Using Knowledge Networks to support Innovation

Henno Gous

Thesis presented in partial fulfilment of the requirements for the degree of

Master of Engineering Science

at Stellenbosch University

Study Leader: C.S.L. Schutte

March 2009

nature of the list. Items that should always appear at the top of lists, regardless of its age, should also be supported.

4.5. Information System Architecture

In Chapter 4.4 the requirement set presented in Chapter 4.3 is grouped and reworked, and this has produced a more functional view of the Information System Architecture that is illustrated in Figure 49. This view of the Information System Architecture requirements provides a multi-layer approach to designing the system, and will be used as a design guide in Chapter 5. This ensures that the entire Information System Architecture is implemented.

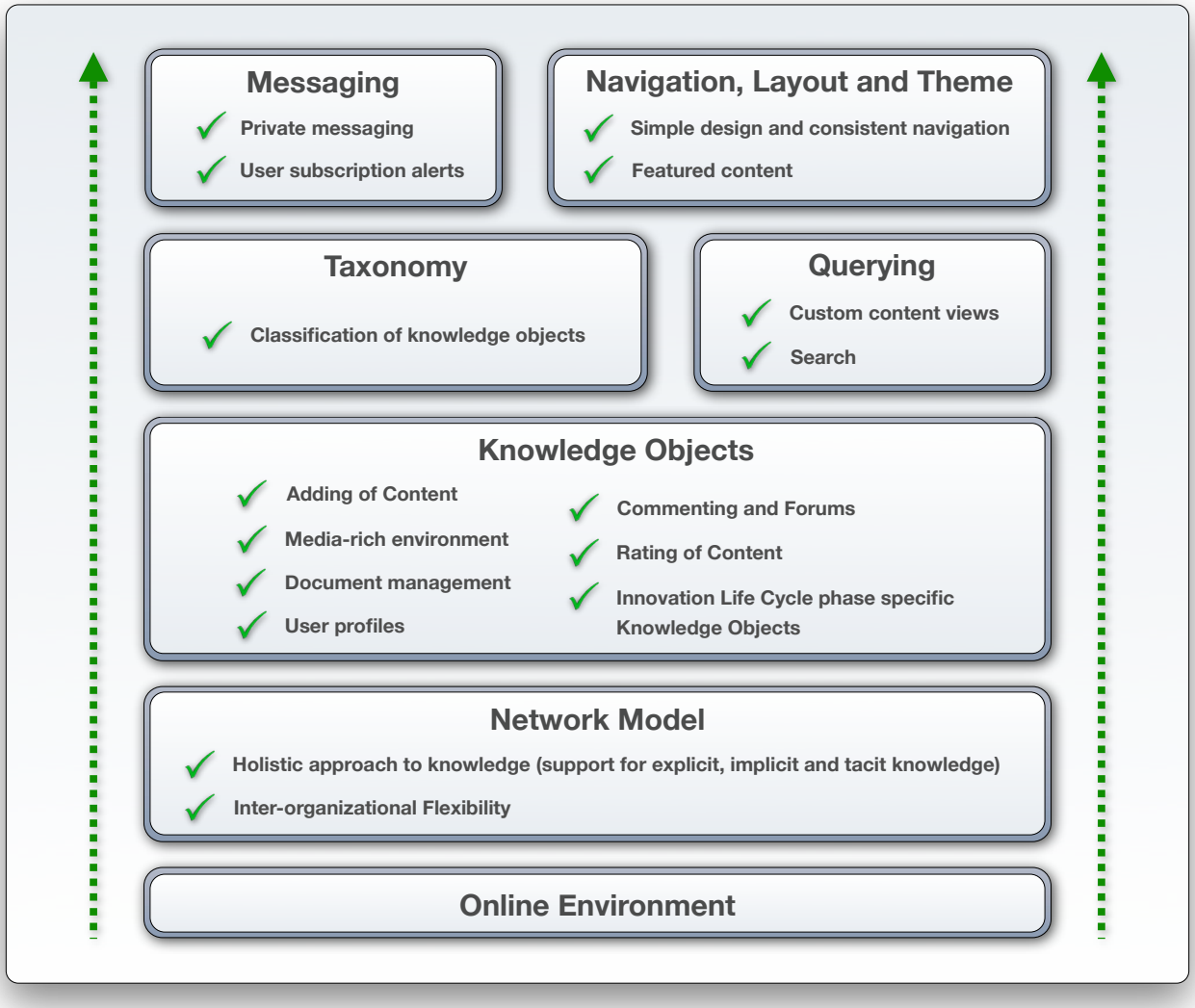

Figure 49 - Information System Architecture

Chapter 5

Design

SDLC Step 5: Design

"Transforms detailed requirements into a complete, detailed system design specification. Focusses on how to deliver the required functionality."

- Wikipedia

Chapter 5 uses the Information Systems Architecture derived in Chapter 4 as a guide for the design of the proposed solution. The technology overview in Appendix A is used as a reference for design decisions. Chapter 5 focuses on how to deliver the required functionality.

The multi-layered presentation of the developed Information System Architecture (as shown in Figure 50) will be used to guide the design process.

Figure 50 - Multi-layered representation of the developed Information System Architecture

Implementation of the Information System Architecture will be approached with six design layers:

1. Network model within Online environment
2. Knowledge Objects
3. Taxonomy
4. Querying
5. Messaging
6. Navigation, Layout and Theme

5.1. Network Model within Online environment

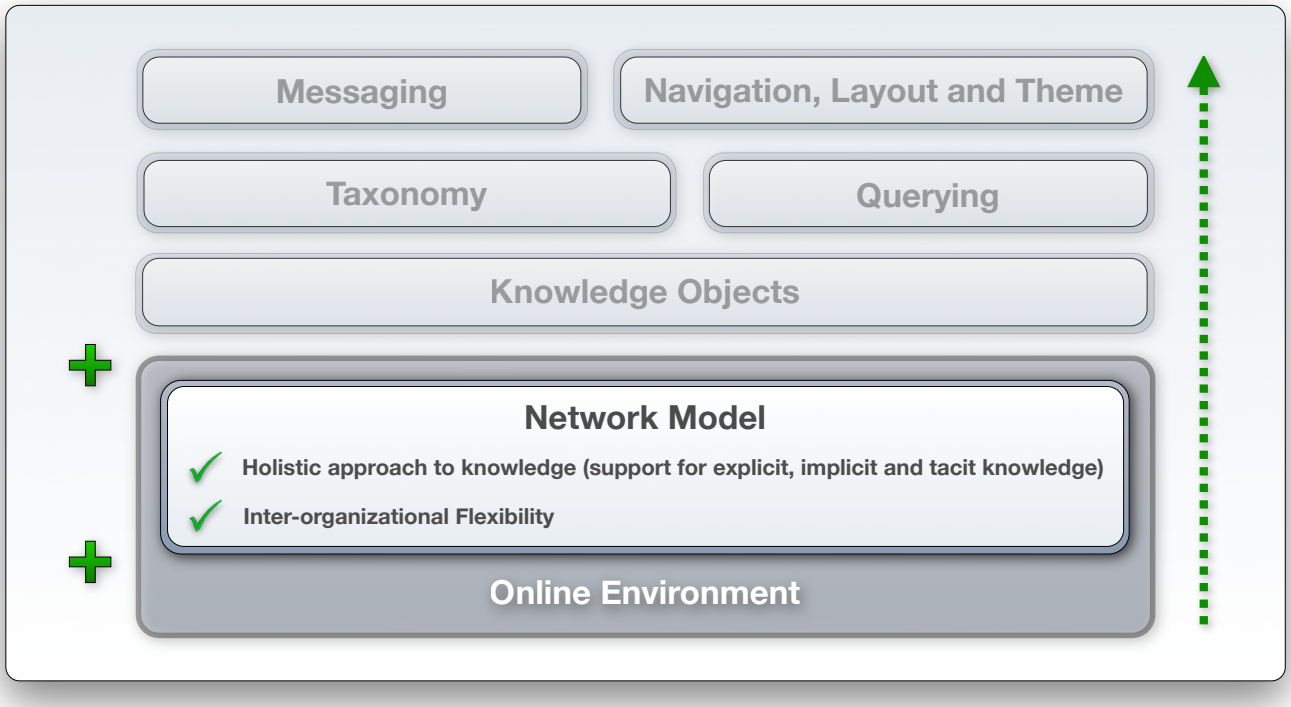

Figure 51 - Design layer 1: Network Model within Online environment

Functionality Specification: Online Environment

Addressing requirement 0, and stated in Chapter 4.4.1 as:

A web-based staging environment for the Information System, allows ease of access from anywhere in the world. By developing the system to be accessed with any standard web browser software, the need for specialized client software is eliminated, thereby making the Information System even more accessible.

The online environment furthermore allows it to be accessed with ease by users from a number of different organizations from around the globe, thereby enabling Integrated Knowledge Network support.

Functionality Specification: Network Model with a holistic approach to knowledge

Addressing requirement 1, stated in Chapter 4.4.2.1:

Designing the network model with consideration for explicit, implicit and tacit knowledge requires the handling of different types of knowledge objects within the network. Knowledge objects will therefore become the nodes of the network.

Functionality Specification: Network model with inter-organizational flexibility

Addressing requirement 3, and stated in Chapter 4.4.2.2 as:

Using a network design as the base of the system enables the system to take advantage of the positive attributes associated with networks, e.g. spanning across organizational boundaries as in the case of an Integrated Knowledge Network.. The network should furthermore be allowed to grow organically, link with other networks and connect nodes without any unnecessary interference.

The solution framework developed in Chapter 4.1.1 as the basis of the proposed Information System Architecture, combines the specifications stated above and poses the following requirement:

An online information system that simulates a network model of actors and relationships in the way it manages content.

Modern Content Management Systems take advantage of the Internet's remote connectivity options and deliver ways to manage users and content in a variety of fashions. The desired network model where actors (users and content) are nodes within a network may be achieved in this way, thereby satisfying the requirement set out by the solution framework stated above.

Several of these Content Management Systems are open-source and available free of charge (refer to a review and comparison of these systems in Appendix A). A vibrant developer community that contributes extensions to the core system supports most of these systems. Choosing the correct Content Management System as basis and extending its functionality with contributions from its developer community will deliver the entire functionality needed to implement the Information System Architecture that was developed in Chapter 4. Extendable Content Management Systems furthermore allow web designers to construct complex web-applications without the need for expert programming knowledge.

The first and most critical requirement is for a network model (refer to Chapter 4.4.2) within an online environment (refer to Chapter 4.4.1) that supports dynamic use of knowledge objects. The solution chosen for this problem is important, as it will influence the solutions to every subsequent system requirement (refer to Figure 29). Choosing WordPress (refer to Appendix A, 10.7.1) as a content management base would mean that only WordPress-compatible technologies could be considered to meet any other requirements. The same argument goes for Joomla (refer to Appendix A, 10.7.2) and Drupal (Appendix A, 10.7.3) as content management bases.

Drupal stores information as units of content normally referred to as ‘nodes’ (refer to Figure 52). Unlike Joomla and WordPress, Drupal does not by default impose any hierarchy or structure on these content units. The approach to content storage is dynamic, with nodes being treated as a “mixed soup of information” and custom queries being used to extract the relevant content on demand. This makes Drupal’s handling of content ideal for the network-based data structure that is needed to manage a knowledge network.

