On the Complexity of Requirements Flow-down Structures

Jeremy Dick and Brendan Jones
Integrate Systems Engineering
{jeremy.dick, brendan.jones}@integrate.biz

Abstract. The process of systematically decomposing and tracing requirements results in graphs in which the tracing diverges and converges as the requirements are re-factored through the layers of design. This paper reports in work in progress to assess measures for the complexity of requirements decomposition graphs, including the use of the standard definition of cyclomatic complexity. It discusses what the complexity of requirements flow-down means intuitively, and establishes some informal assessment criteria. A number of measures are motivated and defined, and then assessed against the criteria. Finally, the paper discusses the use of such measures as viable tools for assisting in the review of requirements development, and the future directions the on-going work should take.

Introduction

Requirements development is the process of systematically decomposing and tracing requirements through multiple levels of abstraction; e.g. developing a user requirement progressively into system requirements, sub-system requirements and component requirements.

The resulting structures are directed graphs in which the decomposition paths diverge and converge as the requirements are re-factored through the layers of design. In general, one expects divergence, since each layer adds design detail to the layer above. Nevertheless, convergence will occur; for instance, each time a single design feature satisfies multiple requirements.

Where complex systems are being developed, different teams may address each layer of design and the requirements decomposition associated with it. With no one team in charge, the complexity of the whole chain of decomposition arising from a single high level requirement may not be evident to an individual engineer. Hence the desire exists to have a tool for high-lighting complexity.

Such measures could be used in the following scenarios:

1. As a metric for the flow-down complexity within a whole requirements layer. This may help answer the question: how complex is the design relating to this set of requirements?
2. As a metric for identifying individual requirements as those that give rise to flow-down complexity over a certain threshold, or significantly higher or lower than the standard variance. This may help answer the question: where should review effort be focused?
3. As a means of assessing the complexity of the relationships between clusters of requirements. This may help answer the question: what would be the cost associated with changing a requirement?

Previous work (Hull, Jackson, Dick 2002) has recommended the use of flow-down statistics to

identify requirements whose decomposition is significantly different from others, and to apply special consideration to such requirements during reviews. This paper amplifies that suggestion by proposing the use of a number of complexity measures as means of assessing flow-down complexity.
The work reported in still in progress, and so only tentative conclusions can be drawn at this stage.

Graph Complexity

McCabe (McCabe 1976) originally proposed his complexity measure in the context of software development; the measure corresponds to the number of linearly independent paths through a piece of code, and is an indicator of the amount of testing required. The approach involves reducing the structure of the code into a strongly connected graph, and applying the measure to the graph. Indeed, the origins of the measure are in general graph theory, and it can be used to measure graph complexity in many other contexts. Therefore, we propose that McCabe's measure can be adapted to requirements flow-down graphs.
In the context of requirements management, we wish to consider flow-down graphs that start from one or more high-level requirements and finish in one or more low-level requirements. In McCabe's formulation, this is equivalent to a software program that has multiple entry and exit points. Such graphs can always be made single entry and exit by adding an imaginary single entry above the existing ones and connecting them all; similarly for the exit points.
Therefore, the formulation of the McCabe measure that we use is:

$$C = L - R + R_E + R_X$$

where C is the measure, L is the number of links, R is the number of requirements, R_E is the number of entry requirements (those with no in-flow in the directed graph), and R_X is the number of exit requirements (those with no out-flow in the directed graph).
The sense of this measure is that, the higher the value of C, the greater the complexity, with $C = 1$ the minimum.
In the context of the flow-down of requirements, the McCabe metric, in effect, counts the number of paths through the structure. While we view this as one useful measure of complexity, we recognize that there are significant differences between the structure of code and requirements flow-down structures.
Firstly, requirements flow-down graphs have no loops – or should not have, since a loop corresponds to a low-level requirement being decomposed into a higher-level requirement.
Secondly, requirements development is expected to introduce detail through systematic design, and so it is normal for there to be more low-level requirements than high-level. One view of complexity is the degree to which lower level requirements are inter-connected – in other words, the extent to which the flow-down is more than just a tree, and some low-level requirements respond to multiple high-level requirements. While McCabe cyclomatic complexity treats divergence and convergence symmetrically, we should also consider measures that tolerate divergence more than convergence.
Thirdly, we are interested in layers of requirements, in the sense that complexity may be measured between a layer of entry requirements and a layer of exit requirements regardless of the number of intermediate requirements (non-entry and non-exit requirements).
These differences suggest that cyclomatic complexity should not be the only measure for requirement decomposition.

