Using the Flag Taxonomy to Study Hypermedia System Interoperability

Uffe Kock Wiil

The Danish National Centre for IT Research
Aarhus University[†]

Kasper Østerbye

Department of Computer Science
Aalborg University[‡]

ABSTRACT

Interoperability between existing systems, program packages, tools and applications with various degrees of hypermedia awareness is a complex and important challenge facing the hypermedia community. This paper presents a general framework (called the Flag Interoperability Matrix) to discuss and examine hypermedia system interoperability based on the concepts and principles of the Flag taxonomy of open hypermedia systems. The purposes of the Flag Interoperability Matrix are to provide a framework to classify, describe concisely and compare different approaches to hypermedia system interoperability, and provide an overview of the design space of hypermedia system interoperability. The Flag Interoperability Matrix is used to examine existing interoperability approaches. Based on a systematic analysis of possible approaches to hypermedia system interoperability, the paper explores one solution to hypermedia system interoperability that seems particularly promising with respect to handling the growing number of applications with increasing but incomplete awareness of hypermedia structure concepts.

KEYWORDS: Flag taxonomy, interoperability matrix, partial hypermedia system, interoperability protocol

1 INTRODUCTION

Interoperability between existing systems, program packages, tools and applications with various degrees of hypermedia awareness is a complex and important challenge facing the hypermedia community. This type of interoperability offers significant promise for future widespread use of hypermedia technology across existing desktop and Internet applications. The overall goal of this research is to be able to create and traverse links just as easily between different applications with different levels of hypermedia awareness as can be done internally in fully hypermedia aware applications. A close examination of this problem domain reveals two general tasks in the quest for interoperability:

- *Intra-application hypermedia services*: adding hypermedia services to applications with limited concept and knowledge of hypermedia. Basic hypermedia services include support for anchors and links, while more advanced hypermedia services also include support for composites, guided tours, trails, etc.
- *Inter-application hypermedia services*: allowing links, composites, etc. to cross application boundaries.

Researchers in the open hypermedia community are currently addressing these issues [20, 22, 25]. An open hypermedia system (OHS) is typically a middleware component in a computing environment offering intra- and inter-application hypermedia services to third party applications orthogonal to their storage and display services. A third party application can be extended to become hypermedia aware by both making the hypermedia services available in the user interface of the application and enabling the application to communicate hypermedia requests to an OHS.

The Flag taxonomy [27] (or simply the Flag), which is briefly presented in Figure 1, can be used to depict the open hypermedia approach to interoperability. An OHS contains a session manager module and a data model manager module that are responsible for managing and storing the hypermedia structures (see Figure 2). Applications are responsible for manipulating (editing and presenting) and storing content, either in a storage manager (e.g., a file system) or in an OHS when possible. OHSs that provide only structural services are often called *link server systems* (e.g., Microcosm [10], Chimera [2] and Multicard [16]), while OHSs that also provide content storage often are called *open hyperbase systems* (e.g., Devise Hypermedia (DHM) [9], HOSS [15] and HyperDisco [23]).

[†] Current address: Aalborg University Esbjerg, Niels Bohrs Vej 8, 6700 Esbjerg, Denmark, ukwiil@aue.auc.dk

[‡] Current address: Norwegian Computing Center, Postbox 114 Blindern, N-0314 OSLO, Norway, Kasper.Osterbye@nr.no

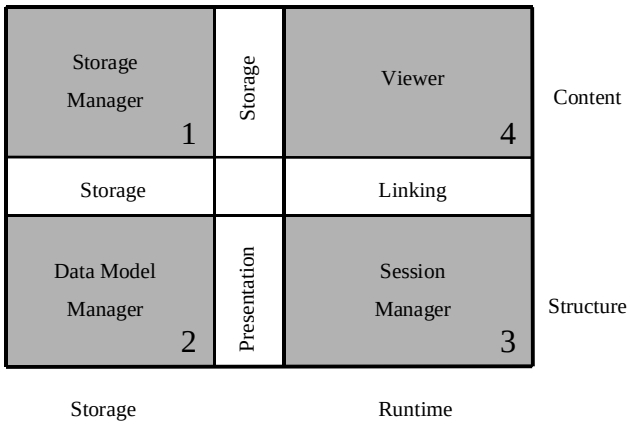

Figure 1: The Flag distinguishes between storage aspects and runtime aspects, and between structure and content. Four functional modules (shown as grey boxes) are specified: storage manager (content storage), data model manager (structure storage), session manager (structure runtime) and viewer (content runtime). Four protocols (shown as white bands) enable neighboring functional modules to exchange information.

Although the approach depicted in Figure 2 is used by most OHSs and has proven quite effective [6, 19], it has also uncovered an important interoperability challenge: *How do we handle the growing number of applications with increasing awareness of hypermedia structure concepts?* As pointed out in [14] and [26], the major underlying premise of the previous work on application integration with OHSs has been that the majority of applications can be characterized as hypermedia unaware (meaning that they have no inter- or intra-application hypermedia services). This rigid view of applications is no longer appropriate, since many applications (such as the individual applications in Microsoft Office 97) have a high degree of hypermedia awareness internally (intra-application hypermedia services). Thus, in these cases, it is only necessary to add inter-application hypermedia services to the applications, and, equally important, make the intra- and inter-application hypermedia services interoperate in a seamless manner.

The remainder of the paper is organized as follows. Section 2 presents a general framework to discuss and examine hypermedia system interoperability based on the concepts of the Flag. The introduced framework is used to examine existing interoperability solutions. Section 3 explores one solution to hypermedia system interoperability that seems particularly promising. Section 4 concludes the paper.

