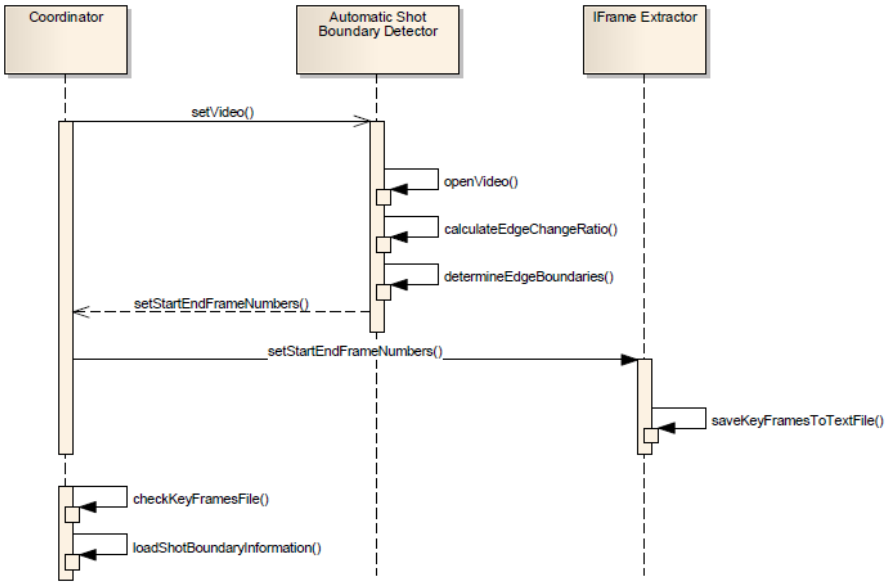


Figure 3-26 Automatic Shot Boundary Detection[13]

Query by Concept and Query by Example are joined together by means of some logical operators to constitute Query by Concept and Content [Figure 3-28]. Here we use concept query and content query for audio and visual. In concept query section, the user starts selecting query constraint with a desired operator and adds them to query list via rich GUI facilities. The constraints can be removed from the list or totally cleared to start a new query. These steps are repeated for any desired modalities along with some logical operators between them.

In audio query by content section, an audio concept selected from database and played then a desired audio selected as an example. In visual query by example, a video is selected from available videos in the database and sent to server, the server in turn responds the client with information about selected video, which includes shot lists, IFrames lists, objects' region annotated n each frame and their temporal information. At this point, a region is selected from an IFrame that belongs to a determined shot [27].

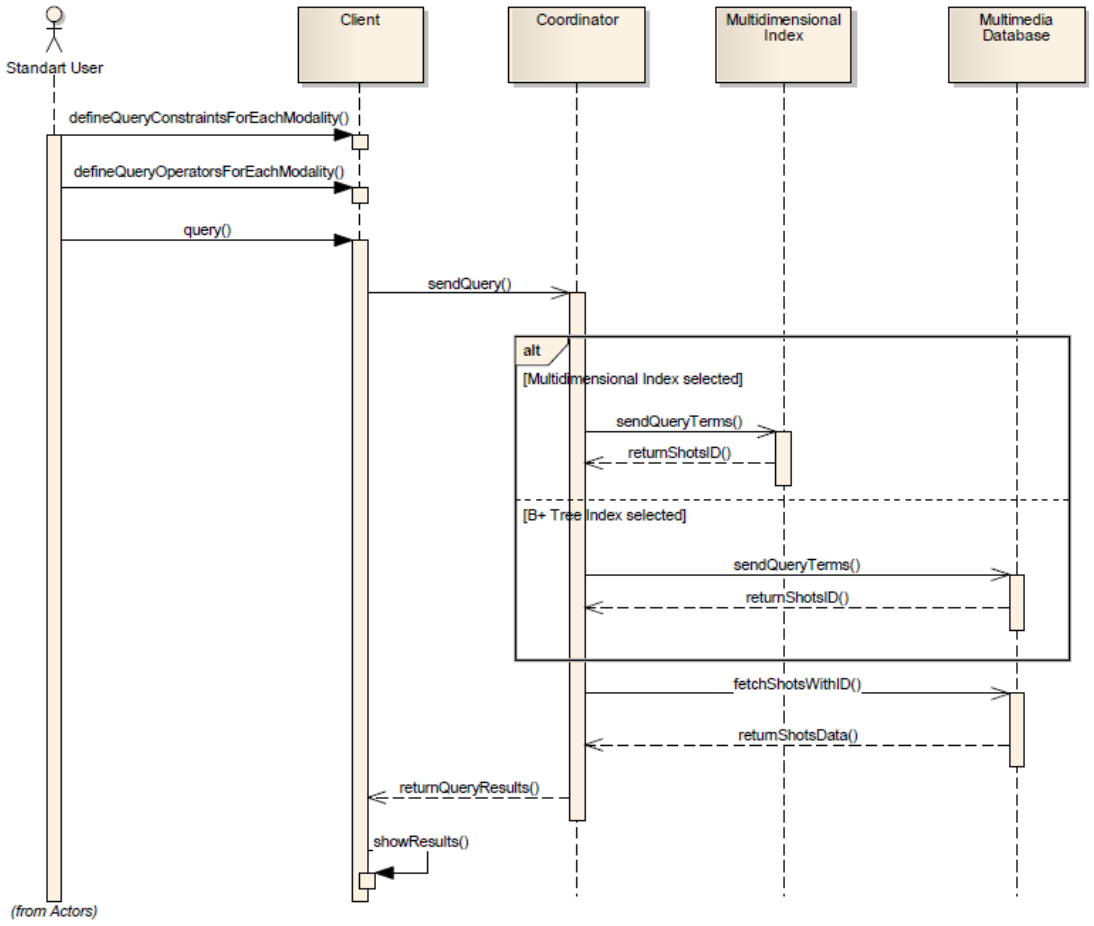


Figure 3-27 Concept Query [13]

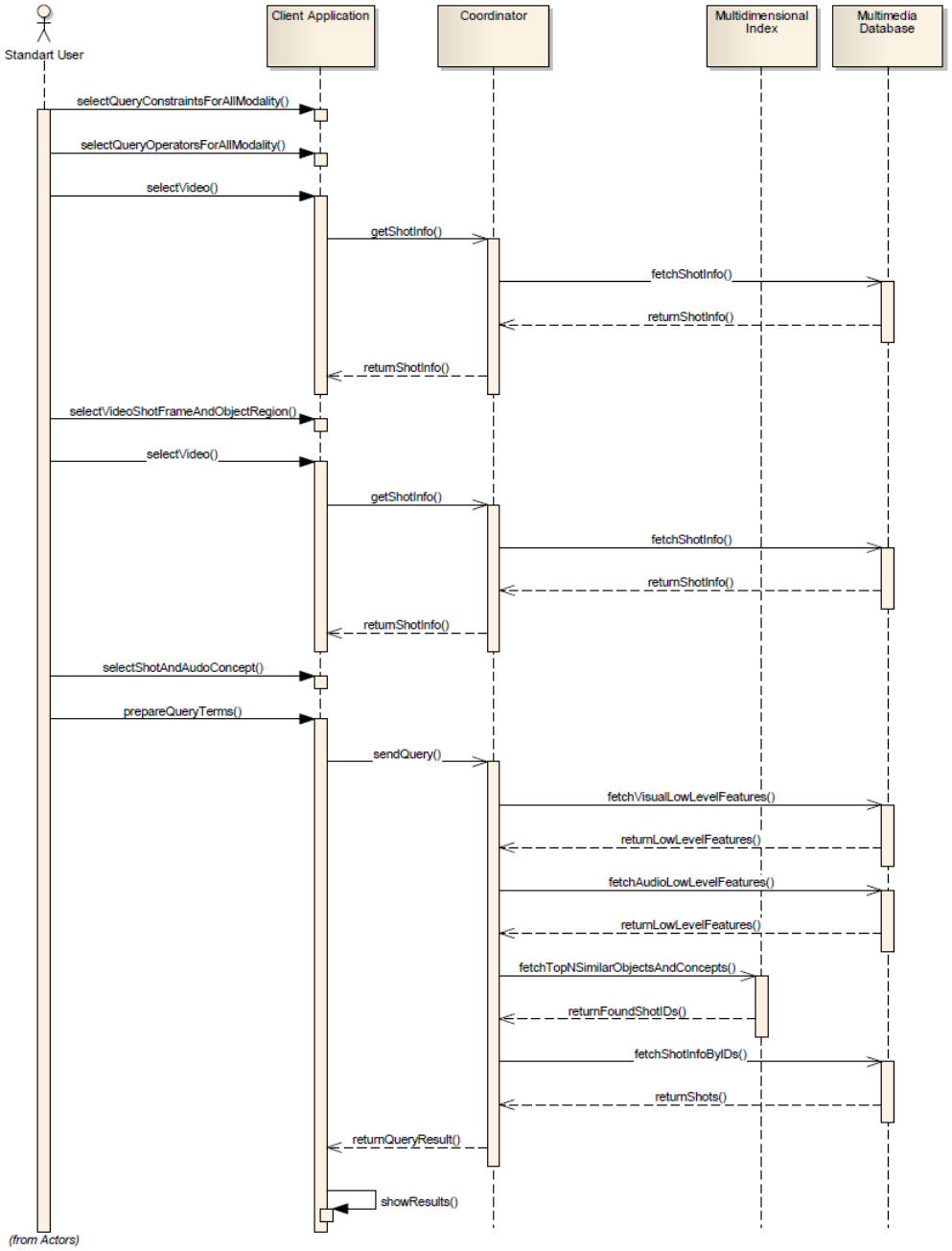


Figure 3-28 Query by Concept and Content [13]

3.9 Use Case Mapping

Visualization of the software units, their behavior and the system's design models enhance the understandability of the system. There are various scenario notations that aim to visualize scenarios that are used for defining the desired behavior of the system, and explain the system's observable behavior. UML Activity diagram, presents system's dynamic behavior via activities. Use cases handle system behaviors from the user's point of view. Message Sequence Charts are often used to model the reactive systems' communication based behavior [34].

Since it includes generic and abstract display elements and can be used to map a scenario to the architectural components, UCM notation is preferred among others. Use case Map (UCM) notation is one of the listed methods, which are used to describe and understand the behavior of complex systems. The aim of UCMs is to map the scenarios that are prepared based on the requirements and use cases to the architectural components and to explain and analyze system behaviors.

It is possible to understand the architecture of a completed system by using UCM [30]. In this work, UCM is employed to check whether there is enough content in a view to carry out a related scenario or not.

3.9.1 UCM Application Process

The UCM activity is carried out to examine the sufficiency of the architectural document. It asks the following question: "Does a view in a document contains enough information to meet a scenario?". When the architectural document is exhaustive, its sustainability is at risk. Therefore, the documentation is expected to have an easily updatable size and to be comfortably understandable ("travel light" principle). We must be sure about having sufficient material in the architectural document and critical details and decisions are not skipped, correspondingly. Otherwise, the document will be useless. To understand whether the documents contain enough information to meet the selected scenarios and to prevent the

document from being proposed with insufficient content, the UCM process is applied to the top-level component and connector view. Architectural evaluation of the METU-MMDMS is conducted separately by application of the ATAM.

In the UCM application process, the architectural components that the scenario is going to be mapped to are initially determined [51]. Afterwards, the scenario steps are mapped to these components via the responsibilities with casual relationships and UCM paths. Finally, for each responsibility, the low-level component which owns the responsibility and, if any, the inputs and the outputs of the activity that fulfills the responsibility are listed on a table as an extra work.

Data extraction and querying are the "sine qua non" functionalities of the METU-MMDMS. There are three types of queries in METU-MMDMS: query by concept, query by content and, query by concept and content. To cover all of these query types, query by concept and query by content are examined ("query by concept and content" is almost the combination of the other two). Further, multimode querying, which is used to improve the accuracy of the system, is among the main goals of the system. Therefore, it should be covered by the architectural views. Keeping the data extraction case in mind, three scenarios are selected for the UCM process:

- Data extraction scenario: Online Concept Extraction
- Multimode query scenario: Query by Concept
- Low Level Features Query: Query by Content

Audio Query by Content UCM application process is explained in the following sections. The other UCMs and corresponding scenarios are presented in [Appendix D].

3.9.1.1 Selection of the Architectural Components

Views and Beyond method is used for the documentation process of the METU-MMDMS. The module views, which model the static structure of the system, are developed in decomposition style. There is one to one mapping between the modules

in module views and components of the component and connector views, which explain the dynamic structure of the METU-MMDMS.

