Conceptual Model and Framework

The task of multimedia generation is to be represented conceptually using knowledge modeling techniques. This proposed approach is aimed at implementing an Intelligent Multimedia Presentation System that can take advantage of the standards and developments around the Semantic Web. To support an implementation, the conceptual model is complemented by a programmatic framework for its interpretation.

To facilitate the automated, dynamic and intelligent delivery of multimedia content we have stated our intention in this thesis to propose a multimedia presentation system which will leverage the machine-understandable knowledge of the Semantic Web to find, adapt and integrate content for delivery as a multimedia presentation.

The limitations encountered in implementing the multimedia generation process have been outlined and knowledge representation techniques introduced as they are increasingly being applied as solutions to these problems. We evaluated prototypes of intelligent multimedia presentation systems – which use knowledge-based approaches to solve the problems of multimedia generation - against the Standard Reference Model for such systems. We have found that no one existing system fulfils all the aims of a generic IMMPS. In this chapter we present a conceptual model for multimedia generation and a programmatic framework for implementing a system which operates from that model.

This system – the **Semantic Web-enabled Multimedia Presentation System (SWeMPs)** – is based around a conceptual framework which is introduced in this chapter. This framework is based on the use of a conceptual model to represent the domain of multimedia presentation generation, a rulebase which interprets a knowledge base instantiated in terms of this conceptual model to realise the multimedia generation task, and a specification for leveraging services for the modular, loosely coupled execution of computational processes to produce the final presentation. It will act as the fundamental basis for the further development work in SWeMPs, which will consist of a domain model (ontology), a demonstration execution environment (application) and a specification of guidelines for the components of the system.

4.1 Overview of SWeMPs

In SWeMPs, the aim is that “intelligent information services” (IIS) can be authored based on the high level representation of the conceptual world in which the system is assumed to be operating (the core conceptual model of the SWeMPs plus ontologies specific to the IIS application domain) and an information request which can be understood in terms of that conceptual world.

Individual IIS are defined through a knowledge base populated with conceptual instances, which are interpreted through generic rules-based application logic. In other words, IIS specifics (such as the means to determine which resources may be presented or which services may be invoked) are declared explicitly and declaratively in the knowledge model. This knowledge is reasoned with using an ontological reasoner (thus able to infer new knowledge) and rules triggered to select and execute services according to the facts in the system. These services are co-ordinated by the system to answer the information request with resources, whether this is further knowledge or presentable content. The interaction with the services is facilitated by their semantic descriptions. An iterative process results in a resource set where those resources are answers to the user's information request. Based on the knowledge in the system regarding those resources (e.g. which concepts they relate to and what sort of relationship they have to one another), domain-specific presentation rules (which can also take into account execution-specific contextual knowledge such as user preferences or device capabilities) combined with generic presentation rules adapt those resources and integrate them into a synchronized multimedia presentation.

Such a system differs from existing multimedia presentation systems in that it operates fundamentally at the conceptual level rather than the syntactic level. This reflects the reality that what users seek is information (knowledge about concepts) and effective communication of that information (respecting the relationships between concepts). It leverages the potential of using the Semantic Web as a distributed, large scale, accessible repository of knowledge about the world of the user just as systems today use the Web as a repository of content.

4.1.1 Plan for realisation

In order to realise SWeMPs a "plan for realisation" has been defined, which inspires the conceptual model and framework outlined in the following sections (4.2 to 4.4). It consists of three paths:

– Conceptual Model for Multimedia Presentation

SWeMPs will operate on the basis of a high-level representation of the conceptual world in which the service is deemed to be operating. This will require consideration of knowledge modelling and representation techniques to produce a practical model which can comprehensively model all requisite knowledge for the system. This requisite knowledge covers the domain of multimedia presentation, that is, all concepts and relationships necessary to guide SWeMPs in carrying out the end-to-end service execution from information request to multimedia presentation. As a final result, a domain model for SWeMPs will be formally defined.

The methodology for developing the conceptual model will be introduced in section 4.2 and the definition of the conceptual model is described in section 4.4.

– Conceptual Framework for SWeMPs

The domain model is intended as the functional basis for a generic application which knows how the model is to be correctly interpreted in order to realise the end-to-end functionality of producing a multimedia presentation from an information request. This application is intended to be as simple as possible, with the true realisation of the multimedia generation process being explicitly and declaratively expressible through the instantiation of the conceptual model. We expect this to support re-usability, modification, persistence and extensibility of the separate aspects of the multimedia generation process, including its interoperation with the knowledge of the Semantic Web in moulding and guiding the overall process of execution.

The conceptual framework then will be based around rules for interpreting the knowledge given to the system in terms of the concepts of the conceptual model. It will follow these rules in its co-ordination of services implemented in the system or discovered by it at runtime to perform the tasks of information retrieval, adaptation and presentation. It will do this by passing to those services the requisite input and then receiving from them their processing result. It will also handle inconsistencies in its data and process flow with regard to the services it is co-ordinating, potentially by calling mediating services which can provide a consistent view of heterogeneous data or process models.

The framework's core functional aspect, then, is the execution of a rule-set upon the system knowledge, the calling of services to realise specific sub-processes as a reaction to the responses to those rules and the co-ordination of these sub-processes so that data and process heterogeneity is respected, errors and inconsistencies handled e.g. by process backtracking, and as a final result a multimedia presentation is made available to the client.

As a proof of concept the implementation of a working application based on this framework will be undertaken that demonstrates the SWeMPs approach in a concrete way. We aim to specify a generic rules set for the interpretation of the conceptual model for multimedia presentation in order to realise the full multimedia generation process.

The conceptual framework is introduced and described in more detail in terms of its components, rules and process in section 4.3.

– Next Generation Web services

While SWeMPs contains its entire functionality in the framework and corresponding conceptual model, that functionality includes the co-ordination of other processes to carry out an actual multimedia presentation generation task. These other processes could prove to be distributed software components on the

Web, which take data from the application, interpret it in order to be able to use it (possibly with the aid of system reasoning with available knowledge), and respond with some result.

It is clear that SWeMPs is only practically capable of carrying out its multimedia presentation in that there are such other services available on the Web to execute the various sub-processes. We argue for an approach using external, distributed services in that we recognize that the open, distributed nature of the Web itself (and, hence, also the Semantic Web) leads to heterogeneity of data, both in terms of Web resources and knowledge statements. Any particular execution of SWeMPs will require various functionalities which may not even be knowable before the execution, in terms of handling certain data formats, understanding certain semantic vocabularies, making certain queries or interpreting certain responses from Web systems. Rather than enforcing that all of these functional requirements must be expressed to the service before its execution, we expect that services can be discovered during execution as soon as certain functionality is required. This is more error resilient, in that the changeable nature of the Web means we must take into account that in any given execution the response from a Web system may include data or knowledge that was not expected, i.e. in a different format or representation, or that a given Web resource or process may be at any given moment unavailable. Rather than expect the execution to break, we dynamically search for an equivalent alternative e.g. finding a mediator to interpret the data to the system. However we do not expect any of these services to be actually available on the Web at present.

Rather, the final part of the SWeMPs framework is to specify how such services could be deployed on the Web and made available to SWeMPs at runtime. It would be the responsibility of the Web community as a whole to produce the necessary services for the sub-processes of SWeMPs, with the recognition that the breakdown of functionalities into sub-processes will make individual service functionalities general enough to be re-usable i.e. once a service is available for mediating between vocabularies X and Y, that service can be re-used by executions of SWeMPs (or other systems that need this mediation) and does not require anyone to repeat the service implementation. A body of services should be distributed and available on the Web, to support the sub-functionalities of the multimedia generation task while also being potentially usable in other application scenarios. In fact, online mediation services for data and knowledge are a general requirement of the Semantic Web infrastructure.

The idea of using external services is included in the discussion on the conceptual framework in this chapter in terms of the service planner and directory components. We restrict ourselves in this thesis (section 5.7) to defining the interfaces for typical service types.

4.1.2 Summary

Figure 4.1 illustrates the plan for realisation in the form of a house, with a foundation (the Semantic Web), three pillars (which are the paths to realisation outlined in the previous section) and the roof (SWeMPs itself). The contributions of this thesis are highlighted by using dashed lines. While building upon other efforts (Semantic Web and Next Generation Web Services) we illustrate that SWeMPs is only possible if built on top of two further efforts: that of defining a conceptual model for multimedia generation as well as a conceptual framework for a multimedia presentation system which uses this model. By providing those definitions in this chapter, we can put on the roof in chapter 5 and complete the house.