A Measure of Connectedness

In this section, we are going to define a metric that measures the degree of inter-connection that exists in a graph. The metric normalizes cyclomatic complexity for the expansion in the number of requirements resulting from divergence in decomposition, which is the normal case. The intuition driving this is the idea of saturation in the number of links. In other words, the idea that complexity is related to the number of links more than is necessary for a hierarchy. To make a tree, you need $R - R_E$ links (where R and R_E are as defined above.) Any more links than this, and there is added complexity.

Therefore, we define the saturation of a graph, S, as the difference between the actual number of links, L, and the minimum number of links necessary to make each part of the graph a tree, $R - R_E$:

$$S = L - R + R_E$$

where L, R and R_E are as defined above.

The sense of this measure is that the higher the value of S the more complex the graph, with $S = 0$ the minimum.

Our intuition is re-enforced by the observation that the definition of S is identical to the definition of C but without the term R_X, thus removing the divergence of the graph from the equation.

How intuitive S is as a measure will be discussed in the section entitled Comparing Measures below.

A Measure of Expansion

Another simple measure of the complexity of a requirements decomposition graph is how the number of requirements expands in a layer, regardless of the complexity within the layer.

We define the expansion ratio, E, of a graph:

$$E = R_X / R_E$$

where R_E and R_X are as defined above.

Note that intermediate requirements are not taken into account in this measure, making it a kind of "black-box" measure.

Relative Complexity

The measures defined so far are absolute measures, in the sense that they don't take into account the size of the graph, which means that the bigger the overall graph, the higher the complexity is likely to be. In this section, we try to turn these measures into relative measures by normalizing them for the size of the graph.

The size of a graph can be measured by the number of links, the number of requirements, the number of entry requirements or the number of exit requirements.

We choose two ways of making McCabe's measure relative to the size of the graph: by the number of entry requirements, RC_E, and by the number of exit requirements, RC_X:

$$RC_E = C / R_E \qquad RC_X = C / R_X$$

where C, R_E and R_X are defined as above.

These measures, in effect, average the overall complexity of the graph over the entry and exit requirements respectively. They give a real number value greater than 1.0, with higher values representing greater relative complexity. Note that, if $E = 1.0$, then $R_E = R_X$, and thus $RC_E =$

RC_X.
With regard to saturation, this is defined in terms of the number of links, so we choose the total number of links as the measure of the size of the graph, giving Relative Saturation, *RS*, as:

$$RS = S / L$$

This measure gives a real value between 0.0 and 1.0, with those values closest to 1.0 being the highest saturation.

Comparing Measures

In this section, we compare the various measures proposed to assess their performance against our intuition.
Figure 1 shows six simple flow-down shapes showing the complexity measures, *C*, *S*, *E*, RC_E, RC_X and *RS*, of each. The graphs are implicitly directed from top to bottom.