Abstract architectural components will also be used in the context of UCM. However, METU-MMDMS has already an actual architecture. Therefore, real components are used for the application of UCM. By avoiding the visual complexity, to keep the model understandable, METU-MMDMS's top-level components are used for the UCM process. Those components are shown in the [Figure 3-11], the Top Level C&C View. The Client component is found on the client side of the system, which has the client-server architecture. The client component has the video uploading/updating/deleting, querying and presenting the query result functionalities. Semantic Concepts Extractor, Coordinator and the Multimedia Database components are found on the server side. Coordinator component is responsible for the organization of the communication among all other components. The video that is loaded to the system via the client component is forwarded to the Semantic Information Extractor by the Coordinator component. The video processing and extraction of the event (e.g. goal in a football video), object (e.g. ball) and concepts (e.g. foul) are carried out at the Semantic Information Extractor component. The Multimedia Database component takes the extracted events, objects and concepts via the Coordinator, and stores them. The queries, which are constructed at client side, are passed to the Multimedia Database component through the Coordinator and to increase performance, by the use of Multidimensional Index Structure, the related data is transferred from the Multimedia Database to the client again through the Coordinator component.

3.9.1.2 Generation of UCMs

The "Audio Content Based Querying" scenario that is under the "Low-level Feature Based Querying" topic of the "Querying" use case of METU-MMDMS is employed within the description of the sample UCM application process. jUCMNav [31] is used for this process. jUCMNav is an analysis and transformation tool for URN.

The UCM component types can be listed as Team, Object, Process, Interrupt Service Request, Agent and Pool. Since the properties of the components do not match with the properties of the ones listed above, the Audio Query by Content UCM diagram is formed with "other component type" component option. The responsibilities, start and end points are entitled/numbered and the related explanations are expressed as notes.

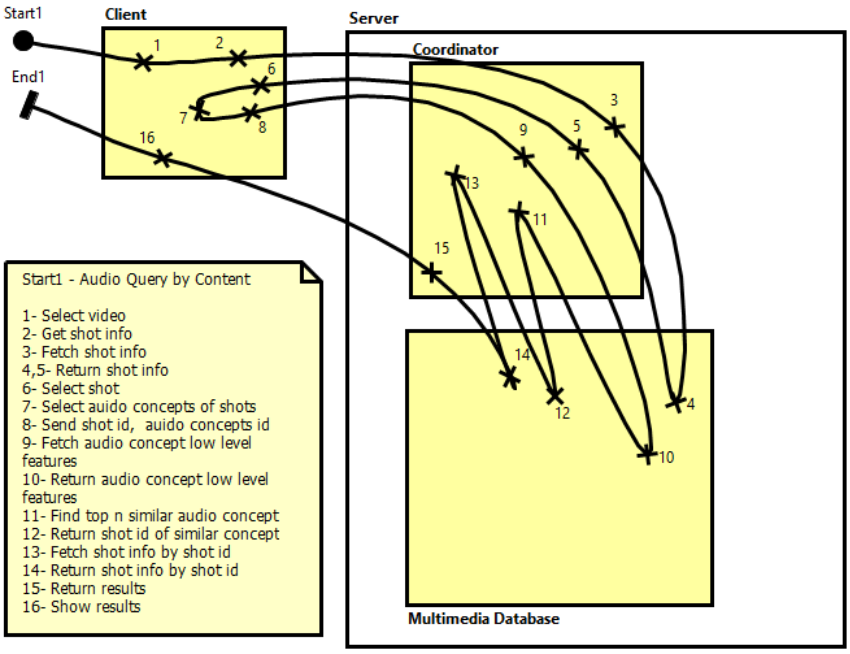


Figure 3-29 Audio Query by Content UCM

The triggering factor of the UCM path in [Figure 3-29] is the start of the application of the audio content-based querying scenario by the user. The user selects the concept from the video, which he/she is going to query (1-7) and the low-level features that belong to this concept are extracted from the system (8-10).

The video shots whose low-level features are most similar to the selected concept's low-level features are presented to the user (11-16). Since the scenario describes a querying process, there is not any important post-condition or effect at the end of the UCM path, matching with "End1" end point. When the UCM path is analyzed, it is expected to be connectors between Client-Coordinator and Coordinator-Multimedia Database. The existence of those connectors can be verified with the Top Level C&C

View in [Figure 3-11]. Furthermore, each responsibility can be mapped to a component, which is responsible for the related activity.

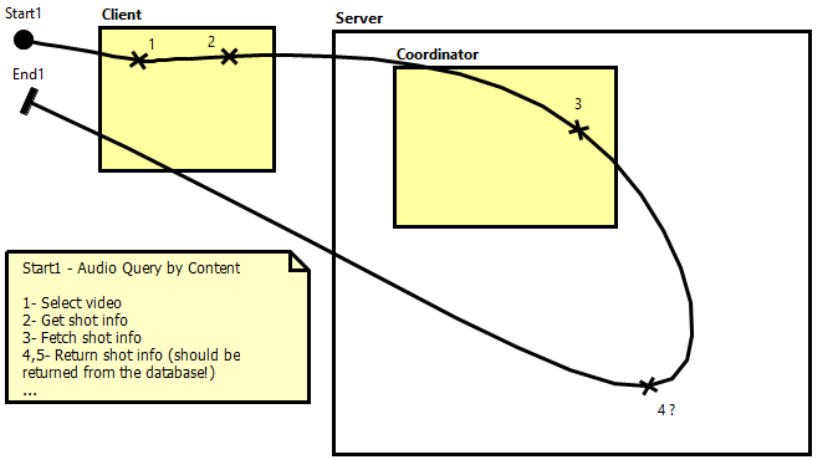


Figure 3-30 Insufficient Architectural Content - UCM cannot be completed

Suppose that in a hypothetical example, the architecture is not reflected to the architectural document completely and the Multimedia Database component is skipped and remove the component from the [Figure 3-29]. In this case, only the Client and Coordinator components are remained in the UCM diagram [Figure 3-30]. The responsibility which corresponds to the fourth step of the scenario to a component, performs the information extraction from database functionality. Hence, the fourth step cannot be shown on the UCM diagram. In this case, there is the insufficiency of the architectural content in the architectural document.

3.9.1.3 Low-level Architectural Mapping

The sample MMDMS's architecture is complete. Therefore, it is possible to give details about the architecture. While maintaining the simplicity of the UCM diagram, A mapping table is formed accordingly to map each responsibility to corresponding low-level architectural components (explained in the previous chapters) and to present the inputs and outputs of responsibilities through this table (see [Table 3-8]).

The low level architectural mapping table for the audio content based querying UCM is shown in [Table 3-8]. The responsibility number, sub-component, input and output are listed at each row, respectively. Each responsibility can be mapped to a sub-level component, which is present in the corresponding view. Therefore, the documentation is sufficient for this scenario.

Table 3-8 Low-level Architectural Mapping

R.#	Sub-component	Input	Output
1	**Client.** *QueryFormulator*	-	Video ID
2	**Client.** *RetrievalHandler*	Video ID	Shot Info
3	**Coordinator.** *QueryFormulatorInterface &* **Coordinator.** *RequestProcessor &* **Coordinator**.*DatabaseOperator*	Video ID	Shot Info
4	**MultimediaDatabase.** *Database*	Video ID	Shot Info
5	**Coordinator.** *QueryFormulatorInterface*	Video ID	Shot Info
6,7	**Client.** *ResultPresenter*	-	Query with Shot ID, Concept ID
8	**Client.** *RetrievalHandler &* **Client.** *QueryFormulator*	Shot ID, Concept ID	Shot info
9	**Coordinator.** *QueryFormulatorInterface &* **Coordinator.** *RequestProcessor &* **Coordinator.** *DatabaseOperator*	Shot ID, Concept ID	Concept's Low Level Features
10	**MultimediaDatabase. Database**	Shot ID, Concept ID	Concept's Low Level Features
11	**Coordinator. DatabaseOperator**	Concept's Low Level Features	Shot ID's of similar concepts

Table 3-8 (continued)

12	**MultimediaDatabase.** *MultidimensionalIndexStructure*	Shot ID, Concept ID	Shot ID's of similar concepts
13	**Coordinator.** *DatabaseOperator*	Shot ID's of similar concepts	Shot info
14	**MultimediaDatabase.** *Database*	Shot ID's of similar concepts	Shot info
15	**Coordinator.** *QueryFormulatorInterface*	Shot ID, Concept ID	Shot info
16	**Client.** *ResultPresenter*	-	Shot info

As it can be seen from the hypothetical example when the sub-level architectural views are not sufficient because the Retrieval Handler sub-component is not present in the Client component, the second row of the table and the UCM diagram cannot be completed. As a result, the insufficiency of the document content can be determined via UCM and low-level architectural mapping table.

Table 3-9 Insufficient Architectural Example

S.#	Sub-system	Input	Output
1	**Client.** *QueryFormulator*	-	Video ID
2	**Client.** *?*	Video ID	Shot Info

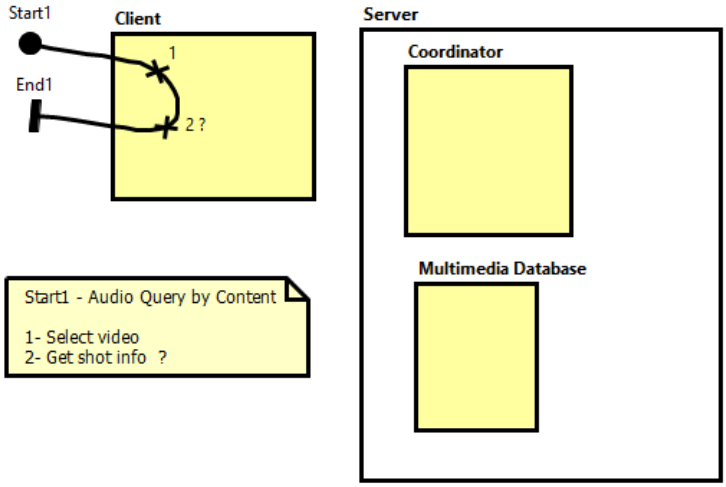


Figure 3-31 Insufficient Architectural Content - UCM cannot be completed

CHAPTER 4

ARCHITECTURAL EVALUATION

To evaluate the software architecture of METU-MMDMS, SEI ATAM is employed. "ATAM is a software architecture evaluation and analysis technique which discovers trade-offs and sensitivity points of the evaluated architecture. ATAM is also a risk identification method which means detecting areas of potential risk within the architecture of a complex software intensive system." [32]. ATAM is preferred among other methods depending on its advantages listed as follows:

- quick and inexpensive
- detailed analysis of measurable quality attributes is unnecessary

ATAM ensures:

- clear and characterized quality attribute requirements
- increase in the effectiveness of the documentation
- design decisions in the documentation

In addition, the typical ATAM steps are:

- Presentation
    - ATAM presentation
    - Business drivers presentation
    - Architecture presentation
- Investigation and Analysis
    - Identification of architectural approaches
    - Generation of quality attribute utility tree
    - Analyze of the architectural approaches
- Testing
    - Brainstorming and prioritization of the scenarios

    - Analyze the architectural approaches
- Reporting
    - Result presentation

Evaluation team, customer representatives and the architecture team do the presentation, investigation, and evaluation. The role assignments for the ATAM process are listed in [Table 4-1].

Table 4-1 Stakeholders

Stakeholders	
Project Manager	Adnan Yazıcı
Architecture Team	Murat Koyuncu Mustafa Sert Saeid Sattari Çiğdem Avcı Salma Merve Aydınlılar Elvan Gülen
Evaluation Team	Halit Oğuztüzün Güneş Uyanıksoy

Brainstorming and prioritization of the scenarios are carried out by all of the stakeholders. Afterwards the architectural approaches are analyzed by evaluation team, customer representatives and the architecture team again. Finally, results are presented to all of the stakeholders. The utility tree, generated scenarios, analysis questions, identified risks and non-risks and identified architectural approaches can be listed as outputs of the ATAM process.

Performance, Accuracy, Conceptual Integrity, Scalability and Maintainability are selected as the important quality attributes for the METU-MMDMS. The definition of the quality attributes are listed in [Table 4-2].

Table 4-2 Quality Attributes

Quality Attribute	**Definition**
Performance	Performance is an indication of the responsiveness of a system to execute any action within a given time interval. It can be measured in terms of latency or throughput. Latency is the time taken to respond to any event. Throughput is the number of events that take place within a given amount of time.
Accuracy	Accuracy includes precise and recall accuracy. Precise means correctness of retrieved data and recall accuracy means ability to retrieve all relevant stored data.
Conceptual Integrity	Conceptual integrity defines the consistency and coherence of the overall design. This includes the way that components or modules are designed, as well as factors such as coding style and variable naming.
Scalability	Scalability is ability of a system to either handle increases in load without impact on the performance of the system, or the ability to be readily enlarged. For the mentioned system, database scalability is in the scope and user scalability is out of concern.
Maintainability	Maintainability is the ability of the system to undergo changes with a degree of ease. These changes could impact components, services, features, and interfaces when adding or changing the functionality, fixing errors, and meeting new business requirements.

Then the utility tree is constructed as in [Figure 4-1]. Scenarios are proposed to show that the system meets the selected quality attributes.

