Towards a Reference Architecture for Open Hypermedia

Kaj Grønbæk
Computer Science Department
Aarhus University
kgronbak@daimi.aau.dk

Uffe Kock Wiil
The Danish National Centre for IT Research
Aarhus University
ukwiil@cit.dk

1. Introduction

The OHS Working Group formed out of the 2nd Workshop on Open Hypermedia Systems at Hypertext '96 in Washington, DC (March 1996). The initial goals of the OHS Working Group were to define open hypermedia systems and to define open link services. The OHS Working Group meeting in Southampton, England (December 1996) showed that it is difficult to discuss scenarios and especially protocols [Davis et al. 1996] without having a common consensus on what the concepts and building blocks (architectural components) of an open hypermedia system (OHS) are. Thus, a third goal defining an open hypermedia reference architecture was added to the OHS Working Group agenda at the Southampton meeting.

This position paper takes a first step towards a reference architecture for OHSs. The purpose of the paper is to "jump start" the work towards common concepts, architectural components and terminology for OHS. A common consensus on these important issues is essentially a prerequisite for being able to identify (and discuss) the necessary protocols in an OHS.

We approach these issues by questioning the use of existing OHS terminology and by critically analyzing existing open hypermedia reference models and architectures. We start in Section 2 by identifying a number of key issues that OHSs must address and by listing a set of requirements for an open hypermedia reference architecture. Section 3 discusses the use of different terminology in existing open hypermedia models and systems. Section 4 discusses four different open hypermedia reference models and architectures that has been proposed in the literature. Finally, Section 5 provides a preliminary attempt at an open hypermedia reference architecture. The basic idea is to try to take the best of existing approaches and work towards a proposal that can provide the identified services and fulfils the identified requirements.

2. Requirements for an Open Hypermedia Reference Architecture
Before discussion and designing a reference architecture for OHSs, we need to make it clear what it is that we are trying to achieve. Our experiences with OHS design, development, deployment and assessment (e.g., [Grønbæk et al. 1997] and [Wiil and Leggett 1997]) indicate that the following services are important to OHSs and should be considered key issues when designing a reference architecture:

**Integration**
OHSs should be capable of integrating (1) existing services such as scripts, programs and software agents, (2) third party applications such as editors, browsers, email systems, calendars and spreadsheets, and (3) third party stores such as databases, hyperbases, link servers, document management systems, file systems, and CD-ROMS.

**Hypermedia**
OHSs should support a rich set of hypermedia services such as anchors, links, file-wrapper nodes, content nodes, composites, pspecs, logspecs and refspecs.

**Distribution**
The OHS itself and the integrated services, applications and stores must be capable of running at different machines on a local area network and across different Internet domains.

**Collaboration**
OHSs should provide support for collaborative authoring of both hypermedia contents and structure. Different modes of collaboration should be supported including asynchronous work (data sharing among a group of authors engaged in individual authoring sessions) and synchronous work (simultaneous sharing and creation of contents and structure by a group of authors engaged in a single shared authoring session).

A core set of OHS services could probably be identified, but in general an OHS should be capable of providing as many of these services as desirable. Thus, a reference architecture for OHSs should reflect the fact that OHSs provide different levels and types of services within the four areas listed above (and potentially other areas as well such as information retrieval). In addition to the already mentioned requirements for services, we assume the following general requirements for an open hypermedia reference architecture:

**Conceptual Architecture**
A reference architecture should provide a high level of abstraction dealing with concepts, architectural components and protocols rather than describing physical processes and services.

**Generality**
A reference architecture should be general enough to be mapped onto several physical implementations of the concepts and architectural components and comprehensive enough to allow the concepts and architectural components of existing OHSs to be mapped onto the reference architecture.

**Simplicity**
A reference architecture should be simple and easy to understand. There should be few architectural components and few protocols between the components.
Extensibility
A reference architecture should be extensible with respect to concepts, architectural components and protocols. It should be possible to augment the architecture to deal with special cases (e.g., special classes of OHSs) and to include support for new types of services and requirements not anticipated at the time of inception.

3. Discussion of Open Hypermedia Terminology

A reference architecture should help in sorting out a consistent terminology. Currently the OHS community uses a rich variety of different terms for essentially the same component of the architectures.

Content handlers are called: viewers, third party applications, participating applications, editors, applications, tools, ...

