Coupling and cohesion, and the Zone of Pain

Henning Torsteinsen

Cohesion

- Cohesion refers to what module can do, internally. It is also called **Intra-Module** binding as it measures the strength of relationship of functionalities inside a module/package/component.
- Cohesion should always be high
  - A module/package/component is focused on what it should be doing, i.e. only methods relating to the intention of the class.

Cohesion

1. **Functional Cohesion:** The execution of the task related to the problem is the only concern from all the elements inside the module.
2. **Sequential Cohesion:** The output of an element is the input of other element in a module i.e., data flow between the parts.
3. **Communicational Cohesion:** Multiple elements in a module operate on same input data and produce same output data.
4. **Procedural Cohesion:** The activities in module are related by sequence, otherwise they are not related.
5. **Coincidental Cohesion:** The activities with meaningless relationship with one another are contributed by the elements in the module.

Coupling?

- **Coupling:** in software engineering is the inter-dependency or degree of relationship between multiple modules/packages/components.
- Multiple modules/packages/components that are **highly coupled** are strongly dependent on each other.
- Multiple modules/packages/components that are **loosely coupled** are not or somehow dependent on each other.

Coupling!

1. Data Coupling: When modules shared primitive data between them.
2. Stamp Coupling: When modules shared composite or structural data between them and it must be a non-global data structure. for example, Passing object or structure variable in react components.
3. Control Coupling: When data from one module is used to direct the structure of instruction execution in another.
4. External Coupling: When two modules shared externally imposed data type that is external to the software like communication protocols, device interfaces.
5. Common Coupling: When two modules shared the same global data & dependent on them, like state management in JavaScript frameworks.
6. Content Coupling: When two modules shared code and can modify the data of another module, which is the worst coupling and should be avoided.

afferent coupling and efferent coupling.

- Efferent coupling tells you the degree to which types in an assembly use types outside of it.
- Afferent coupling tells you how many types outside of the assembly depend on types inside of it.

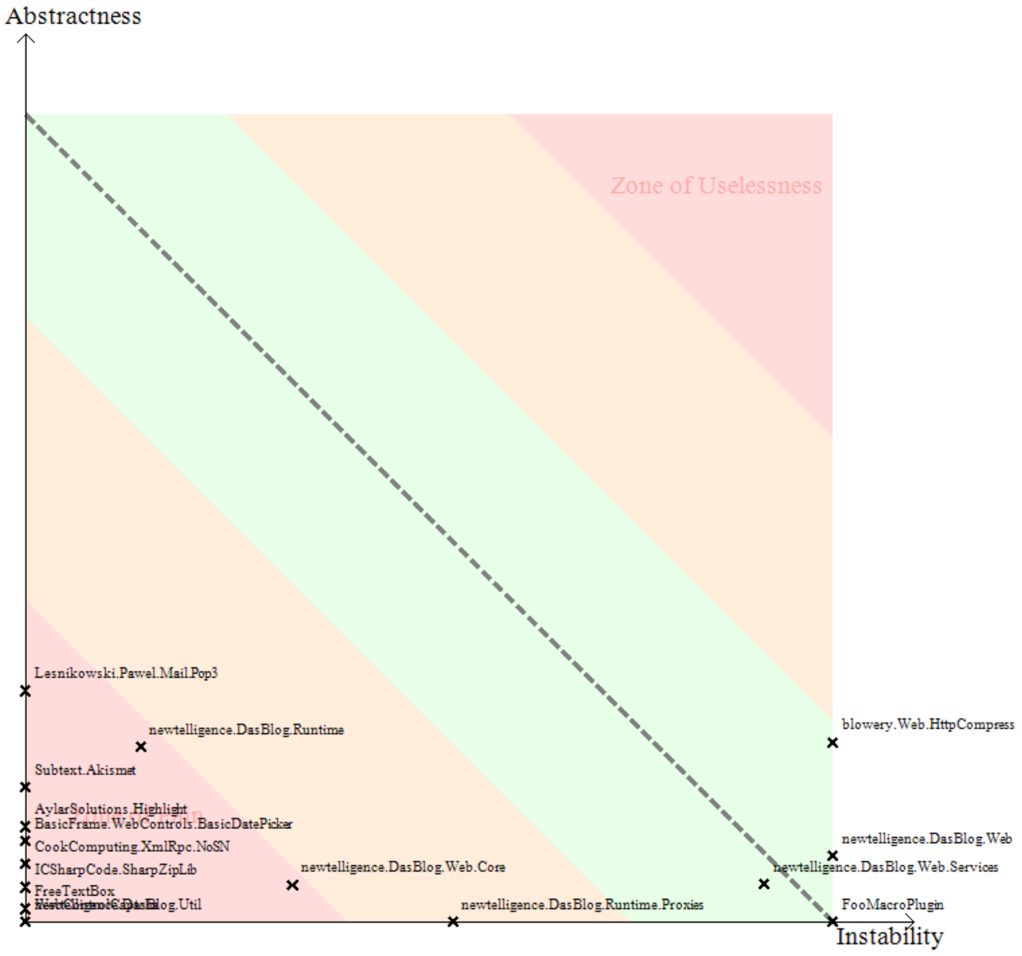
The ratio of efferent coupling to total coupling gives you the instability of the assembly. And low instability (or high stability) contributes to landing you in the zone of pain.