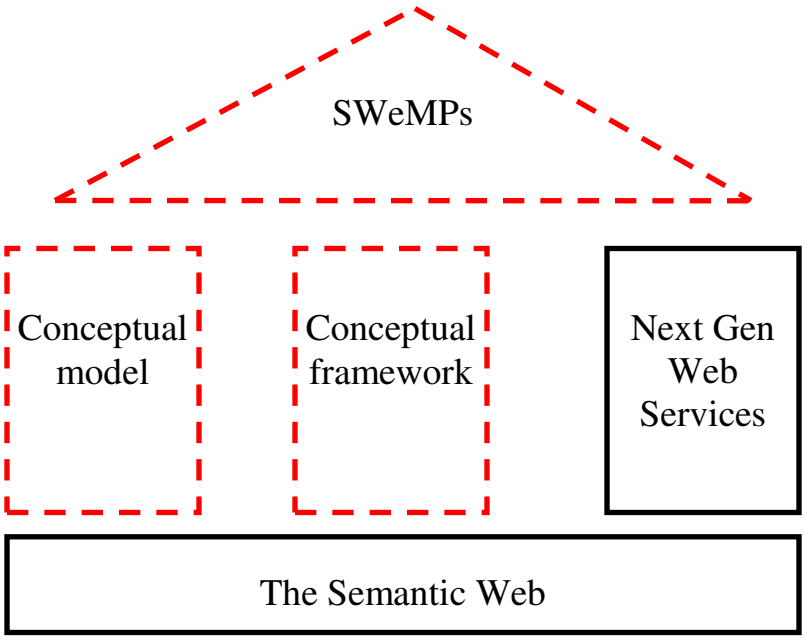

Figure 4.1 Plan for realisation of SWeMPs

4.2 Definition of the conceptual model for multimedia generation

One aim of this work is to build a formal knowledge model of the process of multimedia generation. This model is to be used with a multimedia presentation system to produce multimedia presentations. Conceptual models play an important part in many application areas, such as [Mylopoulos,1998]:

- *Artificial intelligence programs*, which relied on semantic networks and description logics
- *Database design*, which involved a conceptual level schema such as Chen's Entity-Relationship model
- *Software development*, whose initial requirements specification can be seen as a conceptual model

- *Object-oriented software*, which has proposed viewing software components (classes/objects) as models of real world entities. This technique can be seen e.g. in UML.

Fundamentally in this approach the aim is to express individual multimedia generation tasks in a modular, formal, explicit, declarative and interoperable way so that "intelligent information services" can be built upon the generic application rules through the selection and integration of different domain knowledge with a core knowledge base. This core knowledge base, which deals with the domain of generating multimedia presentations independent of any particular topical domain of the service itself, is based upon a formally defined shared conceptual model so that the knowledge base can be authored, validated, maintained, exchanged, and interoperated with on the basis of it conforming to a known ontology.

An ontology is a form of knowledge modelling. Ontology languages are a means for knowledge representation (KR). Ontologies are based upon logic, though the extent of formalism and level of expressivity vary. We want to develop firstly a conceptual model for the multimedia presentation generation domain that is as generic as possible, i.e. expressed through a minimal set of logical constructs for knowledge representation. As a result we have a conceptual model which can be independent of any specific knowledge representation model.

Such a model has a number of advantages for an application which uses it [Brachman,1991]. A knowledge model can exhibit "self-organisation" of its content according to the logical axioms which govern it and specific rules given for the individual application. Information retrieval can become more sophisticated through a more complex classification of the data being stored. A knowledge model is also useful for applications with evolving, changing information. Here a partial, incomplete view of the domain of discourse can be incrementally altered and filled.

We intend to use a knowledge representation approach in the argument that by so doing:

- A multimedia generation process is defined in an unambiguous way as the meaning of the concepts in that process are clearly defined
- That meaning is not just human but also machine understandable, so that the model of the process could be used by an automated computer system without further manual oversight
- The process is modelled in an explicit declarative way, easing authorship (no working with internal logic / programming code), validation (test for semantic consistency) and maintenance (model visualisation, development tools).
- The use of Web based ontology techniques introduces distribution of the model, as well as extension and interoperation with other knowledge sources.

- The functional tasks of the multimedia generation process (retrieval, adaptation, presentation) are able to be based on logical inference mechanisms.

4.2.1 Basic Constructs

Considering the core constructs of KR systems [Brachman,1991], we focus on a *frame-based* approach, i.e. an approach to KR through describing objects rather than asserting logical statements. In such an approach (e.g. implemented in CLASSIC[29]) there are three kinds of formal object:

- **Concepts** – descriptions with a potentially complex structure (one place predicates)
- **Roles** – formal terms for properties (two place predicates); roles filled by a single individual are called attributes.
- **Individuals** – single formal representations of an object, created by asserting that they satisfy concepts ("is-a") and that their roles are filled by other individuals.

As a KR system is logic-based it is possible to impose logical consequences and make logical deductions upon the formal objects of the knowledge base. Logical consequences include inheritance, combination, propagation, contradiction detection and incoherent concept detection. Deductions can be based on concept classification (deducing generalization or specialization of concepts), individual classification (deducing all concepts that an individual satisfies), subsumption (deducing if one concept is more general than another) and rule application (if an individual satisfies the antecedent of a rule, it is asserted to satisfy the consequent as well). A KR system such as CLASSIC allows for partial domain expression through stating that objects participate in roles without stating which particular object, refining incomplete or incorrect information, and by omitting the "closed world assumption" to not allow the drawing of conclusions until all information is known.

The typical approach to describe a new concept or individual in CLASSIC is to give a list of more general concepts and then a list of restrictions that specify how this new concept or individual differs from the more general concepts. To produce a coherent and consistent knowledge base, care must be taken in the building of the knowledge model. Important questions that must be answered for every object to be represented in the model are [Brachman,1991]:

- Is a concept primitive or defined? (i.e. are the conditions for membership inclusive or exclusive of other conditions?)
- Is a property definitional or incidental? (i.e. does the property represent an inherent quality of the concept or simply something that all individuals of this concept would share?)

[29] See http://www.research.att.com/sw/tools/classic/

- Is an object a concept or an individual? (Individuals are unique, countable, and changeable.)

A vast literature of methodologies for identifying objects, classes, properties etc. for a particular application exists in the field of object-oriented software development [Schlaer,1988] but we will use the simple knowledge engineering methodology of [Brachman,1991] which is specifically using the CLASSIC DL-based knowledge model. The steps of the methodology are summarized thus [Borgida,2002]:

- Identify the individuals one can encounter in the universe of discourse.
- Enumerate concepts that group these values.
- Distinguish independent concepts from relationship-roles.
- Develop a taxonomy of concepts.
- Identify any individuals (e.g. enumerated values) that are of interest in all states of the world in this universe of discourse.
- Systematically search for part-whole relationships between objects, creating roles for them.
- Identify other properties of objects, and general relationships in which objects participate.
- Determine local constraints involving roles such as cardinality limits and value restrictions. Elaborate any concepts introduced as value restrictions.
- Determine more general constraints on relationships, such as those that can be modelled by sub-roles or “same-as”.
- Distinguish definitional from incidental properties of concepts, as well as primitive from defined concepts.
- Consider properties of concepts (such as identifiers) and simplify/realign the taxonomy of primitive concepts.

From these steps, we can produce an initial conceptual model for the multimedia generation system SWeMPs. Firstly, in order to enable me to take the first step (identify the individuals one can encounter in the universe of discourse), we turn to defining the conceptual framework of the system. From this definition we expect to be able to extract the concepts and properties that need to be modelled by SWeMPs. The resulting formal specification of the conceptual model is given in section 4.4.

4.3 Definition of the conceptual framework for multimedia generation

A fundamental question at this stage is how to begin to design this system. Here we draw upon the work of the Design Research community in Information Systems[30]. This field is defined as ‘the analysis of the use and performance of designed artefacts to understand, explain and very frequently to improve upon

[30] Design Research in Information Systems http://www.isworld.org/Researchdesign/drisISworld.htm

the behaviour of aspects of Information Systems'. Such artefacts include system design methodologies and languages. As the proposed system is too large and complex for this research work alone to produce a full implementation, we will rely on established design methodologies not only to ground our proposal but also to be able to infer from the design the expected results, i.e. to be able to evaluate the design without a full implementation by using established axioms in system design.The reasoning that occurs in the course of a general design cycle has been analyzed and is illustrated in Figure 4.2.

According to this model, all design begins with *the awareness of a problem*. In the case of this research, this problem has been introduced in Chapter 1. The next stage is to draw suggestions from the problem area – in Chapters 2 and 3 we introduced the issues of multimedia generation and knowledge representation and gave an overview of past and current systems attempting to address aspects of the given problem. This chapter introduces our own suggestion in this field, based on the prior investigation and drawing in particular from the Standard Reference Model of the Intelligent Multimedia Presentation System (IMMPS). The following two chapters describe the development and evaluation stages. A realisation of Design Research is that knowledge is only generated *from the specific act of construction*. In other words, the development acts as the 'proof' of the value of the suggestion.

