Dynamo Architecture Requirements

Geert-Jan Houben and Mark van Doorn

Versie 2.0

This document contains the main requirements for Dynamo's generation system. These requirements represent a common view on the scope of the Dynamo system and they are the basis for the design of (the generation part of) the Dynamo architecture.

The set of requirements contain definitions of the following aspects:

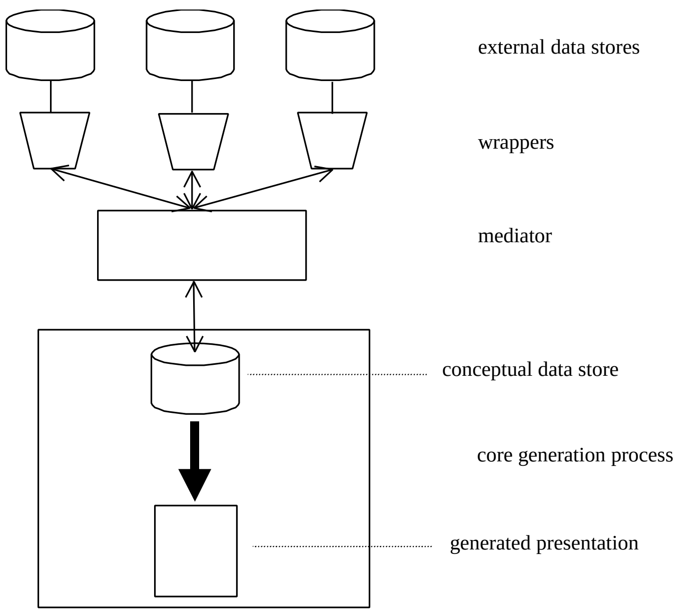
- The scope of the generation process and its environment.
- The intelligence used in the process, including the types of expertise and their functionality.
- The layers in the generation process as steps in the data generation (like in SRM).
- The roles of the data and meta-data used in the different layers of the generation process, where the data can be self-describing or semi-structured.
- The roles of query transformation and data transformation.

Scope of generation process

A complete application can contain/use multiple *data stores*.

It is possible that these multiple data stores are *heterogeneous*. Note that this heterogeneity can have two aspects: the heterogeneity can be of a technical nature (concerning the software or platform involved), while it can also concern the conceptual structure of the contents. While both aspects may be relevant for the practical application, the management of multiple data stores with different internal conceptual structures is within the larger scope of the project: the technical heterogeneity is not.

In order to limit the initial scope of the architecture we assume a wrapping and mediating process to be responsible for the connection between the data stores and the core system.



The functions of these different elements are:

- The *external data stores* contain the data, including meta-data on the data, relevant for the application. Examples are data stored in a RDBMS or in an IR system, or content available from the Web.
- The *wrappers*, generally one per external data store, are responsible for the retrieval of data in their data store, for example a DBMS or IR system. This includes retrieving data from the external data store when asked for, and translating those data into the format and structure used inside the core generation process.
- The *mediator*, generally a single one, is responsible for the transformation of a query (information retrieval request) from the core generation process into queries (requests) for the different wrappers. This includes a translation to the format and structure used in the specific wrappers. The mediator is also responsible for the transformation of the data that result from the different queries into data with the format and structure used by the core generation process.
- The *conceptual data store* contains the data, including its meta-data, as used in the core generation process. The core generation process can abstract from the wrapping and mediating and consider the application data to be available in the conceptual data store. This conceptual data store is controlled by the mediator. It may not be materialized and exist only virtually: then, the generation process could also interact directly with the top-level mediator, to ensure that the data is always up-to-date. If it is materialized, it can be considered as a local cache of (a subset of) the data in the external data stores.

