World Wide Modeling: The agility of the web applied to model repositories

Philippe Desfray – SOFTEAM R&D Director – Modeliosoft

Overview

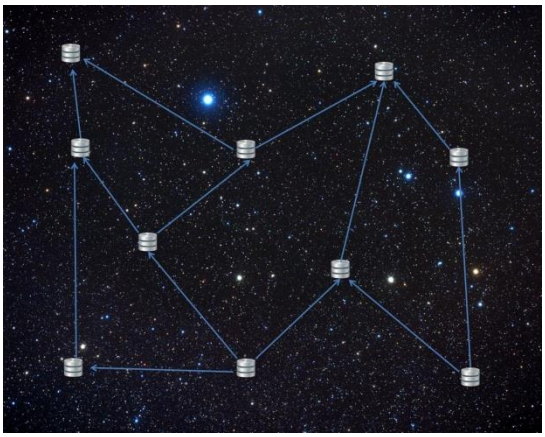

Figure 1 - *Constellation of repositories*

In today's era of data sharing, immediate communication and world-wide distribution of participants, at a time when teams are asked to be ever more agile, the traditional approach of model repositories no longer meets expectations. Centralized organization has become inconsistent with the way in which the world and its companies function.

Consequently, a new repository approach is emerging through the "Constellation" repository technology provided by the upcoming version 3 of the Modelio modeling tool[1]. This is a major change in concept, based on recent approaches and technologies that are better adapted to today's world: Models have to be distributed and shared in as vast and immediate a way as the web. Agile cooperation modes, such as those used in open source projects, also have to be supported at model level.

The existence of the open source distribution of Modelio will enable increased repository accessibility, by removing cost and subscription-related barriers.

[1] Modelio comes in an open source distribution available on www.modelio.org and in a commercial distribution available on www.modeliosoft.com

This white paper presents this new approach, and shows how different organization modes can best use it.

Model repository centralization: The limitations of the standard model

In today's world, it is virtually impossible to set up a model repository for different enterprise entities, large-scale systems or projects, which can be accessed by all participants (readers, contributors, partners, and so on). Standard techniques based on a centralized repository with a designated manager come up against a vast variety of situations, with participants who neither want nor are able to conform to uniform rules and management.

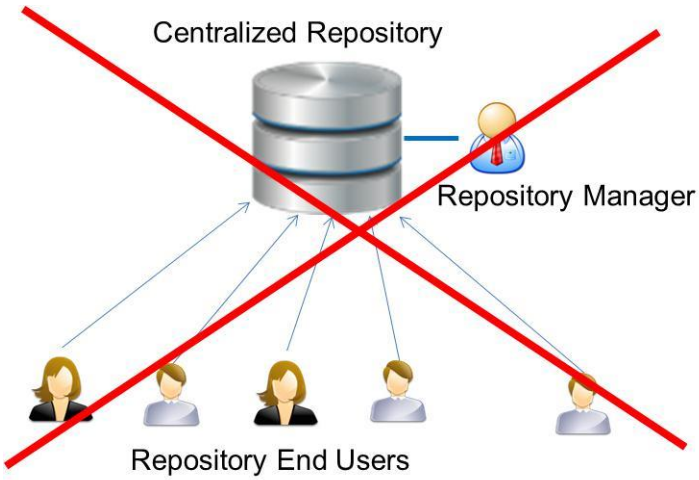

Figure 2 – *Centralized organization: A model which has reached its limits*

This traditional centralization imposes:

- A need for centralized declaration of users and configurations.
- A mandatory organization of repository models.
- A limited repository storage and access mode, which goes against the need for ubiquity and nomadic mode autonomy.
- A costly approach, requiring "server" licenses for each participant.

This kind of approach clearly hinders team agility, as well as team and model distribution. For example, it cannot be considered for contributory open source projects, or for systems and projects involving several partners. As a result, it cannot be used by large companies, whose different departments are often autonomous. This is a real obstacle to model-based knowledge management within organizations.

The use of a centralized repository does present a number of advantages, and has proven its worth over the years. However, it has reached its limits for the adoption and sharing of models, and goes against openness and agility. After presenting a totally decentralized architecture, we will

demonstrate how the two approaches can be combined, so as to benefit from the advantages of both.

Taking inspiration from the web, a widely used and tested approach for information publication

Modelio's new "Constellation" repository technology is based upon two widespread approaches supporting cooperation and information and contribution sharing:

- The web, whose omnipresence and flexibility are required by everyone.
- The distribution of models through "libraries". This approach is widely used for code in open source projects and software development, which are a major source of sharing and re-use.