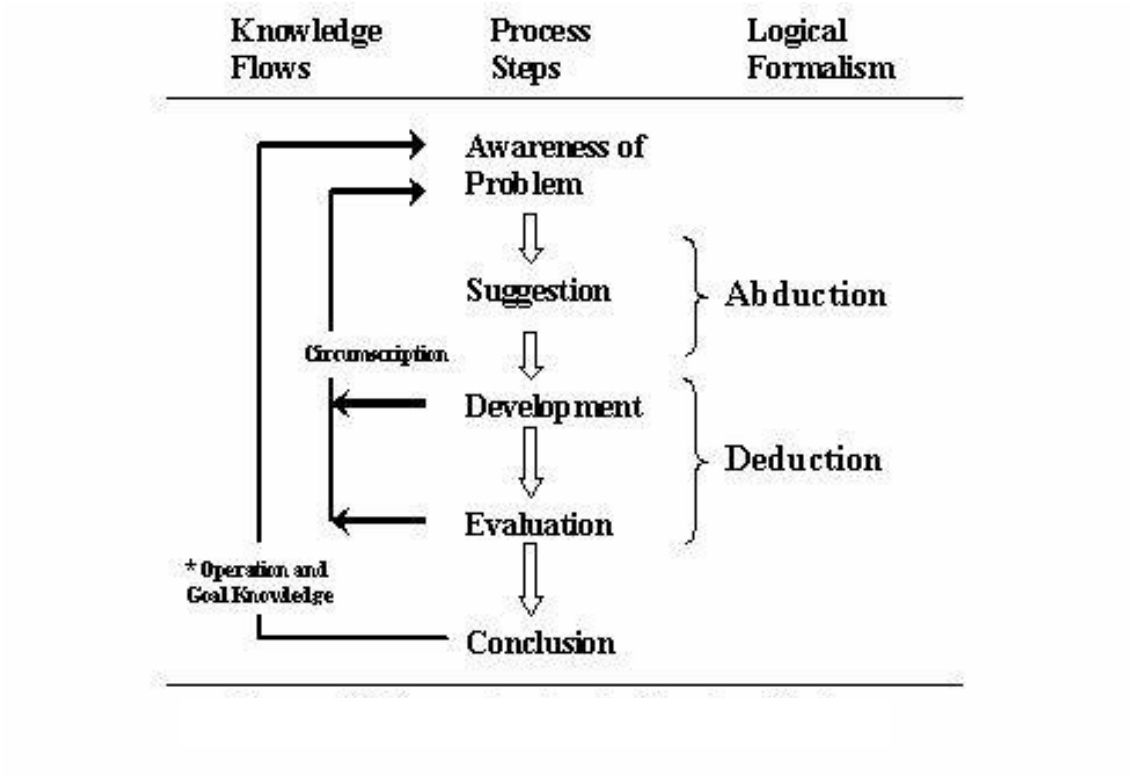

Figure 4.2 The General Design Cycle [Takeda,1990]

It is worth noting that the model and framework presented here can be considered the final design cycle of our research. Earlier design cycles saw the development of other systems based on different prior suggestions, which were then evaluated and on the basis of the generated knowledge the suggestion could be revised. Details of both the XML- and Topic Map-based designs can be found in the published papers given in the appendices.

4.3.1 SWeMPs requirements analysis and system design

The SWeMPs framework proposed here is at its core influenced by the proposed Reference Model for an Intelligent Multimedia Presentation System (IMMPS). However, the latter model incorporates knowledge bases while not specifying how knowledge would be represented in them or how the application would interact with the knowledge bases. It also separates the knowledge for the multimedia generation process into four specific sub-categories. In our approach, in which we focus on Semantic Web techniques for the knowledge representation and interaction (as our aim is to integrate the multimedia generation process with the distributed knowledge that will be available on the Semantic Web), we choose to specify a single knowledge base for the process as a whole, extending it when necessary with specific knowledge for different application-specific domains (which could be domains mentioned in the IMMPS architecture such as user or device, but could also be other domains too).

Similarly, we follow agreed conventions being specified in the Semantic Web effort of the W3C and build our efforts upon a known knowledge representation model and methodology as well as incorporate Semantic Web tools to enable a standards-based interaction between the application and available knowledge.

Hence the resulting framework can be conceived as being a combination of approaches in building Semantic Web applications and in building multimedia presentation systems. We note here some general aspects and decisions made in designing this framework, which form an initial basis for design decisions regarding the system:

- In common with all knowledge-based applications, we conceive an application with a separate knowledge base, noting that the knowledge base can be stored separately, i.e. does not have to reside on the same server as the application. The separation of application and knowledge base is also shown in the Standard Reference Model of IMMPS (Fig 2.4).

- As all knowledge of use to the application may not be contained within the resource-limited size of the local knowledge base and a large knowledge base could be inefficient, knowledge sources distributed on the Web can be referenced and incorporated into the application knowledge dynamically during execution. This necessity can be seen in the

interactive television scenario, where the user has a lot of control over the choice of data and the content of the program annotation may not be knowable in advance, hence it would be very inefficient to try to store all possibly relevant data locally prior to program broadcast, as well as considering that a Set Top Box is limited in available resources.

- As the generation process will be guided by the available knowledge (which may be changing continually), decisions about content and its presentation will be made dynamically during execution and hence resources (for content and presentation) are also referenced rather than statically bound to the application.

- As knowledge and content are, as distributed Web-based resources, likely to be in heterogeneous formats and acquired by different means (e.g. as documents or as queries on a database), the system needs access to functionality which can handle dynamically tasks of resource retrieval and adaptation. As this is not a finite set which can be contained within the application, this also requires the search for distributed functionality over the Web i.e. Web services.

- In common with all multimedia presentation systems, the system also finally needs an internal means of modelling the intended multimedia presentation, firstly in an abstract and flexible form, and then after constraint checking and conflict solving producing the presentation in a format for delivery to the user.

- The application itself contains the functionality for taking an input and producing from it a multimedia presentation, in this case through interaction with the knowledge available to it. This functionality shall be expressed generically as the explicit knowledge declared in the knowledge base of the system shall be the determinant of the process result.

- The IMMPS proposal calls the input to the system the “presentation goal”. In this case we expect that the presentation goal is represented in a way that can be modelled by the high level formulism applicable to semantic knowledge models, in keeping with the intention that SWeMPs is closely integrated with the Semantic Web.

- The functional implementation of the application will itself be based on logic and semantics, so that interaction with external knowledge (including the application knowledge base) will be possible without requiring any internal adaptation to different levels of abstraction (through which semantics could be lost or be omitted).

- That functional implementation will be based upon the process layers proposed for IMMPS.

- As the application will work with knowledge rather than syntax or low-level media, its interactions will take advantage of the ability to reason about knowledge and infer new knowledge from it. For example, in the family tree scenario knowledge about genealogical relationships is necessary and some of these relations may not be expressed explicitly in the knowledge yet can be inferred from it (such as two people being siblings because they share the same parents).

- As the application will also work with Web services to provide modularity and additional functionality specific to individual executions, its interactions will take advantage of the semantics available with Web services to guide its selection and use of available Web services.

- Finally while final multimedia models are necessarily lower level or syntactic, the application will take advantage of higher level semantic representations (e.g. knowledge about the media which have been included in the multimedia model, and about the concepts which that media is to communicate to the user) to guide the final presentation. For example, the organization of the tourism programme and associated additional media (text, images) in the interactive television scenario will be based on the sharing of common concepts.

4.3.2 The SWeMPs conceptual framework

Based on the above requirements, an initial conceptual framework of the architecture can be envisaged. We use the UML component diagram notation above to represent this architecture (Figure 4.3). As well as some UIs providing the initial presentation goal and displaying the presentation result[32], the framework contains these components:

- Rulebase
- Query interpreter
- Reasoner
- Service planner
- Multimedia modeller
- Presentation formatter

The individual components of the framework and the process which operates within it are described in fuller detail in the next sections.

[32] The UI may be, but is not necessarily, the same application.