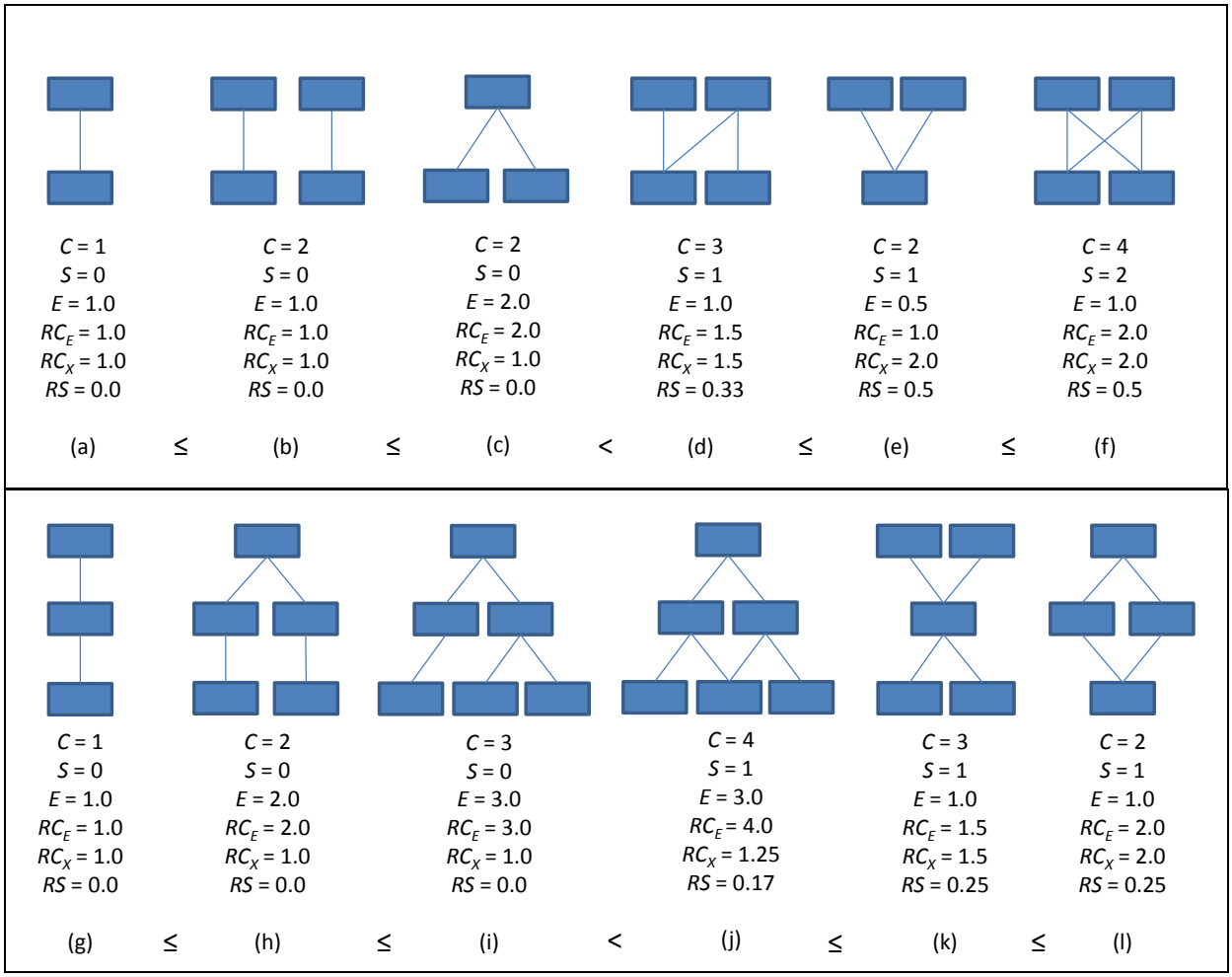

Figure 1. Simple flow-down graphs

Graphs (a) to (f) and (g) to (j) are arranged in intuitive order of complexity. Graph (k) is considered less complex than (l) in that the number of intermediate requirements is higher relative to the number of entry and exit requirements. A similar argument applies to (d) and (e). It can be seen that cyclomatic complexity, *C*, does not respect this intuitive ordering. The

saturation measure, *S*, which accepts divergence as normal, does better, but does not distinguish between (d) and (e), nor between (k) and (l).
The measures that best reflect our intuition are RC_X and *RS*.
Figure 2 shows some more complex examples of clusters of requirements that demonstrate more clearly the difference between the absolute and relative complexity values.