Like the web, there is no "central server" with Constellation. The organization is highly decentralized, which allows the most open and agile cooperation modes. In its own way, the Modelio modeling tool plays the role of an internet browser. Its open source distribution, which also includes "Constellation", provides open access to the repository.

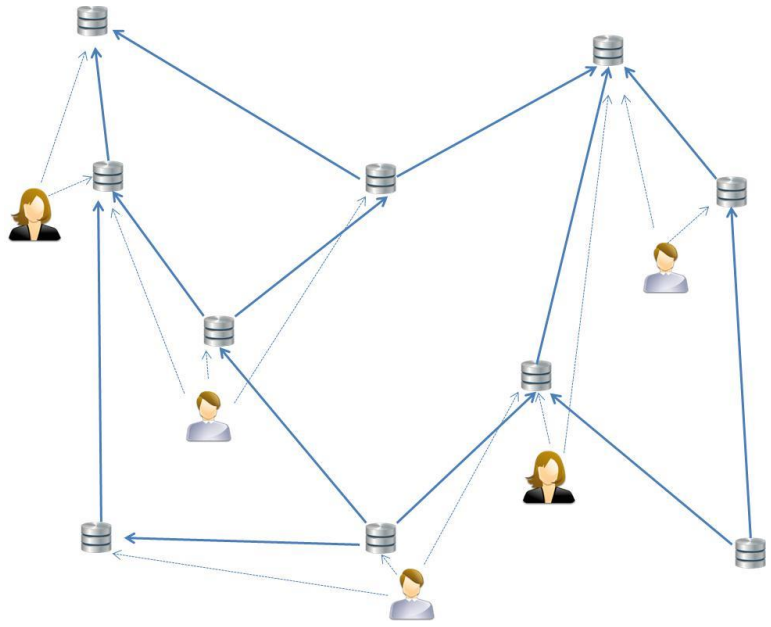

Figure 3 – *Decentralized organization: Distribution of users and model fragments*

Constellation is a network (or web) of "*model fragments*", which constitute autonomous groups of model elements. Each model fragment has its own independent storage, and can have several access modes (local, shared, versioned and configured, library, web, secure web). The model elements in a fragment can be linked to other model elements from different fragments (Figure 4), and the "model" then has a range which includes several distributed fragments.

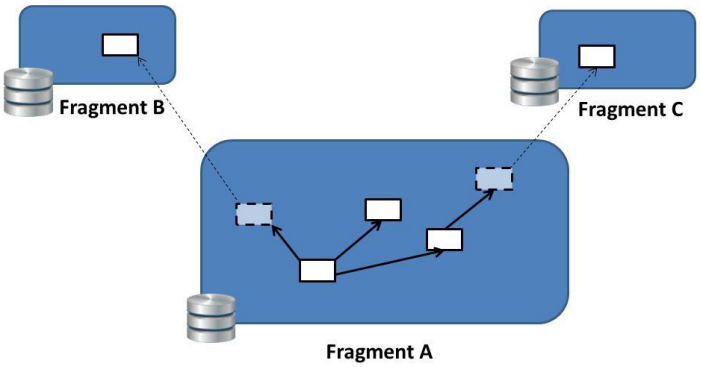

Figure 4 – *Transparent referencing between fragments*

A "model" groups fragments together in order to meet a specific goal. For example, this can be a project, a system, architecture, and so on.

Using the Modelio modeling tool, which acts as a sort of fragment and model browser, the user therefore edits a model which is a set of fragments whose components are interlinked. These fragments can be transparently local or remote via the web or any other network. The user edits or browses a model of unlimited size, without having to worry about the actual location of the elements handled, which are scattered across different fragments in the Constellation.

Defining a model and its components in a constellation

A model is an abstract representation in the form of images and text. It represents something (a system, a problem, a solution, concepts, procedures, …) for a particular community of participants (designers, developers, business analysts, users, …), and has a clearly defined objective (pedagogical example, business view, design, analysis, implementation, …).

The images and texts that a model presents are views of the model elements which constitute it. Model elements break a model down into elementary information, for example a class, an attribute, an attribute's type, an operation's parameter, an association end, and so on.

Handling an individual model element rarely makes sense for a user, who will usually work with a more macroscopic agglomerated element. The following model element aggregation structures can therefore be identified:

- *Configuration units* are groups of model elements which are individually managed by the user. Their definition can vary, but in practice they constitute a management unit for the user, who handles them and manages their version and configuration. They typically correspond to a Class, a Use Case, a Component, a Process or a Package. A configuration unit can also be locked to control and avoid concurrent access to it: it is a user's work unit.
- *Model fragments* are groups of higher level model elements. Materialized as packages, they group configuration units. Model designers attribute logical consistency to model fragments, but their primary function is to autonomously store a part of a model. Their components can be linked to model elements belonging to other model fragments. A model element is also

not definitively linked to a fragment. It can be moved to other fragments, according to the organizational needs of the designers.

