www.ModelingConcepts.com

Do not be afraid to ask!

A Quick Chat about two of many Service-Oriented Modeling Framework (SOMF) Capabilities: Simplicity and Traceability

For architects, business analysts, system analysts, software developers, modelers, team leaders, and managers

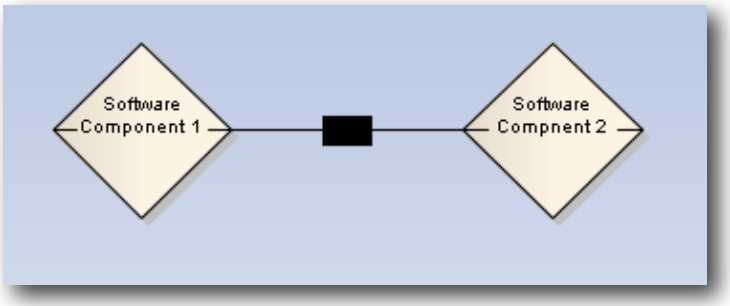
Use the SOMF modeling capabilities for enterprise architecture, application architecture, service-oriented architecture (SOA), and Cloud Computing projects.

SOMF is empowered by Sparx Systems Enterprise Architect modeling platform

Simplicity

Think about how often we try to explain in simple words how a software entity is linked to another, describe dependencies of one component on others, or illustrate the affiliation between applications in a production environment.

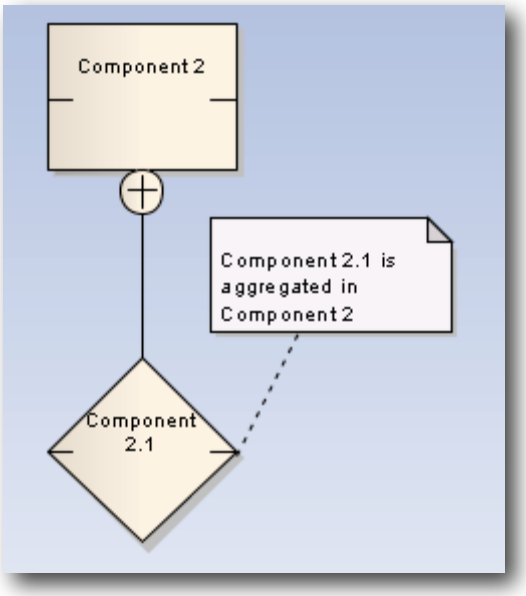
So, here is how the service-oriented modeling framework (SOMF) advocates expressing coupling between two software entities:



Service Coupling Example

Easy? Yes, this is simple because SOMF both promotes simplicity and enables architects, managers, analysts, modelers, and developers to communicate by employing a simple language. In this case, we use the "Coupled" symbol to indicate software coupling.

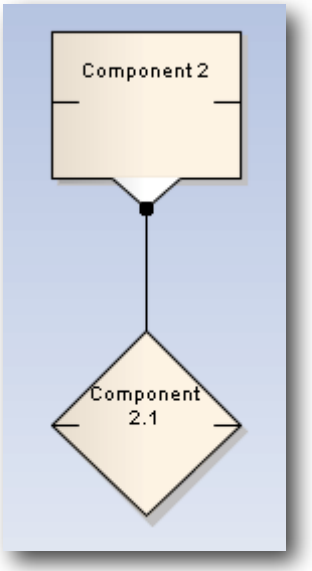
Another example. How can we describe the containment of one software component within another? This is simple to illustrate. See the example below:



Service Aggregation Example

In the above example, Component 2 (parent) aggregates component 2.1 (child) by using the "Aggregated" symbol. Note that the plus sign that is enclosed in a circle points to the parent component – Component 2.

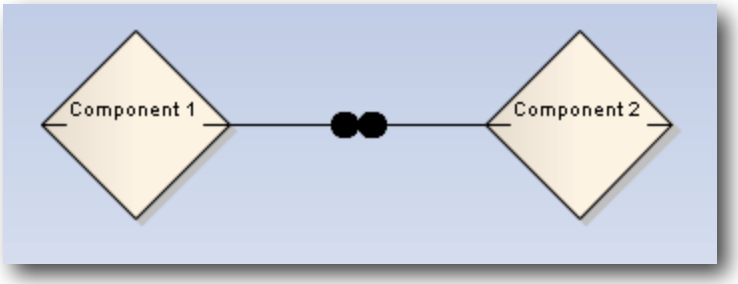
And how do we separate an aggregated software entity? This is also simple to illustrate.



Service Decomposition Example

In the above diagram Component 2.1 (child) is separated from its parent aggregating entity by employing the "Decoupled" symbol. Again, note that the fork-like symbol points to the parent Component 2.