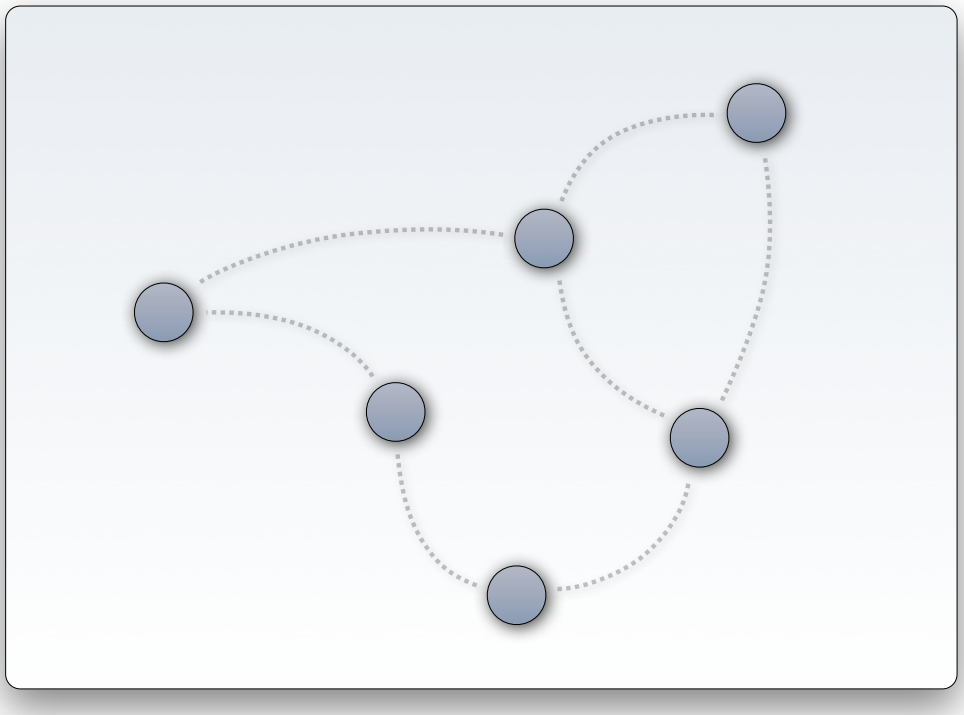

Figure 52 - The Network Model lays the foundation for the Information System

Although knowledge object design forms a separate design specification, ensuring that the content management base can indeed support knowledge objects within a network, also influences the choice of Content Management System. Drupal offers the possibility to store nodes in a number of different ‘content types’. This allows the knowledge network to consist of separate nodes or knowledge objects, which may differ in associated metadata fields, workflow options, etc. Users, documents, discussions or any other object type can thus populate the network and be treated as peers. When distinctions need to be made by object type, topic or any other metadata, a content query can be launched which will return the relevant objects from the data structure.

These arguments present an overwhelming case for [1]**Drupal** as the content management system that will be used to implement the network model that is required. Any subsequent choices for technologies to be used as a part of the system design therefore need to be compatible with Drupal.

Over and above offering the necessary functionality to implement the required network model, the Drupal core distribution does however also include a number of modules that are ideal for meeting several other requirements of the proposed Information System Architecture.

To install and configure Drupal, a **webserver** is needed that supports the PHP programming language and mySQL, mySQLi or pgSQL databases.

5.2. Knowledge Objects

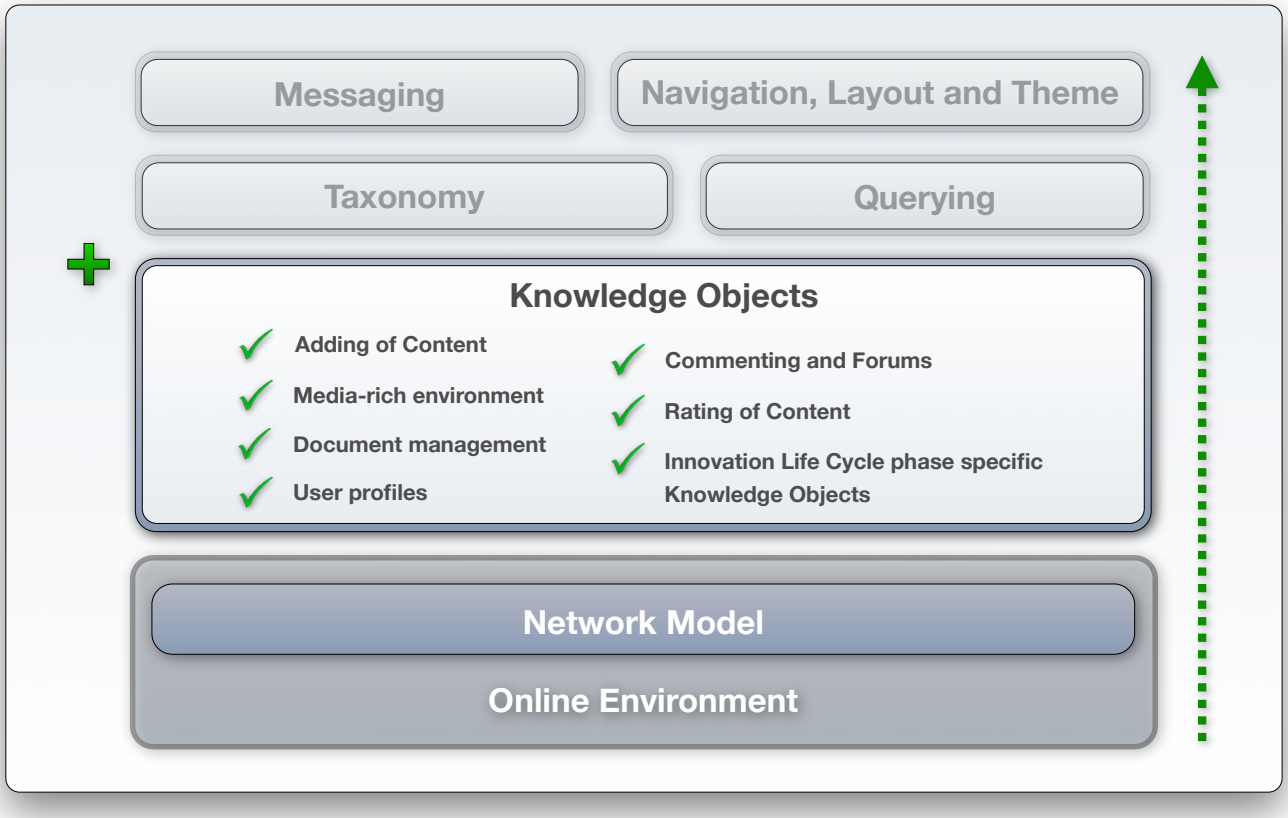

Figure 53 - Design layer 2: Knowledge Objects

1 To enhance clarity in this section all technologies, systems and modules that are approved for use in the development of the proposed Information System, will be printed in bold type.

Functionality Specification: Knowledge Objects

Addressing requirement 2, and stated in Chapter 4.4.3 as:

Knowledge objects form the nodes of the knowledge network and it is essential for the successful implementation of the network model that different knowledge object types are handled elegantly. This implies that users, documents, content posts, ideas, concepts, projects and discussions should all be treated as peers on a data structure level.

Support for knowledge object requirements builds on the initialization of a network model for content management (refer to Figure 52), and entails the ability for different types of content to form nodes within the network model (refer to Chapter 4.4.3). On a functional network level, no distinction must therefore be made between users, content items, ideas, etc. in terms of function. All objects should be easy to add to the network, be compatible with the taxonomy for integration in the knowledge base and be able to establish links with other nodes. Where applicable, users should be able to comment on a node or add a rating.

Providing for all the necessary knowledge object types and allowing for the described functionality will be done as follows:

5.2.1. Adding of Content

Functionality Specification: Adding of Content

Addressing requirement 7, and stated in Chapter 4.4.3.1 as:

Knowledge objects are crucially important to the functioning of the system and it must be easy to add them to the network. A variety of options should be available to contributors to store the knowledge they want to externalize.

Along with treating content as nodes, Drupal has the capability to create and distinguish between a number of different content types (refer to Figure 54). This capability will be used as a starting point in the design of different knowledge objects types. Drupal has built-in support for a number of content types and also provides a simple interface for creating new instances of these types. The design and creation of custom knowledge objects that are not one of these types will be discussed in Chapter 5.2.2.7.

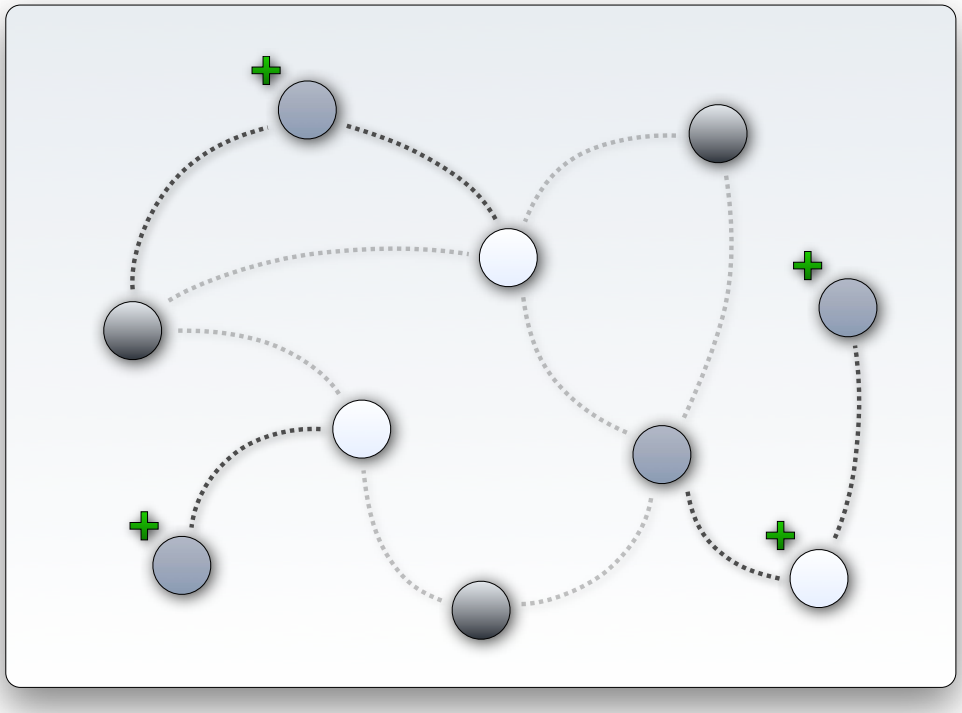

Figure 54 - Knowledge Object support is enabled by allowing nodes of different types to be added to the network

Drupal content types differ in their overall anatomy, but do share a number of common features. These features may be customized for each type, which will automatically influence the corresponding options on all content instances of the specific type. Content items may however also be customized per node, and in this sense will not influence other nodes of the same type.

The author of a node has the option of deciding whether the new content has to be published right away or not. Leaving content unpublished allows the author time to revise and preview his content before allowing other users to see it. This process also allows a moderator to make changes or approve content before it appears on the platform itself.

An extremely powerful feature of the Drupal core's handling of content is its ability to handle version control. Upon enabling the Tracker module, content types can be configured to create new versions each time a node is edited. Selective access control then allows administrators to decide whether they want previous versions to be publicly visible and to which users they wish to grant the power to revert nodes to previous versions.

Authors have the option to promote their content to the front page and to force their items to be displayed at the top of lists. Both these options are important when configuring the presentation of

content along with graphical design and navigation structure to create an efficient interface. Enabling the Upload module gives the very useful option of attaching multiple files to content postings.

The default content types that are configured along with the installation of the Drupal core are the [2]*story* and the *page*. The *book*, *blog* and *poll* content types are also included in the distribution, but should be enabled manually. All of these content types have a number of useful built-in workflow options. It is important to keep in mind that these options are associated with the structure of the content type as a whole and are not limited to specific pieces of content.

The *story* is the simplest content type, consisting of a title, teaser and body. The *story* content type is an ideal content type for simple externalization, as it offers few barriers to easily contributing content to the knowledge base. By default, stories are published immediately, promoted to the platform front page and allow attachments. By opting to not publish a story immediately upon submission, the item can be held back for moderation purposes.

The Blog module does in essence not add a new content type to the system, but makes a copy of the *story* content type and calls it a *blog* content type. Blog postings therefore have the same main characteristics as stories. Enabling the Blog module does however also add the functionality to group and display, in reverse order, all the nodes of the *blog* content type that a single user has contributed, thereby creating a personal blog for each user in the system. Blogging is one of the primary externalization tools in the platform, as it gives users the opportunity to express their thoughts and thereby share their implicit knowledge. Constant updating of user blogs ensures that the platform represents a sustained, vibrant, knowledge network. Extensive externalization also gives an indication of the user's knowledge base – giving valuable insights into his tacit knowledge profile.

The *page* is also a very basic content type, but is used to create static pages that are not by default promoted to the front page. These static pages are normally used to convey content that will stay relevant for a longer period of time than content captured in an item of the *story* content type, e.g. "Contact us" or "About". Pages also allow attachments and can be published immediately.

The Book module creates a content type with the same name and is ideally suited to collaboration. Content items of the *book* content type consists of a set of pages (not to be confused with the *page* content type) that are connected in a specific sequence. This content set can be structured in a number of ways and may contain any number of sections and sub-sections. Coupled with the flexibility of the structure, a variety of access controls for user permissions makes the *book* content type ideal for combined externalization and loosely emulates the functionality of a "wiki". There are no

2 During the discussion of content types, the names of content types that should be implemented will be printed in *italics.*

constraints on the number of users that can collaborate on creating and maintaining a book content item. Permissions assigned to specific users include the ability to author, review, modify and rearrange pages within the book.