2 HYPERMEDIA SYSTEM INTEROPERABILITY

This section is divided into five parts. Section 2.1 provides a theoretical point of view on how the Flag can be used to depict hypermedia system interoperability. The interoperability challenge is further examined in Section 2.2 in an empirical manner by analyzing an interoperability experiment between HyperDisco [23] and Chimera [2]. The theoretical and empirical results are used to develop a framework (the Flag Interoperability Matrix) for systematic analysis of hypermedia system interoperability solutions. The Flag Interoperability Matrix is presented in Section 2.3 and used in Section 2.4 to examine two existing interoperability solutions. In Section 2.5, the results of the previous sections are used to identify one particularly promising solution to interoperability.

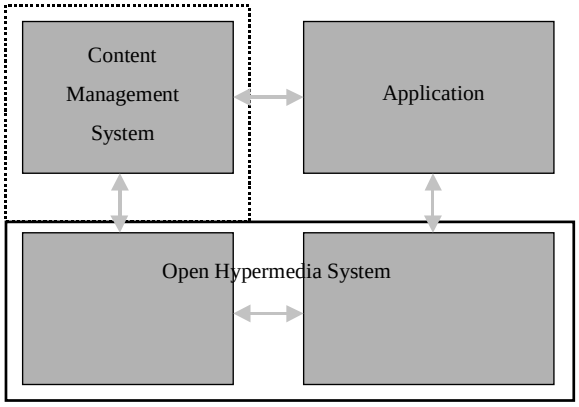

Figure 2: The Flag used to depict the OHS approach to interoperability. An open hypermedia system handles structure and in some cases also content storage on behalf of integrated applications (hence the dotted lines around the content management system).

2.1 Depicting Interoperability with the Flag

This section provides a first view of how the Flag can be used to outline interoperability settings where a number of hypermedia systems interoperate. Two settings (based on Figure 3) and their implications are presented below.

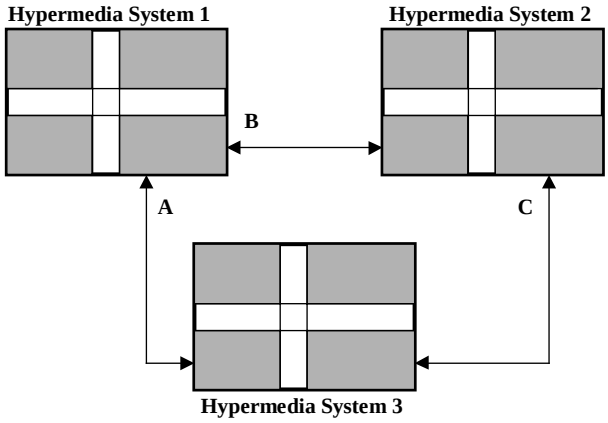

Figure 3: Using the Flag to depict interoperability.

Interoperability Setting 1. Hypermedia System 1 (HS1) interoperates with Hypermedia System 2 (HS2) and Hypermedia System 3 (HS3) through protocols A and B that connect the session manager of HS1 to the data model

managers of HS2 and HS3 respectively. The resulting interoperability setting enables the session manager of HS1 to access a distributed hypermedia storage system. It is important to notice that this setting does not require full functionality of all three hypermedia systems. HS1, HS2 and HS3 could be partial systems that only provide some of the functional modules of the Flag. HS2 and HS3 only need to implement the data model manager (and potentially also the storage manager). HS1 only needs to implement a session manager that interoperates with one or more hypermedia aware applications. This would allow HS1 applications to create and traverse links between content managed by HS2 and content managed by HS3.

Interoperability Setting 2. HS2 and HS3 interoperate through protocol C that connects the session managers of both systems. Compared to Setting 1 where HS1 initiates the interoperability, Setting 2 suggests an equal participation among two systems resulting in shared control over the interoperation. This could apply to a collaborative interoperability setting between two different hypermedia systems or two instances of the same hypermedia system. HS2 could, for example, be a partial hypermedia system consisting of an application providing intra-application hypermedia services and a wrapper implementing a simple session manager that allows the application to interoperate with other hypermedia systems. HS3 could be a full hypermedia system implementing all the functional modules of the Flag and proving both intra- and inter-application hypermedia services. In this case protocol C would allow HS2 to access and use the inter-application hypermedia services of HS3. Thus, this type of interoperability setting could also allow applications with different levels of hypermedia awareness to interoperate.

Two important observations can be made from these two interoperability settings:

1. Previously, the Flag only dealt with integrated applications assumed to have no prior hypermedia functionality and full hypermedia systems. The notion of partial hypermedia systems (and similarly partial Flags) allows us to distinguish applications based on the degree of hypermedia functionality available. An integrated application with no prior hypermedia functionality is depicted as a single functional module, an application providing some degree of inter-application hypermedia functionality is depicted as a partial Flag (e.g., two functional modules), and an application (system) with full hypermedia functionality is depicted as a full Flag.

2. In an interoperability setting it makes a great difference which functional modules of the participating systems are connected. The full range of interoperability connections will be explored in Section 2.3.

2.2 Examining the Interoperability Challenge

This section presents a particular interoperability experiment involving the HyperDisco [23] and Chimera [2] OHSs. The experiment is described fully in [24]. It will be briefly described here and major implications for hypermedia system interoperability will be discussed. Figure 4 depicts the interoperability experiment using the Flag. The HyperDisco tool integrator interoperates with the Chimera server (version 1.2) through a wrapper process. From the perspective of HyperDisco, the wrapper enables the tool integrator to access the Chimera server like it was a HyperDisco workspace. From the perspective of Chimera, the wrapper accesses the Chimera server like any other Chimera enabled application. This allows Chimera enabled and HyperDisco enabled applications to access and update simultaneously structure stored in the Chimera server. The status of the experiment is that a basic level of interoperability has been achieved: HyperDisco enabled applications can use the anchoring and linking services of the Chimera server, and links can connect anchors in a HyperDisco workspace with anchors in the Chimera server. Even though actual collaboration experiments where impossible (HyperDisco supports collaboration, but Chimera 1.2 does not), the experiences suggest that the developed interoperability setting can support collaborative work across different hypermedia systems [24].