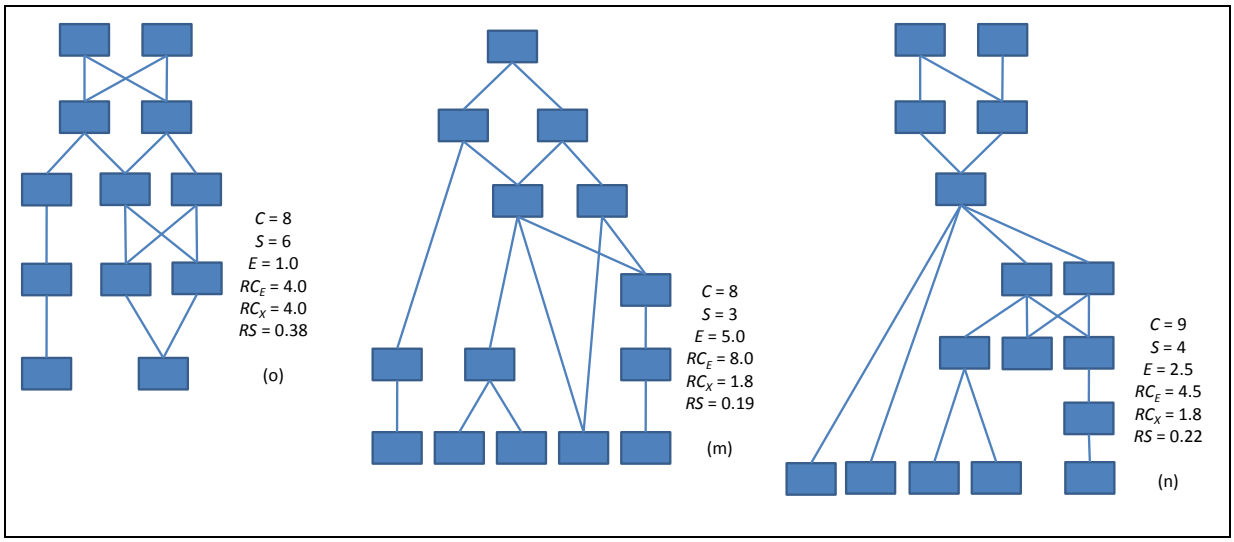

Figure 2. More complex requirements clusters

Using Complexity Measures

Three scenarios for the use of flow-down complexity measures were listed in the introduction. Here is a brief discussion about how the measures discussed in this paper may be deployed in these scenarios.

1) *Complexity within a requirements layer*: Here we suggest the use of relative measures, RC_X and *RS*, both of which are biased towards convergence rather than divergence as the indication of complexity. The expansion measure, *E*, is also a simple but useful measure in this context.

2) *Complexity flowing out of an individual requirement*: Here we suggest the use of one of the absolute measures *C* or *S*. The former gives a measure that takes into account the expansion of the requirement through the layers, whereas the latter takes just convergence into account.

3) Complexity with a cluster of requirements: Here we suggest the use of the absolute measure, *C*, since impact analysis usually has to consider both upwards and downwards connections, suggesting a symmetrical treatment of complexity.

Figure 3 shows an example drawn from some real project data. There are 4 development layers present, marked by the dashed horizontal lines. The eye is drawn to parts of the graph where links are clustered, and which may be construed as areas of complexity. The requirements marked by the letters A through G were carefully reviewed, and one area was found to be erroneous: all but one of the multiple links flowing out of requirement B were redundant. The complexity associated with requirements C to G was considered correct, since there are two groups of closely related requirements whose decomposition is fully interconnected.

The complexity measures *C*, *S*, RC_X and *RS* are shown per layer, and reflect the absolute and relative complexity of each layer. The values are higher where the clusters of links occur.

To assess the complexity of the flow-down associated with each node, we will consider the graph which is the union of the graph looking up from the subject node and the graph looking

down from the subject node. For instance, if the subject node is the one marked "X" in Figure 3, then the graph comprises those nodes filled in white and the links marked in bold.