After the presentation of the ATAM, the business drivers are discussed. It is emphasized that the METU-MMDMS is a research project and its main goal is "providing multimodal querying which increases the accuracy of the retrieval systems by using the data fusion" [32]. The main functions of METU-MMDMS are listed as a summary:

- Concept, object, event annotation
- Storing, indexing, retrieving and managing semantic information
- Content based query and query by example

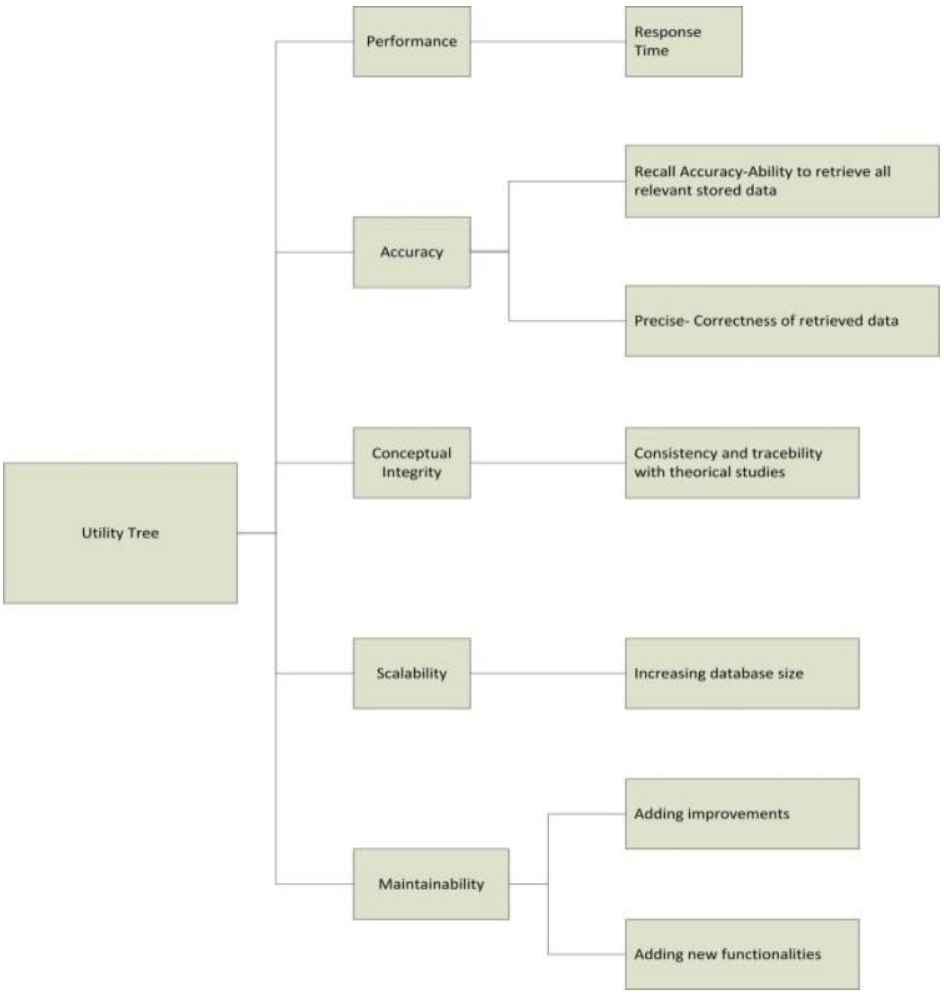


Figure 4-1 Preliminary Utility Tree for METU-MMDMS [33]

Identification of architectural approaches followed the architectural presentation. Component-oriented architecture, high dimensional index structure, object oriented database, coordinator structure, data fusion and the client server approaches have serious effects on the architecture.

The most critical scenarios are selected and analyzed. For each quality attribute, a representative scenario is specified. The scenarios are prioritized according to their "Importance" and "Difficulty to Achieve" (H(High), M(Medium) and L (Low)) (see [Table 4-3] and [Figure 4-2]).

Table 4-3 Scenarios for Quality Attributes

Scenario Number	Quality Attribute	Scenario	Importance (H,M,L)	Difficulty to Achieve (H,M,L)
1.	Performance	Video shots that contain sound of explosion and view of plane at the same time are retrieved from 13000 objects of video, in less than a second . (using the index structure.)	H	M
2.	Performance	A traffic accident scene is retrieved from 13000 objects of video by multimode querying, in less than a second. (Using the index structure.)	H	H
3.	Accuracy	When goal concept in a football game is queried on manually annotated data, 95% of stored goal scenes are retrieved.	M	L
4.	Accuracy	When goal concept in a football game is queried on automatically annotated data, 50% of stored goal scenes are retrieved.	H	M
5.	Accuracy	When goal concept in a football game is queried by using of manual annotation, 99% of retrieved data are correct	L	L
6.	Accuracy	When goal concept in a football game is queried by using of automated annotation, 60% of retrieved data are correct	H	H
7.	Accuracy	Query level fusion is resulted 5% better than corresponding single mode queries	H	M
8.	Conceptual Integrity	Query results are presented as list of shots ranked by relevance for each mode separately and multimode.	H	M
9.	Scalability	Query results from 50 hours of video data are gathered in about one second even if more than one condition is used in a query Query results from 500 hours of video data are gathered in about three seconds even if more than one condition is used in a query (using the index structure.)	H	H
10.	Maintainability	Adding a new learning algorithm to the system is practical	M	L
11.	Maintainability	Adding a new low level feature to the system is applicable	M	M
12.	Maintainability	Incorporating speech to text translation functionality in different languages	H	L
13.	Scalability	Adding a new low level feature to the system is applicable	M	M
14.	Scalability	Efficient execution of type of queries that require higher dimension index	M	M

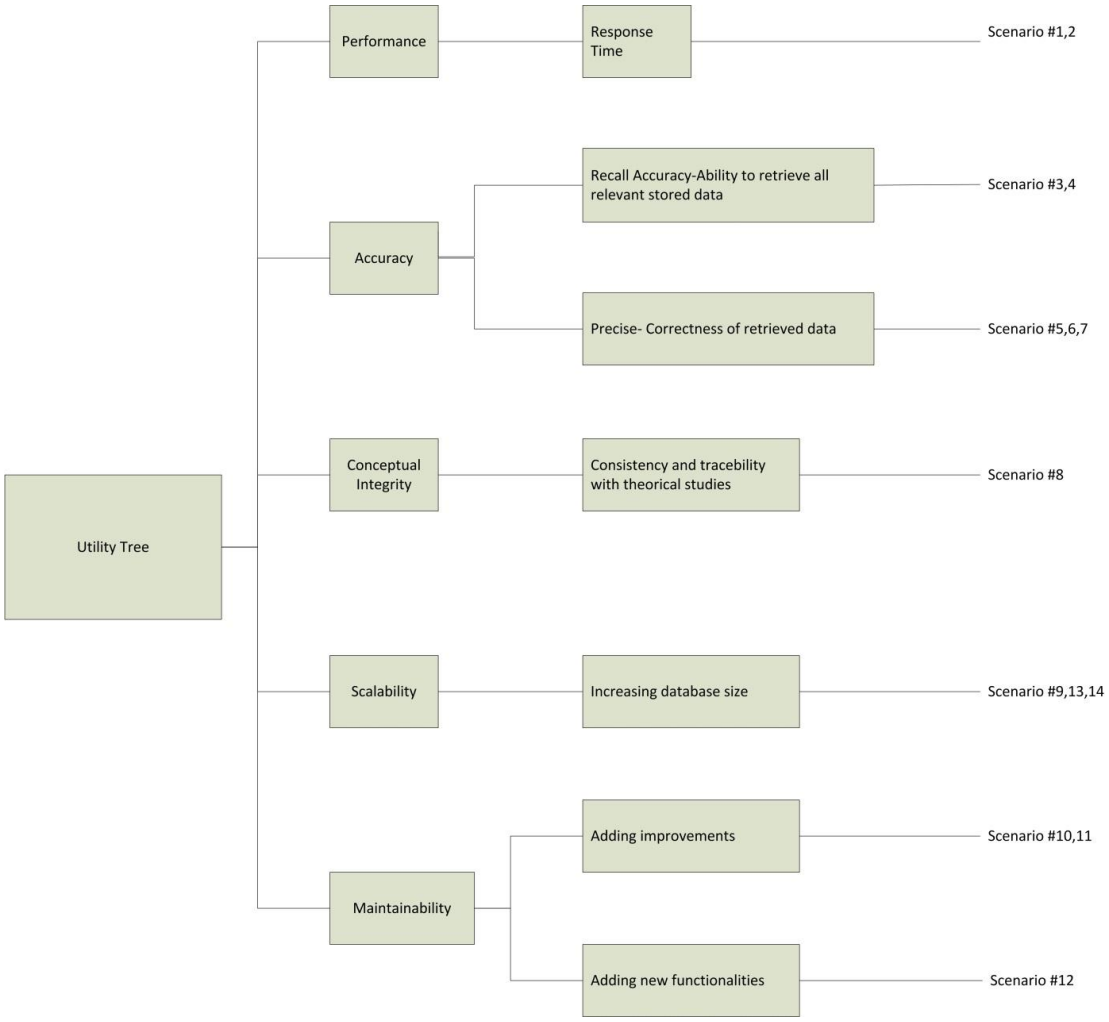


Figure 4-2 Preliminary Utility Tree for METU-MMDMS [33]

The results of the ATAM evaluation are summarized in [Table 4-4] [33] [52].

Table 4-4 ATAM Evaluation Results

Sensitivity Points	• Dimension size of index structure affects the performance. • Index structure should be created for the attributes of concept, which was stored. • Fusion is sensitive to data dependency. • Automated annotation is sensitive to data that is annotated. • Query annotator is sensitive to syntax errors of queries, which are written by user. • Size of index structure depends on size of data. • Success of learning depends on the correlation among modalities. • Query interface depends on output size. • When new low-level feature is added, index structure should be recreated. • Learning mechanism is sensitive to size of training data. • Query level fusion is sensitive to data type. • Scalability depends on dimension size of index structure.
Risks	• Query formulator is not suitable for standard user. • Index structure must be created accurately for system to operate as expected. • Increasing the index size may cause out of memory error. • Failure of multimode querying is unacceptable. • If output size is very large, the system may give timeout error. • It is hard to manage high dimensional data. • Adding new low-level features will decrease system accuracy.
Tradeoffs	• Using fusion increases accuracy, however it causes performance reduction. • High dimensional index structure makes the system more scalable but decreases performance.

CHAPTER 5

DISCUSSIONS AND RECOMMENDATIONS

METU-MMDMS has multiple developers and there are various documents for different components. Even different architectures are documented in different documents or the exact architecture of some component is not reflected to any document, that is why the stakeholders of the architecture are not fully aware of the overall architecture, which is observed during the software architecture discussion meetings. Therefore, the usage of common tools and languages should be planned in parallel to the requirements process and should be encouraged. EA 7.5, UML 2.0 and MediaWiki are decided to be used for this study. EA 7.5 is a comprehensive tool for architectural modeling but it does not provide much traceability options among the architectural models and it is hard to check the consistency with the EA 7.5. MediaWiki is employed for the documentation process and it provides a common platform for all the stakeholders any time, where they can reach, track and edit the contents of it collaboratively and discuss on the same ground. MediaWiki is easy to use and has a low learning curve, therefore software professionals can easily adapt to it. Benefits of wiki for documentation can be listed as follows:

- Translucion: use links instead of information duplication
- Comments from any reader are publicly available
- The user can subscribe any page to be notified whenever a change occurs
- The changes are automatically tracked
- Easy to modify
- By using page referrals the relations between views can be captured

The weaknesses of the wiki can be found in the following list:

- Online access is needed
- Server reliability and security issues should be considered

- It is hard to move wiki between servers (no standard wiki format)
- Sophisticated formatting and printing are not easy
- Access management policy is not rich
- The whole wiki cannot be printed at once

Documenting rationale for architectural decisions is essential to ensure stability and consistency in the evaluation of the architecture. It is also particularly useful to improve communication within the development team and the future developers of the system will probably need to know the rationale behind the architectural design decisions.

During the documentation process of the software architecture, some architectural problems are detected. In [Figure 3-29], it can be easily seen that Multimedia Database and Client components are tightly coupled with the coordinator component which means the Coordinator component uses more information than it should about the Multimedia Database and the Client components. To obtain more modular MMDMS architecture, the design of the Coordinator component can be revised.

Furthermore, the queries for the current architecture should be constructed by expert people, who are aware of the system design. However, in the future systems, the necessity of usability for standard people can occur. Therefore, a query recommendation feature can be added to the METU-MMDMS as a future extension.