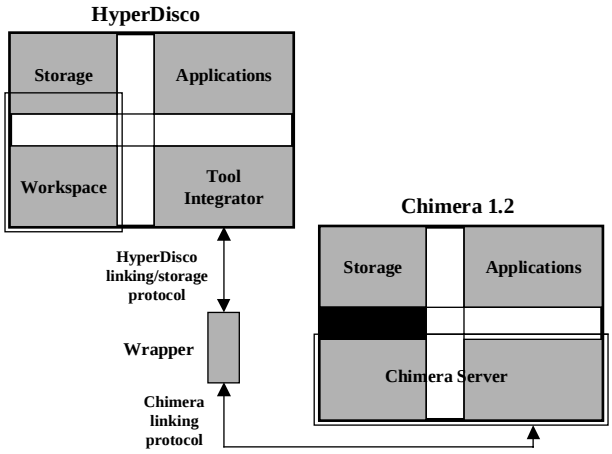

Figure 4: The HyperDisco – Chimera interoperability experiment. HyperDisco workspaces provide both content and structure storage as indicated with the box. The Chimera server handles both session management and structure storage, but does not handle content storage. This is indicated with the box around two functional modules and the black protocol field (which in Flag terminology means that no protocol exists).

The lessons learned from the experiment (and the theoretical interoperability settings of Section 2.1) can be used to observe relationships between the interoperability issue and other issues in hypermedia system design and development such as distribution, multiple users, collaboration and heterogeneity:

- Distribution is a special case of interoperability in which multiple instances of the same hypermedia system interoperate. The more general case (interoperability) allows interoperation between different hypermedia systems. Thus, a general solution to hypermedia system interoperability will encompass the issues of hypermedia system distribution.

- Distribution is a pre-requisite for multiple users and collaboration.[1] To support collaboration, one or more functional modules of a hypermedia system must be replicated across different workstations. Collaborative hypermedia systems of the early 1990s often provided a client-server solution with one server and multiple clients distributed across a local-area network (e.g., Sepia [17] and EHTS [21]). Since distribution is a special case of interoperability, a general interoperability solution can serve as a basis for collaboration.
- Interoperability implies heterogeneity. While hypermedia system distribution is a matter of managing multiple instances of homogeneous functional modules, then hypermedia system interoperability must deal with heterogeneous functional modules of different hypermedia systems. Thus, a general solution to functional module interoperability will address the issues of heterogeneity.

These observations will be used in Section 2.3 to classify hypermedia system interoperability into broad categories.

2.3 The Flag Interoperability Matrix

The previous sections show that the Flag can be used to depict hypermedia system interoperability. Since the Flag has four functional modules, two hypermedia systems can interoperate in sixteen different ways (see Table 1). If one hypermedia system module is considered the initiator of the interoperation and the other the responder, none of the sixteen possible interoperability connections are symmetric. This section briefly discusses all sixteen fields of the matrix and in each case gives an example of what interoperability might be about if realized through a connection between the two given functional modules. Some of the more interesting interoperability connections will be dealt with in more detail. Each field is given a unique number. For example, the (3,2) field corresponds to a setting where a session manager of one hypermedia system initiates communication with a data model manager of another hypermedia system.

The sixteen fields are not completely independent; many share several characteristics. Based on the observations made in Section 2.2, the matrix has been divided into four main categories: *distribution, notification, collaboration* and *undefined*. The *undefined* category results from the fact that two of the six possible protocols have not been defined in the Flag. It is neither possible for a viewer to communicate directly with a data model manager nor for a session manager to communicate directly with a storage manager in the Flag.[2] These four fields will therefore not be further discussed in this paper. Notice that the four corners (1,1), (4,1), (1,4) and (4,4) are less interesting from a hypermedia perspective because the participating functional modules only handle content not structure.

initiator

responder	1 storage manager	2 data model manager	3 session manager	4 application (viewer)
1 storage manager	collaboration (1,1)	distribution (2,1)	undefined (3,1)	distribution (4,1)
2 data model manager	notification (1,2)	collaboration (2,2)	distribution (3,2)	undefined (4,2)
3 session manager	undefined (1,3)	notification (2,3)	collaboration (3,3)	distribution (4,3)
4 appli-cation (viewer)	notification (1,4)	undefined (2,4)	notification (3,4)	collaboration (4,4)

Table 1: The Flag Interoperability Matrix.

Distribution. The fields above and including the diagonal all represent interoperability settings that are well suited for providing distribution.

The (2,1) field corresponds to an interoperability setting in which a data model manager of one system contacts a storage manager of another system. In the DHT system [12], the data model manager is capable of handling content from several storage managers. DHT allows integrated applications to access diverse storage managers in a uniform manner. The DHT data model manager transforms data from the storage managers into nodes and links of the DHT data model and vice versa.

The (4,1) field corresponds to a setting in which an application stores or retrieves its content from a storage manager of another system. This requires that the storage manager of the responder is a self-contained server that can handle requests from the outside and is thus only loosely coupled to the responding hypermedia system.

The (3,2) field represents a setting where a session manager of one hypermedia system communicates with a data model manager of another system. The HyperDisco – Chimera interoperability experiment (Section 2.2) can be interpreted as an example of this. From the perspective of HyperDisco, HyperDisco enabled applications can access the Chimera server via the tool integrator in the same manner as other HyperDisco workspaces.

The (4,3) field represents a setting where an application initiates communication with a session manager of another system. Some applications can have several documents open at the same time. It is possible to imagine that an application simultaneously manages documents that are handled by different session managers. If we ignore the problem of determining which hypermedia system such an application actually belongs to, it is still possible that the application contacts the session manager of more than one hypermedia system. In a setting with several interoperating hypermedia

---

[1] Obviously, multiple users are a pre-requisite for collaboration. A hypermedia system must support multiple users to be able to support collaborative work settings. We will use the more general term "collaboration".