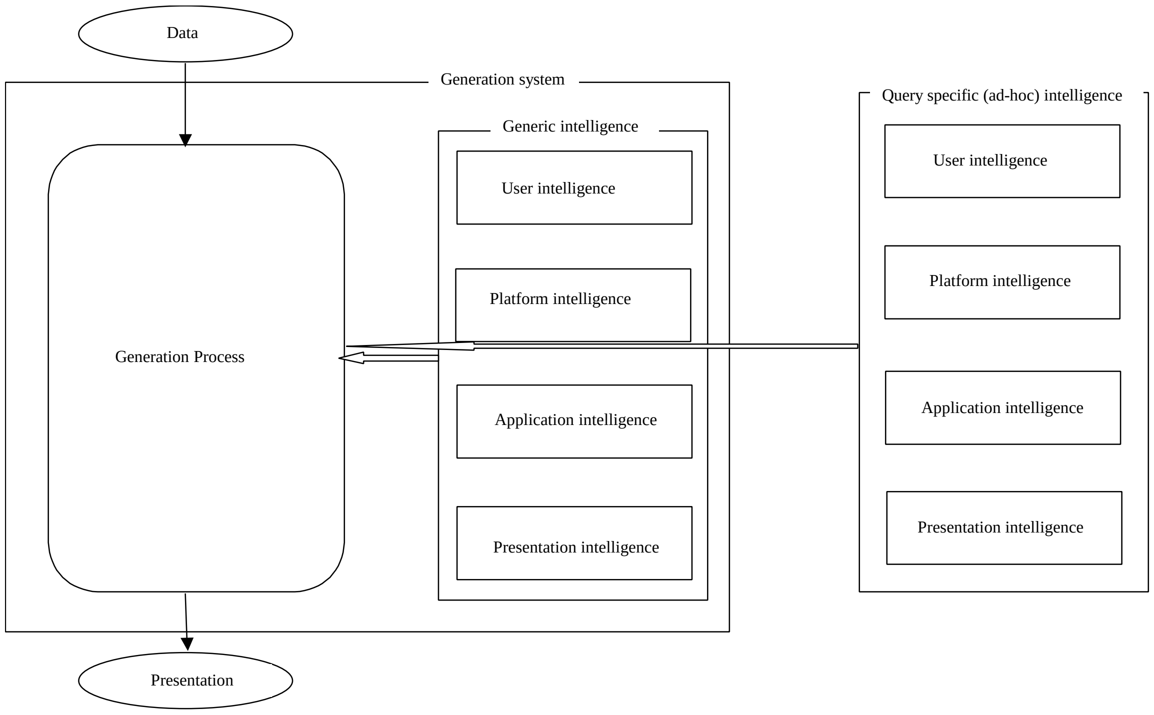
It is important to note that in general there are two (extreme) ways of using multiple data stores (and combinations of those). It is possible to merge the results of different retrieval engines (send a query to multiple databases and aggregate the results) or to use a single database that is filled through many wrappers (i.e. Web crawlers find content on the Web, parse this content and store it in database tables). The nature of the first is known as "pull" (distributed query), while the nature of the second is known as "push" (push content in a central database and query that database later). Note that the subsequent querying by the users is always of a “pull” nature; the above mentioned distinction between push and pull concerns the prior information retrieval. We are not directly interested in the technical aspects of this retrieval process, but we assume the wrapping and mediating process to be conceptually as described above.

For the purpose of this document the application data are available from (stored in) the conceptual data store (CDS). The core generation process, as considered in this document, uses data from CDS in order to generate a presentation.

Generation intelligence

In the (core) generation process for data from CDS a presentation is generated. This means that once some of the data from CDS have been selected, these data are transformed into a presentation that can be shown in the user's browser. In this process information is used that represents "intelligence" on the desired generation process. This intelligence resembles the contents of the Experts in SRM .

Several aspects or dimensions of the intelligence can be distinguished.



The nature of the intelligence leads to different kinds of intelligence:

- *User intelligence*: For the sake of personalization of the generation process knowledge about the user is maintained and incorporated in the process. This includes:
    - *user preferences*:
        - either *explicitly set* by the user
        - or *implicitly derived* by the system (note that the learning process, e.g. derivation by social interaction, is initially outside the scope)
    - *user history*: data on the previous actions (behavior) of the user, usually automatically registered by the system; perhaps divided in:

        - *short term* knowledge (derived from session preferences)
        - *long term* knowledge (derived from complete user history)
- *Platform intelligence*: Knowledge on the platform or network that the user is using is used in order to perform customization to the platform and network. Knowledge on the platform and network can be used to select and schedule media objects: if the network permits broadband communication, high-quality video can be presented to clients otherwise we would have to fall back on low-quality video or even only textual data. This intelligence is tied to the user intelligence.
- *Application intelligence*: The provider of the information, the party presenting the data to the user, has its own goals and therefore its own intelligence on how the presentation should be generated. This knowledge is referred to as the application intelligence, or author intelligence. In the case of an educational application, this intelligence is sometimes referred to as the *Teaching Model*, while in the case of a commercial application the name *Business Model* is being used. Two illustrative examples of application intelligence are:
    - *domain knowledge*, general background information on the domain, perhaps provided as meta-data by the provider of the data thus specifying the domain-dependence, e.g. the schema of the data involved
    - teaching or *domain interaction knowledge*, describing how the learning process is performed by the consumer of the data, e.g. by guiding the user towards data relevant for the user's current state and away from data not relevant (yet)
- *Presentation intelligence*: Besides the different influences for user, platform or application, there is general knowledge on how a presentation should be generated. This can include heuristic principles that guide the system in its generation process. An illustrative example of this intelligence is the intelligence that specifies how data are presented in terms of a web presentation.

