Quantifying requirements volatility effects
G.P. Kulk, C. Verhoef
Summary
Elen Kivi
Mirjam Rauba

Requirements describe what stakeholders want. Stakeholders wishes and budget, schedule are in conflict often. Requirements volatility is regular. It have to be kept under control because otherwise requirements volatility could be effect project fail. But how to control it? We have to calculate when is requirement volatility healthy and when it causes chaos. Requirements volatility is healthy if changed requirements describe better client’s needs.

Volatility in three environments
In this paper we look three case studies:
- bancassurance sector
  ○ low risk subportfolio
  ○ 84 projects 16 500 functions point together
- governmental projects
  ○ high risk subportfolio
  ○ 6 projects
- software product
  ○ low risk subportfolio
There are three forms of requirements change in this paper:
- Creep - requirements are added
- Scrap - requirements are left out
- Churn - requirements are changed
One way to measure change rates of requirements is calculate monthly requirements volatility rate. This method expresses the time aspect.
Calculate monthly volatility rate with duration of t months

\[ \text{SizeAtEnd} = \text{SizeAtStart} \times \left(1 + \frac{r}{100}\right)^t \]
Calculate the compound monthly volatility r

\[ r = \sqrt[12]{\frac{\text{SizeAtEnd}}{\text{SizeAtStart}}} - 1 \times 100 \]
To calculate this rate the size of the requirements has to be determined at least twice. Third variable is time.

Example: we use volatility 2% per months, it is calculated on a 36-month project with size of 10 000 function points

\( (1 + 2 \%/100)^{36} = 1.02^{36} = 2.04 \)
It is 10000 * 2.04 = 20400 function points after three years
Obviously it is an unhealthy project because that doubles its requirements during development
It could be calculate when the size at the end being 2 times the size at the start.
\[(1 + \frac{r}{100})t = 2 \Rightarrow t = \frac{\log 2}{\log(1+r/100)} \Rightarrow t = \frac{\log 2}{r}\]

Also could be calculate project maximum length if we know rate \(r\)

\[b = 1 + \frac{r}{100} \Rightarrow t = \frac{\log(\log(b))}{\log(b)} \quad b > 0\]

Next we look \(p\)-proportional growth also we can call it as tolerance factor. It is easy to understand that if growth is 100% then the project is failing.

The equation for calculating \(p\):

\[p = p(b,t) = \log b^* b^t\]

and maximum \(p\) can calculate as

\[P = \max(p)\]

Maximum \(p\) indicates which growth is acceptable.

Next we can calculate which projects are healthy and which are unhealthy. For that we have to compare project’s durations and size.

\(\pi\) - ratio - it indicates how close a project has approached the danger zone.

If \(\pi > 1 \Rightarrow\) project has experienced more than proportional requirements growth

\[\pi_p = \pi_p (r_{act}^* t, p) = \frac{r_{act}}{r_{\pi}} = \frac{W(p^* t)}{\log f} = (e^{W(p^* t)} - 1) * 100\]

\(\rho\) - ratio - this metrics give higher weight to larger project, it takes account size

\[r_{\rho} = (e^{\frac{W(p^* t)}{\log f}} - 1) * 100 \quad (1 + \frac{r}{100})^t > 0, \quad f > 1\]

Next we introduce formula, where is used duration and size

\[\rho_p = \rho_p (r_{act}^* t, p, f) = \frac{r_{act}}{r_{\rho}} = (e^{\frac{W(p^* t)}{\log f}} - 1) * 100\]

\(f\) - size of the project

Now we know how calculate maximum volatility rate is not so easy we have count project size, durations. From there formulas shorter project could have higher requirements volatility then longer projects and project is under control.

**Volatility variations for outsourcing**

Outsourcing is usual and many organizations use it. Does it depend volatility? It should decrease. Reasons:

- better management
- development assignments try complete as soon as possible
- partners won’t charging more hours than estimated
- contract

Is there effect to volatility rate than if the bancassurance portfolio is outsourced or not? In real life outsourced projects are usually larger because it is quite difficult to compare them. To
further analyse there are not statistically significant difference. In paper outsourced projects and in-house projects display similar requirements volatility characteristics.

The boomerang
After comparing different things it shows that larger project have lower productivity than smaller ones. Let’s compare spent hours on the project and productivity. We don’t have data about outsource project so we look only in-house projects. To calculate productivity function
\[ f_{ppm} = \frac{677.6572}{f} \]
\[ f - function points \]
this functions based on the bancassurance portfolio and expressed function points per staff month. These projects have high monthly growth and high productivity.
Next we divide bancassurance portfolio projects to three sub-portfolios. These have following characteristics:
- high volatility, but low productivity (It contains very complex projects)
- high volatility, but high productivity (Back-office systems)
- low volatility, but moderate productivity (International payments \(\rightarrow\) clear requirements)
In next table are values corresponding, it explains previous characteristics.

<table>
<thead>
<tr>
<th></th>
<th>Low volatility</th>
<th>High volatility</th>
</tr>
</thead>
<tbody>
<tr>
<td>High productivity</td>
<td>( r &lt; 15% ) fppm &gt; 35</td>
<td>( r &gt; 15% ) fppm &gt; 35</td>
</tr>
<tr>
<td>Low productivity</td>
<td>( r &lt; 15% ) fppm &lt; 35</td>
<td>( r &gt; 15% ) fppm &gt; 35</td>
</tr>
</tbody>
</table>

*Table 1 (In original paper page 166)*

With table 1 and function it is possible to find projects which need more management.

Monitoring the volatility of an ongoing portfolio
Next we look how assess the volatility of ongoing projects at low cost. Doing two measures using total number of lines of code. It is possible to consider the size of a project through backfiring. Backfiring is function conversion rate for the lines of code. If we know daily backfiring then we can calculate volatility rate:
\[ r = (\frac{\text{SizeAtDay}_{n+1}}{\text{SizeAtDay}_{n}})^{30.5} - 1\)\)* 10C
\[ 30.5 \text{ is average number of days in a month} \]
This method is really cheap one because it is easy calculate lines of code. Further the average volatility should have been lower because there could be also requirements shrink. Example code removal or replaces clones. But in bancassurance portfolio chance on growth is larger than shrinkage. There is no problem if functionality is added.
By applying backfiring we monitored the volatility of a product line. If we add data line we established volatility benchmark.
Requirements volatility dashboard

Next we introduce volatility dashboard for IT governance. It contains four volatility methods:

- requirements volatility $r$
- volatility tolerance $p$
- $\pi$-ratio
- $\rho$-ratio

There is used $p=0.68$ which is maximum tolerance factor so if $p$ is bigger than project colored as red. Creating dashboard for all projects gives good overview of the project out of control.

Conclusion

Volatility requirement are fact of life but it have been problem for decades. We looked different events:

- Projects increase
- Projects change
- Project management and development reporting

If you need way of interpreting the data for project managers than this article can be helpful. It helps to understand and calculate project volatility.

In this paper by using accurate size estimates, function point analyses, executed at different moments in the project life cycle, could be calculate requirements volatility.

We did not find difference in volatility in-house and outsource projects but these projects which were counted thrice had higher volatility then projected which counted twice.

We showed formulas how calculate tolerance for volatility and can say when project is healthy and when it is unhealthy. We could take into account the size and duration of project. After that we also introduced volatility dashboard what helps easily monitor the volatility.

The most important is that these methods help manage software project little bite better.