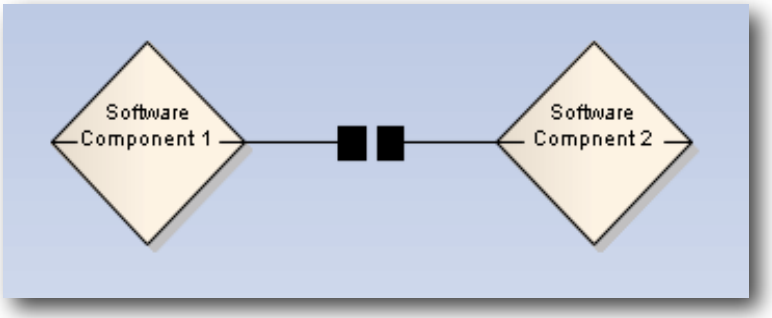
Finally, how do we signify a binding contract between a service provider and its corresponding consumer? This is really easy to do. See the below diagram employs the "Bound" symbol to tie these software entities (Component 1 and Component 2) by a stipulated contract.



Service Binding: Service Contract Example

Traceability

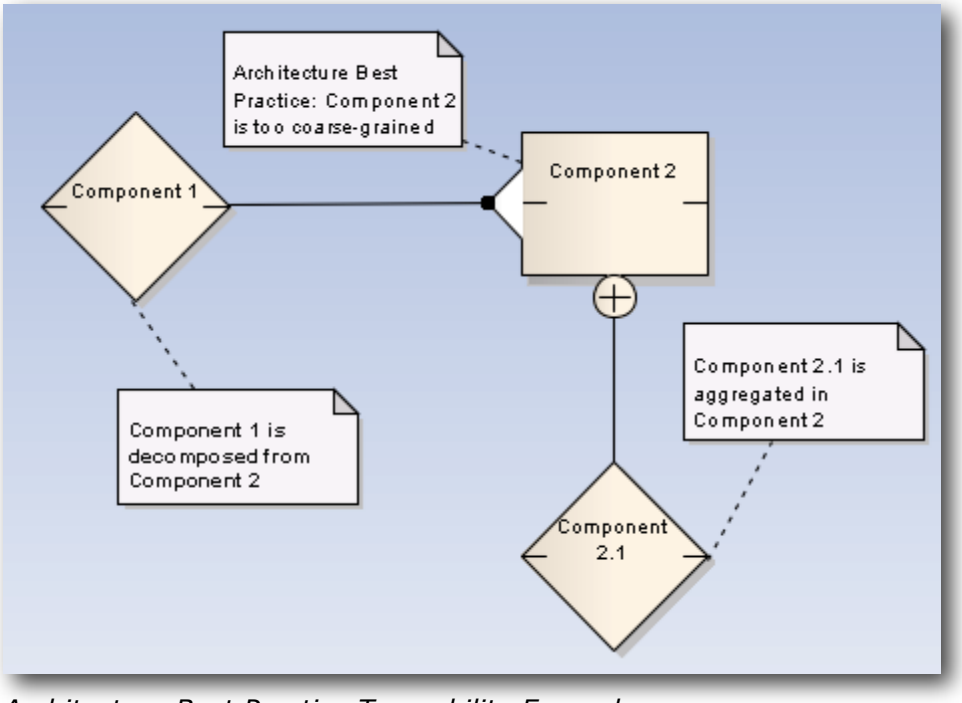
Next, if you want to illustrate past relationships between two software pieces employ the SOMF "Decoupled" symbol, just as you see in the below diagram.



Service Decoupling Example

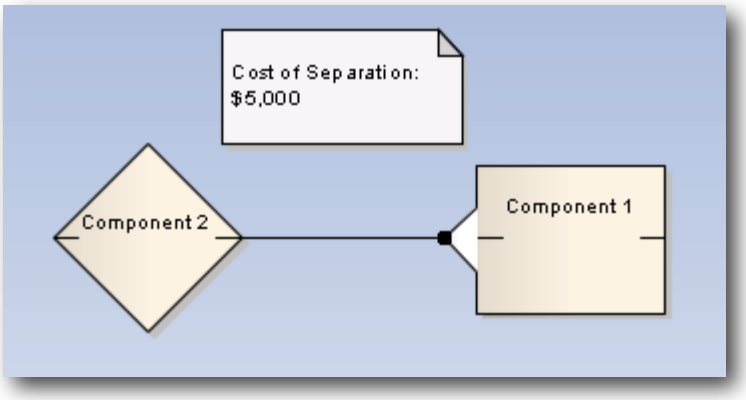
Traceability of software architecture is the reason. In this case, SOMF calls for preserving past implementation decisions and allows traceability of architecture best practices and even expenditure.

So let us take a look at an architecture traceability example below. Here the reason for separating Component 1 (child) from its parent Component 2 is indicated in the comment box: "Architecture Best Practice: Component 2 is too coarse-grained". Component 2.1, however, is still aggregated in its parent Component 2.



Architecture Best Practice Traceability Example

Business traceability can also be expressed by employing this method. Note that the below diagram indicates that the cost of software entity separation is $5,000.



Business Traceability Example

To learn more about many SOMF capabilities, modeling methods and formal notation, and patterns for enterprise architecture, application architecture, service-oriented architecture (SOA), and Cloud Computing refer to these books: