Decomposing model

Model decomposition has a package level granularity. Smaller elements cannot be split into separate projects. Basically, each project package could be partitioned into a separate projects, however this is excessive.

The decision on how to split a model into parts should be made carefully. You should isolate model parts which form logically complete pieces of a structure (e.g., a subsystem, code library, or profile) and are not very dependent on each other.

When there are many one-way dependencies to some model part (e.g., parts **A**, **B**, **C** depend on a part **D**, but part **D** does not depend on any of the parts **A**, **B**, **C**), this part is a good candidate for a placement into a separate project.

When one big project is used to store all the modeling information (e.g., use case models, high level architectural models, detailed implementation level class, sequence, state, and other information), it may be useful to partition the models according to the modeling domains (i.e., use cases in one used project, architectural models in another, implementation level models in yet another). This allows unloading unnecessary used projects while working on one part (and improving the performance), but still retain the relationships between domains and load used projects on demand.

Avoid decomposing a model into parts which have circular dependencies. That is, A B or A B C A situations.

Usually programmers are very adept at splitting large code bases into libraries. The very same criteria should be applied for splitting large models into separate projects.

The MagicDraw project decomposition functionality allows two important possibilities:

- Working without all used projects loaded.
- Using the project in the read-write accessibility mode.

Used projects are often exploited for storing profiles, however a used project is not a profile, and it is important not to mix them. Any model part can be stored in a separate project.

Related pages

Unknown macro: 'list-children'