The documentation stayed at high levels of the METU-MMDMS architecture, because of the aim of proposing a generic architecture in the future. Uses view is decided to be detailed for a high-level architecture at the meetings. The project team is concerned that increasing the amount of details can make the architecture documentation hard to maintain. In the module views, the level of detail is kept at package level and the components in the C&C view has one-to-one mapping with the module view. The ports and delegations should be well constructed in the C&C view and should be consistent with the actual system. Through the same port, the connectors with compatible interfaces can aid the communication among components. If there is a significant data flow in a C&C, it should be supported with a sequence diagram in the C&C View.

The context of a module/component can also be documented with V&B and for hierarchical diagrams; the super level model is used as the context diagram whereas the Use Case diagram is considered as the context diagram of the whole system. Use case diagram can also be used as a behavioral diagram.

When the selected models/diagrams are constructed for architectural documentation, the architectural review process, which is carried out as meetings together with the stakeholders, is started. After the review of the use cases in the architectural meetings, the necessity of adding the "create index structure" use case appeared. Therefore, the architectural documentation benefits from extensive discussions.

According to [51], “Conceptual integrity is the underlying theme or vision that unifies the design of the system at all levels”. When the content and output are common, there is a common language among the components and the following items ensure the conceptual integrity of METU-MMDMS:

- Different modalities uses common concepts
- Different modules use video shot as the video unit
- The query results of different modalities are in the same type

V&B method, which is used during the documentation of the multimedia database system, takes into account different stakeholders, and is comprehensive and convenient for the documented system, and suitable for documenting complex systems. Nevertheless, the backwards modeling is not easier with this method; still there is the need of a long process of analysis. System stakeholders should agree on not only the models, but also the meaning they infer from those models. Stakeholders and the quality attributes they emphasize, affect the whole architectural structure and documentation of the system. During the modeling process, the act of writing the architectural design decisions down is vital for the advancing stages to understand the sense of modeling. What works well in applying V&B in METU-MMDMS case are listed as follows:

- The component and connector view of the semantic information extractor is formed in the pipe and filter style, then meeting expected that the information

from the annotators should be piped to the fusion component when the corresponding diagram is presented. Both because of the nature of this style and the information presented in the diagram helped the attendees of the meeting in understanding the view and after discussions, it appeared that the information is passed to the multimedia database from annotators, instead of the fusion component. Therefore, by the help of V&B documentation, the actual system is correctly understood.

- There is not a well-worked out reference architecture for the MMDMS systems. V&B makes documenting the variation points and rationale behind the design of the METU-MMDMS necessary. The documented variation points and rationale can be beneficial during the reference architecture construction.
- The styles are chosen based on the stakeholders' concerns in the V&B method. Choosing among styles eased the documentation process of an already developed system because METU-MMDMS inherently had notable explicit styles such as shared-data and pipe and filter.
- It is easier to check the conceptual integrity among the system components with views and beyond. Because the templates provided by the V&B organizes the information about a view. E.g. different modules use shots as the video units. The coordinator, which works as a mediator, uses shots as data in its interfaces to the other components; it can be checked from the data types section in the interface template.

Difficulties in applying V&B for METU-MMDMS case are included in the following list:

- Mapping between modules and components is listed as a rule of thumb for mapping between views in [1]. For METU-MMDMS architecture, this action was not much informative. Because in this case, there was a single documenter, and he documenter used same names for the mapping modules and components. It can be said that, if the mapping between modules and components are one-to-one and there is a single documenter, mapping between modules and components is not necessary.

- The rationale is captured in V&B templates, but there were specific important quality properties of METU-MMDMS such as accuracy and functionalities like multimode querying. The V&B does not enable the documenter to emphasize such quality properties. It would have been more effective if there were a system quality properties section. There is a system overview section which include the system wide information but it is not explained enough in [1]. System Overview can also include links to the document where important system wide quality properties are met.
- It will be hard to find a rationale for a component in METU-MMDMS document because, there is no specific component based or rationale based index. The most important design decisions are not emphasized in the rationale section, either. An index which is colored depending on the significance of the design decision could have been useful.
- How much detail should be captured for a design rationale is not discussed but some details are important (e.g. design constraints, assumptions). For example, multidimensional index structure improves accuracy. But what are the constraints and assumptions about that rationale? There is no guidance about how the rationale should be documented in [1].
- Knowledge-based artifacts, such as ontologies and vocabularies about the MMDMS domain are never explained in the documented V&B views. It would have been helpful to have such an explanatory view about the domain. The element catalog is not a convenient section to explain the ontologies and vocabularies because they are not architectural elements like modules or components. They should be presented separately.
- The multimode nature is dominant for the METU-MMDMS, it would have been informative to categorize the views depending on the modes or depending on the query types. But, V&B does not provide enough guidance such free moves. For example, for an audio query, the system uses audio related components such as audio annotator. If there was an audio view, we could have shown the components which are used for audio functionalities in it and give specific information for audio domain.

Finally, the most remarkable design decisions of the METU-MMDMS are discussed in the following table, [Table 5-1]:

Table 5-1 Critical Design Decisions

Decision	**Pros**	**Cons**
Coordinator module with a mediator role	Loose coupling among the components other than the coordinator itself, improves modularity Tight coupling helps in performance increase	Tight coupling between the coordinator and other modules makes the component harder to maintain
Multidimensional Index Structure	Efficient data access Multidimensionality increases the performance comparing to having a separate for each dimension	It is hard to increase its dimension. The performance decreases drastically when the dimension is increased.
Object Oriented Database	Object orientation models real world, multimedia has the ability to capture real world. Therefore, "it is well suited to support multimedia database objects" but "it does not mean that "it automatically satisfies multimedia database requirements".	The size of the objects affects the insertion and retrieval time (performance).

5.1 Recommendations

To improve the design of the system, a couple of recommendations are listed in this section.

- Authorization/Authentication functionality can be added to the Client Application. The users can be specified as Standard User, Power User and Administrator. The Standard User can only do the querying activities while Power User can modify the available queries like adding new concept type to the list of concept that can be queried. For administrative issues like maintaining database and creating new index structure, the Administrator can be employed.

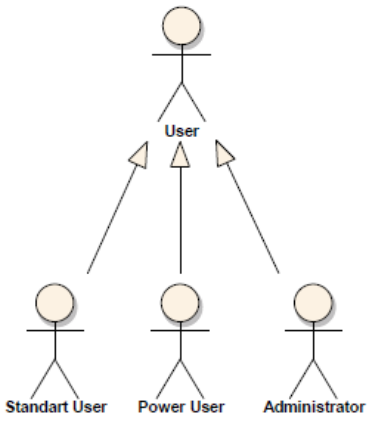


Figure 5-1 User Levels

5.1.1 Recommendations on the Use of Design Patterns

The suggested design patterns for the modules are listed in [Table 5-2] together with the explanations.

Table 5-2 Recommended Design Patterns

Module	**Pattern**	**Explanation**
Fusion	Factory	The Factory design pattern can be used to have a uniform and concise way of object creation. In fusion component, the audio/visual/textual concepts are created extensively so the factory design can be useful.
Audio/Visual/ Text Annotator	Strategy	The strategy design pattern enables selecting the algorithm behavior at run time by encapsulating the required algorithms and interchanging among them when necessary. To make use of various interchangeable classification algorithms at the annotator modules, the strategy design pattern can be employed.
Client Application	State & Observer	When an Object's behavior changes based on its internal state, state design pattern can be used. The client switches between querying, query result presentation, annotation and similar states. State pattern can be beneficial. In order to notify state changes to the client application GUI, observer pattern can be employed.

Presenting a prescriptive architecture is not in the scope of this thesis but it can be considered as a future work. Nevertheless, a couple of recommendations can be made on the possible architectural patterns that can be used in METU-MMDMS architecture. The layered architectural pattern, in which any layer provides services to the layer above it, can be employed for the architecture of METU-MMDMS (the recommended layered architecture is in [Figure 5-2]). The layered architectural pattern is used when building new facilities on top of existing systems or when the development is spread across several teams. In the layered architecture in [Figure 5-2], "Database Replication Middleware" [53] is added among the layers. It is provided for high availability and performance. Different replication techniques by using middleware are explained in [53]. In order to form the layered architecture, the sub components of the coordinator component, which has a two-way communication with the other components, are distributed between the database accessor and application layers.

The Model-View-Controller architectural pattern is not recommended for the METU-MMDMS because it is used when there are multiple ways to view data and the possible future data presentation options are unknown but there are not different complex ways to present the data in METU-MMDMS. Therefore the application of the MVC pattern is not essential for METU-MMDMS.

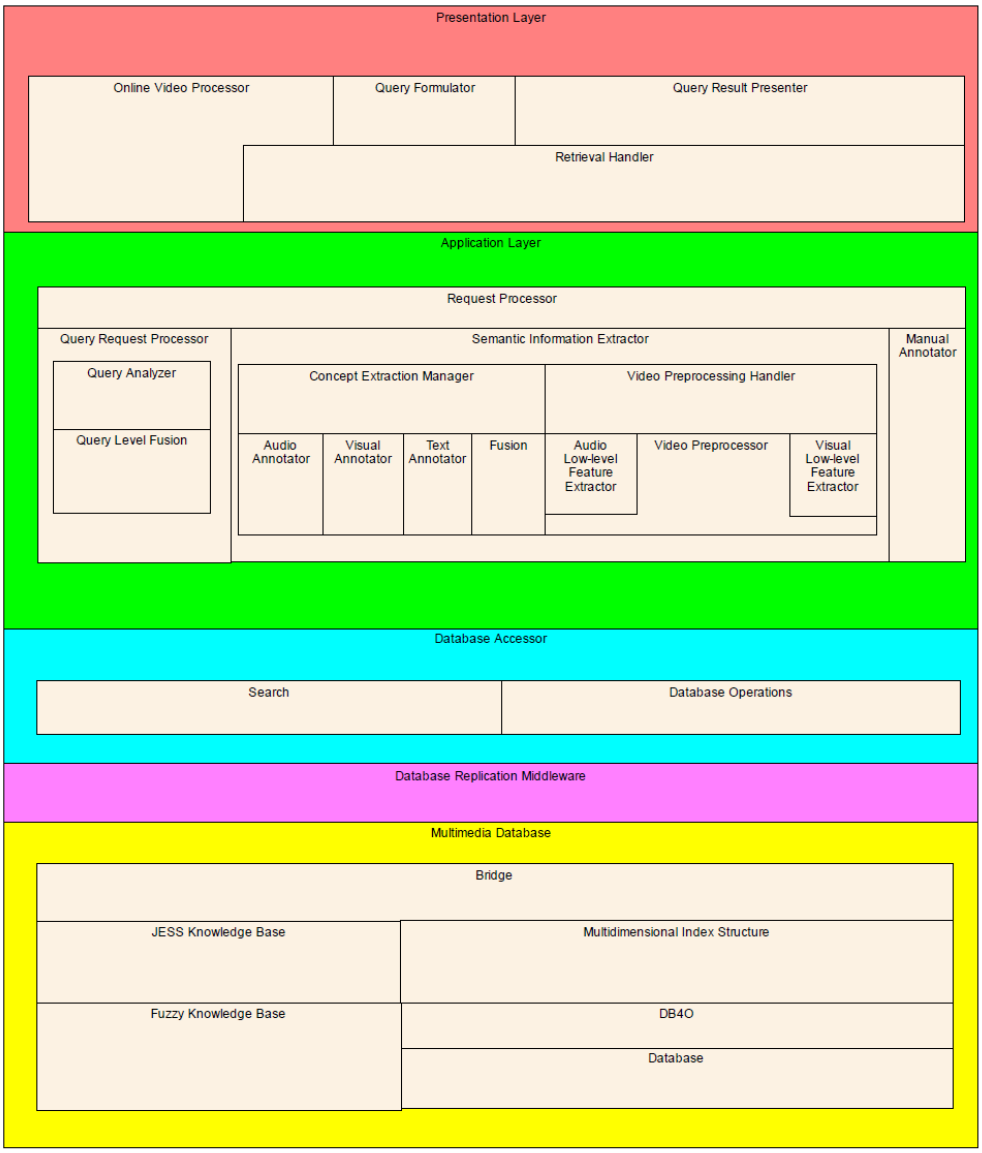


Figure 5-2 Recommended Layered Architecture

CHAPTER 6

CONCLUSION

METU Multimedia Data Management System descriptive software architecture is presented in this study. During the architectural documentation process, the V&B method is employed. Decomposition and uses module view styles, pipe and filter, client-server and shared data C&C view styles, and deployment allocation view style are used throughout the documentation. The sufficiency of the views is questioned by using the Use Case Maps. Moreover, the documented architecture is analyzed by ATAM architecture evaluation and analysis method.

29 views are constructed to represent the architecture (15 module views, 13 C&C views, 1 allocation view). More than 50 diagrams are drawn to represent the METU-MMDMS. The sufficiency of the architecture is checked for 3 selected scenarios with 3 UCM diagrams, more scenarios and UCM diagrams should be used to obtain a stronger result. 14 diagrams are used for the ATAM activity.