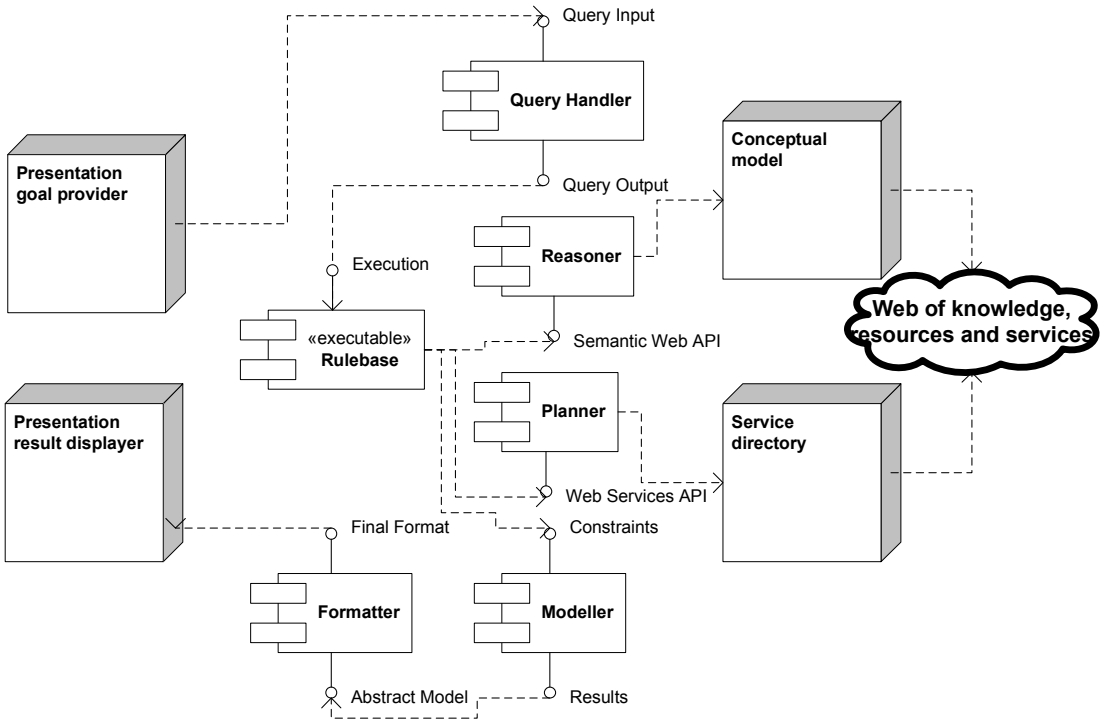

Figure 4.3 SWeMPs conceptual framework as UML component diagram

4.3.3 SWEMPS components

In the conceptual framework it can be seen that there are six components which exist as part of the system. The rulebase is the central execution component and will be detailed separately in the next subsection.

Of the remaining five components:

- The **query handler** receives the information request from the presentation goal provider. It is the purpose of this component to convert the information request from the form it is expressed in by the executor to a logic-based formulism that can be interpreted by the application's internal logic.
- The presentation **formatter** responds to the presentation result displayer with the final multimedia presentation generated by the application. It is the purpose of this component to convert the abstract multimedia model produced in the multimedia generation process to a final format multimedia scene that can be displayed by a target device.
- The **reasoner** interacts with the conceptual model through queries supporting logic-based knowledge representation (i.e. supporting reasoning upon the knowledge such as subsumption and inference).

- The **planner** interacts with the service directory, providing the functionality to the application to discover appropriate services for the task required by the process, negotiate with the selected service (to be able to interact with it) and to finally execute the service, preparing and sending the request message to the service and receiving the response message, passing it back to the process in an appropriate form. The executed services will often be used to act upon an identified resource, providing some extended functionality not defined explicitly within the application (note: it is deliberate that functionalities are called from external services in order to separate the general functionality of the multimedia generation process from specific functionalities required from individual multimedia generation tasks).
- The **multimedia modeller** maintains the multimedia model, building it and altering it according to the activities of the multimedia generation process. It is charged with converting "conclusions" drawn from knowledge in the multimedia generation process into (e.g. spatial and temporal) constraints in the abstract multimedia model which can then be formatted into a final multimedia presentation.

The data structures used by the components differ that in practice, the conceptual model and service directory will be represented by structures that can be stored separately from the application itself, while the resource directory and multimedia model are stored in memory during the execution of the system, hence they are not shown explicitly in the architectural diagram. We introduce these data structures in some more detail:

- A knowledge structure called the "**conceptual model**". This acts as the knowledge base for the application. Its core is the instantiations of the concepts that have been formally defined for the multimedia generation process. This structure may be local to the application (for efficiency) or a distributed knowledge space upon which the application has a single view (as a means to incorporate transiently additional knowledge during the process). It is extended with knowledge specific to the individual multimedia generation tasks, as instances of domain models included into the knowledge structure (either prior to execution as part of an IIS, or dynamically during execution). In the architecture, the conceptual model is based upon the SWeMPs ontology.
- A syntactic structure called the "**multimedia model**". This acts as the abstract representation of the multimedia presentation generated by the application. It shall be able to model the existence of different media items (which may be distributed across the Web) and their spatio-temporal positioning in a multimedia space. It shall also support specifying media characteristics (e.g. size, duration) and interactivity (either programmed or user-controlled). The model should be abstract enough to be able to flexibly represent the intended multimedia presentation, allowing a final formatting phase to resolve conflicts

between media items and their specification. In the architecture given, the model is stored within the multimedia modeller component.

- A distributed space called the "**resource directory**" which maintains references to the retrievable resources available to the application in building the multimedia model. This space is abstract, i.e. it does not represent an actual physical space. Rather it can be thought of as a conceptualisation of a subset of the Web, containing those resources which are deemed relevant to the current multimedia generation task. It may have an initial definition but is changeable during execution, as other activities of the process identify other resources or excludes existing resources in the space. The resources are defined as any digital representation retrievable by the application, not only media content but also syntactic (XML), semantic (RDF) or simple (e.g. a text string). In the architecture given, the resource directory will be incorporated into the conceptual model.
- A distributed space called the "**service directory**" which maintains references to executable programs available to the application in enable interaction with resources. These programs are expected to be distributed across the Web, though they may also be local (to the application they will appear the same), and are selected and invoked as necessary during the multimedia generation process. This space is also abstract, i.e. it does not represent an actual physical space. Rather it is a conceptualisation of a subset of Web services made available to the application in order to perform the current multimedia generation task. It also is changeable during execution, assuming the application has a means to dynamically discover appropriate services at runtime. It may also be updated by the application, either from its own determinations or by checking some third party source (e.g. when a service is known to be down, or another is currently operating more efficiently). Services are expected to be standards-based Web services, making available a description of their interface for the application to determine the appropriate invocation. In the architecture, the service directory is expected to be a semantics-enabled Web Service registry.

As shown in the architectural diagram and explained in the descriptions, the three "internal" components (reasoner, planner and multimedia modeller) mentioned above function together to realise the multimedia generation process through the use of particular structures in the SWeMPs framework. They isolate the logic of the application from the specifics of the interaction with knowledge structures, Web services and the abstract multimedia model. The remaining two components (query handler and formatter) also function as interfaces in the application, but in this case they interface with the executor of the application (i.e. the application that is making an information request and seeking a multimedia response).

4.3.4 SWeMPs rulebase

Within the proposed framework the remaining aspect of SWeMPs which has not been detailed is the rulebase, i.e. the internal application logic realising the execution of the multimedia generation process. The actual implementation of application logic will be considered in the next chapter. At this stage we can specify an abstract (i.e. not tied to any type of implementation) representation of the multimedia generation process as a basis for the later implementation. This process is based upon the typical form of multimedia generation specified in earlier multimedia presentation systems (and particularly the layers of multimedia generation specified in the reference model for an IMMPS). It is however updated for the aims of SWeMPs, i.e. to support and interoperate with the Semantic Web.

The Reference Model for the IMMPS specified five layers for the multimedia generation process. It also notes that these layers are not necessarily linear, i.e. backtracking may be employed where one linear execution results in some presentation conflict. The layers are:

- The *control* layer - organizes and filters presentation goals
- The *content* layer - selects appropriate content, maps it to appropriate media and chooses the appropriate order of communication
- The *design* layer - transforms the internal media decisions to specifications of media objects and their arrangement in the presentation
- The *realisation* layer - realizes the media and layout design in concrete terms
- The *presentation* layer - renders the concrete presentation in a form perceivable by the user

In an earlier paper [Nixon,2003], we have specified a similar set of phases for the production of a multimedia presentation (Figure 4.4):

- A semantics phase, selecting the relevant concepts
- A resource phase, relating resources to concepts
- A transformation phase, adapting resources for presentation
- An integration phase, integrating resources into a template
- A presentation phase, returning a final format presentation

The key difference in these approaches to that intended by SWeMPs is that in these every layer may access knowledge as a means to decision-making in what is otherwise essentially a lower level, syntactic process. For example, [Nixon,2003] conceptualised the multimedia generation process as a set of XSLT transformations on XML data extracted from Web content, accessing knowledge in the form of Topic Maps (using its XML serialization format, XTM). The SWeMPs process can be seen as a refinement of this work, taking into account the use of semantic models (RDF/OWL) rather than syntactic (XML).