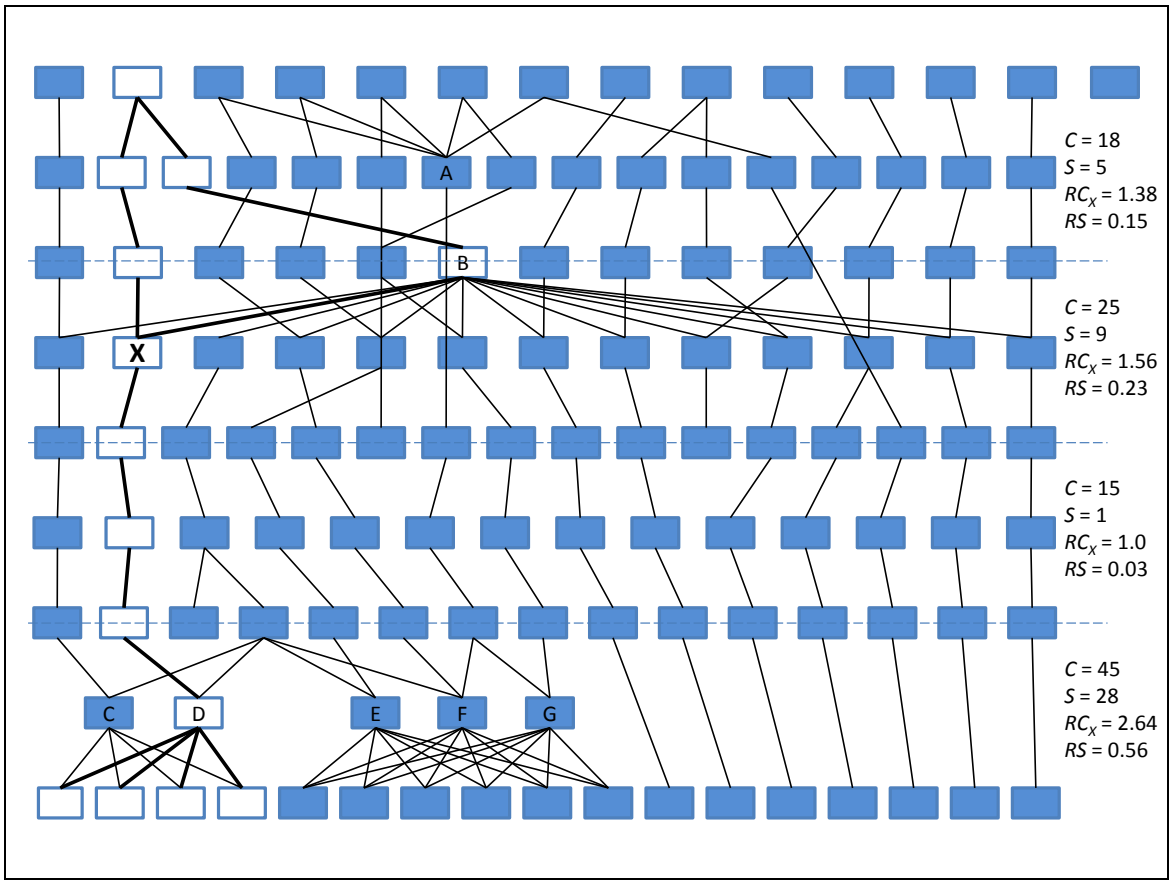

Figure 3. Real flow-down example

In Figure 4, the measure *C* is given for such a graph flowing out of each individual requirement. Figure 5 shows the same for the *S* measure.