Through that process, the researchers gained more awareness about the overall architecture and the components that they are not responsible for. In order to implement the software architecture accurately, the developers and researchers should completely understand it first.

There is not any well presented Multimedia Data Management Architecture in the known literature. With this study, a methodological presentation of the METU-MMDMS is constructed via Views and Beyond which the future MMDM systems can make use of as a detailed sample.

Similarly, a well-defined reference architecture for Multimedia Data Management systems is not present. The composed document is aimed to be the source of information about a sample multimedia data management system and similar systems, for the generation process of a MMDMS reference architecture.

The output of this study is the software architecture document of METU-MMDMS. The architectural documents are evaluated according to different criteria so far such as compatibility to the standards, grammar and spelling. But, the sufficiency of the view in a document is not questioned, until this study. The METU-MMDMS architectural views are documented by considering the "travel light" principle. The views should include enough details to explain the functionality of the system but not more. While minimizing the contents of the views, the necessary details can be inadvertently omitted. During the formation of the Use Case Map diagrams, it is checked that whether the METU-MMDMS architecture documentation meets the selected scenarios or not. As a result, based on the UCM studies, the METU-MMDMS architecture is capable of automatic information/concept extraction of all modalities, and querying the stored information based on concept and content. The more scenarios and UCMs take part in the analysis process, the stronger the sufficiency of the architecture.

Furthermore, as an outstanding design decision, the usage of information fusion both at data level and query level enhances the performance of METU-MMDMS significantly. Thanks to METU-MMDMS's well-developed modular structure, less number of tradeoffs is encountered during the ATAM process.

As a future work, based on the information that is documented about the METU-MMDMS software architecture, and requirements of other systems in the field, a reference architecture will be composed to form a basis for the future systems.

REFERENCES

[1] Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., Stafford, J., "Documenting Software Architectures: Views and Beyond" The Benchmark Handbook for Database and Transaction Processing Systems, Ed., Addison Wesley, 2010, ISBN-13: 978-0-321-55268-6, ISBN-10: 0-321-55268-7.

[2] K. C. Nwosu, B. Thuraisingham, and P. B. Berra, Multimedia Database Systems: Design and Implementation Strategies. Boston: Kluwer Academic Publishers, 1996.

[3] Arjen P. de Vries, Mark G. L. M. van Doorn, Henk M. Blanken, & Peter M. G. Apers, The MIRROR MMDBMS architecture. Proc. of the International Conference on Very Large Databases, Edinburgh, Scotland, 1999, 758-761.

[4] Moscato, V., Indexing Techniques for Image and Video Databases: An Approach Based on Animate Vision Paradigm, Phd Thesis,UNINA, 2006.

[5] May, N. "A Survey of Software Architecture Viewpoint Models", The Sixth Australasian Workshop on Software and System Architectures, March 2005

[6] L. Bass et al. Software Architecture in Practice. Addison Wesley, Boston, MA, USA, 2nd edition, 2003.

[7] IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Institute of Electrical and Electronics Engineers, Sept. 2000. IEEE Std 1471-2000.

[8] Buhr, R.J.A., "Use Case Maps as Architectural Entities for Complex Systems". In: Transactions on Software Engineering, IEEE, December 1998, pp. 1131-1155.

[9] B. Phillips, Mediaway presses access to multimedia database. PC Week, 13(7), 1996, 39-40.

[10] Arjen P. de Vries, Mark G. L. M. van Doorn, Henk M. Blanken, & Peter M. G. Apers,

The MIRROR MMDBMS architecture. Proc. of the International Conference on Very Large Databases, Edinburgh, Scotland, 1999, 758-761.

[11] Saushik Chakrabarti, Kriengkrai Porkaew, & Sharad Mehrotra, Efficient query refinement in multimedia databases. Proc. of the IEEE International Conference on Data Engineering (ICDE), San Diego, California, USA,2000, 196.

[12] Harald Kosch, Distributed Multimedia Database Technologies supported by MPEG-7 and MPEG-21 (CRC Press. 280 pages. November 2003. ISBN: 0-849-31854-8).

[13] K. C. Nwosu, B. Thuraisingham, and P. B. Berra, Multimedia Database Systems: Design and Implementation Strategies. Boston: Kluwer Academic Publishers, 1996.

[14] Moscato, V., Indexing Techniques for Image and Video Databases: An Approach Based on Animate Vision Paradigm, PhdThesis, UNINA, 2006.

[15] C. Ogescu, C. Plaisanu and D. Bistriceanu, "Web Based Platform for Management of Heterogeneous Medical Data, Automation", Quality and Testing Robotics, IEEE, (2008) May 22-25: Cluj-Napoca.

[16] S. C. Stoica, "A Multimedia Database Server Funtionality", Second International Conference on Advances in Multimedia, IEEE, (2010) June 13-19: Athens.

[17] M. M. Shahiduzzaman, N. Mahmuda and U. R. A. Ashfaque, "Portable and Secure Multimedia Data Transfer in Mobile Phone Using Record Management Store (RMS)", 3rd IEEE International Conference On ICCSIT, (2010) July 9-11: Chengdu, China.

[18] D. Emery, R. Hilliard, M. W. Maier. Software Architecture: Introducing IEEE Standard 1471, IEEE, Vol. 34, No. 4, April 2001, 107-109.

[19] /IEC, ISO/IEC 42010:2007: Recommended Practice for Architecture Description of Software-Intensive Systems, 2007.

[20] P. Krutchen: The "4+1" View Model of Software Architecture. IEEE Software, 12, pages 42-50. November ,95.

[21] Hofmeister, C., Nord, R., Soni, D., 1999. Applied Software Architecture. Addison-Wesley, Boston.

[22] Soni, D., Nord, R., Hofmeister, C., 1995. Software architecture in industrial applications. In: Proceedings of 17th International Conference on Software Engineering (ICSE-17). ACM Press, pp. 196–207.

[23] Amyot, D., Use Case Maps: Quick Tutorial. 1999. http://jucmnav.softwareengineering.ca/twiki/bin/view/UCM/VirLibTutorial99.

[24] ITU-T, URN Focus Group, Draft Rec. Z.150 - User Requirements Notation (URN). Geneva, November 2002.

[25] C4ISR Architecture Working Group, C4ISR Architecture Framework (Version 2.0), USA, 1997.

[26] Demir, U., "Integration of Fuzzy Object-Oriented Multimedia Database Components", MS Thesis, METU, 2010

[27] Sattari, S., "Multimodal Multimedia Database Architecture Model Integration", MS Thesis, METU, 2013.

[28] Enterprise Architect, Sparx Systems, 14 Jan 2011, <http://www.sparxsystems.com.au/>

[29] D. Pilone and N. Pitman. UML 2.0 in a Nutshell. O'Reilly Media, second edition, 2005.

[30] Amyot D, Mussbacher G., and Mansurov N., "Understanding Existing Software with Use Case Map Scenarios". In 3rd SDL and MSC Workshop (SAM'02), Aberystwyth, U.K., June 2002. LNCS 2599, pp. 124-140.

[31] Mussbacher G. and Amyot D., "Goal and Scenario Modeling, Analysis, and Transformation with jUCMNav", in 31st Int. Conf. on Software Engineering (ICSE-Companion), ACM, Canada, pp. 431–432, May 2009.

[32] Kazman, Rick; Klein, Mark; & Clements, Paul. ATAM: Method for Architecture Evaluation(CMU/SEI-2000-TR-004). Software Engineering Institute, Carnegie Mellon University, 2000.

[33] Uyanıksoy, G., "Architecture Evaluation: A Case Study with ATAM and Suggestions for Architecture Evaluation Roadmap", MS Project Report, METU, 2013.

[34] Amyot, D. and Eberlein, A.,“An Evaluation of Scenario Notations for Telecommunication Systems Development", Telecommunication Systems Journal, 2002.

[35] Avcı Salma, Ç., Oğuztüzün, H, Yazıcı, A, “Bir Çoklu Ortam Veri Yönetim Sistemi Yazılım Mimarisinin “Views and Beyond” Yaklaşımıyla Belgelenmesi:Durum Raporu”, 4. Ulusal Yazılım Mimarisi Konferansı, December 2012.

[36] Gülen, E., "Fusing Semantic Information Extracted From Visual, Auditory and Textual Data of Videos", MS Thesis, METU, 2012.

[37] Okuyucu, Ç., "Semantic Classification and Retrieval System For Environmental Sounds", MS Thesis, METU, 2012.

[38] Yıldırım, Y., "Automatic Semantic Content Extraction in Videos Using A Spatio-Temporal Ontology Model", Phd Thesis, METU, 2009.

[39] Küçük, D., "Exploiting Information Extraction Techniques for Automatic Semantic Annotation and Retrieval of News Videos in Turkish", Phd Thesis, METU, 2011.

[40] Arslan, S., "Indexing Content and Concept Together for Multidimensional Multimedia Data Access", Phd Thesis, METU, 2012.

[41] Yıldırım, Y., "Fusion of Multimodal Information for Multimedia Information Retrieval", Thesis Monitoring Report, METU, 2012.

[42] Merson, P., "Data Model as an Architectural View", Technical Note, SEI, October 2009.

[43] P. Clements. Comparing the SEI’s views and beyond approach for documenting software architectures with ansi-ieee 1471-2000. Technical report, CMU/SEI, 2005.

[44] May, N. (2005), A survey of software architecture viewpoint models, in J.-G. Schneider, ed., ‘The Sixth Australasian Workshop on Software and System Architectures (AWSA 2005)’, Swinburne University of Technology, Melbourne, Australia., pp. 13–24.

[45] S.Sauer. OMMMA: An Object-Oriented approach for modelling multimedia information systems. Multimedia Information Systems 1999: 64-71.

[46] Roshandel, R. and N. Medvidovic, Modeling multiple aspects of software components, in: Proceedings of SAVCBS'03, Helsinki, Finland, 2003, pp. 88–91.

[47] J. Rumbaugh, I. Jacobson, and G. Booth. Unified Modeling Language Reference Manual. Addison-Wesley, 1998. To appear.

[48] Information technology - Open Distributed Processing - Unified Modeling Language (UML) Version 1.4.2, ISO, Switzerland, 2005.

[49] D. Pilone and N. Pitman, UML 2.0 in a Nutshell (In a Nutshell (O'Reilly)). O'Reilly Media, Inc., 2005.

[50] Sparx G., Enterprise Architect User Guide, Sparx Systems, 1998-2009.

[51] Avcı Salma, Ç., Oğuztüzün, H, Yazıcı, A, "Kullanı Eşlemesiyle Mimari Görünümlerin İrdelenmesi Üzerine Bir Örnek Çalışma", 7. Ulusal Yazılım Mühendisliği Sempozyumu, İzmir, September 2013.

[52] Uyanıksoy, G., Oğuztüzün, H., Yazıcı, A., "Bir Çoklu Ortam Veri Yönetim Sistemi Mimarisinin ATAM ile Değerlendirilmesi", 7. Ulusal Yazılım Mühendisliği Sempozyumu, İzmir, Eylül 2013.

[53] Cecchet E., Candea G., and Ailamaki A., "Middleware-based database replication: the gaps between theory and practice," in SIGMOD 2008: Proceedings of the 28th ACM International Conference on Management of Data, Vancouver, Canada, June 2008, pp. 739–752.

APPENDIX A

ARCHITECTURE VIEW TEMPLATE

Table A-1 Architecture View Template

Architecture View Template

From SAD

Primary Presentation

Add here the diagram (or non-graphical representation) that shows the elements and relations in this view. Indicate the language or notation being used. If it's not a standard notation such as UML, add a notation key.

Element Catalog

This section can be organized as a dictionary where each entry is an element of the Primary Presentation. For each element, provide additional information and properties that the readers would need that would not fit in the Primary Presentation. Optionally, you can add interface specifications and behavior diagrams (e.g., UML sequence diagrams, state charts).

Context Diagram

Add here a context diagram that graphically shows the scope of the part of the system represented by this view. A context diagram typically shows the part of the system as a single, distinguished box in the middle surrounded by other boxes that are the external entities. Lines show the relations between the part of the system and the external entities.

Variability Guide

Describe here any variability mechanisms used in the portion of the system shown in this view, along with how and when (build time, deploy time, run time) those mechanisms may be exercised. Examples of variability include: optional components (e.g., plug-ins, add-ons); configurable replication of components and connectors; selection among different implementations of an element or different vendors; parameterized values set in build flags, .properties files, .ini files, or other .config files.

Rationale

Describe here the rationale for any significant design decisions whose scope is limited to this view. Also describe any significant rejected alternatives. This section may also indicate assumptions, constraints, results of analysis and experiments, and architecturally significant requirements that affect the view.