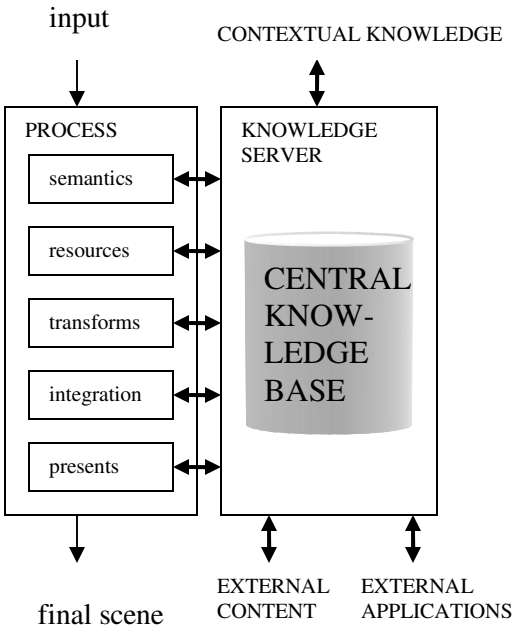

Figure 4.4 Multimedia generation process according to [Nixon,2003]

Following the conceptual framework for SWeMPs, and noting the five-fold division in the components used by the application in realising the multimedia generation process, we maintain the policy of dividing the process into five functionally interdependent activities, and define them in terms of the component that they will use to realise their activity:

- **Activity 1**, using the query handler: read in the information request, set up any necessary preconditions for the execution, break down the request into knowledge requirements for the individual multimedia generation task;
- **Activity 2**, using the reasoner: iteratively resolve the knowledge requirements by making requests upon the conceptual model through the reasoner. The reasoner will handle inferences upon the available knowledge in order to come to required conclusions. To support the inference, the activity may also make requests for acquiring new knowledge from identified resources.
- **Activity 3,** using the service planner: iteratively determine media representations for the knowledge conclusions derived from the previous activity. This determination is made by using the functionality of available services to resolve knowledge deadlocks and to derive content and its presentation from that knowledge. In other words, the previous activity may make functionality requests to this activity, passing control back to the previous activity when the functionality is found and successfully executed. For example, as knowledge deadlocks we can consider knowledge gaps (leading to a functional request for knowledge acquisition) or ontology mismatches (leading to a functional request for ontology matching). Once the previous activity is finished, the functional requests switch to handling the transition from knowledge to content. This means resource acquisition (finding media representing a concept or

relationship between concepts) and adaptation (resolving media characteristics to fit aspects of the information request).

- **Activity 4**, using the multimedia modeller: iteratively model the multimedia presentation by inserting the media found and adapted by the previous activity and determining the appropriate constraints that shall hold for the media items within the model. These constraints (e.g. spatially position media closer together which represent more closely related concepts) are based upon the semantics of the inserted media items and the semantic relationships between the concepts being represented by those media items.
- **Activity 5**, using the formatter: when all prior activities are complete, take the resulting multimedia model and format it into a final multimedia representation.

The process will now be defined more formally in the next section in the form of a rule-set which defines the operation of the application in terms of a generic logical representation of the multimedia presentation generation process. The rulebase is the component which implements this rule-set. The specifics of the interaction of the individual components with the data structures of the framework are detailed then in the section after that. As a result, we begin to make the abstract framework more concrete in the next sections towards a system architecture that can begin to be implemented (the implementation is covered in the following chapter).

4.3.5 Rules for the multimedia generation

We have determined that the application shall be logic-based so that it operates at a conceptual level and can interact with the conceptual model to realise the multimedia generation process whilst avoiding any semantic loss or shortfall as a result of mapping from higher level to lower level representations or vice versa. While the actual implementation of the application will be done later, at this stage we need to express in a logical form the multimedia generation process as an implementation basis for the concrete application. As an initial high level abstraction we use an UML activity diagram (Figure 4.5) to illustrate the multimedia presentation generation process.

Rules-based systems [Hayes-Roth,1985], as illustrated in Figure 4.6, have as a typical architecture:

- Rule set
- Working Memory (the fact base)
- Inference Engine (the rule engine)

The rule set can be considered the ‘program’ or ‘procedural knowledge’ of the rules system and the working memory the ‘data’ or ‘declarative knowledge’.

Rules are triggered through a matching operation between facts asserted in the working memory and the content of rules – simplistically, given the fact *a* and the rule $a \rightarrow b$ the rules system determines there exists a match and infers the fact *b*. In a Closed World scenario[34], the system may also need to resolve conflicts that could arise, e.g. if there already exists the fact $\neg b$ (the negation of *b*).

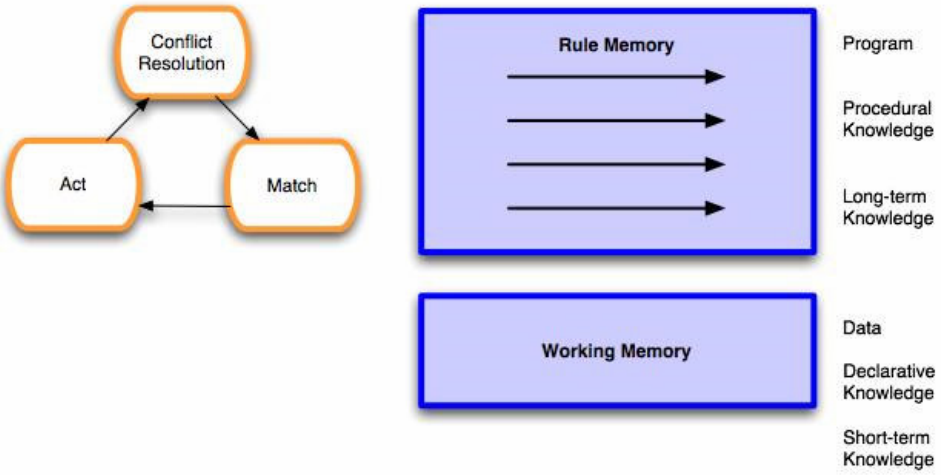

Figure 4.6 General form of a rules-based system[35]

We consider a rules-based system approach relevant as we conceptualise the conditions in which a multimedia presentation generation process occurs as a set of facts (which apply to an individual task of multimedia presentation generation) and model the generation of a multimedia presentation in terms of solving a stated goal with respect to that set of facts. In rules-based system terminology this is considered a *forward-chaining system* in which a stated goal is considered as the "working memory". The multimedia generation process, represented at this stage as an UML activity diagram, could be mapped into a rulebase, allowing the application code to be modelled as rules, supporting the decoding, modification and easier development of the SWeMPs implementation.

We detail at this stage the rules of the process in natural language, though it is already possible to make reference to the other components which were defined to control system interaction with the various data structures (section 4.3.3) and these interactions are considered in more detail in the next section. A more formal specification of the rules is given in section 5.6, where we can draw upon the concepts and relations defined in the SWeMPs conceptual model.

[34] Issues concerning Open and Closed World reasoning will be discussed in section 5.2
[35] Diagram from http://www.igda.org/ai/report-2003/aiisc_rule_based_systems_report_2003.html
[38] Communicative abstractions are mentioned in the linguistic literature, e.g. [Melnikov,1988].