[2] On a number of occasions, Randy Trigg has pointed out that we should consider these two connections in the Flag as well (which would make the Flag look like the Union Jack rather than Dannebrog – the Danish Flag). We have not yet seen a need to do this, but invite useful examples.

systems, a "save as" operation might allow the user to store the structure in a different hypermedia system than that in which it was first created. The HyperDisco – Chimera interoperability experiment can also be interpreted as an example of this setting. From the perspective of Chimera, HyperDisco can be perceived as a Chimera enabled application that accesses the Chimera server in the same manner as other enabled applications.

Notification. The fields below and including the diagonal represent interoperability settings that naturally arise between systems where the initiator needs to inform the responder about changes - that is, in *notification* based collaboration. Each notification field complements a specific distribution field such that functional modules engaged in a specific interoperability setting can act as both responder and initiator. This two-way asynchronous communication between functional modules is necessary to support collaborative interoperability settings.

The (1,2) field corresponds to settings in which a storage manager *notifies* a data model manager of another hypermedia system that a document has been changed in one way or the other. If the responder has a node that wraps a file in the initiator, the responder can be interested in changes to e.g., the file contents or file name in order to maintain its own state as a true model of the file. This setting complements the (2,1) setting.

The (2,3) field represents a setting in which a data model manager communicates with a session manager of another hypermedia system to distribute events. Thus, this setting complements the (3,2) setting.

In a **(1,4) setting**, a storage manager communicates with an application of another hypermedia system. This might occur in link server systems, where distribution of documents is achieved through distribution of storage managers. In such a case, locking and event distribution to applications will come not only from the session manager but from the distributed storage managers as well. This setting complements the (4,1) setting.

In **a (3,4) setting**, a session manager could notify an application in some other hypermedia system that some of its cross application links have been changed. This setting complements the (4,3) setting.

Collaboration. The four fields along the diagonal are characterized by supporting both distribution and notification. Both aspects are a pre-requisite for collaboration. Because the two interacting functional modules are of the same type, the interoperability setting is typically not a client-server relation, but communication between peer systems.

In **a (1,1) setting**, a storage manager directly accesses a storage manager of another system. Several interpretations are possible. The storage manager of the initiator could allow transparent access to content from its own storage as well as the responder's. Alternatively, they might need to be kept synchronized over a distributed environment.

The (2,2) field represents a setting where a data model manager exchanges information with a similar component in another hypermedia system. An example of this setting is a set of hypermedia systems sharing one logical but physically distributed hypermedia database. The (2,2) communication can be used to keep the distributed hypermedia database in a consistent state.

The (3,3) field represents two session managers that interact. One way to view it is as a single logical hypermedia system composed of a number of hypermedia systems, which interoperate through their session managers. The HyperDisco – Chimera interoperability experiment can also be interpreted as an example of this setting, since the tool integrator (session manager) interoperates with the session manager part of the Chimera server through the wrapper, which transforms the HyperDisco protocol to the Chimera protocol and vice versa.

The (4,4) field represents the setting where two applications communicate directly. This could happen when multiple instances of a collaborative application are used in different hypermedia systems at the same time.

The different interpretations of the HyperDisco – Chimera interoperability experiment shows that a particular interoperability setting can be viewed from different perspectives: from the perspective of each participating system and from a more general, system-independent perspective based on the actual connections between functional modules of the participating systems. (3,2) is a HyperDisco perspective, (4,3) is a Chimera perspective, and (3,3) is a general, system-independent perspective. The latter perspective is the generic way to classify the experiment using the Flag Interoperability Matrix. However, the two system perspectives are useful as a supplement because they give additional details on how the interoperability setting is constructed (i.e., what communication protocols are used, etc.).

2.4 Case Studies

A number of OHSs such as DHM [8], Microcosm [4], Chimera [1], Hyper-G [11] and HOSS [15] are currently interoperating with the World Wide Web (WWW) [3]. Anderson [1] provides a taxonomy of possible integrations between OHSs and the WWW, which is constructed by considering the intersections between the four major architectural elements common to both systems: clients, servers, protocols and data formats.[3] This section shows how the Flag Interoperability Matrix can be used to analyze, classify and compare two of these approaches, namely DHM and Microcosm.

Case Study 1: DHM and the WWW. Figure 5 depicts the interoperability approach used in the integration between DHM and the WWW, which is called DHM/WWW [8]. Like HyperDisco, DHM can be classified as an open hyperbase system, which is why these two systems are

[3] While Anderson's taxonomy focuses on OHSs and the WWW, the Flag Interoperability Matrix provides a framework to examine interoperability among hypermedia systems in general (i.e., the example in Section 2.2).

depicted in the same manner using the Flag (compare Figures 4 and 5). However, these systems use different names for their functional modules. The WWW consists of servers, browsers, helper applications and storage. The browser can retrieve local files directly from the local file system and remote files through remote servers. The browser launches helper applications and browser plug-ins to display data formats that cannot be displayed by the browser itself such as PDF files (Acrobat Reader plug-in). Several possible implementations are described in [8]. The platform-independent solution extends the WWW browser with an applet that handles communication to the WWW server via HTTP, communication with DHM through common gateway interface (CGI) scripts, and presentation of links in HTML documents in the WWW browser. The applet has a separate user interface that allows DHM links and anchors to be created in existing HTML documents.

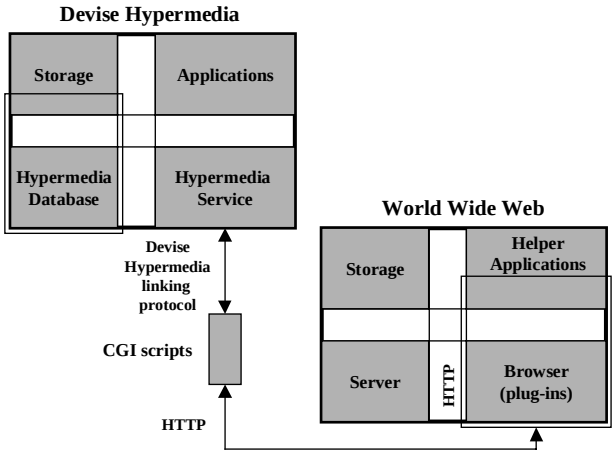

Figure 5: Interoperability between DHM and the WWW. The CGI scripts are invoked via an ordinary WWW server that runs in the same Internet domain as DHM. The CGI scripts transforms HTTP requests into DHM linking protocol requests.