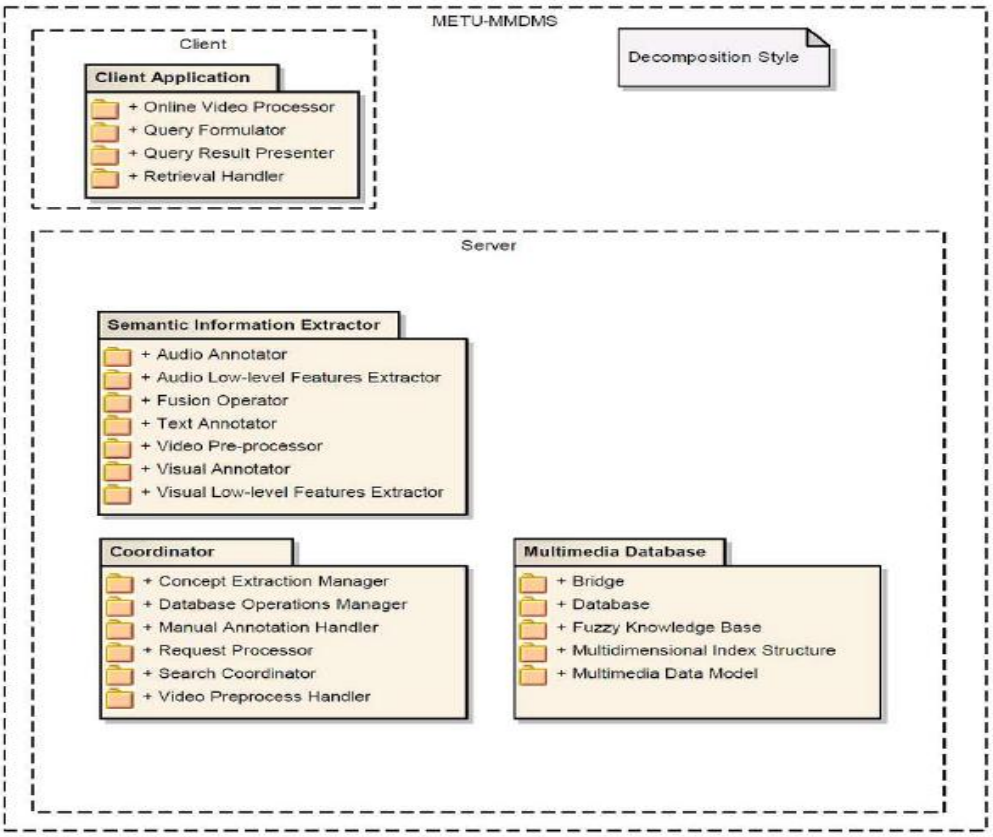
APPENDIX B

SAMPLE ARCHITECTURE VIEWS

Top Level Module View

From Thesis Wiki

Primary Presentation



Element Catalog

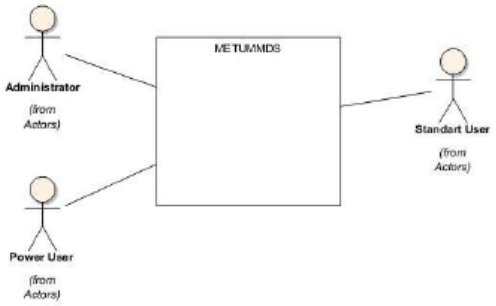
- **Client :** A prototype client application is developed for testing and evaluating presented architecture. *JMF* and *JavaWS* technologies are used in client module.
- **Server :** A server framework, which tightly couples the multimedia database modules, is implemented by using *Java*. The server covers the Semantic Information Extractor, Multimedia Database and Coordinator modules.
  - **Semantic Information Extractor :** It is the module which implements object and event extraction. Semantic Information Extractor includes Audio Annotator, Fusion Operator, Audio Low-level Features Extractor, Visual Low-level Features Extractor, Text Annotator, Video

Figure B-1 Top Level Module View

Preprocessor and Visual Annotator modules.

- **Multimedia Database :** *DB4O* database and *JESS* knowledge base are used in core database structure as storage and fuzzy inference engine. A multidimensional index structure is presented for efficient data access.
- **Coordinator :** Semantic Information Extractor and Multimedia Database components are tightly coupled through a coordinator structure. Coordinator module organizes the data flow.

Context Diagram



Variability Guide

Client application can be re-designed in order to meet new queries, which are not described in the scope of this SAD. Furthermore, the fields of queries can be rearranged.

Rationale

System has a modular structure and the boundaries between modules are defined clearly (**maintainability**). The classification algorithms will be replaced with other suitable algorithms. The user interfaces will easily be modified. (**modifiability**).

Related Views

Retrieved from "http://localhost/thesiswiki/index.php?title=Top_Level_Module_View&oldid=904"

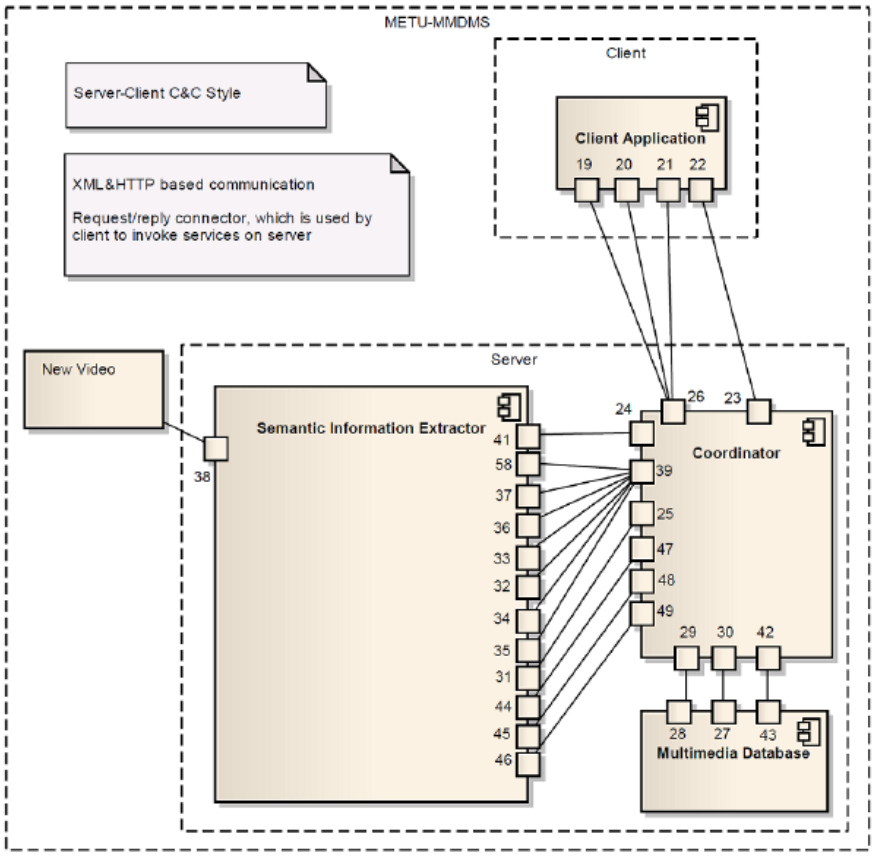
- This page was last modified on 17 August 2013, at 14:42.

Figure B-1 (continued)

Top Level C&C View

From Thesis Wiki

Primary Presentation



Element Catalog

Coordinator

The coordinator component is the interface of the architecture to the outer world.

Semantic Information (Concepts) Extractor

Semantic Information extraction component, provides information to be stored in Multimedia Database.

Mutimedia Database

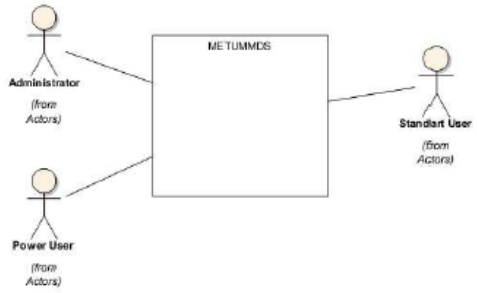
Multimedia Database is an intelligent information management system, which offers an efficient interaction between database and knowledge base technologies.

Client Application

Figure B-2 Top Level C&C View

Textual and similarity based queries are built using the interfaces of client component. Also, object and event annotation functions are invoked from relevant interfaces in this part. Sending the constructed query to the server and receiving the results coming from coordinator is also client's responsibility. A video player is present as part of the client application.

Context Diagram



Variability Guide

Each component will be replaced by alternative components with same interfaces. Due to minimized communication messages between client and server framework, the implemented system can also be used in mobile systems.

Rationale

In METU-MMDMS architecture, as a requirement of thin client technology, all the operational functions are recommended to be executed in server side components and the supplementary functions, which mostly requires user interactions, are suggested to be performed on client side applications.

Since even processing of a short video requires a huge amount of time, and some components of the object annotator requires user interaction, this module is modified to work online on web. Because key frame extraction and segmentation functions require user interaction, these functions of object annotator module are moved to the client side, and feature extraction and classification operations are left to be executed in the server.

Since multimedia data usually exists in huge sizes, transferring resulting objects from server to client may cause problems in slow connections. Wherever lesser quality data provides hi-speed transfer rates, but worse object classification performance, higher quality visual data slows-down the system, but ensures high accuracy in object classification. Therefore an ability to select network speed for image quality is added to the client application.

Query interfaces of client application are formed at runtime according to an XML document, defining domain and semantic entity information. Therefore, domain modifications or changes in server framework require very few modifications for prototype client application.Since the proposed METU-MMDMS architecture recommends using XML based communication methods, interaction between server and client instances is established using both XML and HTTP messaging standards. Therefore, any other client application can easily connect to the server framework and use the implemented sample multimedia database framework so long as they satisfy the requirements of messaging protocol. Due to minimized communication messages between client and server framework, the implemented system can also be used in mobile systems. For **interoperability** with other systems, some web services are also developed and presented.

Web services simplifies the integration of another system to IFOOMMDS (**interoperability**).

The system is easy to test (**testability**).

The system is easy to use (**usability**).

System is .. % accurate (**accuracy**).

Index structure query & information extraction efficiency is improved for high **performance**.

Depending on the flexibility and extandibility of the coordinator structure, the system will be more scalable (**scalability**).

Suggestion: Mediator design pattern will be used in coordinator, because many classes communicate through coordinator component and to preserve decoupling and provide **interoperability**, mediator pattern is usefull. **Strategy** design pattern suits for the semantic concepts extractor component. By the help of strategy pattern, depending on the incoming data, the related algorithms will be selected at runtime. **Adapter** design pattern will be necessary between the storage and semantic concepts extractor or client and coordinator. The data format that is flowing between these component should somehow match by the help of adapters. **Visitor** design pattern will be used in bridge to handle multimodality. All the components should be designed as **singletons**.

Related Views

Retrieved from "http://localhost/thesiswiki/index.php?title=Top_Level_C%26C_View&oldid=908"

- This page was last modified on 17 August 2013, at 14:57.

Figure B-2 (continued)

APPENDIX C

TRACEABILITY OF FEATURES

Table C-2 Feature Traceability Matrix

Feature ID	**Section**
[Feature-1]	[3.5.1.1] [3.5.2.4]
[Feature-2]	[3.5.1.2] [3.5.2.1]
[Feature-3]	[3.5.1.1] [3.5.2.4]
[Feature-4]	[3.5.1.2] [3.5.2.1]
[Feature-5]	[3.5.1.1] [3.5.2.4]
[Feature-6]	[3.5.1.2] [3.5.2.1]
[Feature-7]	[3.5.1.1] [3.5.2.4]
[Feature-8]	[3.5.1.4] [3.5.2.2]
[Feature-9]	[3.5.1.4] [3.5.2.2]
[Feature-10]	[3.5.1.1] [3.5.2.4]
[Feature-11]	[3.5.1.1] [3.5.2.4]
[Feature-12]	[3.5.1.3] [3.5.2.3] [3.5.1.2] [3.5.2.1]
[Feature-13]	[3.5.1.3] [3.5.2.3] [3.5.1.2] [3.5.2.1]
[Feature-14]	[3.5.1.4] [3.5.2.2]