Books are also equipped with an efficient internal navigation structure. A contents page is automatically created and links to next and previous pages, as well as to the above level in the structure are included at the bottom of each book page. The structure of the book itself is based on parent-child relationships between nodes, which may be altered by users with sufficient permissions. Nodes within the structure also exist autonomously, allowing users to visit any page in the book directly. Nodes of other content types than book pages may also join the structure, which makes the *book* content type even more effective as a tool for collaborative externalization.

The Poll module introduces simple polls to the network as individual objects of the *poll* content type. Polls typically consist of a question and a number of options for answers. In the context of Knowledge Work Processes, the poll is a tool that can be used to simulate externalization on a group level by determining the collective opinion on a topic. Polls can also be grouped to form a more comprehensive quiz on a topic. Poll results can be made publicly available when it is deemed appropriate and could stimulate a user to externalize implicit knowledge on an individual level.

The ability of a basic installation of the Drupal core to handle such a variety of content types goes a long way to meeting the platform requirements for knowledge object support. This validates the decision to choose Drupal as the Content Management System used to form the backbone of the proposed system.

5.2.2. Media-rich environment

Functionality Specification: Media-rich environment

Addressing requirement 7, and stated in Chapter 4.4.3.2 as:

A media-rich environment is necessary to allow users freedom of expression. This implies that users should be able to embed images, audio and video in their contributions and have the option to attach files to nodes as well.

Drupal has built-in support for the creation of a media-rich environment for users to experience freedom of expression during externalization (refer to Figure 55). This support is packaged in the form of input formats that are used to interpret input provided by users. There are four default options for input formats:

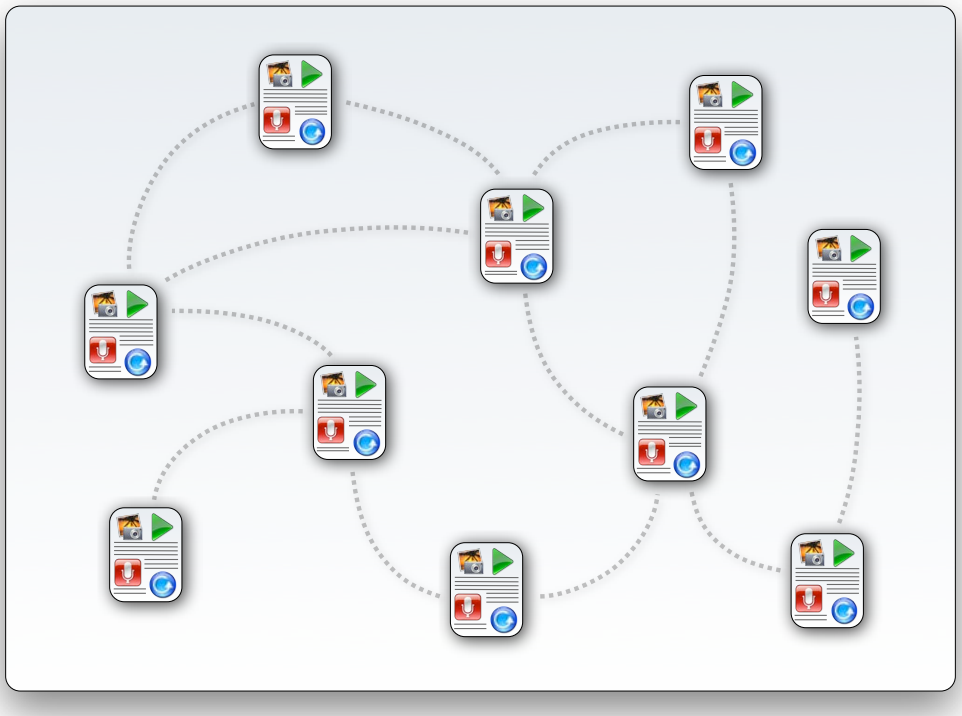

Figure 55 - Creating a media-rich environment transforms knowledge objects

a) Filtered HTML

Users may use a limited selection of HTML-tags in their input to bold or italic text or to embed images, audio or video items. Any tags that are not in the selection will not be interpreted and will be rendered as plain text output. The tag selection may be narrowed or expanded by administrators.

b) PHP code

Users may enter full PHP code as part of their contributions. This allows for an extremely dynamic environment, but could potentially raise security concerns.

c) Full HTML

Users may use the full set of HTML-tags in their input and this offers an extremely rich environment which is more secure than that produced by using PHP code as an input filter.

d) Messaging plain text

No markup tags will be interpreted and all text input will be rendered as-is. Offers a secure but media-void environment.

Given the nature of the content that will be presented on the platform, an environment is needed that is secure but offers all the necessary media options. An expanded version of the Filtered HTML input format provides this option. This however raises the problem of users not being accustomed to HTML tags and not being able to take advantage of the media-rich environment that is at their disposal.

This problem is overcome with the implementation of a WYSIWYG (What You See Is What You Get) editor in the Drupal content-adding interface. These editing interfaces offer a familiar environment that resembles popular word-processing interfaces, e.g. that of Microsoft Word. Users are then free to tailor their content in the editing interface, which inserts the appropriate tags into the contributed content, in order to achieve the output that the author had in mind, before submitting it to the content management system. There a number of these editors available, all of which offer a balance between simplicity and versatility. TinyMCE is a popular example of such an editor, and is easily embedded in the Drupal interface. The TinyMCE interface can be configured to include a variety of functions, each linked to a specific kind of tag that can be inserted into the content. Synchronizing the functions that TinyMCE offers with the tags that the Filtered HTML input format recognizes, creates a contribution environment that is rich in media-support, but offers very few barriers to entry.

5.2.3. Document management

Functionality Specification: Document Management

Addressing requirement 2, and stated in Chapter 4.4.3.3 as:

Documents are key knowledge objects in knowledge networks and should be able to form autonomous nodes within the proposed network model. This will create a repository of explicit knowledge that needs to be integrated with the knowledge base.

Drupal does not provide a document management facility as part of the core configuration, but it is a function that can be simulated with relative ease. Creating a *document* content type, which is primarily used to attach documents to, will solve the problem (refer to Figure 56).

Along with including several default content types, the Drupal core allows administrators to create new custom content types. Newly created content types include a number of parameters that will be used to form the basis of the functionality of the *document* content type. When submitting a new document to the platform, a user will have the opportunity to add a title and body text to the node, as well as options to publish immediately, promote to the front page or attach a file. The Drupal interface for adding content to the system includes several text areas that may be used to display submission guidelines, and may differ between content types. The node submission guidelines for items of the *document* content type are therefore customized to encourage the user to paste the document abstract in the node body and to ensure that the proposed contributed document is indeed attached to the node.

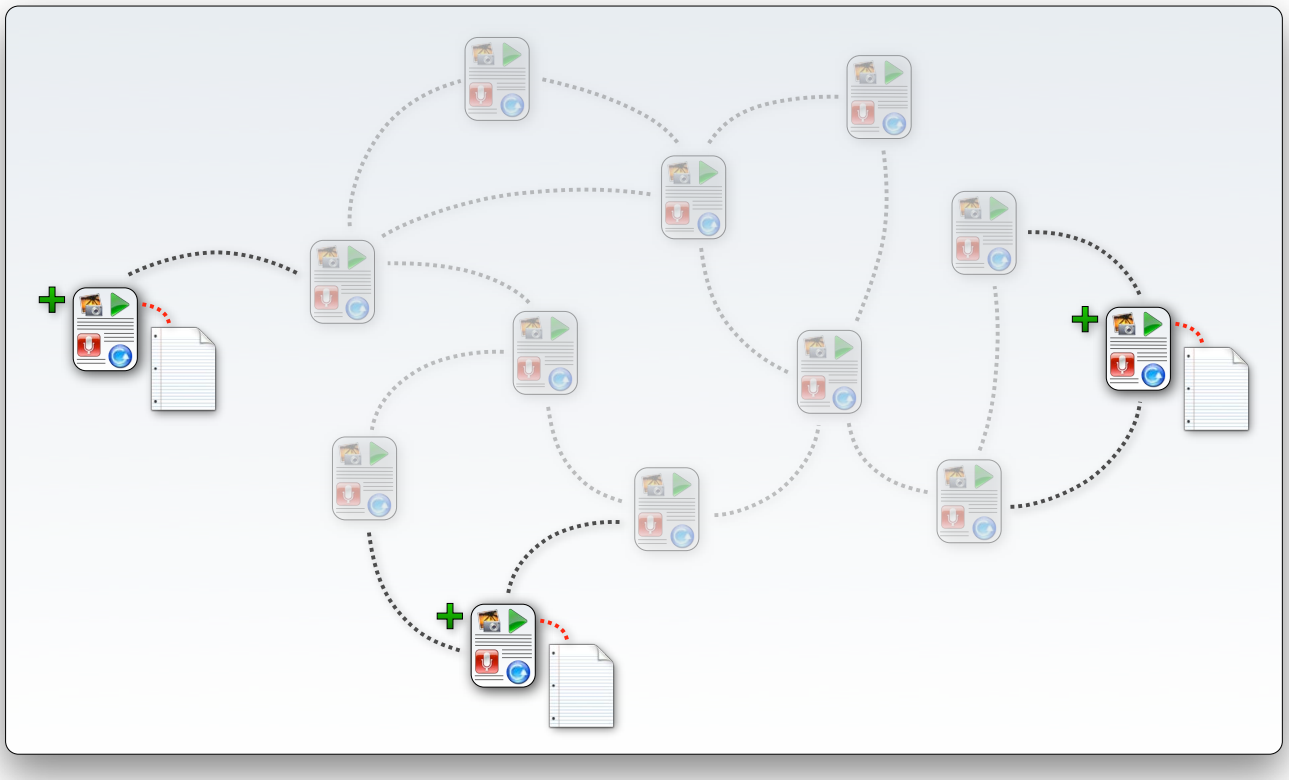

Figure 56 - Documents are introduced to the information system as knowledge objects

Enabling the option for creating new revisions of the node each time an edit is made, means that the document management function is not only effective in presenting finished documents, but offers valuable support for document development and version control.

Using nodes as carriers for documents within the knowledge network instead of using a standard document management approach provides the advantage of documents being treated as standard

knowledge objects within the knowledge network. Submitting an appropriate query to the Drupal system (refer to Chapter 5.2.4) can however provide a list of all the documents within the system. The query can also be customized to incorporate some of the custom parameter fields that were appended to the content type, e.g. requesting a list of documents by a certain author and to show the output in reverse chronological order. This shows that the approach to document management within the Drupal system is not only appropriate in terms of knowledge object support, but offers power and flexibility in managing the documents themselves.

5.2.4. User Profiles

Functionality Specification: User Profiles

Addressing requirements 2 and 6, and stated in Chapter 4.4.3.4 as:

Users should be able to join easily and maintain their profiles and biographies, thereby creating and updating their own nodes in the network. Users will also be able to use their accounts to manage their communication and alerts within the network.

Drupal includes a module called Profile in its core distribution and it creates user profiles that have a number of uses within the system. User profiles are used to provide each user with a username and password with which they can log in to access content that is invisible to anonymous guest users. They are also used to track user activity within Drupal, e.g. to determine how often a user visits the platform. The Profile module can also be used to configure the structure of profiles by specifying fields that users can complete to provide information on themselves. These fields are of generic types and can be configured to capture birthdays, qualifications and business background. User profiles will also be used to coordinate communication with the system and between users in Chapter 5.2.5.

Access to content can be restricted in a much more sophisticated manner than by only denying access to anonymous visitors to the platform. The Profile module introduces the option to define a number of user roles to the platform. Permissions are assigned to each of these roles by the platform administrator. Most of the modules that are installed in the Drupal system have a number of permission options assigned to them, allowing for an intricate design of custom privilege sets. As roles are assigned to users, a number of user groups start to appear within the user community. In the discussion of the Book module (refer to Chapter 5.2.2.1), in which a user may have the option to add new book pages, but cannot create whole new books or add other content types than book

pages to the book structure. Drupal therefore offers a powerful built-in access control function that is lacking in many other content management systems, e.g. Joomla.