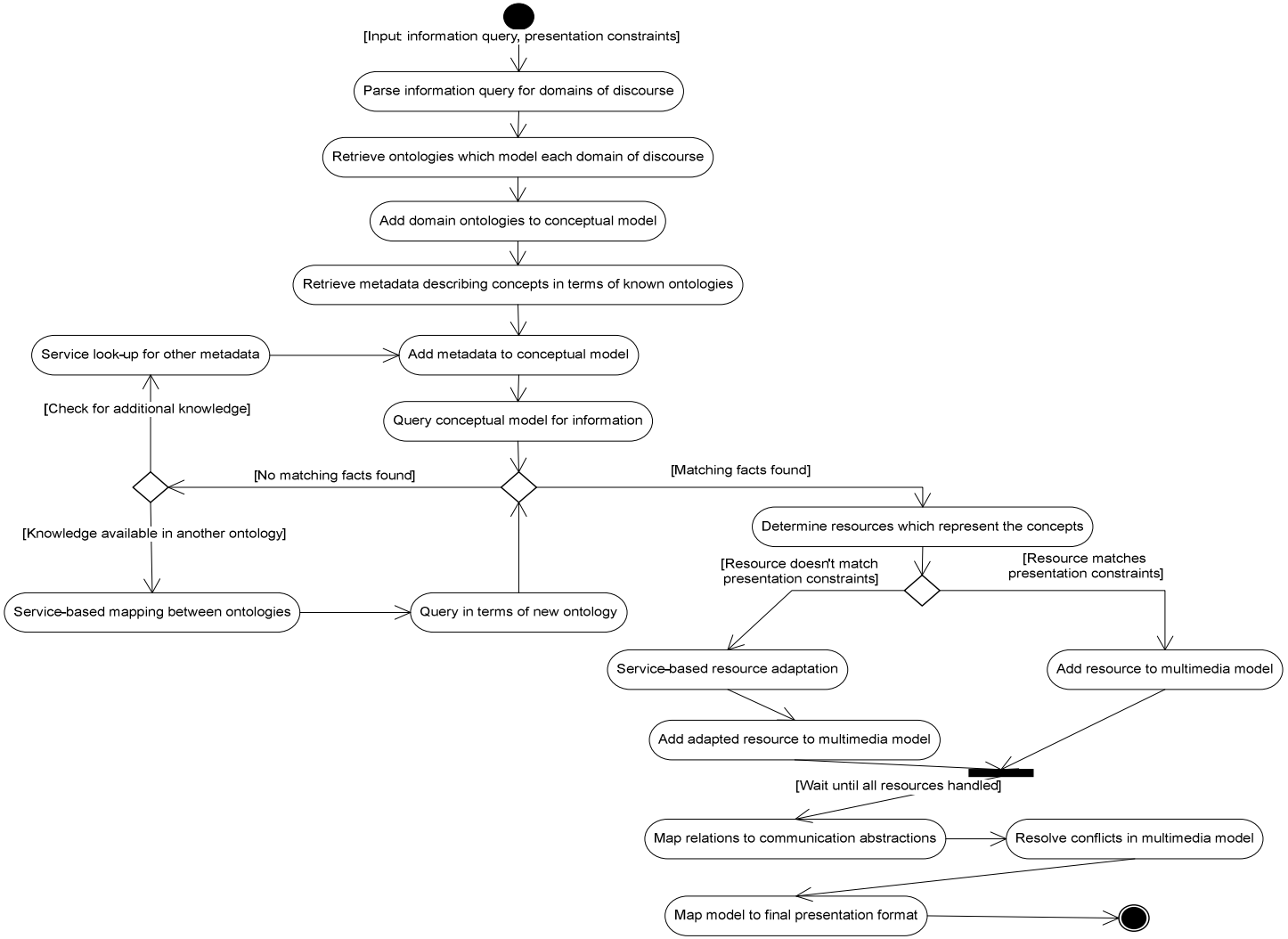

Figure 4.5 Multimedia generation process as UML activity diagram

Firstly, the process receives as input an information request and some execution-specific presentation constraints (e.g. details of the user and device and the context of usage). The query handler is then called to interpret the information request into semantic queries (i.e. the handler acts as an interface to the entity requesting the information so that the format of its request can be independent from and mapped to the logical representation used internally by SWeMPs).

It then sets up the conceptual model for the actual process by loading the necessary ontologies (SWeMPs plus domain-specific ontologies) and knowledge bases i.e. ontology instantiations (of the SWeMPs ontology, defining the individual task of this execution, and the domain-specific knowledge relevant for that task). The choice of ontologies and knowledge bases is determined from the information request, e.g. acquire the ontologies which define the domain of a concept in the information request and acquire the metadata which describes the concept itself.

For each semantic query generated by the query handler, the query is executed upon the conceptual model using the reasoner component (so that logical reasoning may be supported). The result of the query will be passed back to the rulebase in the form of a set of matching facts (statements fulfilling the query). However this procedure is seen as too simplistic to ensure good results on the Semantic Web. Rather, two potential problems are identified that must also be handled:

- That the conceptual model does not contain sufficient knowledge to answer the query
- That the conceptual model's representation of the knowledge does not match the representation used in the query

In the case that there is insufficient knowledge available to the application, tested by checking if the set of knowledge results from the query is null, the planning component is charged with executing a service to acquire additional knowledge that may answer the query, integrating it into the current conceptual model of the process. The query can then be repeated, and as long as it returns null, new knowledge introduced into the knowledge base until the service planner reports that no new knowledge can be extracted by the available services.

In the other case that the knowledge representations do not match, tested by comparing the ontologies being used to define concepts in the conceptual model with the ontologies being used to define concepts in the query (which would be expected to be a subset), the planning component is charged with executing a service to resolve this mismatch by determining possible mappings between the concepts in the query and concepts defined in ontologies existing in or referenced from the conceptual model. Note that this case is not expected to occur in the first iteration of semantic querying as all the ontologies used in the query should be imported into the conceptual model (provided they are available, i.e. were referenced from the conceptual model or acquired through a service).

However in successive iterations, after new knowledge is dynamically acquired, the possibility occurs that the new knowledge may introduce concepts from ontologies not known to the conceptual model. Without this resolving of the mismatch it is not possible to know if the acquired knowledge answers a given query or not. Also note that we choose not to automatically import ontologies when acquiring knowledge as this could be very resource-intensive and inefficient when it is determinable that there is no semantic relationship between a potentially large ontology and the semantics of the knowledge queries.

Reasoning on the knowledge in the model is carried out iteratively until all queries have either been answered through the available knowledge or no further knowledge is available to the system which may be able to answer the query (at this stage, the system gives up).

The application is now working with a set of knowledge results, i.e. statements that answer the request for information which was made to the application. The reasoner acquires through queries metadata identifying resources which provide representations of the concepts which answer the information request (avoiding duplicate queries for the same concept). Each conceptualisation of a resource is passed to the multimedia modeller which is able to make decisions about the presentation of the actual resource through querying the resource metadata and testing if it matches the actual presentation constraints. The relationships between concepts represented by selected resources are also mapped to presentation constraints which are passed to the multimedia modeller, again checking to avoid duplication.

When a resource is to be presented in the abstract multimedia model, an adaptation stage may be necessary to conform to the presentation constraints, which consist both of the ‘static’ constraints originally provided to the process at execution and the ‘dynamic’ constraints arising from the current abstract multimedia model (e.g. the resources already being presented). This is handled as a semantic query where the knowledge base to be queried is the appropriate resource metadata and the test to be made is against the currently valid presentation constraints, and a match is defined as a case where each query returns true. Just as with a knowledge query, knowledge acquisition may be done where the resource metadata is insufficient to determine if a constraint is met, and concept mapping is done where the ontology used by the constraints does not match the ontology used by the resource metadata. Where there is not a match, an attempt is made to adapt the resource to meet the constraints. This adaptation can be divided into two types: simple, defined as an adaptation that can be defined directly in the multimedia model, or complex, defined as an adaptation that requires some manipulation upon the resource representation generating a new representation that can be referenced from the multimedia model. An example of the former would be resizing (as size can be explicitly specified in the multimedia model, and scaling done by the client), and the latter would be format conversion. In latter cases the planning component is charged

with executing a service which adapts the given resource (assuming there are no problems with regard to digital media rights in directly manipulating a resource and saving an altered version). The reference from the multimedia model is then updated to the new reference of the altered version.

Retrieving resources and adding them to the multimedia model continues iteratively until all knowledge results have been processed.

Besides the resources, the semantic relationships between the concepts they represent are mapped into presentation constraints upon the presentation of the resources in the model, not on the basis of factors 'outside' of the model (such as the screen size) but in terms of their display in relation to other resources (e.g. grouping resources that deal with the same concept together). The modeller is called upon to determine a final abstract multimedia model, resolving the constraints that have been collected. Once the multimedia model is complete, it is transformed to an end format for display (using the formatter component) and is output.

4.3.6 Proposed model of interaction

In the SWeMPs architecture, various aspects of the multimedia generation process are handled by other components which act as interfaces to data structures (so that if a data structure changes or we wish to alter how we interface with it we only need to change the component, and do not need to alter the core application code i.e. the rule-set). Here we detail the interactions of these components between the application and the data structures which they interface. This will be further formalized in the following chapter, particularly in section 5.7.

The first call is to the **query handler**. This is made as soon as an input information query is received. It is expected that the query handler is able to:

- Know the format of the incoming information query, either predefined or by analysis.
- Transform from the format of the incoming information query to a format which supports description logic-based reasoning and that can be parsed and interpreted by the reasoner component.

The mapping from information query to knowledge query is 1:n, that is, we expect that the information query can be expressive enough to allow for simple expressions which are mapped into more complex sets of knowledge queries. This supports services which allow specialised requests for information without placing the complexity of the query modelling on the side of the client (i.e. the requesting application). No specific information query format is specified by SWeMPs, only that the query handler will be able to resolve any queries in that format to knowledge queries in a reasoner-supported format.

The **reasoner** executes the queries over the conceptual model. It will be a specialised component which contains the functionality for reasoning with the knowledge representation used by the conceptual model. It will require an API for access from the application, allowing the passing to it of the queries, and once the queries have been resolved returning the results to the application. Threading will be required as each query-result pair represents a separate sub-process. Additionally the reasoner can be called from the service planner or the multimedia modeller when knowledge needs to be checked from the conceptual model. There will need to be a defined means of enabling this, e.g. the other components pass a particular message to the application which, according to a rule in the application logic, fires a particular query to the reasoner. For example, once the set of knowledge results is complete, a rule shall request the reasoner to query the conceptual model for resources which represent the concepts given in those results.

The service planner handles the execution of external services for additional functionality required by the application. Services will have defined input and output data structures (specified in their service descriptions). The service planner will receive four parameters when executed from the application: the type of service required in terms of the type of input the service takes and the type of output it returns, input data from the application and the data structure for the output data that is returned to the application. The planner will then find and execute the requested service, possibly having negotiated to use it (in complex cases this could involve access rights, security, or payment to use a service). Primarily service negotiation will involve examining the input data structure requested by the service, the input data structure supplied by the application and resolving differences (which, suitably, might be done using another service). It will also wait for and receive the response of the service, and again examine the received output data structure in comparison with the requested data structure for the output data from the application and resolve the differences, so that the service response is returned to the application in the desired form.