- *Models* are groups of model fragments with a defined goal (the model of an application, of system architecture, of a particular problem, and so on). Models identify the fragments which compose it and group them in a specific context. Fragments exist independently of models, and can be referenced by several models.

When a model is edited, the location of fragments is transparent: they can be local to a machine, on a local company network, or published on the web. Decisions on breakdown into fragments, physical location and accessibility are driven by participants' organizational, cooperation and accessibility needs.

As with internet browsers, access to other fragments via the web cannot be guaranteed. The remote element may be absent, the internet connection can be interrupted, and so on. The architecture of Constellation guarantees that links from origin elements to remote elements remain visible, and that reconnection with absent elements happens transparently, as with broken links and absent pages in internet browsers.

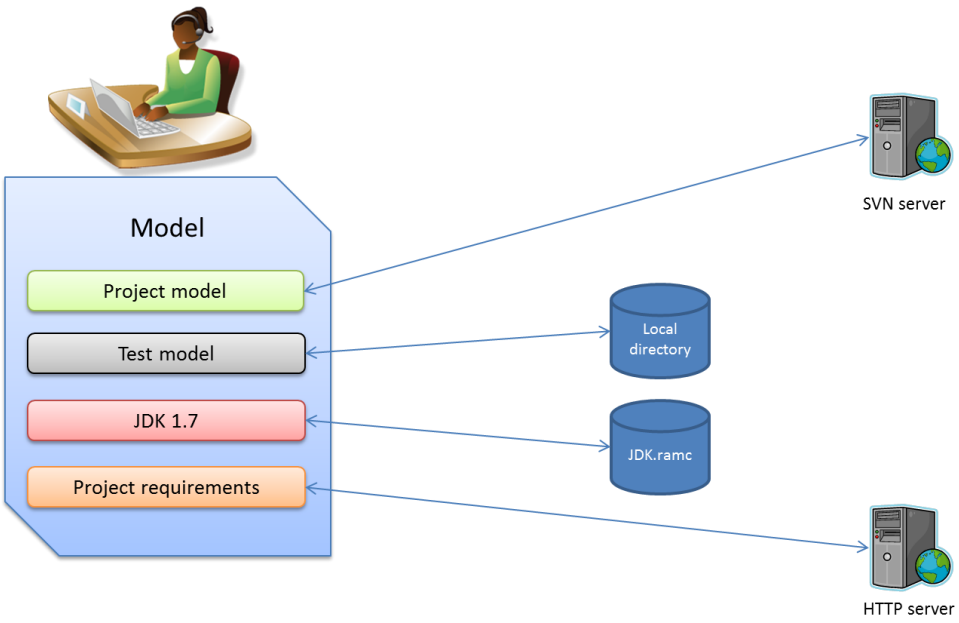

Figure 5 – *Example of model configuration*

In Figure 5, a model has been defined, targeting the software development of a Java application made up of several fragments:

- The model developed during the project. This model is shared by all project participants. Access to the model is controlled by SVN, which manages versions, configuration and concurrent access.
- A local test model. This fragment is only visible to the current user, who stores it locally on his/her workstation.
- The JDK (Java) library model: This reversed model is distributed in the form of a "model component" (or model library), and is accessible in read-only mode.

- The requirements model resulting from business analysis. This model can be accessed in read-only mode via http. It is used to trace the application model to requirements and to run impact analyses.

This example shows us that a model (dedicated here to a project) is configured by assembling fragments which can have different statuses and access modes (remote and managed in teamwork mode and in configuration, local to a workstation, in the form of a library, in open access mode via the internet).

Modelio servers: Governing modeling for a system or organization

Like the web, the Constellation technology is naturally distributed and has no centralizing element. This characteristic ensures openness for participants.

To combine the advantages of the openness of decentralization and the control afforded by a centralized repository approach, servers must be put in place.

In a "Constellation" architecture, model servers provide a central access point to a set of distributed model fragments. A server defines a community that will use a set of fragments and takes care of making sure that access rights, conventions and rules are respected. In a constellation of model fragments, an unlimited number of servers can exist, with each model fragment accessible by zero or several servers.