Case Study 2: Microcosm and the WWW. The interoperability approach used in the integration between Microcosm and the WWW is depicted in Figure 6. The research version is called Microcosm's Distributed Link Service [4], while the commercial release is called Webcosm [18]. This example will examine Webcosm. Like Chimera, Webcosm belongs to the link server system category of OHSs (hence the Flag depicts Webcosm and Chimera in a similar manner – compare Figures 4 and 6).[4] Webcosm consists of a link server, a WWW server extension and linkbases. The Webcosm extension can either be applied to an existing server or a proxy server (the Webcosm extension can also be used as a standalone server that basically acts like a proxy server). The link server, which can manage several different linkbases, is accessed via CGI scripts. WWW browsers need not be extended in any way. Instead, Webcosm provides a separate user interface that handles selection a linkbases, presentation styles for Webcosm links in HTML documents, and creation of Webcosm links in HTML documents.

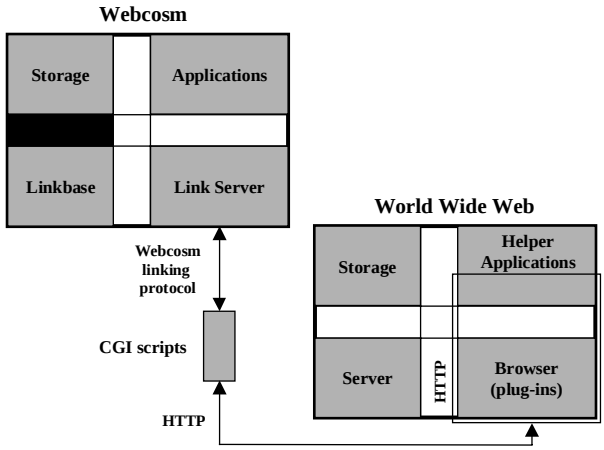

Figure 6: Interoperability between Microcosm and the WWW. The CGI scripts are either invoked via a proxy server or through an extension to an ordinary WWW server. In both cases the CGI scripts (and the server activating them) must reside in the same Internet domain as the Webcosm link server. The CGI scripts transforms HTTP requests into Webcosm linking protocol requests.

Both case studies use the (3,3) interoperability setting[5] and are depicted in the exactly same way using the Flag. From a user's point of view the two resulting systems have many features in common. Both solutions add external links to the WWW, provide a separate user interface for the creation of these links, and benefit from existing WWW infrastructure, which provides Internet distribution, scalability and a large user base. The differences lie in the way the interoperability is implemented and in the resulting hypermedia features. DHM/WWW extends the WWW browser, while Webcosm extends the WWW server to achieve the interoperation. The hypermedia models of DHM and Microcosm are quite different (DHM provides a richer hypermedia model), which is reflected in Webcosm and DHM/WWW. Webcosm adds external links to the WWW, while DHM/WWW adds external links, composites and a potential for collaborative authoring of HTML documents and external structure (composites and links).

2.5 Towards a Single Interoperability Protocol

The previous sections show that hypermedia systems can interact in many ways. We believe that it would be counter-productive to specify and develop twelve protocols (not counting the four undefined). A more interesting question is: "*Can we narrow the number of protocols down to one?*" Such a candidate should be sought at the diagonal of the

4 Except that Chimera implements the data model manager and session manager as one process (hence the enclosing lines) and Webcosm implements these functional modules as two separate processes.

5 The actual communication takes place between the session managers of each system, but both DHM and Webcosm use the existing WWW infrastructure to activate the CGI scripts. Requests are routed through a WWW server, which makes the settings resemble a (2,3) setting.

Flag Interoperability Matrix, since these four protocols are all potentially collaborative and peer to peer distributed. The two corners, (1,1) and (4,4), can be ignored, as they do not address hypermedia structure directly. This leaves two choices: (2,2) and (3,3). The (2,2) approach represents interoperation through the structural storage module. While this can lead to interoperability settings with powerful collaboration capabilities, it is less obvious how runtime information can be shared and exchanged. To create a link that spans two hypermedia systems, the user has to select link endpoints in both systems. This simple observation implies that both systems must have a notion of "current endpoint selection", which is an archetypecal runtime concept. This leaves the (3,3) approach as the most promising candidate for an interoperability protocol for hypermedia systems. In fact, all the interoperability solutions examined in this paper are based on the (3,3) approach. Section 3 defines the necessary operations of such a protocol.

3 A PROPOSAL FOR INTEROPERABILITY BETWEEN HETEROGENEOUS HYPERMEDIA SYSTEMS

In Section 2, the Flag was used to describe different ways in which (partial) hypermedia systems can interoperate. The OHS working group [13] is currently creating a standard linking protocol (called the Open Hypermedia Protocol or OHP [5]) for interaction between hypermedia aware applications and OHSs. In essence, the OHP allows different OHSs to use the same set of applications. However, the OHP does not address interoperability among peer hypermedia systems. This section introduces a protocol for interoperability between heterogeneous hypermedia systems. The protocol is named the T3 protocol after the matrix entry three-3. It has been an explicit goal of ours to reuse existing OHP operations whenever possible. In order to analyze what new operations are required in the session managers, we will examine two typical hypermedia system services: following a link (navigation) and creating a link (link authoring). In order for the T3 protocol to work, it was necessary to introduce a new component into the overall system, the Hypermedia System Manager (HSM), which maintains information about the location of hypermedia systems and services. The HSM can be implemented as a separate component or as part of an existing hypermedia system. In this section, we will show the HSM as a separate component for illustrative purposes. Section 3.2 discusses the HSM in more detail.

