Tracking Stability of a Software System

May 2015

What is Stability?

The stability of a software system is a measure of how much of the system is affected when a change is made. A system that is less stable is generally fragile and breaks easily when a small change is made. As a result, a system with lower stability is often harder to maintain because every change requires greater testing and validation. This also means that a system with lower stability is likely to be buggy or error prone. By contrast, a system with higher stability is robust, changes are often localized, and it is easy for developers to understand.

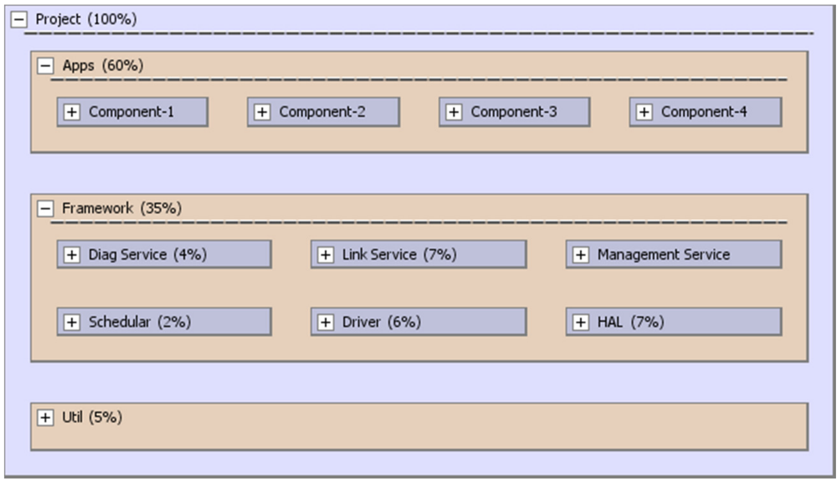
Measuring Stability

We measure stability by examining what the impact of change is on the system for every element. The overall stability number is an average number over all the elements of a system.

For a precise definition, we will first compute a number called **Average Impact**. Since Lattix projects contain dependency information for every element, we can calculate through transitive closure the number of elements that are potentially affected when an element is changed. This number is then averaged over all the elements. The Average Impact value is converted into a percentage of the total number of elements. **Stability** is the inverse of the percentage Average Impact. For instance, if the stability of system is 70%, it means that 30% of the elements on the average are affected when an element is changed and 70% are unaffected. Note that stability is computed as a percentage of the size of the system, and, therefore, does not increase or decrease simply because the system is larger or smaller.

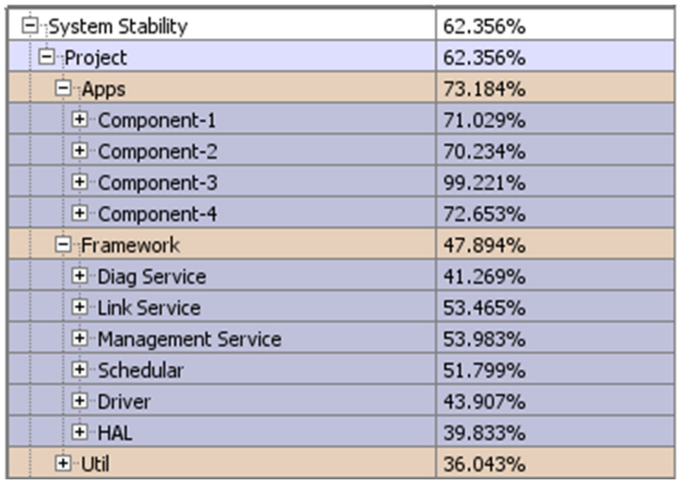
Stability of Software Systems

The architecture of a software system has a profound impact on its stability.



For instance, if we create a set of applications on a common framework and then change the framework we will affect all of the applications. On the other hand, if we change an application then the impact will be much lower.

Therefore, it is essential to understand not just the Stability value of the entire system, but to understand what the source of stability is.

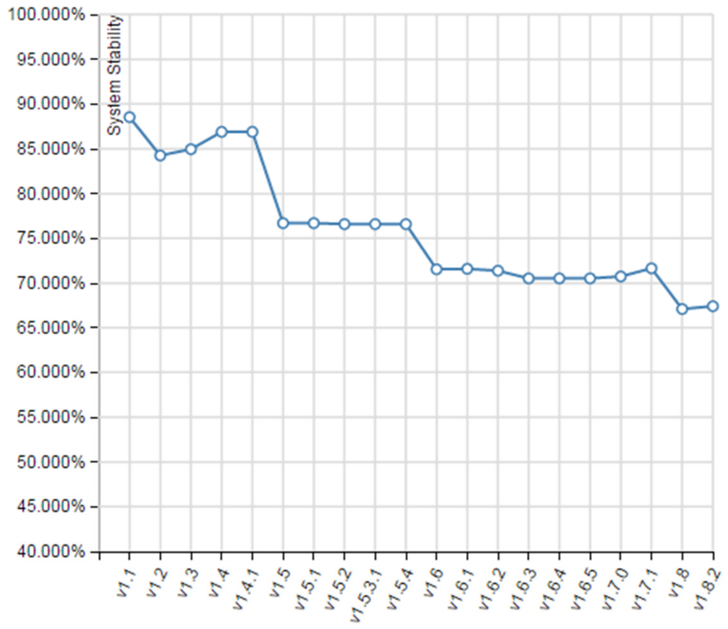


System Stability	62.356%
Project	62.356%
Apps	73.184%
Component-1	71.029%
Component-2	70.234%
Component-3	99.221%
Component-4	72.653%
Framework	47.894%
Diag Service	41.269%
Link Service	53.465%
Management Service	53.983%
Schedular	51.799%
Driver	43.907%
HAL	39.833%
Util	36.043%

The total stability of *Project* is about 62%, but the *Apps* layer has a stability of 73%, while *Framework* and *Util* have lower stability. Notice that *Component-3* has the highest stability. Upon investigation it was found that *Framework* did not have a cyclic dependency on *Component-3*. This meant that changes to *Component-3* did not affect the rest of the system. On the other hand, *Framework* had dependencies on the other components, which meant that changes to those components had a much larger impact.

Tracking Stability

It's possible to track how the stability of software changes over time. We have heard how software architecture erodes over time and how software becomes harder to maintain. The Stability metric gives us a measurable insight into this phenomenon.



This project was tracked over 20 versions. Notice how the stability was the highest with the first release. Since then it has edged down in slow increments. The net result is that an incremental change to the system requires ever increasing testing and verification.

Conclusion

The stability of most systems tends to reduce over time. The notion of stability corresponds closely to the idea that design erodes over time and becomes harder to maintain as systems grow older. Monitoring stability can help us recognize the importance of maintaining a clean and modular design. The erosion of architecture is often considered a key part of technical debt and serves as a tax on the cost of future improvements. Lattix makes architecture management a part of your continuous integration and makes architecture management easy and integral to the development life cycle.

Try Lattix Architect on your Project

Is your software complex, buggy and hard to maintain? Lattix provides support for a wide range of technologies including C/C++, Java, .NET, Fortran, Ada, Javascript, Actionscript, Pascal, Python, UML/SysML, Rhapsody, SparxEA, Oracle, SQLServer, SyBase, LDI and Excel.

Do you want to see what your architecture looks like and what you can do to modularize it? We have helped companies all over the world improve the quality of software and we can help you achieve the same results.

Contact Lattix at sales@lattix.com or call 978-664-5050.