The interface between content handlers and hypermedia service is called: shim, wrapper, gateway, proxy, ...

The hypermedia service application is called: link service, runtime, tool integrator, session manager, hypermedia service, ...

The hypermedia database is called: link service, structure server, filter, OODB, HBMS, hyperbase, hyperbase system, workspace, ...

To state a reference architecture for open hypermedia we need to agree on a consistent set of concepts which provide a common understanding of the elements of the architecture. The choice of hypermedia data and process models such as Dexter and others should be orthogonal to the reference architecture. We are not in a position to dictate a decision on the use of terminology, but it is one of the purposes of this paper to put the discussion on the agenda for the OHS Workshop.

4. Discussion of OHS Reference Models and Architectures

Only a few open hypermedia reference models and architectures has been proposed in the literature. This section presents four proposals, the Flag model [Østerbye and Wiil 1996], the layered DHM architecture inspired by the Dexter model [Grønbæk et al. 1994], the Shim architecture [Davis et al. 1996], and the extended HyperDisco model inspired by the interoperability experiment in [Wiil and Whitehead 1997]. Each of these four candidates are examined and important contributions of each proposal are identified. We believe that a reference architecture which fulfils the ideal requirements described in Section 2, should be developed as synthesis among these architectures.

4.1 The Flag Model
The Flag model [Østerbye and Wiil 1996] modifies and extends the terminology of the Dexter model to explicitly deal with integration and use of third party applications. The main idea is to distinguish between storage aspects and runtime aspects and between structure and contents. These distinctions make the Flag model a conceptual reference model which identifies four functional modules and four protocols in OHSs (see Figure 1).

Figure 1. The Flag model (from [Østerbye and Wiil 1996]).

The Flag model makes it very explicit that third party applications (1) can participate in the hypermedia services and (2) can store contents in one place (e.g., the file system) via the storage protocol and store structure (and in some cases contents) in the OHS via the linking protocol. Another merit of the Flag model is that it is completely system independent and provides a general framework to classify, describe and compare (open) hypermedia systems.

4.2 The Layered Architecture inspired by the Dexter Model

In [Grønbæk et al. 1994] an architecture for the Devise Hypermedia framework is presented as a modified and extended Dexter-based layered architecture. Figure 2 depicts an updated version of this architecture which features four conceptual layers: Application Layer, Communication Layer, Runtime Layer, and Storage Layer.

Architectural elements of a suite of DHM integrated applications in a multiuser setting are located relative to the conceptual layers. The architectural elements in Figure 2 suggests a scenario, where two users are connected to multiple Hypermedia Databases and multiple Document Management systems possibly distributed on several hosts in a network.
Figure 2. A layered architecture inspired by the Dexter model.

An advantage of this model is the combination of a diagram for the actual relationships among architectural elements of a specific system configuration and conceptual layers. A different system configuration can be mapped to exactly the same conceptual layers thus illuminating the specific differences in the configuration. While Figure 2 depicts a system configuration with ordinary desktop applications and a document management system typically running on a shared file system, Figure 3 projects another open hypermedia system configuration onto the same conceptual layers in a consistent fashion.

Another advantage is that both an application <-> hypermedia service protocol (OHP) and a hypermedia service <-> hypermedia database protocol is identified.
One disadvantage of the strict layering in this approach is the potential ambiguity in whether it is only the architectural elements seen from the point of view of the hypermedia system which is mapped to the layers or it is also the integrated applications themselves which should be mapped to the layers. This ambiguity comes to the surface when locating the contents of components managed by the applications. Should that be mapped to the (physical) Storage layer or to the Application layer. In fact Figures 2 and 3 make a different choice with this respect. The Document management system in Figure 2 is placed in the Storage layer whereas the WWW servers are placed in the Application layer in Figure 3. The choices being made here are dependent on whether the Hypermedia Service Process communicates directly with the Document Management System/ Web Server or not. This ambiguity could, however be avoided in the Flag model approach as described in Section 4.1, here both Document Management System and Web Server would map into the Content/Storage corner.

4.3 The Shim Architecture

[Davis et al. 1996] propose an architecture that supports reuse of third party application extension modules across different Open Hypermedia Systems. The idea is that specific applications should extended to talk to a standardized protocol called OHP. Having
integrated applications this way, so called shims, which fit the standardized OHP to private protocols implemented by say DHM and Microcosm, would then emancipate the OHP community for writing a lot of nearly identical extension modules for the applications to be integrated with each hypermedia service. The idea is depicted in Figure 4, which is borrowed directly from [Davis et al. 1996].