3.1 The T3 Protocol

Navigation. Figure 7 illustrates the process of following a link. The boxes represent hypermedia systems and the arrows are calls the systems make to each other. The numbers along the arrows describe the calling order, and a small dot at the source of the arrow indicates that the call returns a value of interest. The situation is that the user activates a link on System 1 (S1), the link is resolved through a series of steps, and finally the destination of the link is shown in System 3 (S3):

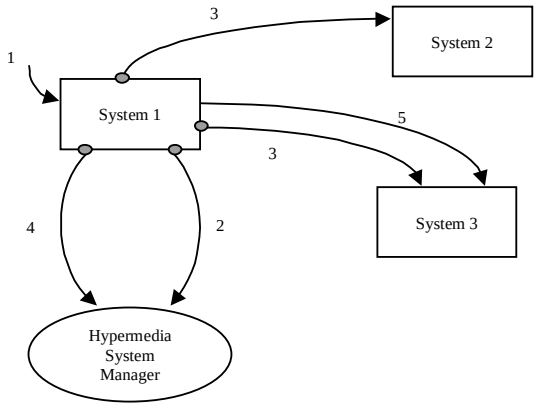

Figure 7: The process of following a link between heterogeneous hypermedia systems using the T3 protocol. Although not explicitly shown, hypermedia systems interoperate via their session manager.

1. The user activates a link marker in S1. S1 looks up the link endpoint and decides that it is either not able to resolve it, or that it will ask other systems to resolve it.
2. S1 calls the operation *whoResolves* in the HSM. The HSM returns a set of system identifications, which can be used by S1 to address these systems.
3. Based on the answer from the HSM, S1 now knows that both System 2 (S2) and S3 resolve. S1 decides (either by built-in logic, or based on user interaction) to ask both S2 and S3 to resolve the given endpoint. This is done using the *resolve* operation that returns the set of endpoints of the resolved link. Because both S2 and S3 were asked to resolve the link, S1 needs to combine the results in some way. We will assume that S1 selects just one endpoint to be displayed. Otherwise, steps 4 and 5 must be repeated for each endpoint to be displayed.
4. Because S1 is not able to present the type of document that is the destination of the link (we assume it is not for illustrative purposes), it needs to find out who is. S1 calls the HSM operation *whoShows*. An endpoint consists of three specifiers: a node specifier, which identifies the node of the endpoint, a location specifier, which identifies a location within the node, and a presentation specifier, which specifies how the node is to be presented. Part of the presentation specification is a description of the media type of the node (i.e., if it is ASCII text, a word document, midi sound format, etc.). S1 is told that S3 can show the endpoint. The HSM might only consider the media type of the presentation specification, or it might be more advanced taking into account other parts of the specifications as well.
5. Finally S1 asks S3 to present the endpoint (using the *show* operation). An interesting issue is whether S3 is able to display the endpoint on the same machine as S1. As some users have several computers running different operating systems in their office, we will not restrict the protocol as to require that the destination(s) of a link should be presented at the same display as its source.

The navigation part of the T3 protocol can be used to provide linking into a system that does not store anything but persistent selections (e.g., a program development environment (PDE)). PDEs normally support a limited internal "hypertext" in the sense that program identifier usage are "linked" to identifier definitions. To enable a PDE to participate in the T3 protocol, it needs, in its simplest form, to be able to respond to the *show* operation. A PDE can handle endpoints in the form of identifiers and map them to their definition, automatically finding the file that contain the definition, and scroll to the appropriate definition. Thus, the identifier definition can be viewed as a persistent selection that is maintained by the PDE. Similarly a PDE can perform a simple *resolve* operation, as it can pretend that an identifier usage is indeed linked to its definition. Thus, a PDE can from the outside be made to look like it maintains a link database and provides a link resolution mechanism.

Link authoring. This section will examine what it takes to create a link that spans two heterogeneous hypermedia systems. Many of the issues of where the link resides are similar to issues addressed in the Neptune system [7], but actually seem clearer in the heterogeneous setting where a context becomes a unique system.[6] Figure 8 illustrates a situation where a user creates a link from S1 to S3 (storing the link in S2). The diagrammatic notation is as before.

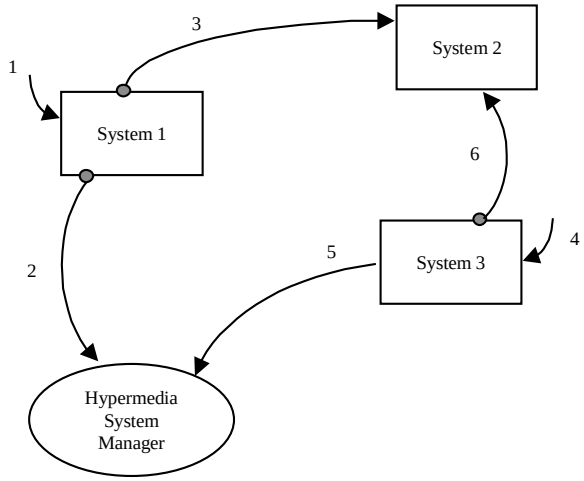

Figure 8: The process of creating a link between heterogeneous hypermedia systems using the T3 protocol. Like before, hypermedia systems interoperate via their session manager.

1. The user makes a selection, and issues a "create link with selection as endpoint" function. S1 is not able to store links, and needs to find some one who can.
2. S1 calls the HSM with the *whoLinks* operation and is told that S2 supports structural storage.
3. S1 then calls the operations *createLink* followed by *addEndpoint* on S2 (S1 created an endpoint based on the selection).
4. Next the user moves to S3, where another selection is made. This time, however, the user is not interested in creating a new link, but adding the endpoint to the newly created link.
5. S3 issues a *whoLinks* call to the HSM, which again gives S2 as the answer. We have not found it useful to have a *whoCreatesLink* and a *whoAddsEndpoints*, as we have found no situation where it makes sense to have one but not the other.
6. S3 calls the *addEndpoint* operation on S2, with the endpoint created from the selection made in (4).