APPENDIX D

USE CASE MAPS

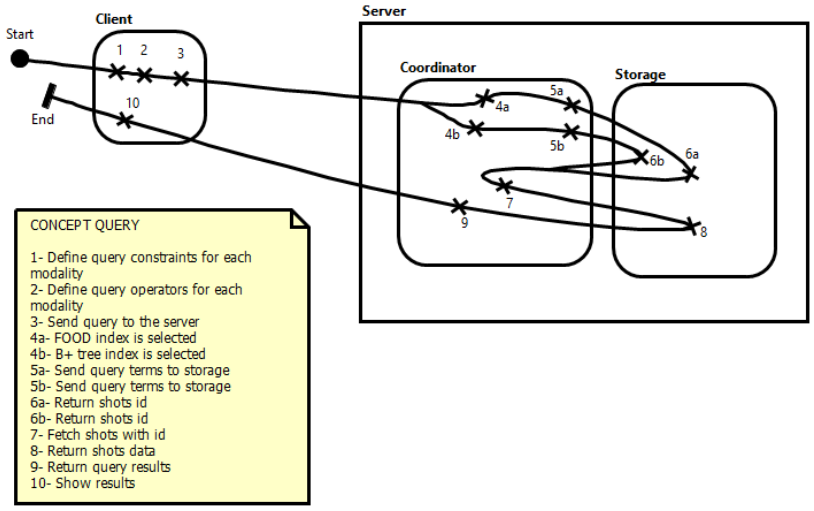


Figure D-3 Concept Query

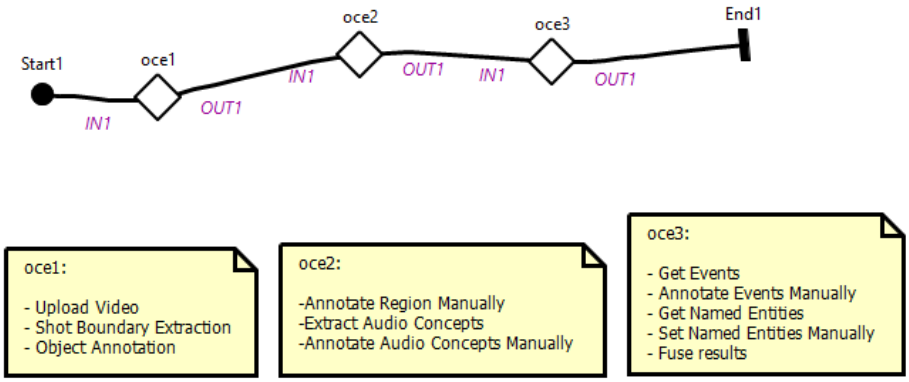


Figure D-4 Online Concept Extraction

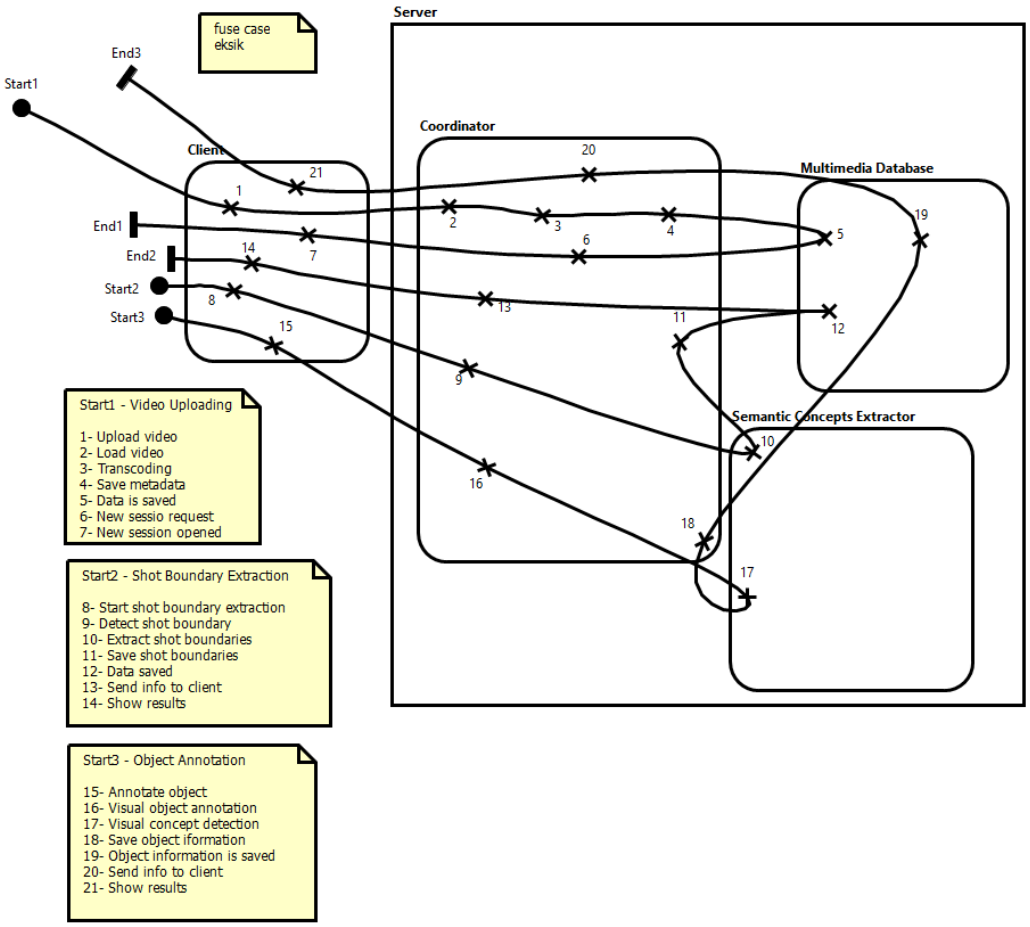


Figure D-5 Online Concept Extraction - 1

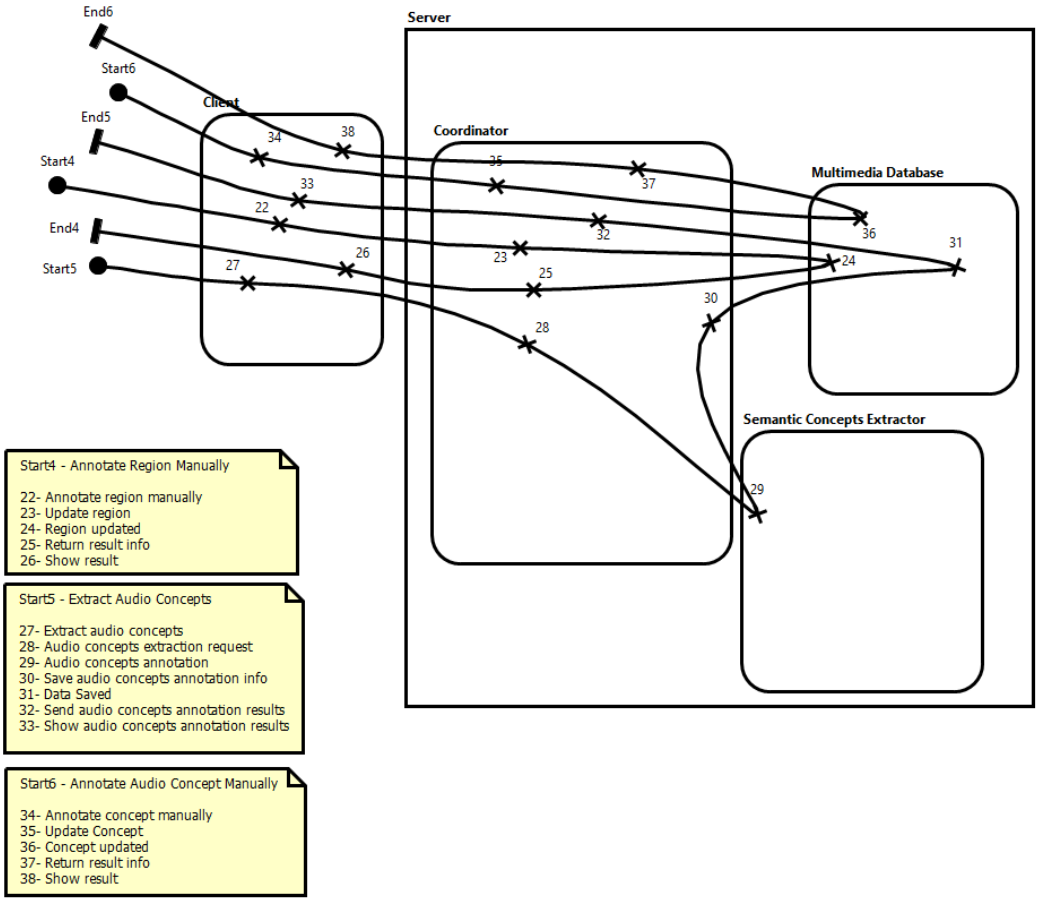


Figure D-6 Online Concept Extraction - 2

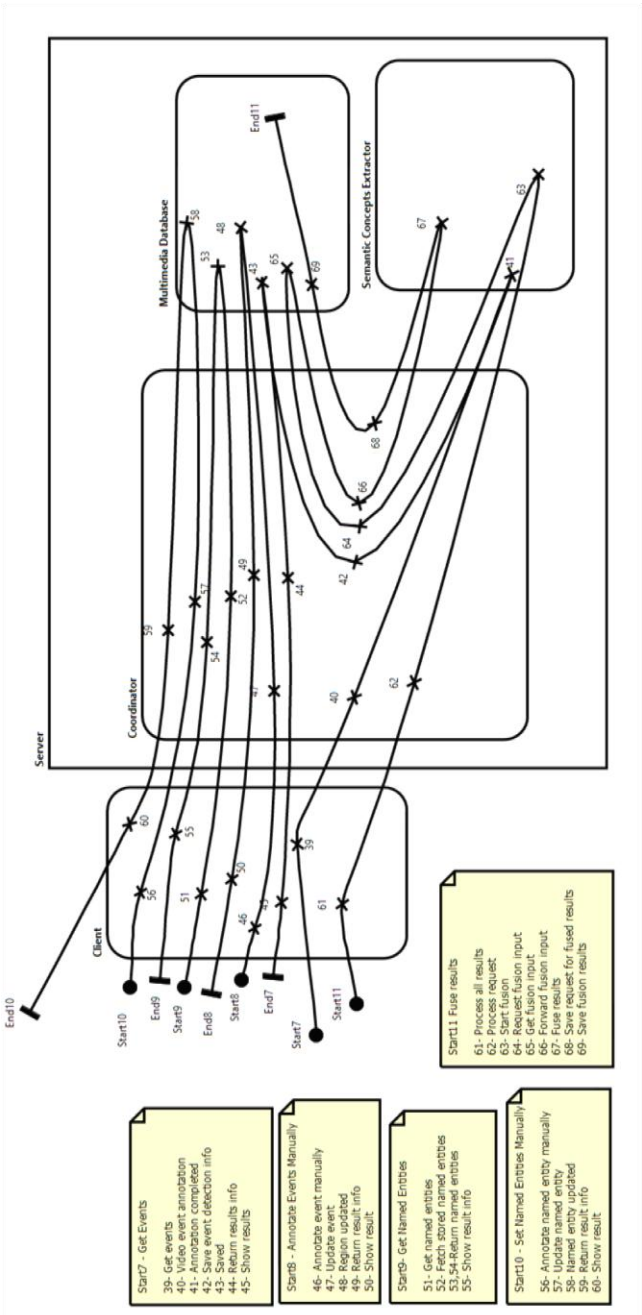


Figure D-7 Online Concept Extraction - 3

APPENDIX E

USES VIEW DIAGRAMS