A system architecture requirement unique to the specifications of the proposed platform is the handling of users as knowledge objects, thereby integrating them as nodes within the network (refer to Figure 57). This need is resolved with the implementation of the **Bio** module to compliment the built-in Profile module. The Bio module creates a custom content type and links a singular content item instance of the *biography* content type to each user's profile. This results in a system that resembles the solution that was implemented to simulate document management within the system. The biographical content item associated with the user profile is compatible with other content in the system and effectively integrates the user profile with the system content. Given its application, it is logical that a user may only maintain a single biography item. Biographies are created upon user registration and the submission guidelines may be customized to encourage users to specifically describe their self-perceived knowledge profile in their biographies.

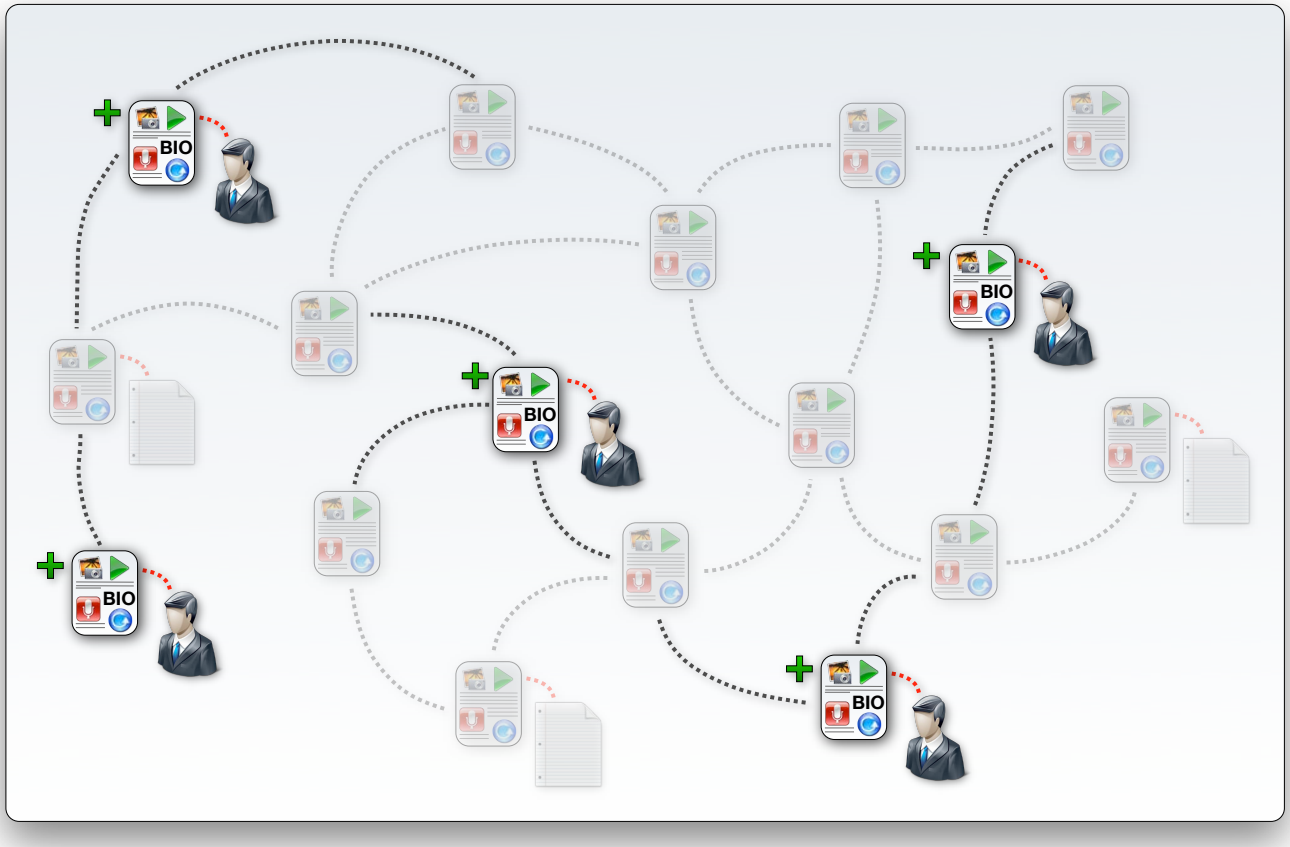

Figure 57 - Users are introduced to the information system as knowledge objects

The Bio module has a number of configurable settings, including the option to show a link to the biography of the author of any content item that is applicable. This is useful, as it creates a closer link

between a user's self-maintained knowledge profile and the knowledge that he or she externalizes by contributing to the knowledge base. The module can also be configured to take over the user profile, which will lead to only the biography item being displayed on the user's profile page. This is however not prudent, as the parameter fields defined by the Profile module will provide valuable insight into the user's background, and can be used to put the biography content into context.

5.2.5. Commenting and Forums

Functionality Specification: Commenting and Forums

Addressing requirements 5 and 8, and stated in Chapter 4.4.3.5 as:

The platform should allow users the option to comment on knowledge objects within the network. These comments should be linked to the objects themselves, as discussion on an object enriches the object. It may also happen that discussion develops without it being based on a specific node in the network, but is rather stimulated by a topic outline – in this case the discussion becomes an object in itself in the form of a forum discussion.

The Drupal core includes a Comment module, which provides a powerful commenting model for enabling discussion on published content (refer to Figure 58). This module is versatile and can be configured to suit specific workflow needs. A prominent feature of the Comment module is that it allows for threaded comments, a view which keeps replies together to help the reader to follow the discussion. Comments can be organized according to date, with an option to display the oldest or newest comments first.

Commenting is a secondary form of externalization, and the same media-rich environment that was created for adding content should exist in the commenting interface. Specifying a Filtered HTML-setting as input format and allowing the TinyMCE editor to be displayed when adding comments is therefore necessary. Comment functions can be controlled on a user access level by setting the appropriate permission combinations according to user roles. Administrators may for instance allow only users of a specific role to post comments without moderation, with comments from all other users forming a moderation queue waiting for approval. Comment settings can additionally be configured for each content type, e.g. allowing comments on blog entries, but not on user biographies to prevent misuse, irrespective of the role of the user (refer to Figure 58).

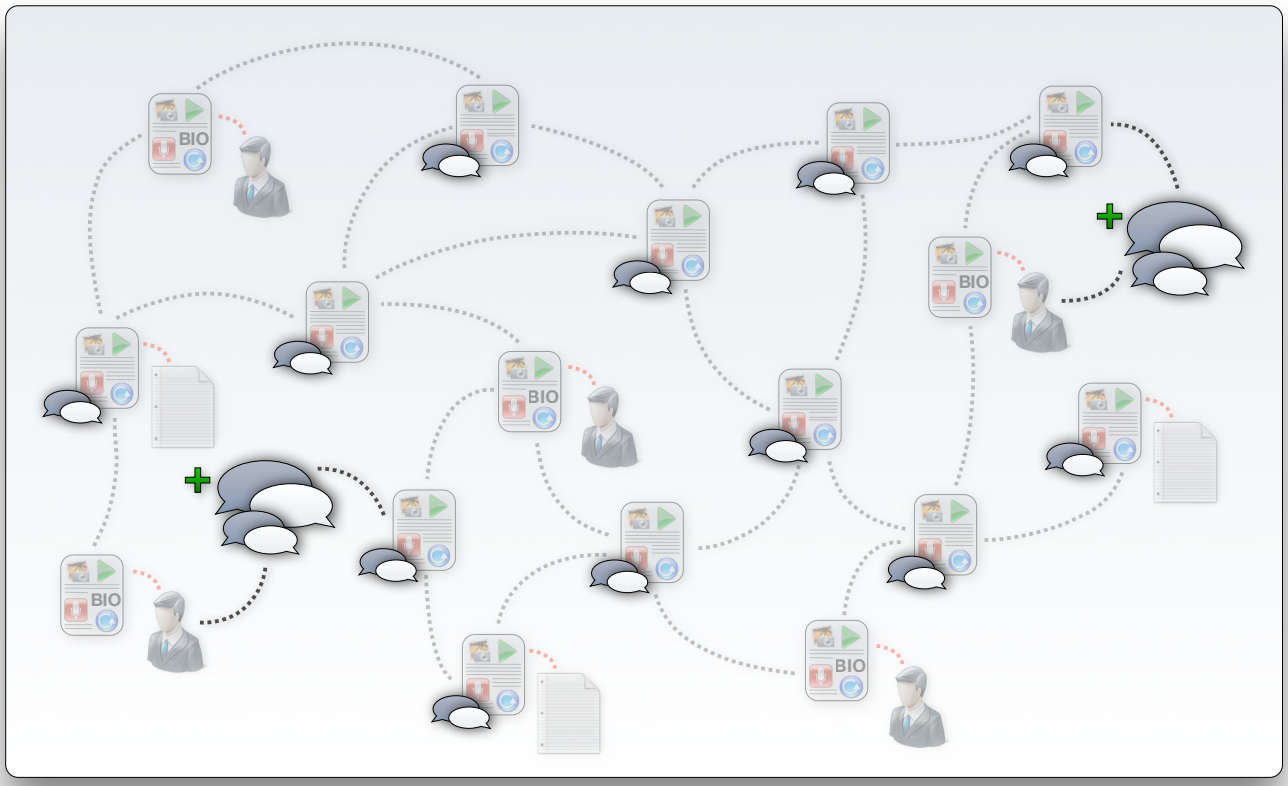

Figure 58 - Forum discussions are introduced to the information system as knowledge objects and commenting is selectively enabled on other objects

Discussion in forums is also supported natively in the Drupal core with the **Forum** module. Enabling the Forum module creates a *forum* content type specifically designed to manage threaded forum discussions. Including a specific content type for forums results in discussions being included in the network as knowledge objects. Drupal forums can be organized using “containers”, offering the opportunity to build a forum structure that resembles the taxonomy scheme of the network knowledge base. Forum discussions have the ability to employ the Filtered HTML input format along with the TinyMCE editor to take advantage of a media-rich environment.

Permission control to forum participation is managed similarly to the approach used for comments. Privileges are granted per user role and include the ability to add or edit a topic, as well as participation in the discussion.

5.2.6. Rating of Content

Functionality Specification: Rating of Content

Addressing requirement 9, and stated in Chapter 4.4.3.6 as:

Allowing users to rate content gives them the opportunity to express their opinion on knowledge objects in a quantitative fashion. Rating should not only be permitted on written content, but should be extended to documents, ideas, concepts, etc.

Drupal does not offer native support for the rating of nodes, but a wide array of modules are available to add this function to a Drupal installation. The **VotingAPI** module does not in itself create a front-end voting application, but allows other modules to take advantage of the framework in which it captures and stores voting information. A module like **Fivestar** links with this framework and creates a voting widget that can be configured and displayed to allow users to easily rate content.

