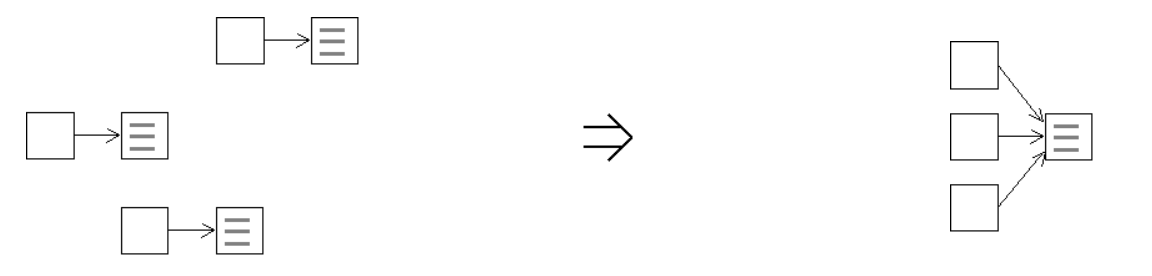
**Consolidate Abstraction**

|  |  |  |
| --- | --- | --- |
| *nearly identical components occur at various places in the design* |  | *unify the identical components and remove the redundant parts* |



**Context:**

One of the reasons code duplication is problematic is because code fragments that remain duplicated for to long tend to diverge over time. A little fix here, a few tweaks there and eventually the code has become so degraded that the similarities are barely recognizable. [Consolidate Abstraction](https://web.archive.org/web/20080916093330/http:/www.refactoring.be/thumbnails/rtn-ca.html) counters such design decay and let's you preserve the higher level design.

Opportunities to [Consolidate Abstraction](https://web.archive.org/web/20080916093330/http:/www.refactoring.be/thumbnails/rtn-ca.html) without the need for further preparation rarely occur within a design. Before being able to consolidate each of the differences among the components needs to be resolved using preparatory refactoring steps. For example: [Extract Variation](https://web.archive.org/web/20080916093330/http:/www.refactoring.be/thumbnails/extract-variation.html).

[Consolidate Abstraction](https://web.archive.org/web/20080916093330/http:/www.refactoring.be/thumbnails/rtn-ca.html) is a contraction step that naturally follows [Eliminate Duplication By Composition](https://web.archive.org/web/20080916093330/http:/www.refactoring.be/thumbnails/rtn-edbc.html) possibly in combination with [Enable Substitution With Interfaces](https://web.archive.org/web/20080916093330/http:/www.refactoring.be/thumbnails/rtn-eswi.html).