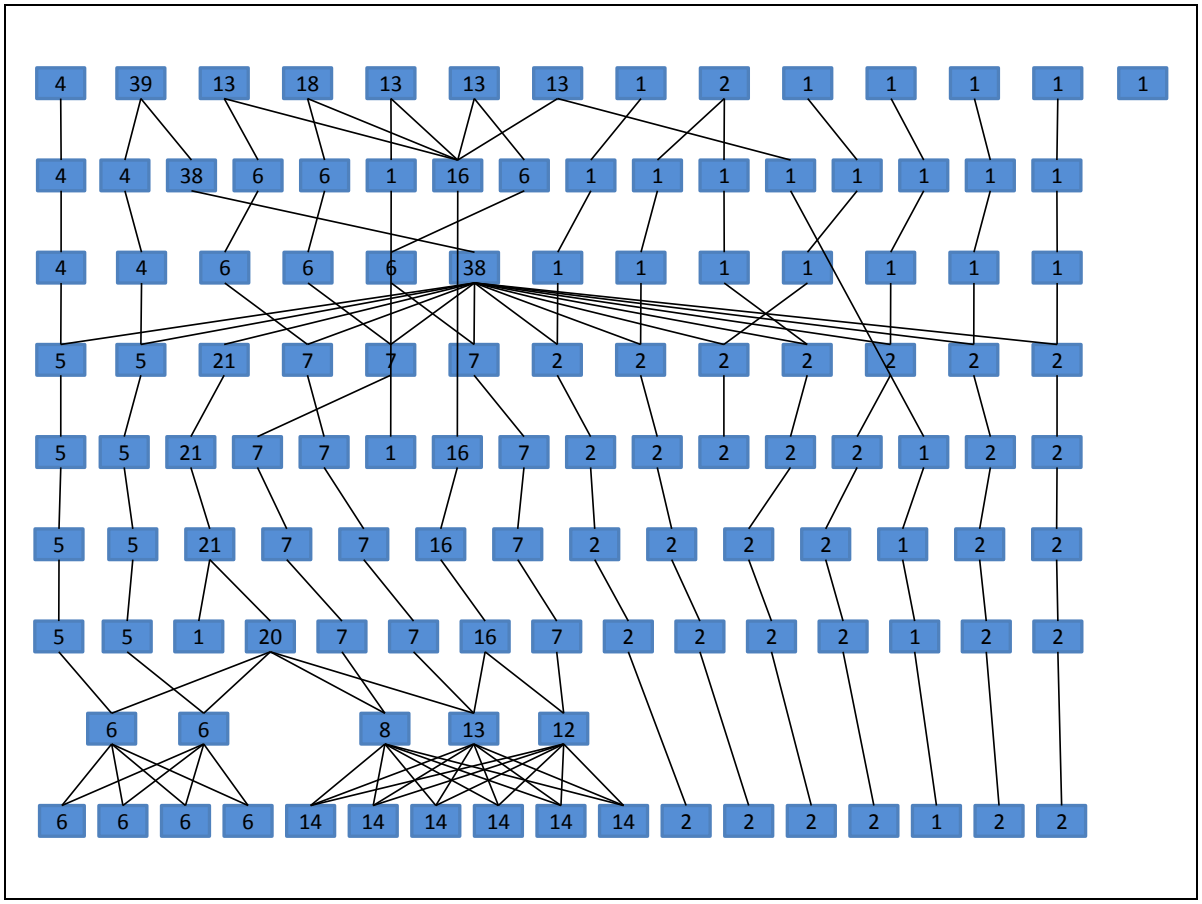

Figure 4. Cyclomatic complexity (*C*) values for real flow-down example

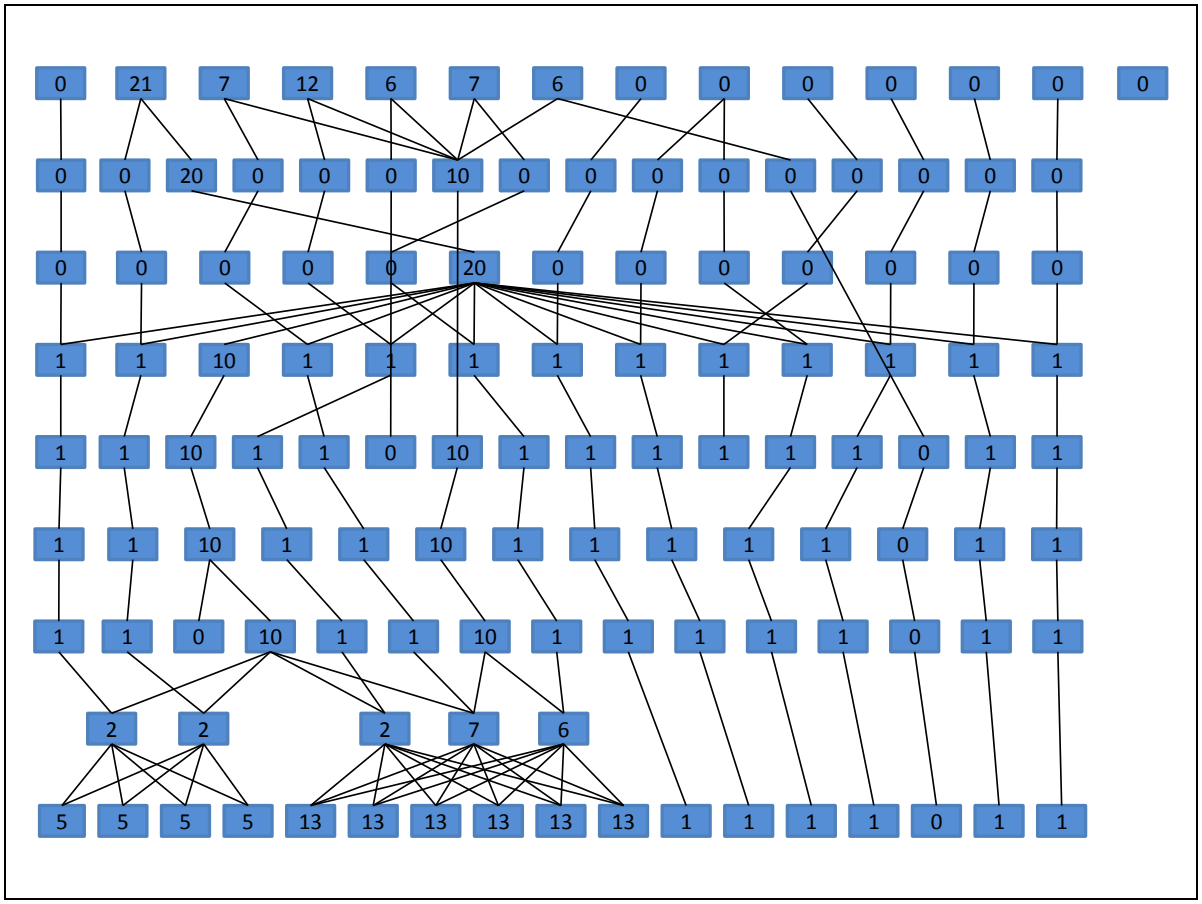

Figure 5. Saturation (*S*) values for corrected flow-down example

Both these properties allow the step-wise location of complexity in the graph by starting with individual requirements in the top layer, and following down the chains of high-complexity in the subsequent layers. The difference is that *C* considers divergence and convergence symmetrically, whereas *S* considers only convergence.