Figure 4. The Shim architecture (from [Davis et al. 1996]).

The Shim architecture proposal is focusing closely on this fitting of different protocols. It does a nice job in this respect, and it does not attempt to cover the rest of the hypermedia system architecture for instance database interfaces and the like.

A critical question one could raise however is: What is the long term prospects of shims if we could standardize an OHP? If all open hypermedia systems agreed on the same OHP, there would be no need for the extra overhead to pay for a Shim module, except for the situations, where a conversion of one basic communication protocol to another is absolutely necessary.

4.4 The Extended HyperDisco Model
The HyperDisco system architecture consists of three levels: tools consisting of both hypermedia tools and integrated third party applications, tool integrators and HBMS (now called workspaces) [Wiil and Leggett 1996] - see Figure 5.

![Figure 5](image)

**Figure 5.** The HyperDisco system architecture (from [Wiil and Leggett 1996]).

Originally, wrappers were only used to integrate third party applications. A recent experiment which integrated the Chimera server to operate as a HyperDisco workspace [Wiil and Whitehead 1997] made use of a wrapper to integrate a third party information repository (in this case the Chimera server) to interoperate with HyperDisco. The results from the experiment were generalized into a proposal for a reference architecture. Figure 6 shows an architecture of a generic integration of open hypermedia systems showing integration of both external applications and external information repositories (External Storage). The open hypermedia system consists of one or more native OHS Applications, one or more OHS Session Managers and one or more native OHS Storage components (hyperbases or link servers).

![Figure 6](image)

**Figure 6.** The extended HyperDisco model (from [Wiil and Whitehead 1997]).
Wrappers are used to enable external applications and external information repositories to conform to the OHS protocols. The OHS Session Manager is to a large extent similar to the HyperDisco tool integrator concept. An OHS Application is an application that directly supports the OHS Presentation Protocol. Likewise, "OHS Storage" is an information repository that directly supports the OHS Storage Protocol. The model in Figure 6 assumes that the OHS Session Manager is capable of using the standard Presentation Protocol and the standard Storage Protocol. This could be achieved by additional wrappers as in the OHP proposal [Davis et al. 1996] or by adopting the OHS Session Manager to conform to the standards.

The merits of the extended HyperDisco model is the introduction of a second open hypermedia protocol (the Storage Protocol) and a second type of wrapper (the Storage Wrapper) handling integration of External Storage such as databases, file systems, and special-purpose storage managers. Thus, integration of third party stores is handled in the same manner as integration of third party applications. The result is a simple and clean model with few protocols and few types of architectural components.

5. Towards a Synthesis Reference Architecture

By attempting to achieve a synthesis of the important features of the above candidate architectures we arrive at a proposal like Figure 7.

Figure 7. A Synthesis Architecture.
We have simplified the layers by eliminating the Communication layer. Since communication takes place in potentially many different directions, we choose to depict communication with the double bar symbol. One could think of these interfaces both as CORBA IDL interfaces and as Interface Suites in the AppleEvent terminology. The interfaces represent peer-to-peer communication. The notation allows us to keep it open whether the bars actually implements a Shim or is a straight interface. We propose names for four protocols: OHP - the protocol for applications; DBP - the protocol for the hypermedia database; DMP - the protocol for document managers; AP - the protocol for agents. But we envision new types of services which we in the future have to interface to, thus we have a couple of dashed double bars.

This architecture still shares the problems with application data with the earlier described Dexter-based architectures. Another general problem with this architecture, and in fact all the candidates is that, distribution and cooperation aspects are hard to express explicitly in the diagrams. Cooperation is handled at the Hypermedia service and Hypermedia database level, depending on the actual distribution of responsibilities among these two elements in a specific system. Distribution is depicted implicitly in that each hypermedia service may communicate with multiple databases, applications and document managers.

Despite the mentioned deficiencies we believe the architectural model may serve as a first step towards a reference architecture which is conceptual, simple, general and extensible.

References

[Davis et al. 1996]

[Grønbæk et al. 1997]

[Grønbæk et al. 1994]

[Wiil and Leggett 1997]

[Wiil and Leggett 1996]
[Wiil and Whitehead 1997]

[Østerbye and Wiil 1996]