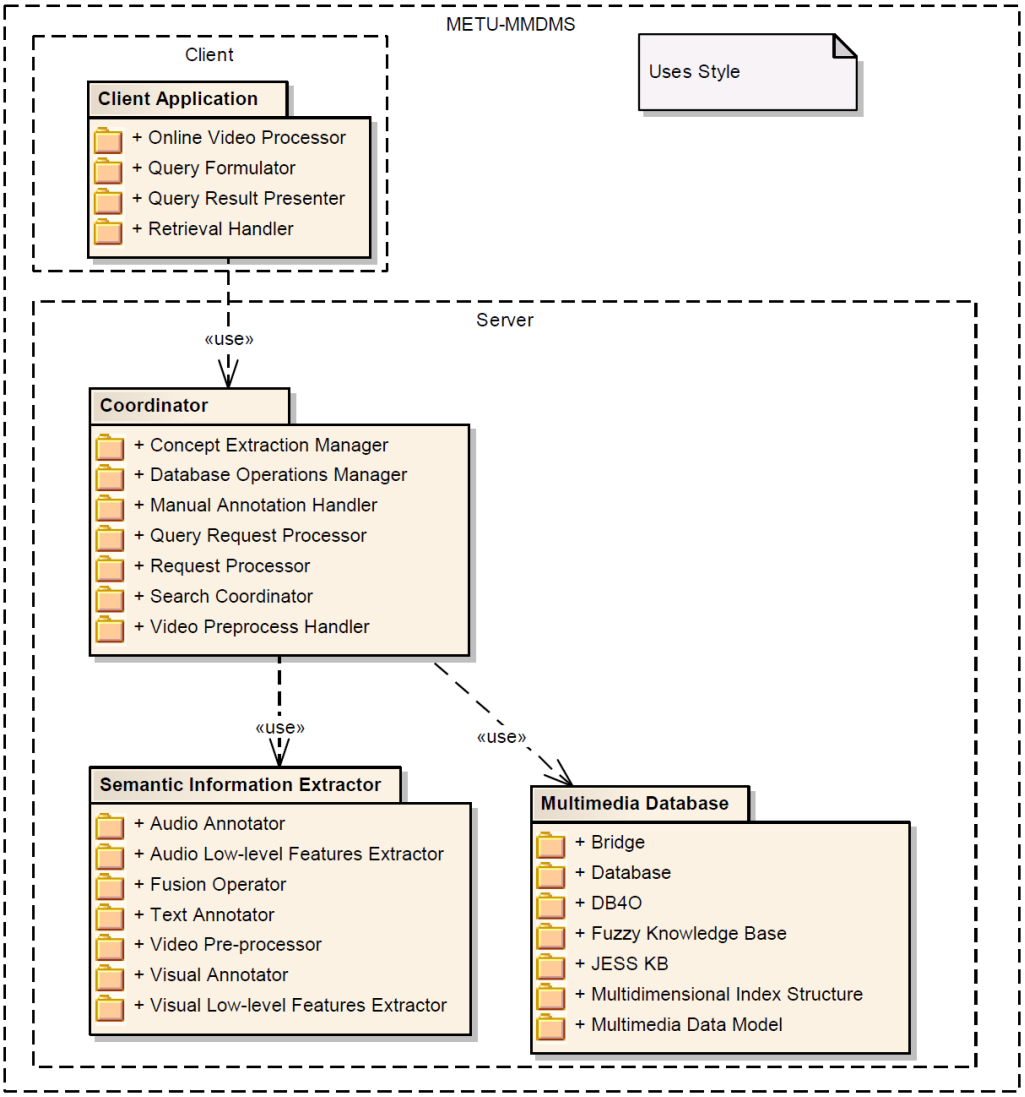


Figure E-8 Top Level Uses Module View

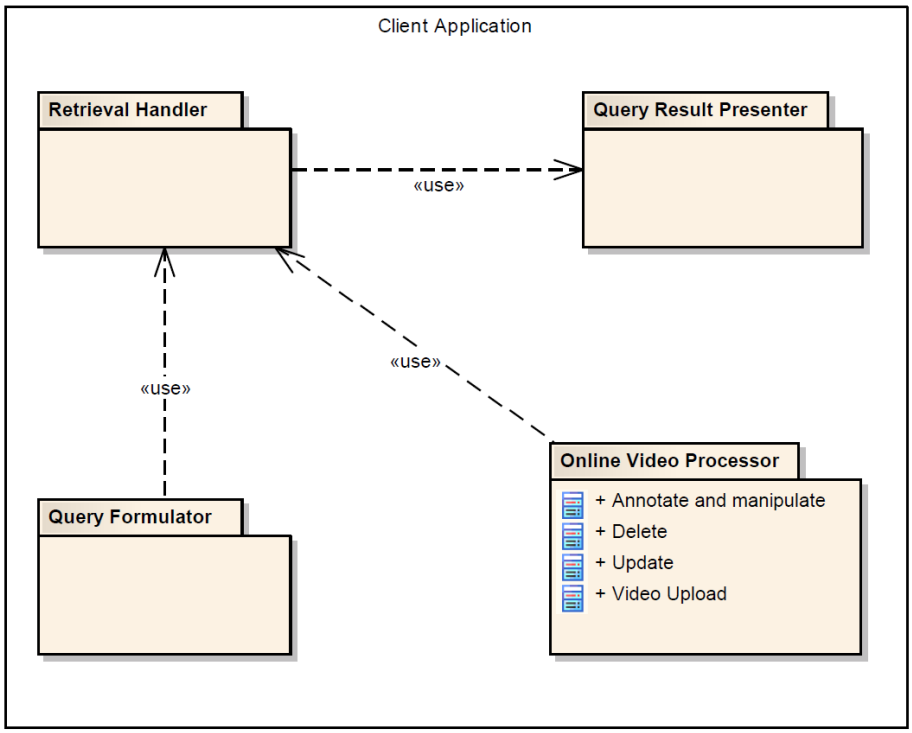


Figure E-9 Uses Module View of Client

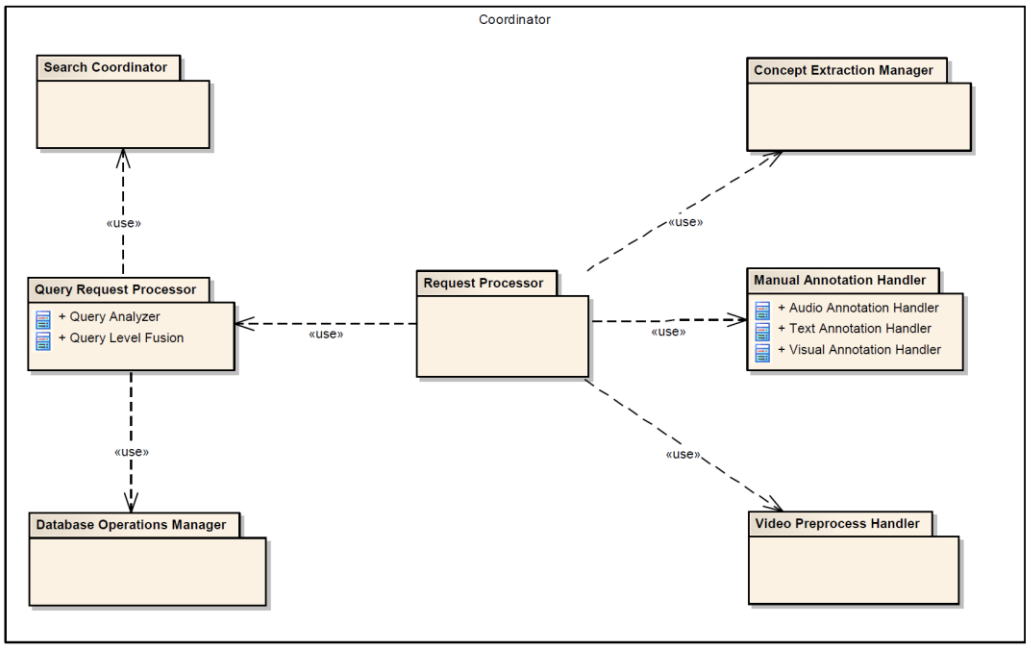


Figure E-10 Uses Module View of Coordinator

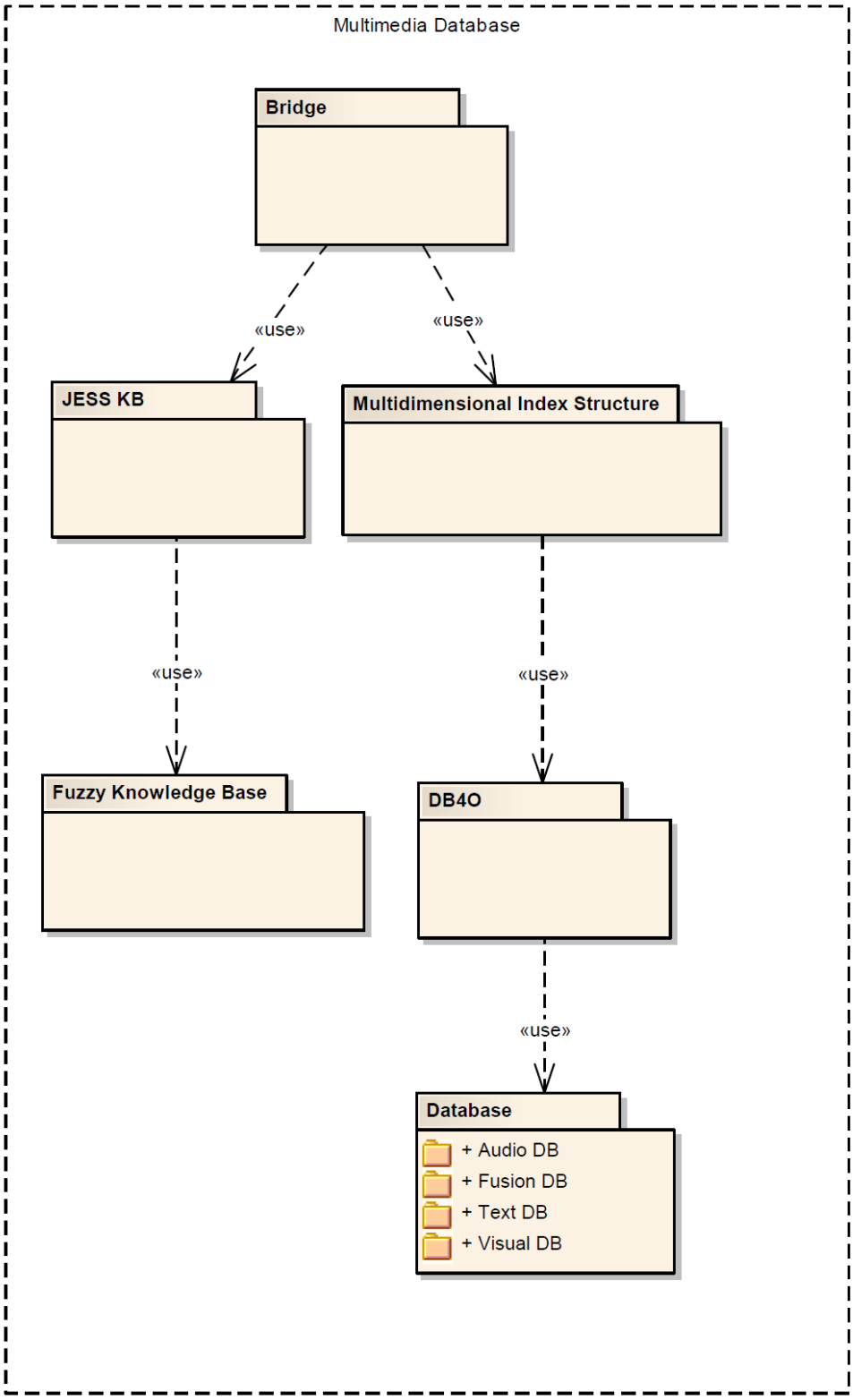


Figure E-11 Uses Module View of Multimedia Database

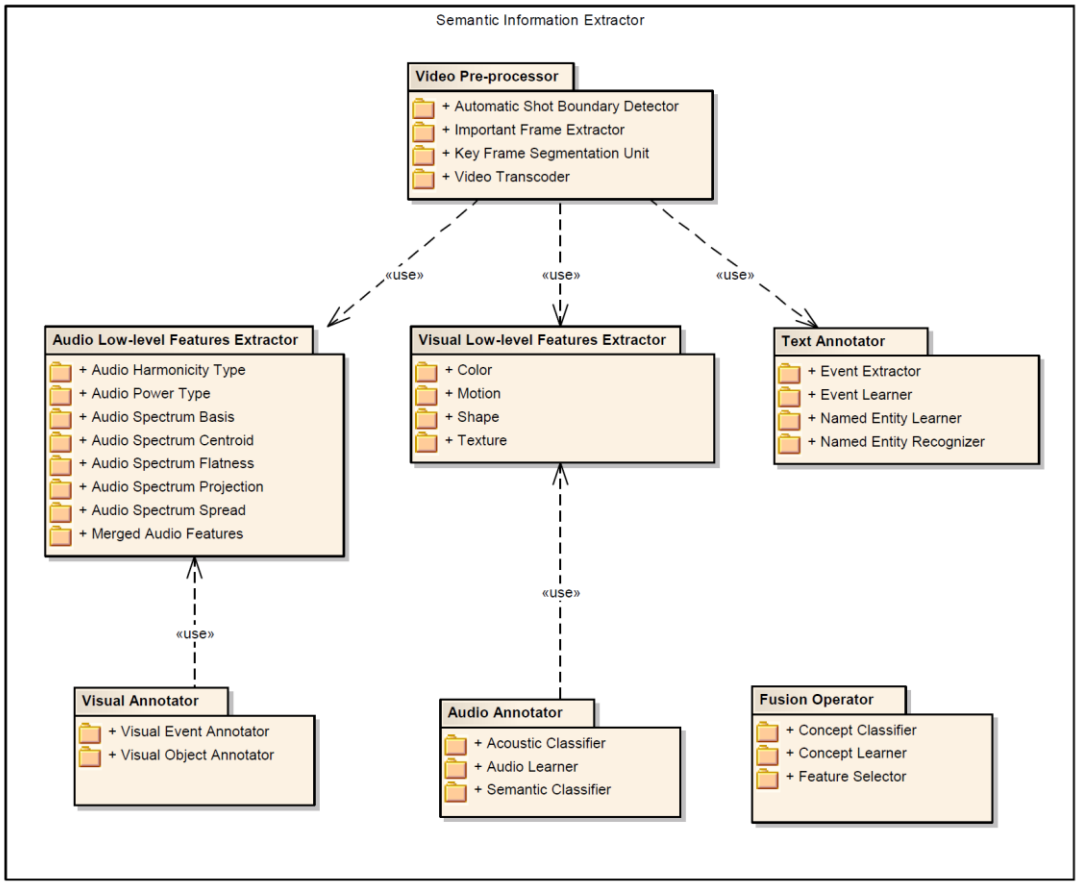


Figure E-12 Uses Module View of Semantic Information Extractor