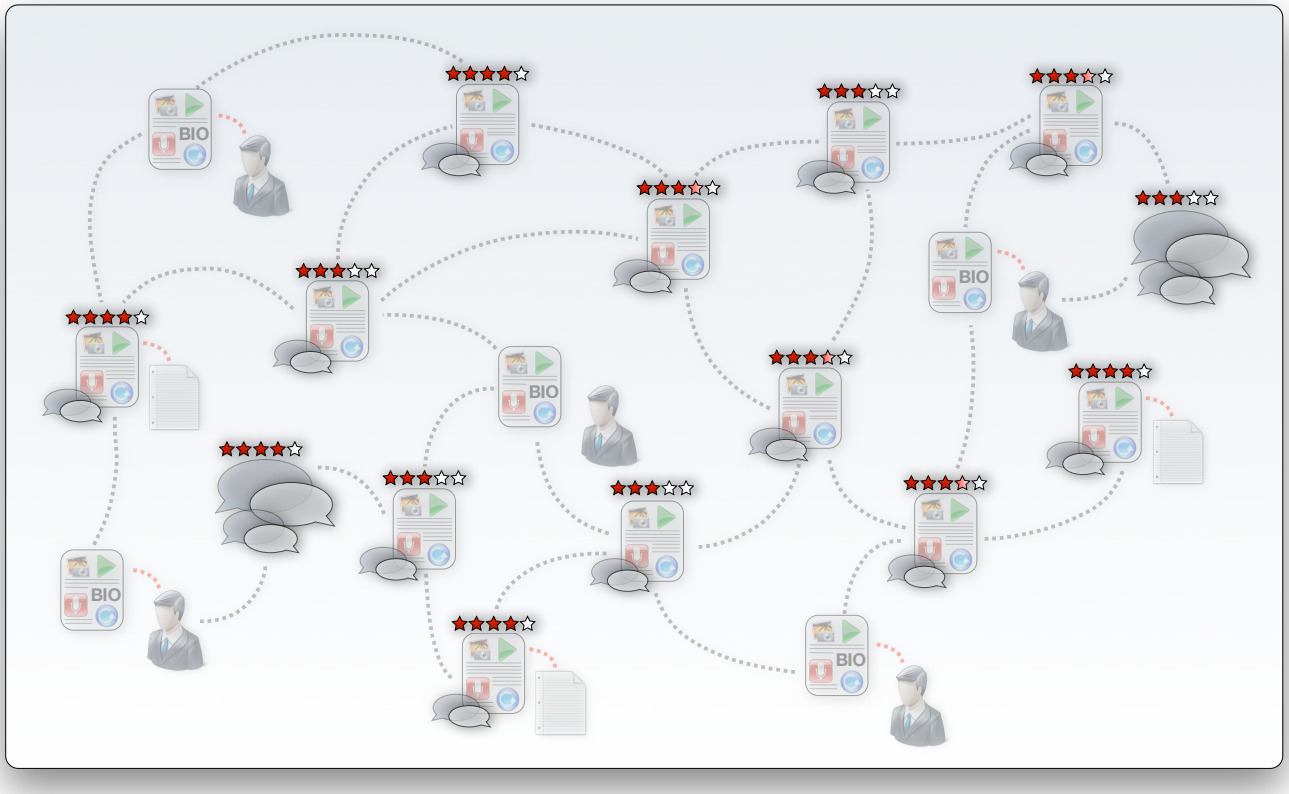

Figure 59 - Rating capability is enabled on selected knowledge object types

Fivestar includes global settings that govern the look-and-feel of the rating widget on whichever node it appears. Access to the rating widget is controlled in the same fashion as the access control employed for commenting. Role-based access control combines with selective enabling of rating for

certain content types. As was the case with commenting, rating will be disabled on user biographies to prevent misuse (refer to Figure 59).

Fivestar settings per content type are however not limited to enabling or disabling rating of content types. Various options are available and include the number of stars to be used in the widget, whether text values should be associated to ratings to give context to the mark. The widgets may also be configured to display in a static format on article teasers, as well as to display the currently logged in user's rating for an item along with the user community average when a node is being displayed in full. Users are also free to undo and edit their ratings, as the context of knowledge objects may change over time and along with it users' perception of the knowledge captured in the object.

Employing the VotingAPI module to create the framework for the storage of rating information allows other modules to access the results. This will be exploited in the design of querying functions to extract more value from the knowledge base (refer to Chapter 5.2.4).

5.2.7. Innovation Life Cycle phase specific Knowledge Objects

Functionality Specification: Innovation Life Cycle phase specific Knowledge Objects

Addressing requirements 16, 18, 19, 20, 21, 22, 23 and 24, and stated in Chapter 4.4.3.7 as:

Ideas, concepts and projects should be represented as knowledge objects within the network, thereby integrating them with the knowledge that was used to generate them. Each phase of the Innovation Life Cycle also has specific information needs and custom knowledge objects must be designed for this purpose.

The knowledge objects that are designed to represent concepts and projects during the Innovation Life Cycle should not only be singular objects. The needs of these phases of the life cycle are of such a nature that a group of network nodes should be combined in a dynamic and organic way, with a central node being used to bind them together.

Throughout the Innovation Life Cycle, all knowledge objects that represent ideas, concepts or projects should clearly indicate their current development status. This will help to guide development along the generic innovation roadmap.

Three custom knowledge object structures have to be designed to offer support for the Innovation Life Cycle within the knowledge network (refer to Figure 60). The idea is the simplest of these three object types and will be discussed first, followed by the concept and project, which are more complex.

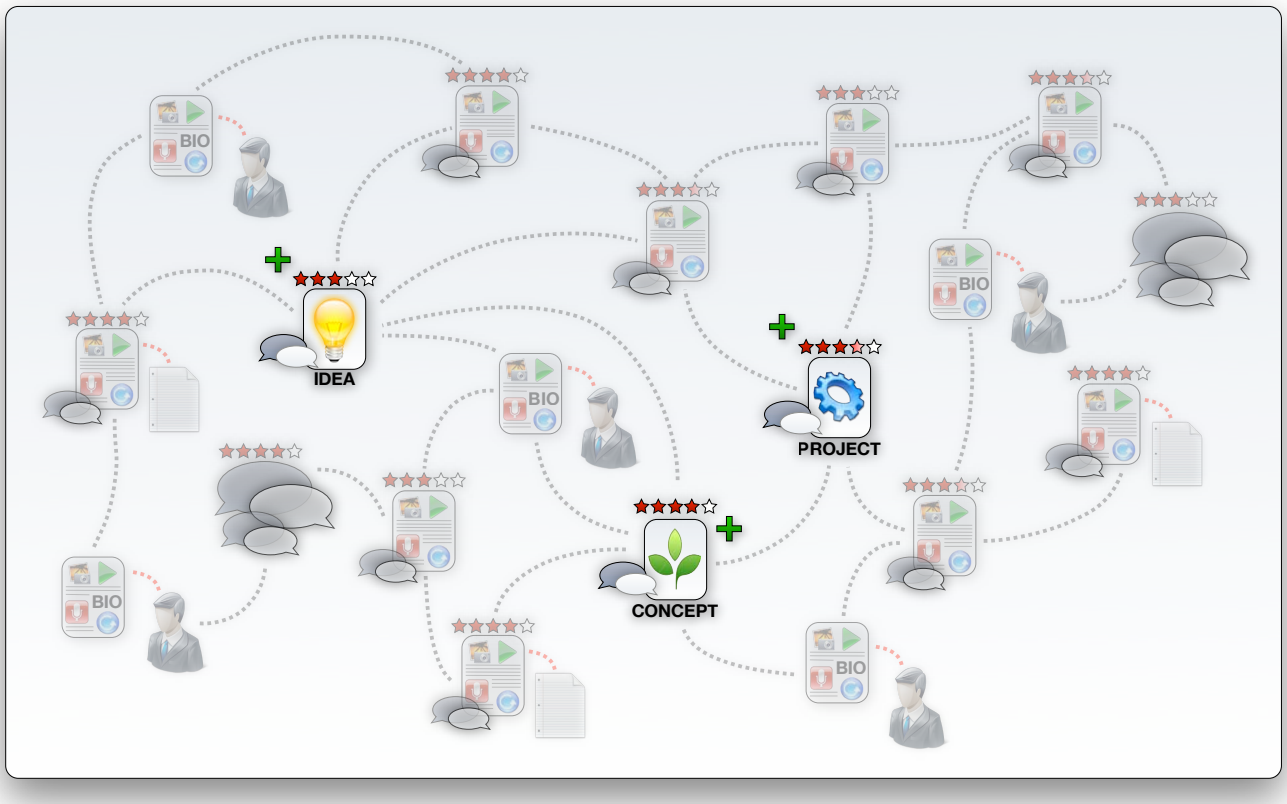

Figure 60 - Custom knowledge objects are introduced to the network to support the Innovation Life Cycle

5.2.7.1. Idea Object

The Drupal core is used to create a new content type that includes a number of default parameter options, which will be used to form the basis of the functionality of the *idea* content type. The interface that is used to create new idea postings within the system also serves as the idea capture form. The importance of this form capturing all the necessary information regarding the idea has been discussed. The option to specify text to be displayed as submission guidelines on the capture form should therefore be exploited to remind the contributing user of the importance of completeness in capturing his or her idea.

As was the case with other features supporting the externalization process, it is important to provide the contributing user with a simple interface that is user-friendly but powerful in terms of media-

richness. Enabling a powerful input format and the TinyMCE editor, as well as the ability to attach files to the idea posting creates an environment that allows for user expression.

Sufficient metadata regarding the idea posting should be captured, and the Drupal core automatically saves information on the identity of the author and the date of creation. Installing the Content Construction Kit (CCK) module provides the ability to add any number of additional parameter fields to the basic structure of a content type. There are several options for parameter field types, including text areas, selections, URL's, images and even references to other nodes within the system. In the specific case of an idea posting, it is important to capture information on other knowledge network nodes that may have stimulated or influenced the generation of the idea, thereby developing a better understanding of its context. Contributing users will be able to select relevant nodes from a number of conditional lists that will each only contain nodes of a certain type. Separate lists will be created for related content, documents, users, ideas, concepts and projects.

As ideas will ultimately be integrated with the platform knowledge base, postings will have a protracted lifespan within the knowledge network. The CCK module will be used to create a development progress indicator that will help to distinguish developed ideas from undeveloped ones. This development progress indicator will reflect all the main generic development states as proposed by the Innovation Life Cycle.

Contributing users may often need some time to clarify their own thoughts on their ideas and therefore they will have the option to leave their idea posting unpublished. Contributors will also have the ability to edit their idea postings after publication if they feel the need to do so.

Idea postings will attract a lot of attention among users browsing through the knowledge network and often these users will want to comment or rate ideas. The Comment and Fivestar modules should therefore be enabled for the idea content type, to allow commenting and rating respectively.

5.2.7.2. Concept Object

The concept is a much more complex knowledge object than the idea. It combines a number of other objects to represent them as a single entity, while still allowing access to the separate objects. The Organic Groups module takes advantage of Drupal's dynamic approach to content storage to achieve exactly this result.

The Organic Groups module allows users to create and manage their own 'groups' within a Drupal website's user community. Each group can have subscribers, and together these subscribers may maintain a group home page where they can communicate amongst themselves. Subscribers may post any content types that are allowed by the group setup into the group. These posts are displayed

on the group home page, which also displays summary information on the group. Several security configurations are available for groups and vary from free, open groups to which users may subscribe without any moderation, to invitation-only groups to which access is controlled strictly. Access to group content is also controllable, allowing group moderators to decide whether their group's content is to be displayed publicly on the site, or whether it will be available to subscribed group members only.

The way in which the Organic Groups module allows a number of content nodes within a Drupal environment to combine to form a group, makes it possible to exploit this functionality to design the knowledge objects needed for the concept and project regions of the Innovation Life Cycle (refer to Figure 61). Organic Groups allows nodes to combine to form a new object without losing their own

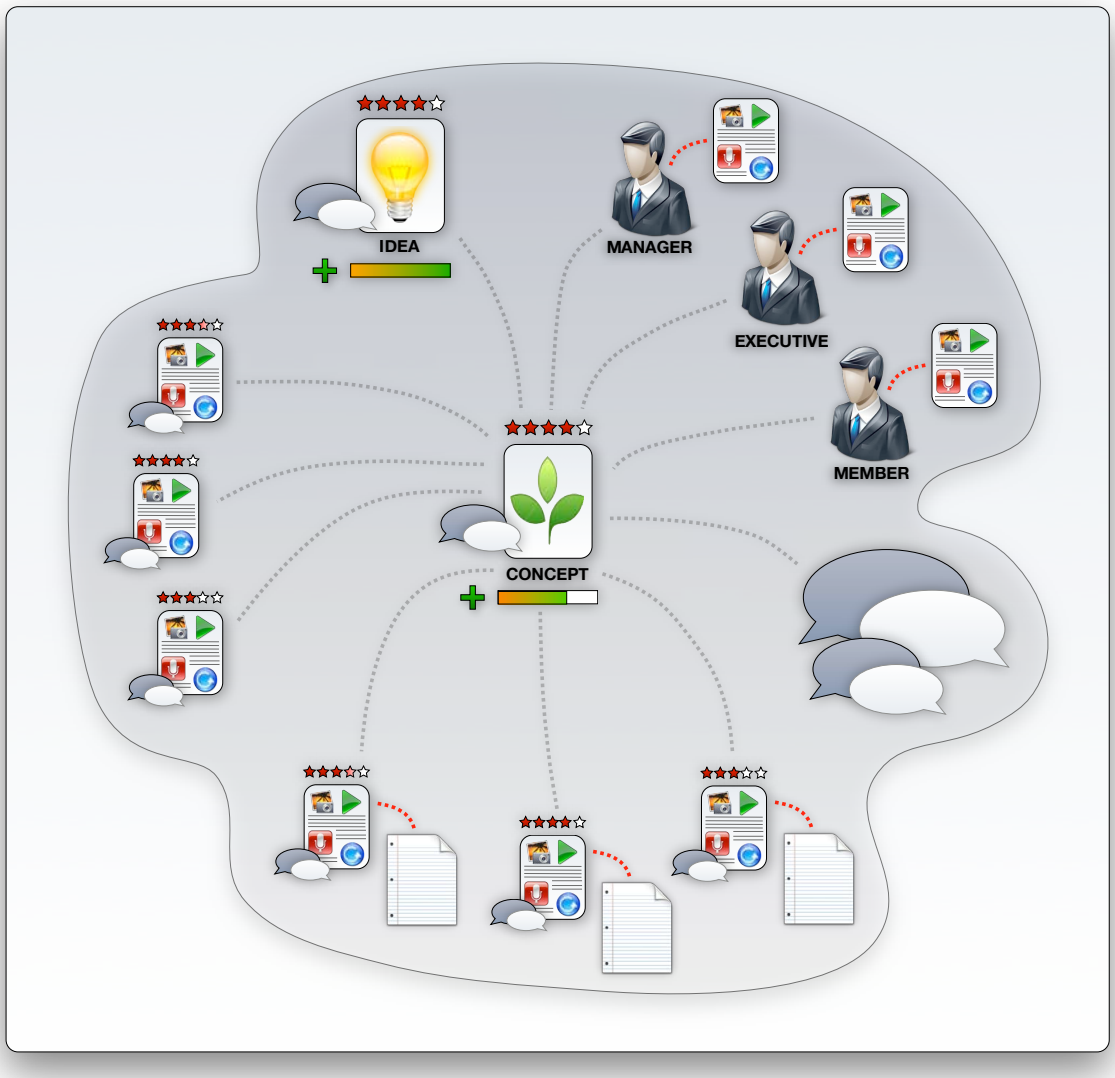

Figure 61 – Concept Objects are node clusters that form around a central node

identity, offers user subscriptions, access control and the ability to add new knowledge objects to the network while specifically identifying a specific group as the primary audience. The Organic Groups module is also extendable to allow for development of functions that are not included in the module itself.

