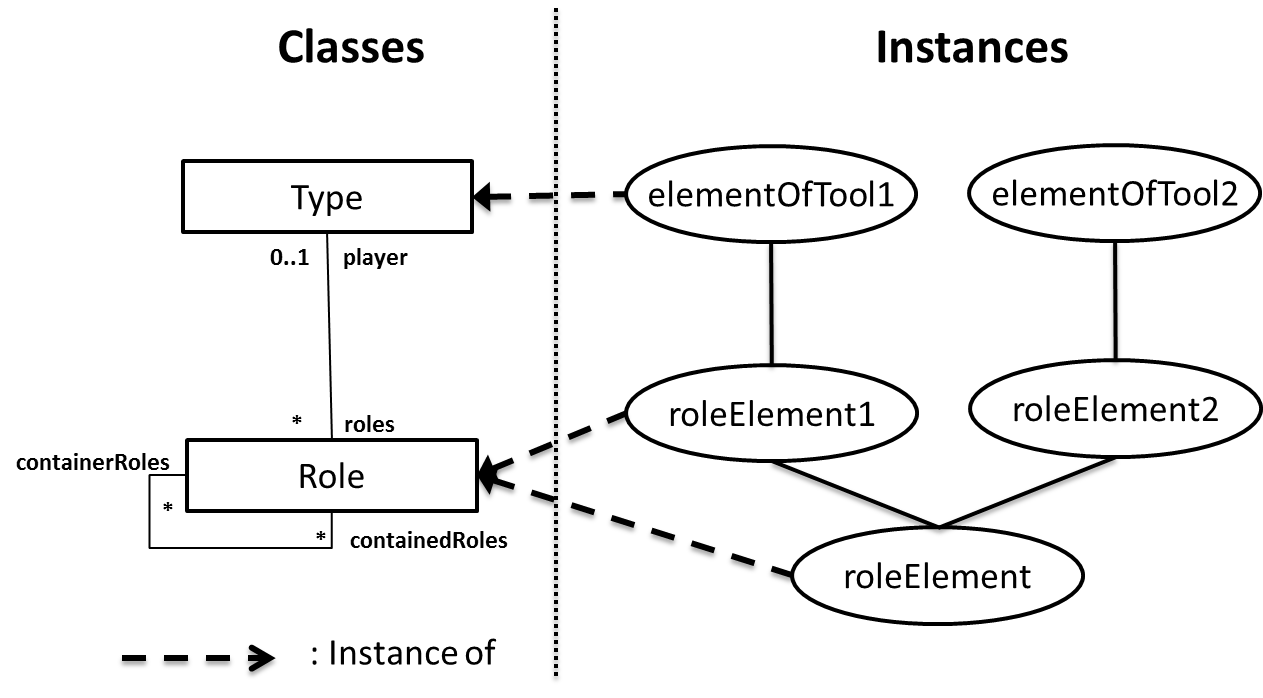
|  |  |
| --- | --- |
|  | Roles => adaptation without models modification  1 Roles medeling for different tools  Associate some roles to create a new one with all the different features  Rols => adapted the point of view according to the context.  All model elements can play rols  Role can play rols  A players (role or type) can play servial rols |

In most projects we used serval specific tools to model systems or manipulate data and sometimes these tools used the same model or data. Therefore a collaborative process is necessary, it is call interoperability. Unification is solutions to create interoperability between two tools that consist on create a pivot model shared by the tools. This solution is simply to implement if all tools were link with the same pivot model. But the unification installing two main problems, the first emerge when the pivot model is modified, by definition of a pivot models all tool models linked with need to be updating too. The second one comes when a new tool is added to the project that is necessary to connect the new tool with the pivot model without modified the pivot model (cf problem one). This task can be a real problem if the tool model and the pivot model are strongly discordant. Moreover today the tools model progress faster than standards considering that we cannot use a pivot model for a dynamic interoperability [thesis of Maud RIO]. Therefore other solutions were imagined to create a dynamic interoperability like the federation. Role4All is an example of federation system, in Role4All a role model is defined this model as a pivot model allows the interoperability between serval tools. For each tool each model element play a role defined in the role model, the connection between the role model and the tool model is formatted by an adaptor. Unlike a pivot model, a role model is independent of the tools, for example if a tool model is updating we need to adapt the adaptor but not the role model himself. Moreover all elements can play roles included an elements of unrelated type therefore with Role4All it is possible to federate all tools type. Furthermore a role can play a role this ability allows to extend a role in order to adapt the role model to a tool without modifications in the tool model or the role model. Role provides others features, we can associate serval existing roles to create a new one it is useful if a new concept emerge in the roles model. Another feature is that element can play several roles, therefor it is possible to create different point of view about the same model according to the context [old article].

In Role4All all elements play roles so it is possible to federate various tools if they "play" the same role [old article]. In addition to the federation Role4All allows synchronizing tools through the concept of role. With Role4All all model elements play a role therefore all model instances are strongly linked with a role instance through an adaptor [old article], consequently to synchronize model instances is equivalent to synchronize role instances. Finally to synchronize tools we need to synchronize role instances.

A role has two instance variables called *containedRoles* and *containerRoles* that are collections of roles. So, a role can contain several roles and be contained in various roles. We use this relation to connect together the roles that will be synchronized.



In the figure XXX the relation between the Role class and the Type class was simplified, the detailed relation was developed in the previous article [old article]. The figure XXX present how was connected the role instances, the instance *roleElement* contain two role instances: *roleElement1* and *roleElement2*. The instance *roleElement* knows which instances it contains through the variable *containedRoles* and the instances roleElement1 and roleElement2 know in which instances they are contained through the variable *containerRoles*. This bidirectional relation is necessary to create synchronization between role instances. Furthermore the container role (*roleElement*) is a facade to the contained ones therefore it is possible to work directly with the containers, this allow to be independent of tools.

The synchronize methods are developed in the container, by default the synchronization is a simply check-out/check-in synchronization [Thesis synchro]. After a save all changes are immediately reflected in all the synchronized tools. A user can and should define various rules and checks when he creates synchronization between tools with Role4All. More over some security systems can be apply on synchronization, as examples a prioritization of the synchronizations according to the role of the user. Finally it is possible to change the default synchronize method to another one, like the composition model [thesis synchro]. The main idea with the synchronization in Role4All is to be independent of the tools and easily editable.

Relation between a role instance and a elements is defined according to an adaptor [old article]. The synchronization between these instances should be defined in the role instance.