



Presentation prepared by Michael Egan for US Federal  
CASIC Workshops, March 2005

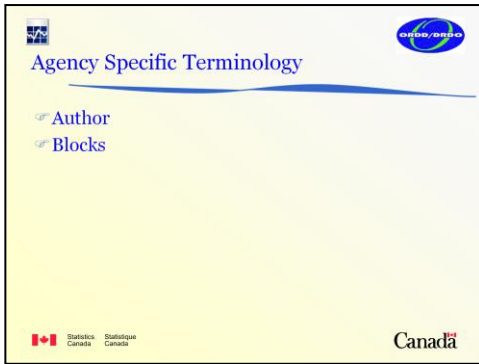
---



## Presentation Contents

- ☞ Why
- ☞ Examples
  - ♣ Problem logs
  - ♣ Technical design
  - ♣ Tracking development
- ☞ Résumé





An author is a person who converts questionnaire specifications into Blaise code. Synonyms would be "developer" and "programmer."

A block is a collection of questions, answers, edits and flows, normally relating to the same topic. A block can contain another block, too.

Why metrics?

- Evaluate practices
  - Better understand processes
  - Measure impact of changes

Statistics Canada / Statistique Canada


Canada

Metrics can help you understand your development practices. The more that development processes can be quantified, the better able one is to estimate one's services and timelines. Also, you would be well placed to measure the impact of new practices.

If you had to sell senior management on a new development practice, what would you tell them? You'd tell them that, using this new practice, development will take less staff time, or less calendar time. They'll want to know how much less. If you've got the appropriate metrics in place (or you can put them in place), you can provide a reliable answer.

2.856 thoughts on metrics

- 100% accurate measurements are not possible
- 800 > 0
- There's more to it than people, time and money



Statistics Canada / Statistique Canada

Canada

Measurements are estimates, so it is better to use round numbers than three digits of precision.

A picture is worth a thousand words, but a slightly blurred picture is far better than a blank canvas. Over time, the picture can be improved. Which means: an imperfect measurement is better than no measurement at all, and an imperfect measurement can be improved over time.

What else can be measured? Problem logs; Complexity.

1) Problem Logs - Trigger

☞ "We're submitting too many problem logs during block testing."



Statistics Canada / Statistique Canada

Canada

Several years ago, a client said that they were registering too many problem logs during block testing. There was an implication that the authoring effort had been shoddy.

1) Problem Logs – Response

- Let's look at the metrics
- There were lots of problem logs
- There were lots of code changes
- There were lots of specification changes

Statistics Canada / Statistique Canada

Canada

We examined every problem log that had been submitted for the survey (over 700 at the time, it took a couple of days - eventually over 1,000). We categorized logs by source of error (specs/code) and nature of change (logic/text/field def).

We decided to look at code changes, rather than at problem logs, because it was a more accurate measurement of the process (problem logs are requests to change the code). Sometimes a problem log resulted in several code changes, sometimes several problem logs resulted in one code change. In the end, there were about a 1:1 relationship between code changes and problem logs.

For this survey, we found that 90% of the code changes were related to specification changes. 9 times out of 10, we were changing code that was working exactly as specified.

1) Problem Logs – What changed?

- ☞ Rigorous and formal spec reviews
- ☞ Goal 1: Reduce testing burden
- ☞ Goal 2: Reduce authoring burden

Statistics Canada / Statistique Canada

Canada

We continued studying problem logs, eventually covering 11 surveys (varying sizes, clients, topics). We found that 80% of code changes were spec related (never less than 70% at the survey level).

As a result, we introduced rigorous block specification reviews as a standard part of development. **We ensured that clients knew this would be done, during our initial talks with them.** They knew that specs would be considered “final” only after they had been approved - not when they were first delivered.

Authors who have been around for a while tell me that the quality of initial specifications has greatly improved (are they better because clients know they will be reviewed?). We also think that relationships have improved, due to our earlier involvement and due to reduced testing burdens.

Block testing goes much better since we have introduced rigorous block specification reviews.



1) Problem Logs – What's next?

- Ticket system: generic testing reporting tool
- Continue to look at problems
  - Sources of problems; nature of problems
- Look for patterns

Statistics Canada / Statistique Canada | Canada

With a significant reduction in code changes during testing, authors will obviously spend less time on a project. And that will have an impact on our development estimates and on development time lines. A little work up front pays off mightily!

We have just introduced a new development stage, internal QA, and we expect to see differences in the nature of tickets submitted by clients. QA is a round of testing performed by a separate group in ORDD, and we expect that will identify almost all of the authoring errors. Ideally, if there is no change to the specifications, clients will approve blocks on the first round of their testing.

---

To make data comparable across surveys, you need to have some sort of divisor (number of code changes is the dividend). The obvious divisor is "fields", but we prefer to use "objects" (code changes per object).