The service planner will also be required to handle service errors, including time-outs and retries. Hence the planner will be called upon to carry out “planning” tasks to co-ordinate the execution of and the waiting for responses from services, including deciding when to cancel requests, repeat requests or make requests to alternative services. The need to find alternative services offering the same functionality or to build required functionalities through the combination of services suggests the use of a Web Services framework supporting discovery and composition. By extending this to Semantic Web Services, it could also be viable to replace the parameter specifying a desired service in terms of syntactic input/output with a parameter specifying the required functionality in a semantic form (i.e. carrying out semantic discovery of services) and to extend syntactic specifications of input and output with semantic specifications of input, output, precondition and effect (where precondition and effect define the state of the world before and after execution of the service). The process shown in Figure 4.5

contains three specific cases where services may be called to offer needed functionality to the application:

- To find knowledge relating to a given concept. Here, a service acts as an efficient entry point to a metadata repository. Typically the planner will pass a concept to the service and it will return a finite set of metadata which provide information related to that concept.
- To match concepts defined in different ontologies. In this case we expect a service to accept as input a particular concept and an identification of the set of ontologies which define the concepts to which this concept may be matched. It can be expected that this set of ontologies is, at the very most, the set of ontologies referenced from the conceptual model (as it makes no sense to match an unknown concept with another concept that is equally unknown to the application). It returns statements defining mappings between the concept and concepts from the other ontologies. If the service returns null, it means no relationship can be found between the concept and the ontologies passed to the service.
- To adapt resources. Services for resource adaptation can fulfil a much wider range of functionalities, which can be defined in terms of the vocabulary of the presentation constraints. In the simplest case, a single (multi-functional) service could take as input the reference to the resource to be adapted, and a statement defining the mismatch between the resource metadata and the presentation constraints, and returns a reference to an adapted version of the resource which meets those constraints.

The final component which abstracts some interaction between the application and data structures according to the SWeMPs architecture is the **multimedia modeller**. The modeller receives as input from the application a conceptualisation of a resource (i.e. a reference to the concept of a resource in the conceptual model). The modeller uses certain information about the resource so that it can make decisions about the presentation of the resource in the multimedia model. This includes the media type and its characteristics (taken from its metadata) so that the correct type of media object is created in the multimedia model and that its characteristics best match those of the resource being presented through the media object. Additionally, on the application side, a matching procedure against the resource characteristics and the presentation constraints is used to determine necessary adaptations of the resource which may require accessing a relevant service (leading the resource reference in the modeller to be updated to that of an adapted version).

The modeller can also query for relationships in the conceptual model which exist between the concept represented by the resource and other concepts represented by existing resources in the multimedia model. The modeller will check rules that exist internally to it to determine a mapping from these properties to *communicative abstractions*[38] which will be applied between the resources in the model representing the related concepts. At the highest level

these rules can test abstractly the “closeness” of properties (e.g. sub-classing, properties of properties) though it is expected that the rules in the multimedia modeller will be extended by domain-specific rules for individual information services. Communicative abstractions will commonly constrain the spatio-temporal relationships between media objects. Additionally properties on or between concepts could be mapped to action in and interaction between media objects.

The modeller builds iteratively a multimedia model through the insertion of media objects and the specification of communicative abstractions between them. At each iterative step a check is made for any conflicts that occur and conflict resolution is carried out. When the application is finished with processing the knowledge results the modeller checks that all parameters are respected, resolves any remaining conflicts and passes to the formatter the reference to the abstract multimedia model.

The multimedia modeller is a very important component in the SWeMPs framework as it contains the functionality to produce the multimedia presentation which communicates to the requester their information need. The internal model for a multimedia presentation must be suitably abstract and flexible to describe the intended final presentation without tying media objects and their relationships to concrete representations (the mapping from abstract to concrete takes place in the formatter) as the complicated mixture of media object parameters, presentation parameters and constraints on the media objects and the presentation (both implicit to multimedia and explicit to this presentation) will force tradeoffs, backtracking and the search for a “best case” solution to parameter and constraint conflicts.

Mappings between resource metadata (potentially in different formats) and the presentation constraints to the format used internally by the modeller to represent display parameters also need to be defined. The modeller must also be able to handle the knowledge-based rules for deriving communicative abstractions from properties in the conceptual model. The communicative abstractions themselves are internally defined in the modeller.

The final call is to the presentation **formatter**. It receives a reference to the multimedia model being stored by the multimedia modeller on behalf of the application. It transforms this model into a concrete multimedia presentation using an end device format (the necessary format is specified in the presentation constraints, though it may be the case that a format is fixed by the application or determined by some other means).

4.4 Formalising the Conceptual Model

We build a conceptual model using the CLASSIC DL-based grammar and the knowledge engineering methodology of [Brachman,1991]. This model will be refined during the implementation of the SWeMPs application (see Chapter 5.8). An iterative process to produce a final version of a conceptual model is acknowledged as necessary among the KR community e.g. [Noy,2001].

As a first step, it is recommended to list all types of objects that would be "talked about" in the intended universe of discourse. Our domain is "the modelling of the multimedia generation process and the components that participate in it". We have already outlined a conceptual framework for this process which foresees data structures for concepts, resources and services. Considering this framework we make a first-attempt list of the objects in our model:

```
multimedia process
semantic object
semantic object domain
media representation
resource
resource metadata
URL
media type
resource type
service
service description
```

Then we seek to separate the concepts (which can exist as an independent thing) from the roles (which are implicitly dependent on something else). In the roles, it can be noted if we are looking at an attribute (a role with a single filler). To distinguish concepts and roles, we capitalize the concepts and keep roles as lower case.

CONCEPTS

```
MULTIMEDIA PROCESS, SEMANTIC OBJECT, RESOURCE, SERVICE
```

ROLES

```
semantic object domain (attr), semantic object
description (attr), media representation (attr),
resource description (attr), URL, media type (attr),
resource type (attr), service description (attr)
```

The next step is to produce a concept taxonomy. At first, this seems to be trivial as we have four concepts which seem entirely separate. However when we consider the roles that we have, they must have defined concepts to constrain their permitted fillers. These can be modelled as new concepts and we can consider some of those concepts to be subclasses of our (top level) concepts:

```
semantic object domain: ONTOLOGY as sub-class of RESOURCE
semantic object description: SEMANTIC OBJECT METADATA as
sub-class of RESOURCE
media representation: MEDIA TYPE
resource description: RESOURCE METADATA as sub-class of
RESOURCE
URL: URL (datatype)
media type: MIME TYPE
resource type: XML NAMESPACE
service description: SERVICE METADATA as sub-class of
RESOURCE
```

Given the three metadata concepts it seems reasonable to group them as sub-classes of a METADATA class, disjoint to ONTOLOGY. We note that URL is modelled as a literal value (an instance of a datatype). Additionally as the multimedia presentation system (whose individual executions are to be represented by this conceptual model) is intended to realize individual intelligent information services, it follows that in our model MULTIMEDIA PROCESS would be modelled as an instance of SERVICE.

This results in the following concept taxonomy (where → indicates a superclass-subclass relationship and *is-a* indicates a class-instance relationship):

```
SEMANTIC OBJECT;
RESOURCE → ONTOLOGY, METADATA;
METADATA → RESOURCE METADATA, SERVICE METADATA, SEMANTIC
OBJECT METADATA;
SERVICE;
MULTIMEDIA PROCESS is-a SERVICE;
MEDIA TYPE;
URL;
XML NAMESPACE;
MIME TYPE.
```

URL is modelled as an instance of a datatype, which will be dependant upon the support for datatypes within the KR model.

In terms of individuals of interest in all states of the world in this universe of discourse, we can note that semantic objects, resources and services are expected to vary across applications but that some media types, MIME types and

XML Namespaces are expected to be constant (as they are fundamental parts of this universe of discourse). We could specify such individuals in the model, e.g.:

```
Media Type: Application, Audio, Video, Text, Image, 3D
Model

MIME Type:  application/rdf+xml, application/owl+xml,
application/xml, audio/mpeg, image/gif, image/jpeg,
image/svg, model/vrml, text/html, text/plain, text/rtf,
video/mpeg

XML Namespaces: RDF, RDFS, OWL, OWL-S
```

From this it can be seen that MIME Types can be modelled as instances of subclasses of Media Type (Audio Type, Video Type and so on). It must also be noted that the choice of MIME Types and XML Namespaces makes some assumptions about the type of content that will be handled by the service, however given the intention to be Internet-based (and hence using common Internet resource types) and Semantic Web-enabled (and hence using the W3C Semantic Web standards) this choice seems fair.