Removing the erroneous links on requirement A reduces the complexity of the graph as a whole. Figure 6 and Figure 7 respectively show the *C* and *S* complexities of each node.

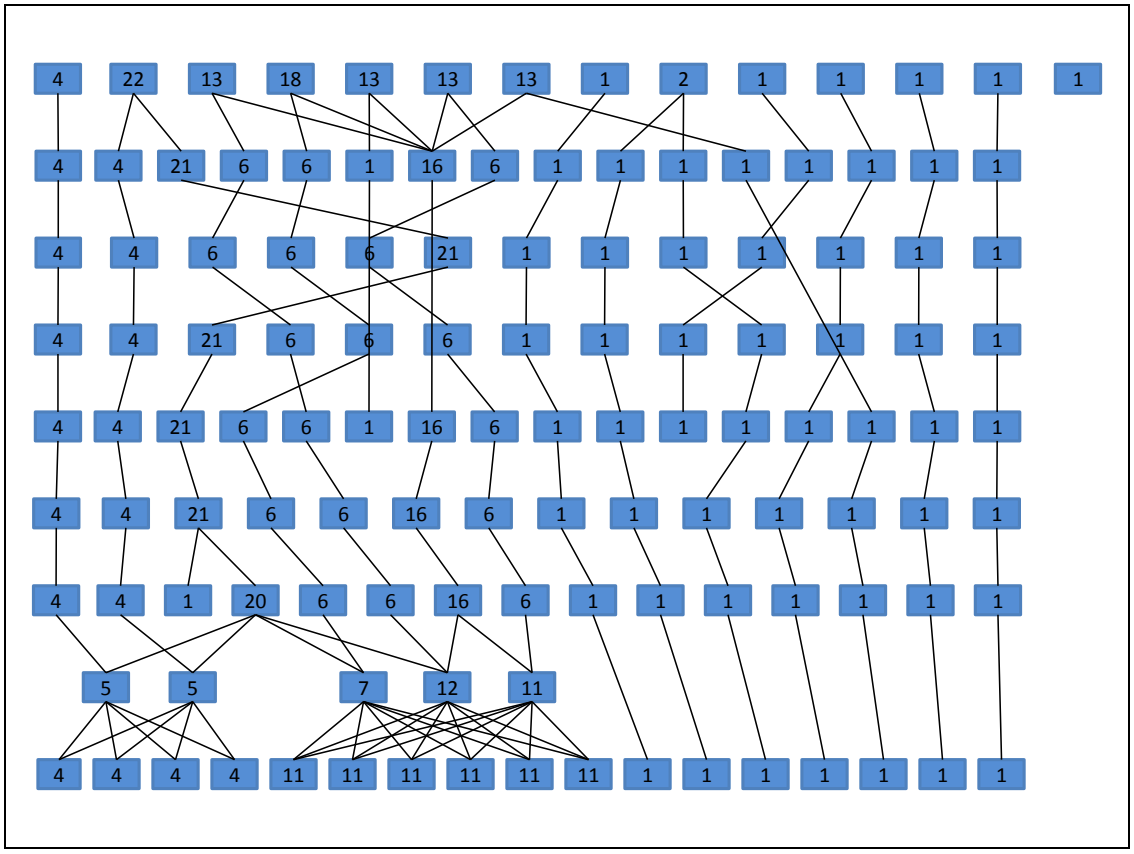

Figure 6. Cyclometric complexity (*C*) values for corrected flow-down example

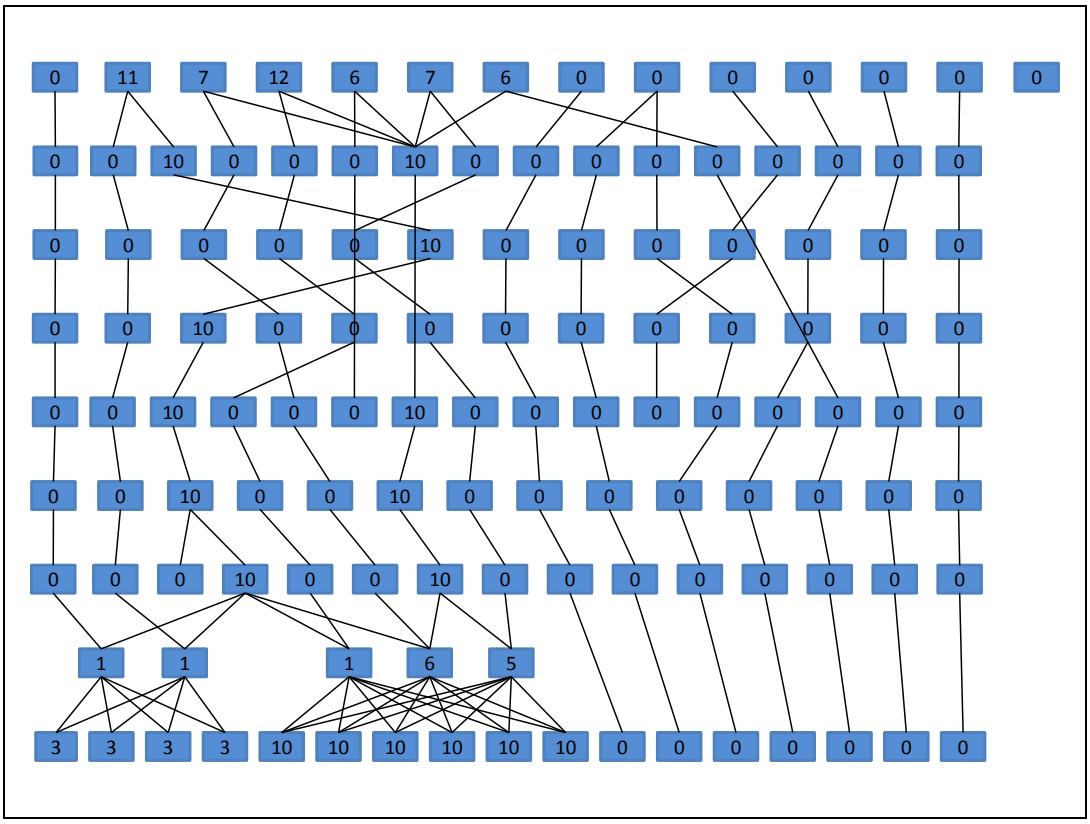

Figure 7. Saturation (*S*) values for corrected flow-down example

Summary and Conclusions

We have reported on the early results of an assessment a number of measures of complexity with requirements flow-down structures, and tentatively concluded that different measures are useful for different assessment scenarios. In particular, cyclomatic complexity offers a symmetrical assessment of complexity, which is well suited to complexity assessment related impact analysis, whereas measures that provide an asymmetrical treatment in favour of convergence are more suited to other scenarios.
The potential uses of these measures are summarized in Table 1.

Table 1: Summary of complexity measures discussed in this paper

	Description	***Absolute and Relative forms***	***Potential use***
C	*Cyclomatic complexity*: the number of independent paths through the requirements flow-down.	$C = L - R + R_E + R_X$	As an indication of the number of potentially impacted requirements during impact analysis.
		$RC_X = C / R_X$	As an indication of the complexity of requirements within a layer, in comparison with other layers.
E	*Expansion*: the growth in the number of requirements between layers	$E = R_X / R_E$	As an indication of the overall growth in the number of requirements through layers, regardless of the complexity within the layer.
		n/a	n/a
S	*Saturation*: The number of links more than is necessary for a tree	$S = L - R + R_E$	As an indication of complexity due to convergence rather than divergence, which is the norm.
		$RS = S / L$	As an indication of the complexity (in terms of convergence) of requirements within a layer, in comparison with other layers.

Clearly further work is needed to validate these measures in live project environments. Comparison of requirements flow-down complexity with existing measures of design complexity would also strengthen the assessment.

References

Hull E., Jackson K., and Dick J. 2002. *Requirements Engineering*, 1st ed., Springer, 2002, pp.152–158. (Hull, Jackson and Dick 2002)

McCabe, Thomas J. 1976. “A Complexity Measure,” IEEE Trans on Software Eng., Vol. Se-2, No.4. (McCabe 1976)