**"Objects" is a lead-in to the next section.**

2) Tech Design - Trigger

“Testing is hard for us and it takes too much time.”



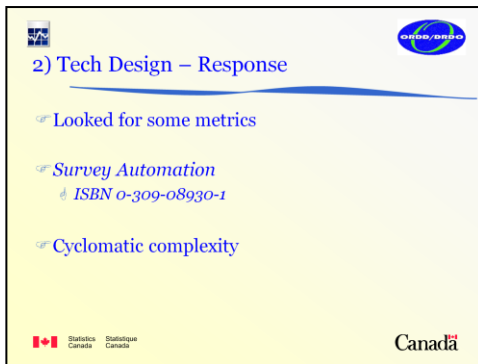
Canada

Statistics Canada / Statistique Canada

One day, a client said that testing was hard and it took too long. It was not the first time a client had said this.

On this particular day, the immediate (unspoken) response was “well, if you want testing to be easier and go faster, make your blocks less complicated.”

How can you estimate how complex a block is? How do you know when enough is enough?



We were looking for some way to quantify complexity, and found this book.

## Survey Automation

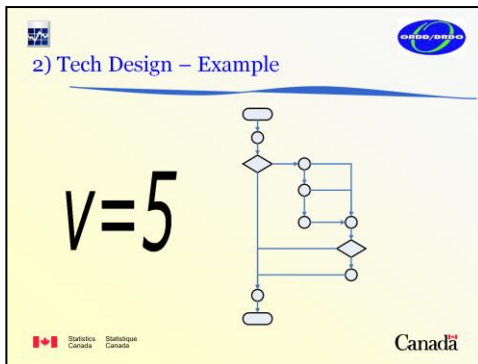
A report on workshop proceedings - April 2002

Robert Groves & William Kalsbeek

Presentation by Thomas McCabe, pgs 116-137.

Mr. McCabe's presentation addressed our actual issue: complexity. Block testing was taking a long time because blocks were extremely complicated, not because they were extremely long.

Mr. McCabe presented a method to estimate the complexity of a set of code, and guidelines for how complex a set of code 'should be'. The measurement is called "cyclomatic complexity"; it is a count of the basis paths through a block (the total number of unique paths).



To do the technical design, we draw a flowchart of the block. We don't need a full and formal specification from which to work; most surveys have planning documents that describe their content in enough detail.

In our flowcharts, circles represent questions, and diamonds represent flow decisions (we have other shapes for edits, blocks, etc.). **Shapes** are **objects**; they are connected by lines (logic flows).

We count the objects (nodes), subtract the number of lines (edges), and add two (to keep the number positive). The result is the cyclomatic complexity of the block.

In this example, there are 11 objects and 14 Lines, so the complexity estimate is  $((14-11)+2)=5$ . The symbol used in the book is a lowercase "v".



The goals of technical design are to reduce complexity where possible, and to isolate the remaining complexity.

Note: if the flowchart cannot be drawn without crossing lines, the requested logic is unstable. The flowchart can also show other types of unstable logic: branching into and out of decision flows. When this happens, we work with the specifiers to stabilize the logic.

---

The benefits of simple blocks are numerous:

- 1)The specifications are easier to understand and review;
- 2)Authoring time is reduced (initial and test support);
- 3)Testing effort is significantly reduced;
- 4)Testing can be done with more accuracy and confidence.

---

Another problem we are trying to address is the order of development. We want to start with the blocks that are going to take the most time (the hardest and longest blocks).

2) Tech Design – How?

- ☞ If  $v > 10$ , consider multiple blocks
  - ☞ No extra overhead for multiple blocks
- ☞ Breaking up is easy to do

Statistics Canada / Statistique Canada

Canada

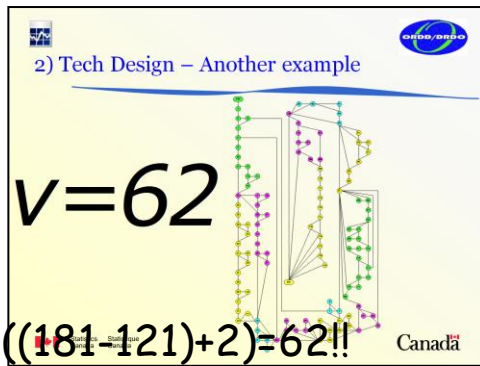
The “10” figure is a strategy, not an unbreakable rule. You can get false positives - lots of simple branches.

(“10” comes from Mr. McCabe - page 126 - - “less reliable and require higher levels of testing”)

It must be stressed that technical design is not something we do “to” clients, it is something we do “with and for” clients. The original order of questions will never change as a result of the technical design: we are adding a layer of design to the original specifications. They will reap huge benefits from a technical design: testing will take much less effort.

---

We have a document on how to do technical design. It is not a difficult process.