The next step in the knowledge engineering is to determine the properties and parts in the model. For each concept, we list its properties – intrinsic (specific to that concept), extrinsic (can occur on other concepts) or parts (structural qualities). In making this list, we consider also the sort of relationships that can exist between the modelled concepts: that semantic objects exist in the model, that resources are intended to represent semantic objects, metadata describes other concepts whether a semantic object, resource or service, and that services are intended to handle concepts in predefined ways.

Properties are categorized as (i) – intrinsic property, (e) – extrinsic property or (p) – part-of property.

```
SEMANTIC OBJECT: exists-in-domain (i), has-url (e)
RESOURCE: represents (i), is-of-type (e), has-metadata (e),
has-url (e), has-namespace (i)
        -> METADATA: references (i)
        -> ONTOLOGY
SERVICE: handles-media-type (i), handles-namespace (i), is-
of-type (e), has-metadata (e), has-url (e)
MEDIA TYPE: has-url (e)
XML NAMESPACE: has-url (e)
```

For each property we now allocate the restrictions that are in effect in terms of cardinality and value.

Property	Cardinality	Value
exists-in-domain	>=1	ONTOLOGY
has-url	>=1	*anyURI (XML Schema datatype)*
represents	>=0	SEMANTIC OBJECT
is-of-type	1	MEDIA TYPE
has-metadata	0 or 1	METADATA
has-namespace	>=0	XML NAMESPACE
references	>=1	SEMANTIC OBJECT
handles-media-type	>=0	RESOURCE
handles-namespace	>=0	XML NAMESPACE

We can see that some significant design decisions have been made at this stage. These are:

- `RESOURCE`s *could* have a `XML NAMESPACE`. That would affect XML resources as well as RDF-based (`METADATA`) and OWL-based (`ONTOLOGY`), but not non-XML resources such as media objects, databases or textual documents.
- `SERVICE`s *could* specifically handle a media type *or* a namespace. In other words, services can be seen as generic to all instances of a (XML, RDF, OWL) namespace or specific to any media type, whether it has a namespace or not.
- `RESOURCE`s and `SERVICE`s *could* have an associated metadata resource but it is not required.
- `RESOURCEs` *must* have a single identified type.
- `MEDIA TYPE`s are uniquely identified by an URL (e.g. using one of the pre-defined MIME type instances or a shared definition of a new type).
- URL individuals are equated with URL strings which identify the concept whose role is filled by this URL individual. Note that this is not unique identification: the same URL string can be used to identify a semantic object, resource and service. The chosen datatype should be able, if possible, to specifically represent an URL string, e.g. the *anyURI* type in XML Schema.
- All other properties *must* exist at least once in each instance of a concept.

It is also important to note at this stage that not all of the model should need to be instantiated manually by a service developer, i.e. one aim of the model is that it is possible to implicitly determine instances of concepts and relationships where possible from available resources and services. For example, semantic Web service descriptions may be sufficient for the application to determine the applicability of the described service for a particular media type or namespace. As a CLASSIC type KR system automatically determines membership of individuals in certain concepts based on essential properties, we can state some

rules C -> Y where an individual is a member of concept C when the statement Y is true, e.g.[40]

```
METADATA -> RESOURCE is-of-type application/rdf+xml
ONTOLOGY -> RESOURCE is-of-type application/owl+xml
```

We see this functionality as being more relevant in an individual application where the ontology is being extended by domain-specific classes and instances and then rules are expressed to determine automated membership of instances in the given classes.

Finally, as CLASSIC type KR systems distinguish between primitive and defined concepts (primitive concepts do not express sufficient conditions for membership while defined concepts do), we consider our modelled concepts and conclude only METADATA and ONTOLOGY express sufficient conditions for membership, i.e. are defined concepts.

4.4.1 Conceptual model in the CLASSIC grammar

Thus we produce an initial conceptual model for multimedia presentation generation, based on the conceptual framework of SWeMPs. We took the knowledge engineering methodology of [Brachman,1991] and hence have used the CLASSIC DL-based grammar that was used in that methodology. The resulting ontology can be expressed so in the CLASSIC grammar:

```
Semantic object (PRIMITIVE
                    (AND (ALL exists-in-domain Ontology)
                         (ALL has-url URL)
                         (AT-LEAST 1 exists-in-domain)
                         (AT-LEAST 1 has-url)))

Resource (PRIMITIVE (AND (ALL references Semantic Object)
                    (ALL is-of-type Media Type)
                   (ALL has-metadata ResourceMetadata)
                   (ALL has-url URL)
                   (ALL has-namespace XML Namespace)
                   (AT-LEAST 1 is-of-type)
                   (AT-MOST 1 is-of-type)
                   (AT-LEAST 0 has-metadata)
                   (AT-MOST 1 has-metadata)
                   (AT-LEAST 1 has-url)))
```

[40] These axioms are however not part of the final SWeMPs conceptual model (section 5.8), as we decide to not insist on RDF/XML and OWL/XML syntax for metadata and ontologies. The application would however have to be able to handle any syntax that is retrieved from the given URL of a metadata or ontology instance.

```
Metadata (AND Resource)

SemanticObjectMetadata (AND Metadata
                          (AT-LEAST 1 references))

ResourceMetadata (AND Metadata)

ServiceMetadata (AND Metadata)

Ontology (AND Resource)

Service (PRIMITIVE (AND (ALL handles-media-type Media Type)
                    (ALL handles-namespace XML Namespace)
                    (ALL is-of-type Media Type)
                    (ALL has-metadata ServiceMetadata)
                    (ALL has-url URL)
                    (AT-LEAST 1 is-of-type)
                    (AT-MOST 1 is-of-type)
                    (AT-LEAST 0 has-metadata)
                    (AT-MOST 1 has-metadata)
                    (AT-LEAST 1 has-url)))

Media Type  (PRIMITIVE (ALL has-url URL)
                    (AT-LEAST 1 has-url))

XML Namespace (PRIMITIVE (ALL has-url URL)
                    (AT-LEAST 1 has-url))

URL (PRIMITIVE (AND URIstring*))
```

* where *URIstring* is a host datatype of type string which is recognised as specifying a Uniform Resource Identifier (URI)

4.4.2 Conceptual Model in a Description Logic Representation

We also express this model using a formal syntax, since the grammar used in the previous section lacks any formal meaning and is specific to the CLASSIC KR system. Furthermore, CLASSIC is derived from the KL-ONE system, and it has been demonstrated that subsumption in this system is undecidable [Schmidt-Schauss,1989]. For a Semantic Web based system, decidability is a necessity; hence the formalisation of the model needs to be re-expressed in a decidable subset of first-order logics, such as Description Logics. By using a commonly agreed and understood DL formulism, we can ensure that any encoding of the model in a DL system can be formally validated as well as be better able to support any reformulating of the model at a different level of expressivity. Both cases will be seen in the next chapter (section 5.8).

We use the Description Logic syntax of **SHOIQ(D)** [Horrocks,1999] as the Semantic Web language OWL is based on this "flavour" of Description Logics.

$Semantic\ Object \equiv T \wedge \exists exists\text{-}in\text{-}domain.Ontology \wedge \exists has\text{-}url.URL$

$Resource \equiv T \wedge \forall represents.Semantic\ Object \wedge =1\ is\text{-}of\text{-}type.Media\ Type \wedge <=1\ has\text{-}metadata.ResourceMetadata \wedge \exists has\text{-}url.URL$
$\wedge\ has\text{-}namespace.XMLNamespace$

$Metadata \subseteq Resource$

$SemanticObjectMetadata \subseteq Metadata \wedge \exists represents.SemanticObject$

$ResourceMetadata \subseteq Metadata$
$ServiceMetadata \subseteq Metadata$

$Ontology \subseteq Resource$

$Service \equiv T \wedge handles\text{-}media\text{-}type.Media\ Type \wedge$
$handles\text{-}namespace.XMLNamespace \wedge =1\ is\text{-}of\text{-}type.Media\ Type \wedge$
$<=1\ has\text{-}metadata.ServiceMetadata \wedge \exists has\text{-}url.URL$

$MediaType \equiv T \wedge \exists has\text{-}url.URL$

$XMLNamespace \equiv T \wedge \exists has\text{-}url.URL$

$URL \equiv u$[41]

4.5 Conclusion

In order to build a Semantic Web-enabled Multimedia Presentation System a key design decision has been to determine a framework for its implementation, based on a set of components and a generic rules-based process, and develop a conceptual model of the multimedia generation process. This conceptual model forms an ontological basis for expressing knowledge about the desired multimedia generation process that is needed for individual "intelligent information services" (IIS). In order to demonstrate the model's validity and to ensure its interoperability with the knowledge that will be acquirable through the Semantic Web, we have formulised this model in Description Logic. The next step will be to demonstrate realising the functionality of multimedia generation on the basis of this conceptual framework and model, as will be outlined in Chapter 5.

[41] u in the DL syntax represents the XML Schema datatype *URIref*