There is an interesting issue related to the *addEndpoint* operation, which is to specify what link the endpoint should be added to. First it is worth noticing that we cannot give a link identifier as parameter, as there is no way for S3 to know the identity of the link created by S1 in S2. One way to solve the issue is to let each system maintain a "current link", which is set using a link editor from the system in question. Thus, the *addEndpoint* operation does not need to have a link parameter at all. However, we feel that this solution is too inflexible, and suggest that link specifiers be used to indicate to which link an endpoint should be added. The only new requirement we make on the OHP is that *addEndpoint* can accept link specifiers (e.g., "current", "last link created by Kasper", etc.). This will also solve the similar issue in (3): *createLink* returns a link identifier, which can be used in the following *addEndpoint* call.

An OHS attempts to provide a framework, which enables linking between parts of information in formats maintained by third party applications. Endpoint resolution and specification is a major issue; the practical problem is to tailor applications to maintain persistent selections to enable anchors to remain in place during changes to the information. It is possible to see the T3 protocol as a middle ground. A PDE will not normally be able to maintain arbitrary selections from one session to another (or even during a session). However, like many other systems, a PDE provides a set of "predefined" endpoints within the information. The link authoring example can be interpreted as follows. S1 could be Microsoft Word with a simple hypermedia session manager wrapper around it, which cannot store links. S3 is a PDE. S2 is a link server system capable of storing links. The link might connect part of a specification (in Microsoft Word) to the procedure, which implements the specification (in the PDE). A link is then created from the Word document to a procedure within a program source file.

Protocol summary. Table 2 summarizes the operations of the T3 protocol and lists equivalent OHP operations (if any).

[6] We refer to the fact that Neptune partitioned the hypertext into contexts and allowed links to cross context boundaries.

T3 operation	Description	OHP equivalent
Navigation		
show	Show the endpoint passed as parameter. The presentation will be based on the presentation specification (node type, etc.) that is part of the endpoint.	*none*[7]
resolve	Return the set of endpoints that are linked to the endpoint passed as parameter. Consider only the links that this hypermedia system maintains.	*followLink*[8]
Link authoring		
addEndpoint	Add the endpoint passed as parameter as endpoint to the link matching the link specification parameter.	*getLink* followed by *updateLink*
createLink	Create a new link.	*createLink*
HMS operations		
whoShows	Answer the set of hypermedia systems that can show the endpoint passed as parameter.	*none*
whoResolves	Answer the set of hypermedia systems that support the *resolve* operation.	*none*
whoLinks	Answer the set of hypermedia systems that support the link authoring operations.	*none*

Table 2: Operations of the T3 protocol.

3.2 The Hypermedia System Manager

It has so far been left unanswered how the HSM obtains the information necessary to answer the *whoX* requests. There are several possibilities, both architecturally and with respect to information flow. The architectural issue is whether the HSM should be a separate component or part of an existing hypermedia system. Both solutions have been implemented in existing hypermedia systems. Both the WWW [3] and HyperDisco [23] implement their HSM as part of their session manager, while HOSS [15] implements its HSM as a separate component. The information issue is how to migrate location and service information from hypermedia systems to the HSM. One solution is to manually keep a configuration file. Another solution is that each hypermedia system registers its information to the HSM. Registration can either occur once when the hypermedia system first starts up or every time the hypermedia system is started. The configuration file solution has some obvious scalability problems. The other solutions are clearly more flexible, but require more functionality of both the HSM and the participating hypermedia systems.

7 It is currently under discussion in the OHS working group how the notion of following a link can be unbundled into link resolution and endpoint display. In the current status of the discussion there is no way in the OHP to ask a session manager to display an endpoint, but the session manager can ask an application to display a document.

8 In the current status of the discussion the *followLink* operation is unbundled, and only resolves the link.

4 CONCLUSIONS

An important challenge is facing the hypermedia community: *How do we handle the growing number of applications with increasing awareness of hypermedia structure concepts?* This paper provides three contributions towards answering this question: the Flag Interoperability Matrix, the notion of some applications being partial hypermedia systems, and the proposed T3 protocol.

The Flag Interoperability Matrix is a general framework to examine and discuss interoperability among hypermedia systems based on the concepts and principles of the Flag. Over the years, several solutions to interoperability (corresponding to entries in the Flag Interoperability Matrix) have been tried out. The Flag Interoperability Matrix can be used to distinguish clearly such approaches from each other. Three different interoperability solutions were examined using the Flag Interoperability Matrix: HyperDisco with Chimera, DHM with the WWW, and Microcosm with the WWW. The analytical results pointed to a particular interoperability solution based on interaction between session manager modules of participating hypermedia systems. The T3 protocol illustrates the highlights of this solution.

A partial hypermedia system is a system (application) that only implements part of the Flag in terms of functional modules (e.g., program development environments, which can resolve function names into function definitions, and help systems, which can resolve index terms to help cards).

The major implication of the T3 protocol is that it allows partial hypermedia systems to participate, since it only requires that a hypermedia system can respond to *show* and *resolve* calls. Thus, to enable a partial hypermedia system for the T3 protocol requires less of an effort than enabling it for the OHP. The fundamental reason is that it has been relieved of much functionality, which now resides in those hypermedia systems that provide structural storage. The T3 protocol combined with the notion of partial hypermedia systems allows for interaction with the built-in links (intra-application hypermedia services) of many applications.