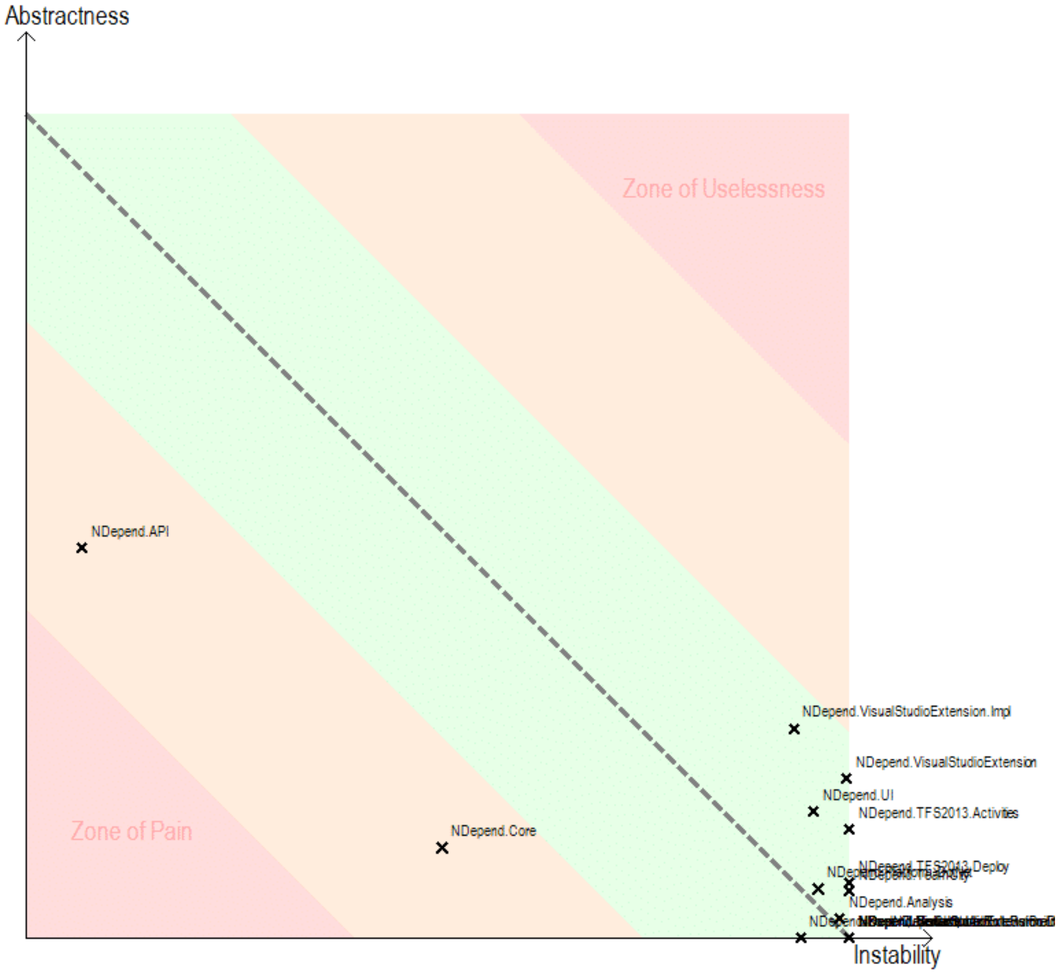
- Maximum stability occurs when an assembly couples entirely in afferent fashion. This makes intuitive sense when you think about it. Imagine an assembly dependent upon by every other assembly in your application, but dependent upon none. The weight of all of that inbound dependency would deter you heavily from changing it, creating a stable assembly. But that same, intense stability creates a situation where needing to change something can bring pain.

Abstractness

- Abstractness, in this instance, simply means the ratio of abstract (interface or declared with abstract keyword) to total types within the assembly. If your assembly contains many interfaces and abstract base classes, it will have a higher abstractness score. Likewise, if it contains none, the ratio will plummet to zero.
- Lack of abstractness pushes you toward the zone of pain. Abstractions provide *seams* that allow for testability and for changing the code by adding members, rather than heavy modification.
- Without abstraction, modification becomes risky and difficult.

Zone of Pain


Abstractness
Zone of Uselessness
Lesnikowski.Pawel.Mail.Pop3
newtelligence.DasBlog.Runtime
Subtext.Akismet
AylarSolutions.Highlight
BasicFrame.WebControls.BasicDatePicker
CookComputing.XmlRpc.NoSN
ICSharpCode.SharpZipLib
FreeTextBox
newtelligence.DasBlog.Util
newtelligence.DasBlog.Web.Core
newtelligence.DasBlog.Runtime.Proxies
blowery.Web.HttpCompress
newtelligence.DasBlog.Web
newtelligence.DasBlog.Web.Services
FooMacroPlugin
Instability


Abstractness
Zone of Uselessness
NDepend.API
Zone of Pain
NDepend.Core
NDepend.VisualStudioExtension.Impl
NDepend.VisualStudioExtension
NDepend.UI
NDepend.TFS2013.Activities
NDepend.TFS2013.Deploy
NDepend.Analysis
Instability

Summary?

- Reduce coupling to make it easier to change.
- Increase cohesion to specialize a class.
- Depend on abstractions to reduce “stability”.
- Stay away from the Zone of Pain ;-)

<End of presentation>

THANK YOU FOR YOUR TIME!

LET’S KEEP IN TOUCH:
GITHUB.COM/HENNINGNT
WWW.LINKEDIN.COM/IN/HENNINGNT/