An introduction to content analysis

Kevin Crowston

Syracuse University  School of Information Studies
crowston@syr.edu  http://crowston.syr.edu/
What is content analysis?
What is content analysis?
Process of content analysis
Examples
Hope everyone is having a good week. :-) 

I have been working on paring down Defines.h for awhile now. Jason and I discussed it several months ago, and I only just got around to it. What I did was break out everything that was in only one file. These are things that shouldn’t have been put in Defines.h to begin with.

It looks like I changed 28 files. The changes should be simply cosmetic and shouldn’t affect functionality. However, since this is such a large change, I wanted to run the changes by everyone before I did something stupid that would have to be backed out.

And so, I give you, the Ginormous Diff From Hell[TM], and the binary that goes along with it:  http://www.kocharhook.com/nick/fire/diff.html

Take a look at the diff, maybe try out some of the changes in the binary. Let me know if you see anything amiss.

If I get no comments, I’m going to check this in tomorrow.

--
Nick Kocharhook  --  <avpx@xb...> -- Rot-13
# Example code book

## Code family: Emotional expression

<table>
<thead>
<tr>
<th>Code</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emoticons</td>
<td>Expressions of emotion or emphasis using emoticons</td>
<td>:)</td>
</tr>
<tr>
<td>Capitalization</td>
<td>Expressions of emotion or emphasis using conspicuous capitalization</td>
<td>“EVERYONE ON THE LIST”</td>
</tr>
<tr>
<td>Punctuation</td>
<td>Expressions of emotion or emphasis using (repetitious) punctuation,</td>
<td>“!!!”; Underline; “?!?”</td>
</tr>
</tbody>
</table>
## Example code book, continued

<table>
<thead>
<tr>
<th>Code family: Positive Politeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colloquialisms or slang</td>
</tr>
<tr>
<td>Spelling out phonological slurring, using colloquialisms or slang; beyond group specific; used to show familiarity.</td>
</tr>
<tr>
<td>“Saturdayish”, “yep”, “BTW”</td>
</tr>
<tr>
<td>Vocatives</td>
</tr>
<tr>
<td>Referring to participants by name, or specifically addressing part of a message to an individual. If there is a “you” or “your” specifically referring to a particular single person, we’ll code it.</td>
</tr>
<tr>
<td>“As sean said”, “Martin,”</td>
</tr>
<tr>
<td>Phatics</td>
</tr>
<tr>
<td>Personal greetings and closures, including communication for purely social reasons</td>
</tr>
<tr>
<td>“Hi”, “regards”, “Thanks,” (at end of a message)</td>
</tr>
<tr>
<td>Encouraging participation</td>
</tr>
<tr>
<td>Encouraging all the members of the group to participate</td>
</tr>
<tr>
<td>“Any comments welcome.”</td>
</tr>
</tbody>
</table>
### 1. A Shape at rest

- [□] Arrow
- [□] Parallel
- [□] Spread
- [□] Tail
- [□] Tent
- [□] Underside
- [□] Up
- [□] Z No photo

![Images of different moth shapes](image1.jpg)

### 2. A Forewing main color

- [□] Black
- [□] Brown
- [□] Gray
- [□] Green
- [□] Orange
- [□] White
- [□] Z Unknown

![Images of different moth colors](image2.jpg)

### 3. A Forewing distinctive color

- [□] Blue
- [□] Green
- [□] None
- [□] Orange
- [□] Red
- [□] Yellow
- [□] Z No photo

![Images of different moth color variations](image3.jpg)
What is the **Forewing Distinctive Color?**

Drag the photos onto the spaces below to answer. Click the question marks for help.
Theme and variations
Variations

Overall research design

Style of analysis

Unit of analysis

Role of theory

Nature of evidence

Explicitness of codebook

Source of codebook

Unit of coding
Inductive (vs. deductive) coding
Nature of evidence

Manifest
- Surface text (e.g., use of words of different categories)
- Evidence is in the text; coders just record it

Latent
- Pattern of content (e.g., phrases representing concepts)
- Coders must recognize evidence

Projective
- Meaning underlying the text (e.g., hermeneutic reading)
- Meaning comes from interaction of person and text
Evaluating content analysis
Reliability

周六 16 6月 2012
Reliability
Construct validity
Internal validity
Tools for content analysis
From: http://www.provalisresearch.com/wordstat/