The colours represent how we created 13 sub-topic blocks from the original topic block.

We have a formula for estimating the authoring time for a block, based on objects and cyclomatic complexity. It produces very reasonable estimates. Our estimate for developing this topic in 13 blocks is about half (55%) of what it would have been as one block.

Formula ( $cV$  = cyclomatic complexity):

$$Hrs = ( ( 1.5 * Objects ) / 60 ) + ( 15 * ( cV^{1.4} ) / 100 )$$

Any formula for estimating block development time must address field definitions, type definitions and flow logic. The first part of this formula addresses field and type definitions, and assumes the author is using *CodeBuilder* to create the fields and types. Using *CodeBuilder*, an author can produce field definitions at the rate of 40 per hour.

The second part of this formula addresses how long it takes to write the RULES section, with the understanding that complexity has an exponential impact on how long it takes to do it.

2) Tech Design – What changed?

- Technical design has become a standard part of development practices
- Testing burden has been reduced
  - Testing is easier, because blocks are less complex
  - Authoring burden has also been reduced
- Use complexity information for estimates

Statistics Canada / Statistique Canada

Canada

We have seen a significant reduction in code changes per object (as we continue to review problem log metrics).

Our original idea was to use the authoring time estimate to prioritize blocks for development. Where timing permits, we use the results of this formula in our development estimates.



2) Tech Design – What's next?

- Expand the expertise
- Compare estimates against actual

Statistics Canada / Statistique Canada

Canada

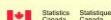

We need to get more people doing this type of work (expand the expertise base).

We don't currently track time at the SURVEY-BLOCK-TASK level, so we can only take a very good guess at how accurate the estimation formula is. It would be too much of a burden on the authors to collect data at this level. We'll try to do this in a non-burdensome way.

3) Tracking Dev - Trigger

Block ID	Initial				Round 1			
	Spec for Authoring Done	Authoring Done	QA Test Done	Block to Clients	Tickets Received	Update Done	QA Test Done	Block to Clients
EO								
GI	20050125	20050126	20050201	20050201				
EV	20050125	20050201	20050201	20050201				
LU	20050125	20050218	20050218	20050218				
INR	20050127	20050209	20050215	20050215	20050218	20050218	20050218	20050218

What if we could use this data to determine how much time was used to go through each stage?

We were already tracking (DevTracker):

- Blocks in a questionnaire
- Stages of development
- Date stages were achieved

DevTracker (excel spreadsheet) was used as a record of development. It showed what had happened when, and was used during weekly with clients. It is an internal tool.

3) Tracking Dev – Response

- Added some formulae to DevTracker
- Added “complexity” data for each block
- Now know how much work is in what stage
  - Based on amount of work (not number of blocks)

Statistics Canada / Statistique Canada

Canada

The most difficult part was figuring out how to measure the elapsed number of working days between stages.

Using estimated initial authoring hours (from technical design data!), we can track development by the amount of work rather than by the number of blocks.

When tracking development, it is misleading to treat a 4-2 (objects-complexity) block in the same manner as a 24-8 block. The latter is eight times more work than the former.

---

What we measure with DevTracker is elapsed working days, not staff effort. It does not keep track of how many people are working on a project; it does not record overtime.

It answers the question: where did all the time go? If you want to reduce the amount of time it takes to get an application into the field, you need some way of knowing where that time is spent.



This is a relatively new practice for us, so the development practice changes resulting from this data are what's next.

Having duration data with "initial authoring estimate" data allows for comparisons across surveys. A three-hour block is a three-hour block.

Results from one survey show that specifications took 70% of development time, development activities took 18%, and client testing took 12% of the time. From experience, we understand that client testing is normally three or more times longer than development, so our new practices are clearly having a positive impact. (This survey went through technical design, specification reviews and QA testing.)

Note: it had always been my impression that clients spent more calendar time testing than specifying, and this data shows that that was wrong. Client testing efforts are obvious to us, and they take place at a time when authors are doing heavy work, too. All of the time spent the client spent creating the specifications "got lost" in the past. Tracking activities from the start produces a much better picture of all development: the time didn't "get lost".

The slide is titled "Resumé" and features a blue wavy line below the title. It contains three bullet points: "Metrics can be meaningful and useful", "Better understand development process", and "Focus efforts for improvements". The bottom left corner has the Statistics Canada logo, and the bottom right corner has the word "Canada".

Resumé

- Metrics can be meaningful and useful
- Better understand development process
- Focus efforts for improvements
- Measure impact of new development practices

Statistics Canada / Statistique Canada

Canada

One could do this ...

How much better is this new practice? Authoring was reduced by 6%, and testing was reduced by 23%.

Metrics help you understand your development processes, which can then help you look for improvements in appropriate places. When you make changes in your development processes, metrics can help you measure how effective they were (or how ineffective they were!!).