In setting up Organic Groups to meet the design requirements, a number of options will be configured to deliver the appropriate functionality to the platform. Firstly, a content type needs to be designated to act as group home node – the node that forms the center of the node grouping. A *concept* content type is created specifically for this purpose, which is then edited with the Content Construction Kit (CCK) to include all the parameter fields that is necessary for the concept as a knowledge object. These parameter fields include a selectable list of the ideas that combined to initiate the development of the concept, as well as a development status indicator that makes provision for all the possible generic states that the concept may reach during its lifespan. Other content types like the *page, story, blog, book, poll, document* and *idea* are designated as group posts, which means that nodes of these types may be posted directly into the group.

The Organic Groups module automatically creates a directory of active groups within the site content and groups are configured to be listed in this directory by default. Group administrators will however be allowed to remove their groups from the directory. Groups may also be listed on the platform registration page for users to join upon registration. This option will however be disallowed as in most cases new users will need an amount of time to get accustomed to the network knowledge base before taking part in the development of a concept. All groups will however be set to public access be default, which means that they will be displayed as objects within the knowledge base and users may browse through them like they would with any other content item. As concept development is usually of a collaborative and non-secretive nature, registered users will be able to subscribe to groups freely. Group posts will also be available as separate public knowledge objects within the network to allow direct access to them. Their relationship to groups to which they belong will however be indicated. Contributors will also have the option to post new content into multiple groups at once, as may be the case when group activity motivates a user to externalize, and he or she feels that the new explicit knowledge may be relevant to multiple concepts.

The knowledge object that represents concepts within the knowledge network further requires a dedicated forum discussion where users involved with the concept exchange views and opinions. Binding a forum discussion to a group does not form part of the core functionality of the Organic Groups module, but extending it with the Organic Groups Forum add-on module expands the group functions to include a forum. This forum will only be visible to group members, unless a group administrator sets the forum to allow public access.

A dynamic group of nodes that combine to form a concept offers the flexibility that is needed for efficient development, but will not operate optimally without users accepting responsibility for the development process. Assigning specific roles to users within the group will contribute towards group members taking ownership of the group. Site-wide user roles will however not suffice for this requirement, as a single user may fulfill several different roles in different groups that he or she is involved in.

The Organic Groups User Roles module allows the assignment of group-specific roles to users. Users may have a different role in each group that he or she is a member of. The roles of 'manager', 'executive' and 'member' are created as site-wide user roles, which Organic Groups User Roles then assign within groups. The creator of the group home node is automatically promoted to the role of 'manager', and is allowed to manage all features available within the group functions. One of these features is the ability to assign roles to all the other members of the group. Any executive decision makers who are involved in the group are assigned the 'executive' role, and will be allowed exclusive access to certain views of the content (e.g. the concept filter and funding gate) and will have specific privileges like changing the state of the development status indicator. All other members of the group will be assigned the 'member' role.

By using the clustering capability of the Organic Groups module and enhancing its functionality with a custom central node, a dedicated forum and designated roles for users, a knowledge object design is created that will handle the complex needs of the Concept Object.

5.2.7.3. Project Object

The structure of the Project knowledge object is almost identical to that of the Concept, and once again the Content Construction Kit, Organic Groups, Organic Groups Forums and Organic Groups User Roles will be employed to deliver the desired functionality.

The Drupal core is used to create a new content type, namely the *project*, which is then tailored to meet the requirements by the Content Construction Kit. The Organic Groups module uses nodes of this *project* content type in the same way it uses nodes of the *concept* content type to form the central node of node clusters (refer to Figure 62). As with the *concept* content type, a multi-state development status indicator is appended to the basic content type structure. However, in the case of a project object, a selectable list of combining concepts are made available, rather than a list of ideas, as was the case with the concept objects. Group members are allowed to *post page, story, blog, book, poll* and *document* items into project groups.

Projects are more formalized than concepts and group membership will often reflect the project team charged with operating the project. Group subscription may therefore be on invitation-only basis, but this can be configured per project as is needed. Each project group will also be configurable in terms of whether the group will be listed in the groups listing, and whether separate nodes will be displayed in the global knowledge base as knowledge objects that allow direct access. These security settings may be tightened during project operation and then relaxed upon project completion to allow knowledge created during the project operation to disseminate and enrich the knowledge base.

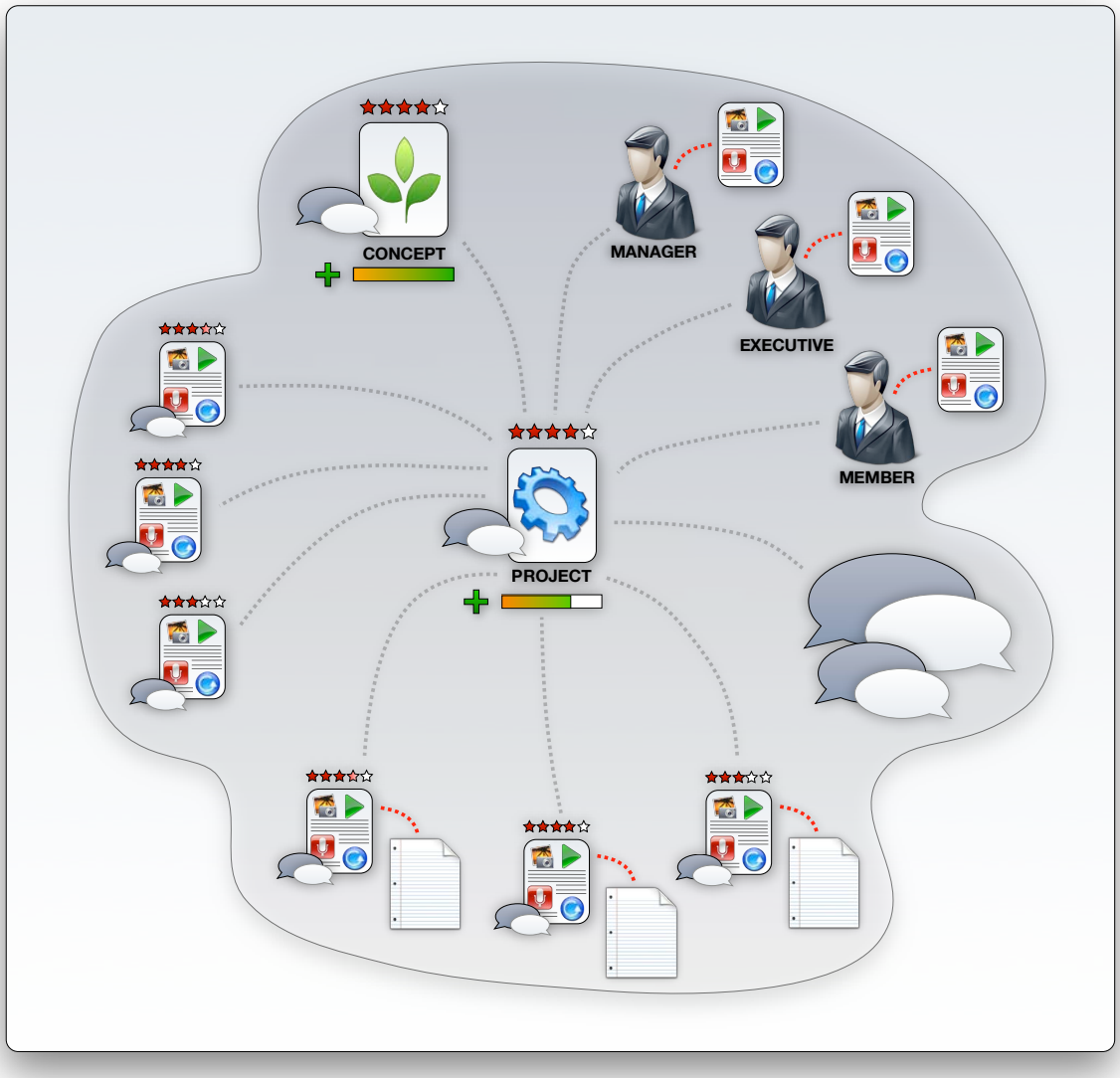

Figure 62 - Project Objects are node clusters that form around a central node

A dedicated discussion forum is added to the group functionality with the Organic Groups Forums module and Organic Groups User Roles is used to create member roles. New roles could be created to reflect the project team structure, or the more generic roles that were set up for concept development could be utilized again. As Organic Groups User Roles can manage different user roles

for each group, these role setups may differ from project to project to suit each endeavor's individual needs.

A user role that will however almost always be present is that of the 'executive' user, as these users will be tasked with evaluating projects on the grounds of overhead views of projects within the network. Their decisions on project performance and progress will be reflected in the project development status indicator.

By using the clustering capability of the Organic Groups module and enhancing its functionality with a custom central node, a dedicated forum and designated roles for users, a knowledge object design is created that will handle the complex needs of the Project Object.

The Concept and Project Objects' approach to node clustering is made possible by and takes advantage of the network model used to manage knowledge objects within the Information System. Adding the Idea, Concept and Project Objects to the knowledge network allows the Information System to support the entire Innovation Life Cycle while still enriching the network knowledge base (refer to Figure 63). Drupal and its Organic Groups module provides a dynamic approach to content management and generates value in a way that is unparalleled among Content Management Systems.