The concepts introduced in this paper add a new dimension to open hypermedia integration. Currently, an OHS assimilates existing third party applications (assumed to have no knowledge of hypermedia) into an open hypermedia world through integration (e.g., using the OHP). In contrast, the proposed T3 protocol accommodates applications with intra-application hypermedia functionality. Thus, the T3 protocol does not replace the OHP, but instead adds extra value to the OHP.

ACKNOWLEDGMENTS

We wish to thank Peter Nürnberg for his helpful comments. This research was supported in part by the Danish Natural Science Research Council through Grant no. 9400911.

REFERENCES

1. Anderson, K. M. Integrating Open Hypermedia Systems with the World Wide Web. In Hypertext '97 Proceedings, (Southampton, UK, Apr. 1997), ACM Press, pp. 157-166.

2. Anderson, K. M., Taylor, R. N., and Whitehead, E. J., Jr. Chimera: Hypertext for Heterogeneous Software Environments. In ECHT '94 Proceedings, (Edinburgh, Scotland, Sep. 1994), ACM Press, pp. 94-107.

3. Berners-Lee, T., Cailliau, R, Luotonen, A., Nielsen, H. F., and Secret, A. The World Wide Web. Communications of the ACM, 37, 8, (Aug. 1994), 76-82.

4. Carr, L., De Roure, D., Hall, W., and Hill, G. The Distributed Link Service: A Tool for Publishers, Authors and Readers. In WWW4 Proceedings, (Boston, MA, Dec. 1995), O'Reilly & Associates, pp. 647-656.

5. Davis, H. C., Reich, S., and Rizk, A. OHP - Open Hypermedia Protocol. Working Draft 2.0, 20th June 1997. http://diana.ecs.soton.ac.uk/~hcd/ohp/ohp.htm

6. Davis, H. C., Knight, S., and Hall, W. Light Hypermedia Link Services: A Study of Third Party Application Integration. In ECHT '94 Proceedings, (Edinburgh, Scotland, Sep. 1994), ACM Press, pp. 41-50.

7. Delisle, N. M., and Schwartz, M. D. Contexts - A Partitioning Concept for Hypertext. ACM Transactions on Information Systems, 5, 2, (Apr. 1987), 168-186.

8. Grønbæk, K., Bouvin, N. O., and Sloth, L. Designing Dexter-based Hypermedia Services for the World Wide Web. In Hypertext '97 Proceedings, (Southampton, UK, Apr. 1997), ACM Press, pp. 146-156.

9. Grønbæk, K., Hem, J. A., Madsen, O. L., and Sloth, L. Cooperative Hypermedia Systems: A Dexter-based Architecture. Communications of the ACM, 37, 2, (Feb. 1994), 64-74.

10. Hall, W., Davis, H., and Hutchings, G. Rethinking Hypermedia - The Microcosm Approach. Kluwer Academic Publishers, 1996.

11. Maurer, H. Hyper-G now HyperWave - The Next Generation Web Solution. Addison-Wesley, 1996.

12. Noll, J., and Scacchi, W. Integrating Diverse Information Repositories: A Distributed Hypertext Approach. IEEE Computer, 14, 12, (Dec. 1991), 38-45.

13. Nürnberg, P. J. (editor). Open Hypermedia Systems Working Group. http://www.csdl.tamu.edu/ohs

14. Nürnberg, P. J., and Leggett, J. J. And now for the Tricky Part: Broadening the Applicability of Open Hypermedia Systems. In [20], pp. 93-95.

15. Nürnberg, P. J., Leggett, J. J., Schneider, E., and Schnase, J. L. Hypermedia Operating Systems: A New Paradigm for Computing. In Hypertext '96 Proceedings, (Washington, DC, Mar. 1996), ACM Press, pp. 194-202.

16. Rizk, A., and Sauter, L. Multicard: An Open Hypermedia System. In ECHT '92 Proceedings, (Milan, Italy, Dec. 1992), ACM Press, pp. 4-10.

17. Streitz, N., Haake, J., Hannemann, J., Lemke, A., Schuler, W., Schütt, H., and Thüring, M. SEPIA: A Cooperative Hypermedia Authoring Environment. In ECHT '92 Proceedings, (Milan, Italy, Dec. 1992), ACM Press, pp. 11-22.

18. Webcosm. http://www.multicosm.com/webcosm

19. Whitehead, E. J., Jr. An Architectural Model for Application Integration in Open Hypermedia Environments. In Hypertext '97 Proceedings, (Southampton, UK, Apr. 1997), ACM Press, pp. 1-12.

20. Wiil, U. K. (editor). Proceedings of the 3rd Workshop on Open Hypermedia Systems, Scientific Report 97-01, The Danish Natl. Centre for IT Research, 1997.

21. Wiil, U. K. Issues in the Design of EHTS: A Multiuser Hypertext System for Collaboration. In HICSS '92 Proceedings, (Kauai, HI, Jan. 1992), IEEE Computer Society Press, pp. 629-639.

22. Wiil, U. K., and Demeyer, S. (editors). Proceedings of the 2nd Workshop on Open Hypermedia Systems. UCI-ICS Technical Report 96-10, University of California, Irvine, 1996.

23. Wiil, U. K., and Leggett, J. J. Workspaces: The HyperDisco Approach to Internet Distribution. In Hypertext '97 Proceedings, (Southampton, UK, Apr. 1997), ACM Press, pp. 13-23.

24. Wiil, U. K., and Whitehead, E. J., Jr. Interoperability and Open Hypermedia Systems. In [20], pp. 137-145.

25. Wiil, U. K., and Østerbye, K. (editors). Proceedings of the ECHT '94 Workshop on Open Hypermedia Systems. Department of Computer Science, Technical Report R-94-2038, Aalborg University, 1994.

26. Østerbye, K. Fred the Programmer. In [20], pp. 146-149.

27. Østerbye, K., and Wiil, U. K. The Flag Taxonomy of Open Hypermedia Systems. In Hypertext '96 Proceedings, (Washington, D.C., Mar. 1996), ACM Press, pp. 129-139.