The function of the model server (Modeliosoft solution) is therefore to organize the repository, govern access to it, assist teams in a particular cooperation mode and manage model and project portfolios. Servers also manage project configuration, by updating and maintaining the consistency of all participant configurations (versions of Modelio "plug in" extensions or "modules", fragments and libraries)[2]. A server is used to define user communities, which share organization rules and conventions. The server presents its users with a portfolio of models (or projects). In a typical model Constellation, several servers co-exist and share model fragments, by governing and controlling the models accessible to the community. This co-existence of several servers which share model fragments is the most important new development with regard to standard server schemas. It enables openness and sharing between the different "worlds" that each server represents within a "universe" which is the Constellation architecture.

[2] Here we find traditional server services, already provided in the current version of Modelio

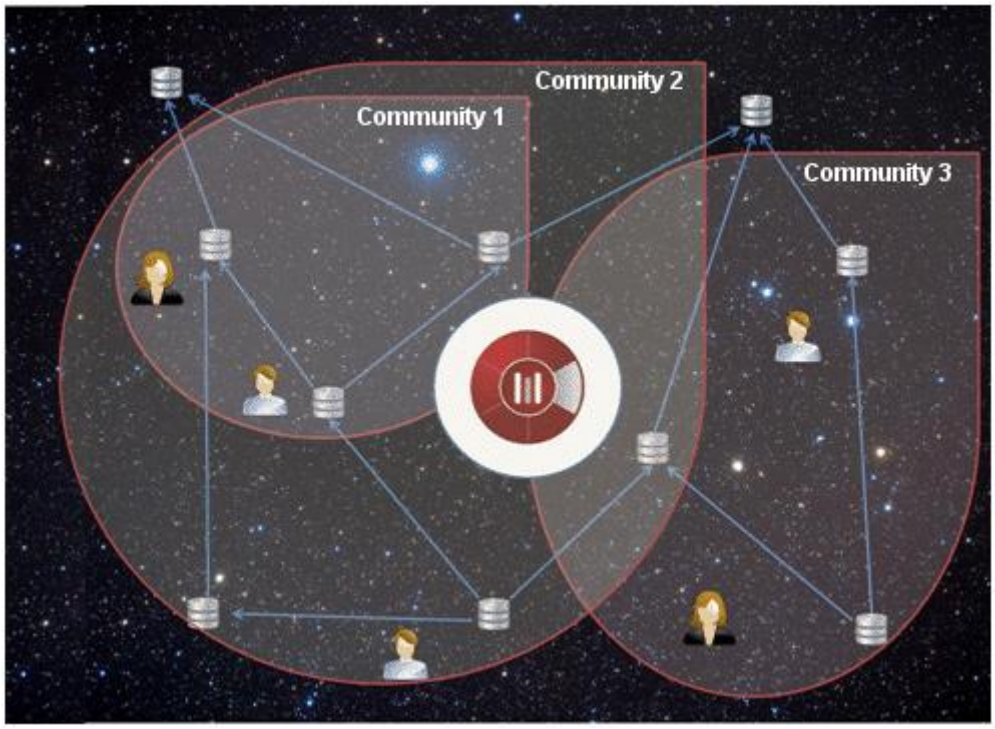

Figure 6 – *Servers organize different communities in a constellation of model fragments*

The fragments managed by a server can be accessible from other servers or in the absence of servers, which allows the most openness and the widest sharing. Conversely, they can be visible to only a restricted community, in accordance with strict rules and security and confidentiality constraints.

The use of Modelio servers with the Constellation technology thus enables the combination of the agility and openness required by certain cooperation modes with the organization rules dedicated to certain groups of participants.

Conclusion

This new approach to organizing repositories removes the obstacles of the current centralized view, both in terms of the organization and distribution of participants and date, and in terms of the volume of data handled.

It provides repositories with great agility, and enables the use of organization modes which are in tune with current cooperation modes.

This approach provides large-scale organizations with a means of building a real, universally accessible model repository, while retaining the means of governing and sharing conventions within participant communities. In this way, it supports the diversity of entities which exist within an organization, and their autonomy, while facilitating sharing and cooperation. This widening enables modeling support to be applied to the "extended enterprise", which incorporates its eco-system (providers, partners, and so on).

This approach also provides a solution to tool limitations with regard to the support of large-scale models (due to the size and number of concurrent accesses), by breaking them down into fragments and allowing "load balancing" strategies and data replication/synchronization strategies between servers.

Finally, it allows a more lightweight, more flexible organization for all sorts of communities: open source, small to medium-sized companies, and so on.

The new technical and functional capabilities are recognized, and the time has come for organizations to take ownership of this approach, in order to put in place open and distributed cooperation modes which correspond to their goals.