My proposal :

Component Object Model (COM) COM is a component model and runtime infrastructure built into the Windows operating system . It forms a powerful extensibility mechanism for many applications, including influential programs such as Excel. COM supports a hierarchical model, and composition of instances is via a registry based approach for indirectly locating service providers.

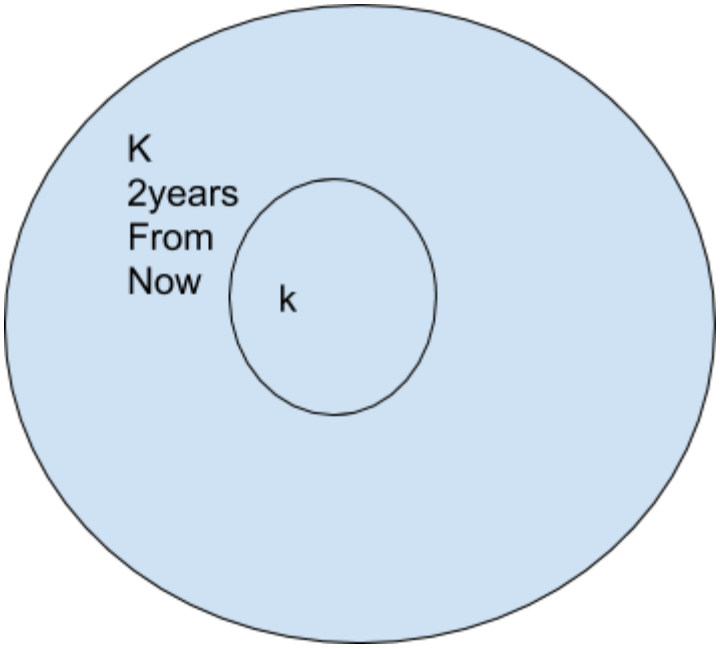
We Can take COM as the base for registering the plugin dependency and plugin list. By implementing **Advanced plugin systems ..** In the time bootstrapping the application they will always maintain the level, which implies lower level plugin must load before higher level plugin

Registry
Level 0 plugin
Level 1 plugin
Level 3 plugin

Plugin will be a mathematical function F(x)=Y , where Y $\subset$ C (will provide requirement: ***AppropriateLevel, DetectAndCorrect)***

C = {set of channel }
And here x is the argument received through a channel or produce by some functionality.

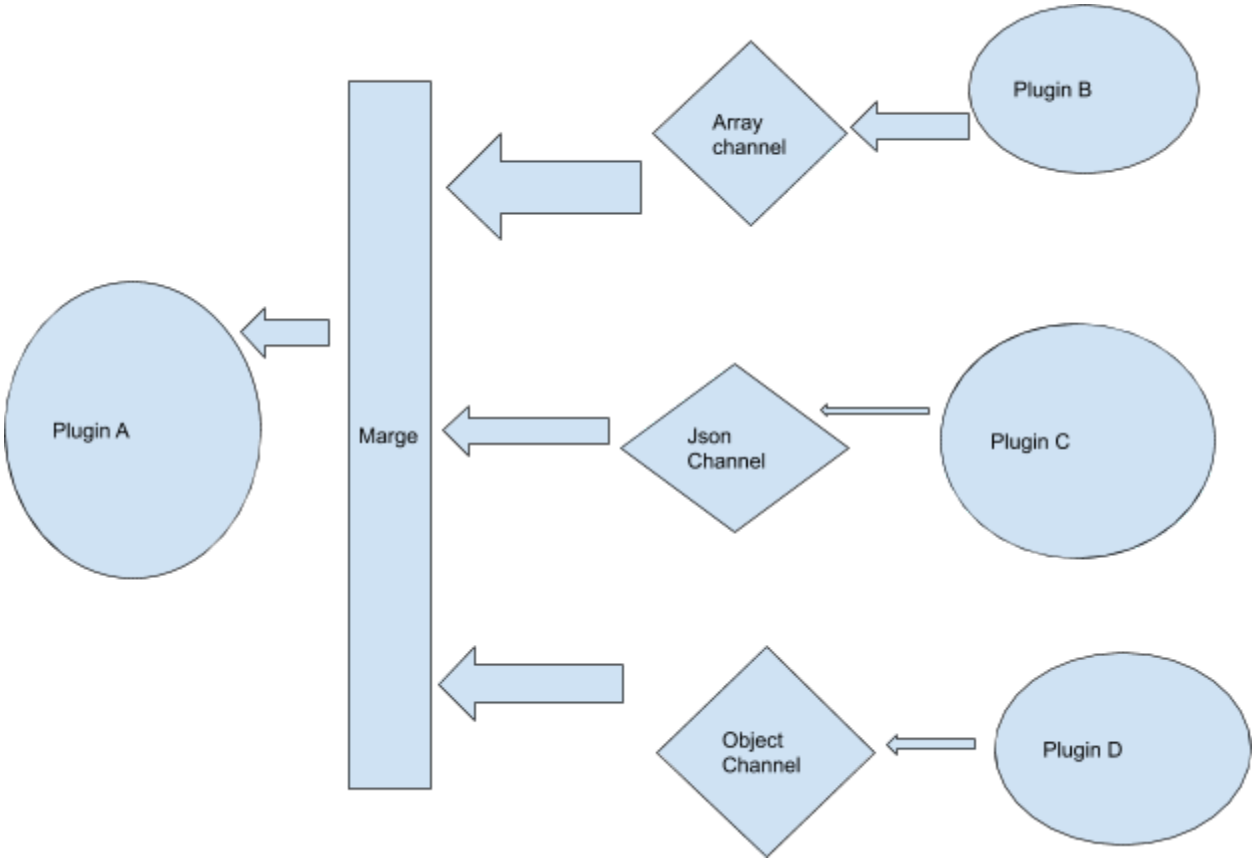
So the whole system will be a composite set .
Ex. output = F (G (K(x))) , where K, G , F are plugin , here k is the lower level plugin and F is the higher level plugin ..
Now K,G,F are and always will be super set of k,g,f



This will satisfy the requiremtns [***Alter , NoSource , NoImpact, Upgrade]***
This feature will be accomplished through **Open-Close principle** of S.O.L.I.D

Layer Merger and Channel:

Since Plugin has level / layer and a higher level plugin are depends on it's lower level plugin so there are a transection of information between layers. Now this transaction are done through a channel and there will never be a direct transaction of information between any 2 layers. There will always a sub-layer which will be responsible to merge information/ conflict resolution / authentication between the set of plugin ..



Inter Transection Model

There will be a dependency hierarchy

Inter Dependency Hierarchy which is inspired from COM..