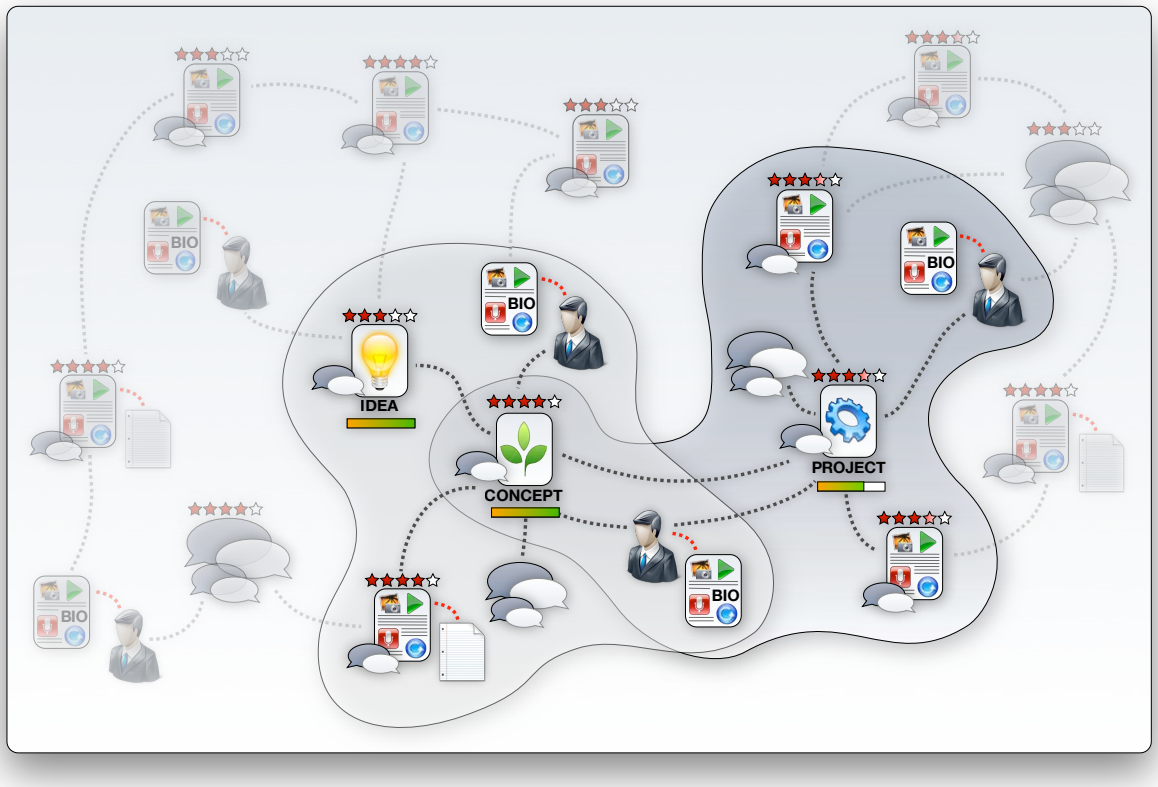

Figure 63 – The Concept and Project Objects form organic node clusters within the knowledge network

5.3. Taxonomy

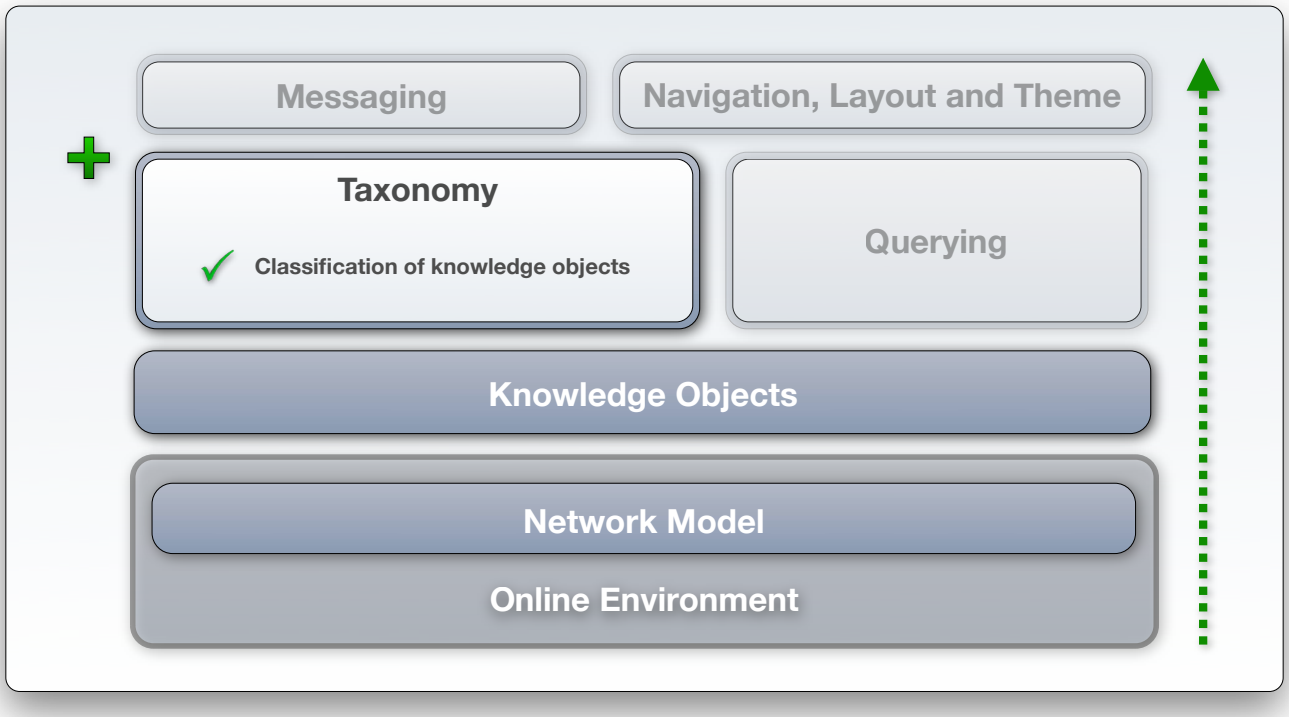

Figure 64 - Design layer 3: Taxonomy

Functionality Specification: Classification of Content

Addressing requirement 10, and stated in Chapter 4.4.4.1 as:

To generate the maximum amount of value from the knowledge network, nodes should be systematized. This entails creating categories in which to classify objects, as well as including the functionality to tag objects with keywords. These tags will be used to determine relevance between nodes (proximity within the network) and provide suggestions to users when they are navigating the knowledge base. Navigation of the knowledge base via the taxonomy should be made as simple and intuitive as possible.

A network model that supports knowledge objects is of limited value to an organization if those objects are not systematized to provide a structured knowledge base. Knowledge objects should therefore be classified according to a taxonomy scheme to provide an indication of their content (refer to Figure 65). This classification will also serve as a guide to navigation of the knowledge base.

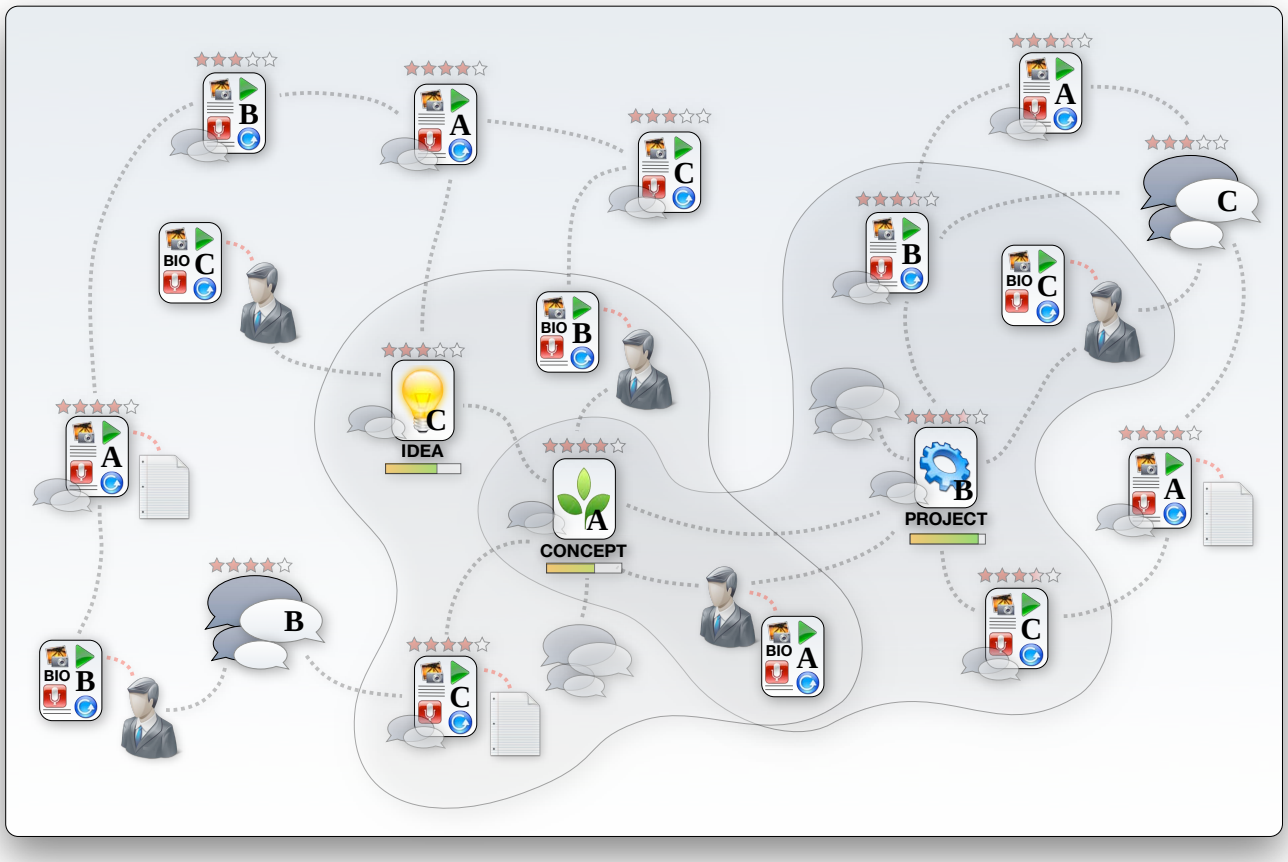

Figure 65 - Introducing a taxonomy scheme to the information system allows for systematization of the network knowledge base

The Drupal core includes a Taxonomy module that helps to classify content in sites by allowing vocabularies to be set up as either user defined tags (folksonomy) or administrator defined terms (taxonomy). Content nodes may be described in several ways, ranging from multiple terms from a single vocabulary, to single terms from several different vocabularies. Each vocabulary consists of a set of terms, and Drupal sites may have an unlimited number of vocabularies, each with an unlimited number of terms. The power and flexibility of Drupal's Taxonomy module is unparalleled amongst content management systems (Canlas [7]).

Predefining vocabulary terms provides controlled structure to the network knowledge base and produces a taxonomy scheme. Vocabularies 1 and 2 in Figure 66 illustrate this type of vocabulary setup. If "free tagging" is however allowed within vocabularies, users may add terms to the vocabulary as they are contributing content, instead of having to choose from a list. This creates the dynamic classification structure that is known as the folksonomy and is illustrated in Vocabulary 3 in Figure 66. Taxonomies provide overall structure and guidance, while folksonomies keep the knowledge base up to date with latest developments. By allowing for the development of folksonomies within the

boundaries of controlled vocabularies, Drupal taxonomies avoid the “pigeon-holing” system that most other content management systems employ without compromising accuracy and consistency.

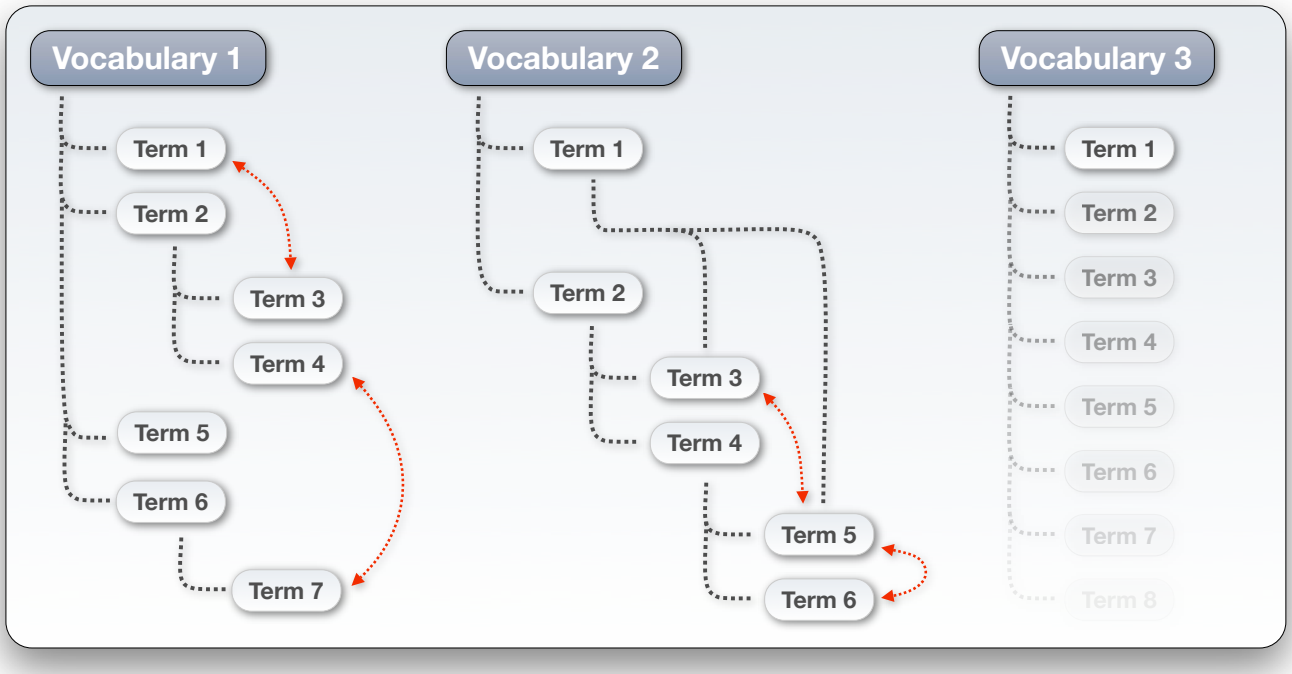

Figure 66 - Vocabularies may be set up with predefined terms or to allow free tagging

Vocabularies can be set to allow hierarchies of terms. Disallowing this option will keep all terms on the same level, while single parent-child relationships result in nested hierarchy of terms. Allowing for multiple parent-child relationships creates a flexible vocabulary that indicates context by association. With either single or multiple hierarchy options, each vocabulary can have as many levels as desired. Allowing for “related terms” within vocabularies enables horizontal relationships between terms, thereby creating thesauri within vocabularies, further indicating the context of nodes that are classified. This functionality is illustrated with the red arrows in Figure 66. Vocabularies may further be configured to allow users to classify a post with one or more terms from a single vocabulary. Classifying content in terms of vocabulary terms may be set as a requirement or optional extra feature for post submission.