Note that *data intelligence*, specific intelligence on the data that is not covered in the application intelligence (schema), e.g. their internal structures or their data types, is treated as an integral part of the data (subject of the generation process), and not as some explicit kind of "outside" knowledge: this in the spirit of semi-structured data.

Note also that the intelligence described above contains general intelligence. However, specific domains can have specific intelligence that represents the "best practice" for that domain. Usually, the intelligence will be a combination of *general* and *domain-specific intelligence*. Besides the domain-specific intelligence another kind of specific intelligence can be involved, namely the intelligence that is explicitly added to a query by the user and that is only applicable for this one query.

The applicability leads to another dimension of the intelligence:
- *Generic intelligence*: The main body of intelligence should be applied in every generation process. It is assumed to be included in every instantiation of the generation process.
- *Query-specific*, or ad-hoc, *intelligence*: Sometimes, intelligence may be entered by the user specifically for the current instantiation of the generation process. If this facility is provided to the user, then the assumption is that the (experienced) user has an insight in the use of the intelligence, and is able to specify specifically for this one query intelligence that extends or overrides the generic intelligence.

This dimension considers the functionality of the system that allow the user to provide query-specific, ad-hoc directives that guided or influence the system's generation process, for example the possibility to ask the system to use a guided tour and not the index that the intelligence would suggest. Note that the distinction between generic and query-specific knowledge is orthogonal to the different kinds of intelligence described before (and to the distinction between general and domain-specific intelligence). Also, the different aspects of intelligence have no relationship with the way in which this intelligence is obtained. Possibly explicit, through a user that specifies it in an appropriate interface, or implicit, through a part of the system that derives the intelligence by learning from the previous behavior. For this document this distinction between explicitly and implicitly obtained knowledge is outside its scope.

At the start of the project the modeling (interface) of the intelligence is not a main research topic. In order to restrict the scope, specifications of the intelligence are used as input for the generation process.

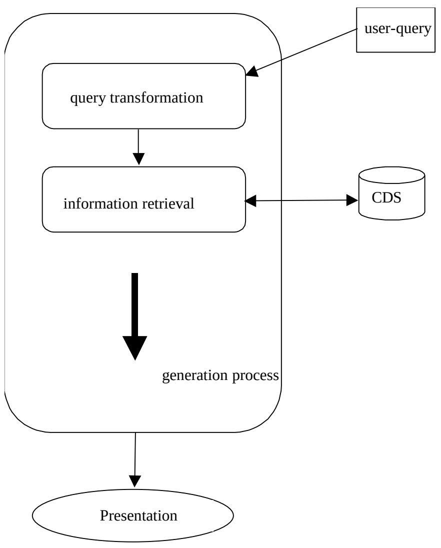
Data and meta-data

The generation process transforms data from CDS to data that are suited for the user's browser (as a presentation). Note that the actual generation process depends on the intelligence (described above). Here, we concentrate on the data (and meta-data) that play a part in the generation process.

CDS contains data, available for retrieval and subsequent presentation. Only a part of this data is retrieved for presentation. A *query* is a specification of data from CDS. The goal of a query is to specify which data should be retrieved from CDS and how these data should be organized into one query result.

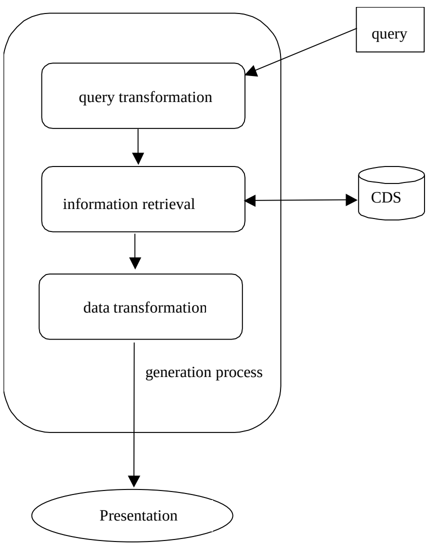
The generation process starts with a query that represents the data the users wants to retrieve (the user's information need): we call this the *user-query*. Note that for this definition of user-query it makes no difference whether this user-query is (somehow) specified explicitly by the user, or derived by the system (learning from the user interaction). For the moment we assume that the user-query is specified ad-hoc by the user. A consequence is our focus on *ad-hoc queries* that are minor deviations of standard queries, for which it is assumed that presentations have been designed and available: this implies that we do not specifically target a general purpose query facility.

The nature of the user-query in this process is such that the information retrieval process does not simply retrieve the data thus specified by the user. The generation process starts by *transforming* or adapting this query on the basis of the available, relevant intelligence. This means that, in several steps, the original user-query specification is transformed into a (query) specification of the data that are actually retrieved.



Both for the original user-query specification and the subsequent query transformation, it is necessary to have knowledge on properties of the data. Just like for traditional query mechanisms, the assumption is that some meta-data is available on the data. These meta-data capture the main properties of the data in a so-called *schema*. This schema specifies the conceptual contents by describing at the type-level which elements and relationships are available. It is important to note that we both need the structure or properties of the data (schema) and the semantics of the data to transform the query using the knowledge we might have about the user, domain, platform, network etc.

After information retrieval, when the retrieved data are available for further processing, the generation process continues. The data are transformed to a presentation based on the intelligence that is available and relevant. These transformation steps transform the data themselves, and do not just consider the meta-data but also the information contained in the data themselves. An example may be a filtering step to select only the data suited for the platform in use.



Note that the data transformation may be data store dependent. This means that the effects of a transformation, e.g. filtering, may depend on whether or not the data from that store is suited for that transformation. In the context of semi-structured data, the data from a data store may lack some structure: in that case some transformation may not be applicable.

The entire generation process thus consists of three main subprocesses:

1. *query transformation*
2. *information* or *data retrieval*
3. *data transformation*

Together, query transformation and data transformation construct the presentation for the data on the basis of the relevant intelligence. In between information retrieval is responsible for executing the query that results from query transformation to deliver the appropriate data to data transformation.

Both query transformation and data transformation could be done on a *top-down* or *bottom-up* basis or in combination. Part of the research will try to establish methodologies for the construction of presentations from the retrieved data objects. Note that it can depend on the domain which methodology is more suited.

For the specification of the transformation processes we need to specify the original query and its subsequent query transformations (as suggested by the intelligence), and the subsequent data transformations. This implies the need for a language or model for describing data (including all its internal structures) and transformations on those data.

This language/model consists conceptually of different parts:

- For the specification of the schema of the data, as relevant for the query formulation and transformation, we use a variant of (Extended) RMM's Application Diagrams. Their concept of m-slice allows for the specification of data and their hypermedia structures. The m-slice is very close to the abstract concept "concept" as used in earlier Dynamo discussions. This RMM approach is more specific than a model like RDF, which could possibly fit also. (Note that we could express this also in Dexter, for academic purposes.) Essential elements in this data model are concepts, concept relationships, describing and identifying values, and concept composition.
- For the formulation of the queries, we would need to invent a new language, that specifies operations on m-slices: a language for manipulating "concepts". Such a language would look like a complex object language or web query language. Possibly, we could even wait for a standard for a web query language. (Note that a language like XML-QL is a candidate for the practical aspects, but not (yet) a standard. Also, for academic reasons a special "slice manipulation language" targeted at our specific goals appears helpful.)
- Both query transformation and data transformation could be formulated in a new language, that allows for the specification of rules on how to transform a query into another query, or data into new data. The main difference is that with data transformation the data instances themselves can be considered, while query transformation only looks at the meta-data (schema).

Remember that query transformations lead to "new", "volatile" concepts that play a role within the presentation. These new concepts can play a role at the conceptual level (as a new concept in the contents), but they can also play a role in the presentation to represent some derived temporal or spatial object. It is an implementation issue which of these new concepts are materialized and which are computed on-the-fly.

Note that presumably the queries, data and meta-data will technically be expressed in XML, while currently XSLT is used to express the transformations.

Implementation layers

The description, given above, of the generation process covers only the conceptual part of the process. In terms of SRM this would cover the Control Layer, the Content Layer, and (possibly a large part of) the Design Layer. The remaining layers would also be present in our system architecture, but outside the process described above.

An issue that has not been mentioned before, but that does play a role here is the separation of server-side adaptation (computation) and client-side adaptation. While this may not be prominent at the start of the development of this conceptual architecture, this issue deserves attention. Perhaps we can include this infrastructure knowledge also in the generation process: then, we could dynamically change the infrastructure and the system would adapt.

Domain dependence

The target in this project is not to come up with a generic approach that is perfect for all applications. We have defined a class of target applications, e.g. real-estate databases, mail order catalogs, or electronic program guides: for these target applications we want to consider a generic approach. For some aspects it might be the case that we need to become more specific in order to be able to develop concrete prototypes. For those aspects we want to focus on a particular domain. Currently, we choose the *WebEPG* as that domain and therefore as the prime application for which prototypes are developed. It is important to realize that we mean EPG in the broad sense here: creative combinations between e-commerce, EPG, agenda etc. could be attractive illustrations of our work.

Note that a domain dependence can also be expressed in terms of “bridge laws”: they are rules that specify which relationships implicitly exist and when they are materialized (during the generation process) in order to offer the appropriate navigation mechanisms (e.g. hyperlinks). In an associated project like DHymE this approach is used to make domain-dependent decisions in the generation process.