Vocabularies may be configured in a number of different ways, enabling administrators to set up a combination of vocabularies that creates the necessary taxonomy scheme and functionality. Vocabularies can be named and described to clarify their use. These descriptions may be displayed when content belonging to that vocabulary is listed on the site to provide additional context. Vocabularies can furthermore be tied to specific content types, allowing for separate dedicated vocabularies for classifying e.g. ideas or documents.

Once the Taxonomy module is set up, its power becomes evident when submitting and retrieving content, as vocabularies make it possible to classify an individual node in multiple ways. For example a taxonomy scheme (refer to Figure 67) may include a “Industry” vocabulary and a “Location” vocabulary including terms such as: “Africa”, “Europe” (with sub-terms like "Britain" and "France"). A single node containing a market survey might be identified using the term "Mobile" from the Industry vocabulary and Location vocabulary term "Europe" and sub-term "France". Adding a vocabulary for "Brand” might lead to the following combination of terms: a "Mobile” product from “Nokia” aimed at “Europe”, allowing users to locate it by any of these three terms. A further vocabulary may support user defined tagging, and enables authors to add new model codes (e.g. “E51”) as terms without it being predefined.

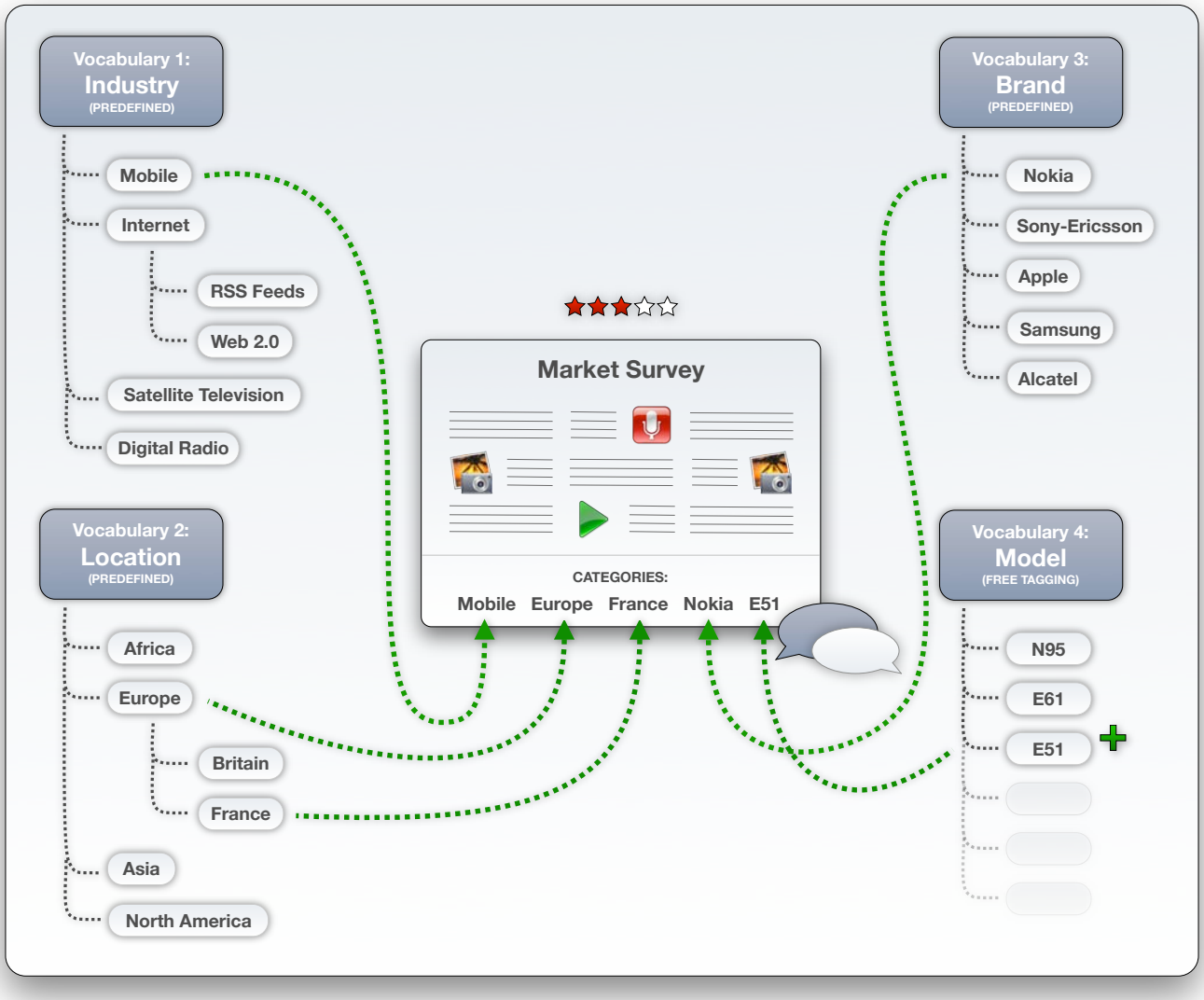

Figure 67 - An example of a content item being tagged with multiple terms from different vocabularies in the taxonomy scheme

5.4. Querying

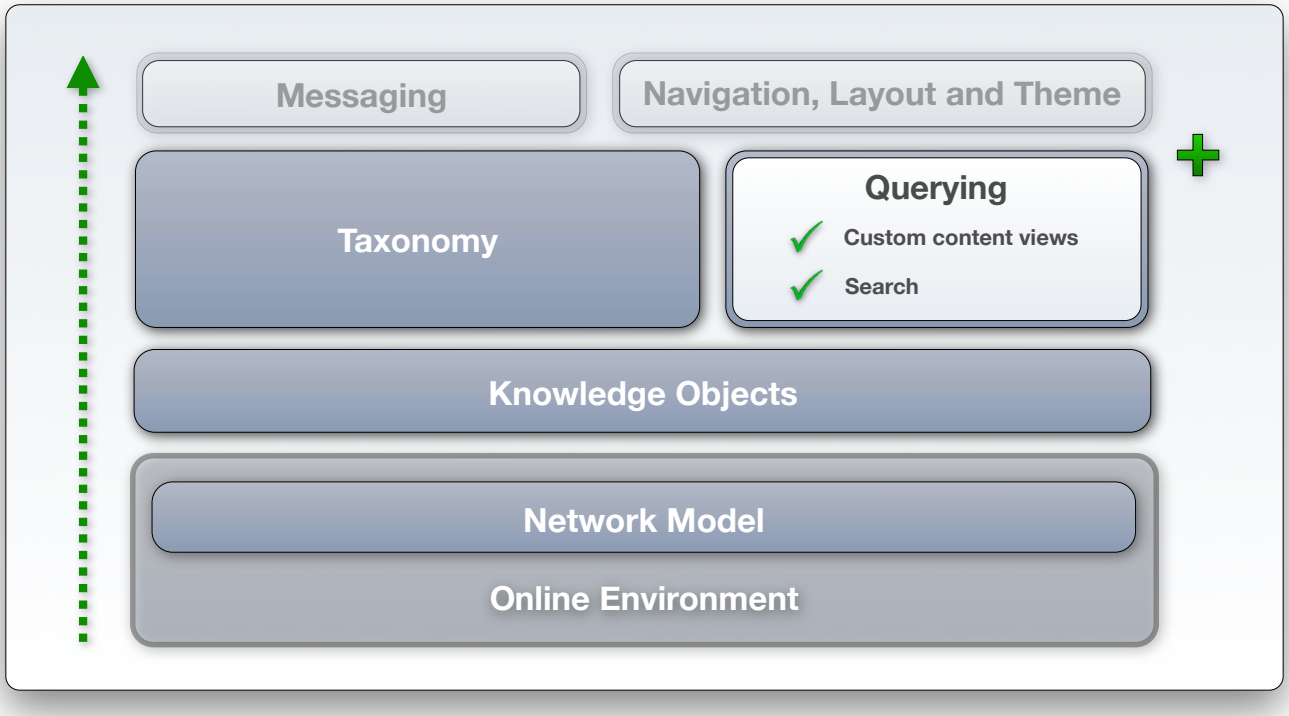

Figure 68 - Design layer 4: Querying

Functionality Specification: Extract custom user views of content

Addressing requirement 17, and stated in Chapter 4.4.5.1 as:

The implementation of knowledge objects results in all network nodes being treated as peers on a data structure level. Categorizing and tagging these objects provides a rich knowledge base from which custom queries can be made. These queries can be used to enhance the user experience or to play an important role in the Innovation Life Cycle, e.g. generating a view of all concepts that have been identified as being "refined" for consideration in the funding gate.

Functionality Specification: Search function

Addressing requirement 12, and stated in Chapter 4.4.5.2 as:

Along with being able to navigate through the taxonomy with ease, users should be able to search for terms throughout the network and receive full results. The implementation of knowledge objects will result in users, documents, posts, ideas, concepts and projects as possible output as search results.

It can be said that an information system that support a knowledge network and ultimately innovation, consists of two main functions: the first being the capture of data, the second being presentation of this data. A modified Drupal core that handles knowledge objects, along with the Taxonomy module is extremely adept at handling the capture, classification and storage of data. A further extension of the Drupal core is however necessary to handle the value-adding presentation of data.

The Views module is the most popular extension for the Drupal core and is employed in roughly 50% of all Drupal installations. The Views module is a powerful query builder for Drupal that allows you to fetch and present lists of content to the user by filtering the data that is displayed by a variety of options, including but not limited to: node type, author and taxonomy term. Views also allows sorting of these lists by a variety of different options, including post date, updated date and number of page views (refer to Figure 69). The Statistics module that is included in the Drupal core keeps logs of all activity on the site and records useful information on node popularity and usage patterns.

Each content filter that is constructed with the Views module consists of a number of overlapping parameters and is saved as a “view”. These parameters gradually narrow down the nodes that will be displayed and eventually adds a guideline for what the output should look like. Views can also be reused and displayed in multiple locations and in different formats. Several basic views are set up when the Views module is installed, but custom views can be built from the ground up.

Parameters are grouped in terms of filters, arguments and fields. Filters allow a subset of nodes to be selected for display, with multiple filters being combined with an AND-operator. Arguments can parse information from the URL of the page that is navigated to and include it in the subset parameters to make the view more specific, e.g. .../taxonomy/term/1 will only display nodes tagged with taxonomy term 1. Fields are used to specify which elements of the node subset that is ultimately selected, will be displayed to the user. These fields may also be used to sort the nodes that are displayed.

Figure 69 - Introducing querying functionality to the information system allows for custom views of the network knowledge base

Display options include a list view that by default displays only the titles of nodes, but could be configured to include more information. A table view can display multiple fields from each node on a single row, which is good for comparing nodes with each other, especially when fields are made sortable. A teaser view delivers the standard, modern blog layout, by displaying the title and first paragraph of nodes. The front page of Drupal website by default employs a teaser view of nodes that were designated as being promoted to the front page.

Access to views can be controlled using user roles, and this approach is used to display special content views (e.g. idea filter and concept filter) to certain users (e.g. executive management). The views especially needed for the Innovation Life Cycle may be constructed as depicted in Figure 70.