Monday 18 June 2012
From http://www.yoshikoder.org/
# LIWC example

<table>
<thead>
<tr>
<th>LIWC Dimension</th>
<th>Your Data</th>
<th>Personal Texts</th>
<th>Formal Texts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-references (I, me, my)</td>
<td>6.51</td>
<td>11.4</td>
<td>4.2</td>
</tr>
<tr>
<td>Social words</td>
<td>4.73</td>
<td>9.5</td>
<td>8.0</td>
</tr>
<tr>
<td>Positive emotions</td>
<td>1.18</td>
<td>2.7</td>
<td>2.6</td>
</tr>
<tr>
<td>Negative emotions</td>
<td>1.18</td>
<td>2.6</td>
<td>1.6</td>
</tr>
<tr>
<td>Overall cognitive words</td>
<td>7.10</td>
<td>7.8</td>
<td>5.4</td>
</tr>
<tr>
<td>Articles (a, an, the)</td>
<td>5.92</td>
<td>5.0</td>
<td>7.2</td>
</tr>
<tr>
<td>Big words (&gt; 6 letters)</td>
<td>14.20</td>
<td>13.1</td>
<td>19.6</td>
</tr>
</tbody>
</table>

Monday 18 June 2012
Why You Should Read This Bug Writing Guide

Simply put, the more effort you put into writing a useful bug report, the better. These bug writing guidelines are the result of consulting experienced developers, testers, project managers, and others who have shared their wisdom about what developers expect and appreciate in a bug report. While a bug report should not be a complete technical documentation of the bug, it should be enough for a developer to reproduce and fix it. There are many ways you can improve the quality of bug reports, including:

1. **Reproducible.** If an engineer can't see it or conclusively prove that it exists, the engineer will probably stamp it "WORKSFORME" or "INVALID", and move on to the next bug. Every detail you can provide helps.

2. **Specific.** The quicker the engineer can isolate the issue to a specific problem, the more likely it'll be expediently fixed. (If a programmer or tester has to decipher a bug, they spend more time cursing the submitter than fixing or testing the problem.)

How to Write a Useful Bug Report

Useful bug reports are ones that get bugs fixed. A useful bug report normally has two qualities:

1. **Reproducible.** If an engineer can't see it or conclusively prove that it exists, the engineer will probably stamp it "WORKSFORME" or "INVALID", and move on to the next bug. Every detail you can provide helps.

2. **Specific.** The quicker the engineer can isolate the issue to a specific problem, the more likely it'll be expediently fixed. (If a programmer or tester has to decipher a bug, they spend more time cursing the submitter than fixing or testing the problem.)

Let's say the application you're testing is a web browser. You crash at foo.com, and want to write up a bug report:
How to Enter your Useful Bug Report into Bugzilla:

Before you enter your bug, use the Bugzilla Query Page to determine whether the defect you've discovered is a known bug, and has already been reported. (If your bug is the 37th duplicate of a known issue, you're more likely to annoy the engineer. Annoyed engineers fix fewer bugs.)

Next, be sure that you've reproduced your bug using a recent build. (Engineers tend to be most interested in problems afflicting the code base that they're actively working on, rather than those in a code base that's hundreds of bug fixes obsolete.)

If you've discovered a new bug using a current build, report it in Bugzilla:

1. From your Bugzilla main page, choose "Enter a new bug".
2. Select the product that you've found a bug in.
3. Enter your E-mail address, Password, and press the "Login" button. (If you don't yet have a password, leave the password text box empty, and press the "E-mail me a password" button instead. You'll receive an E-mail message with your password shortly.)

Now, fill out the form. Here's what it all means:

Where did you find the bug?

**Product:** In which product did you find the bug?  
You just filled this out on the last page.

**Version:** In which product version did you find the bug?  
If applicable:
### Hypothesis 1 - Purpose

<table>
<thead>
<tr>
<th>Name</th>
<th>Sources</th>
<th>References</th>
<th>Created On</th>
<th>Created By</th>
<th>Modified On</th>
<th>Modified By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypothesis 1</td>
<td>10</td>
<td>12</td>
<td>14/10/2011 8:43 PM</td>
<td>DVS</td>
<td>07/05/2012 6:01 AM</td>
<td>IK</td>
</tr>
<tr>
<td>Hypothesis 1 - Purpose</td>
<td>17</td>
<td>22</td>
<td>14/10/2011 8:47 PM</td>
<td>DVS</td>
<td>07/05/2012 6:01 AM</td>
<td>IK</td>
</tr>
<tr>
<td>Hypothesis 2</td>
<td>0</td>
<td>0</td>
<td>14/10/2011 8:47 PM</td>
<td>DVS</td>
<td>26/01/2012 1:05 PM</td>
<td>IK</td>
</tr>
<tr>
<td>Hypothesis 3</td>
<td>0</td>
<td>0</td>
<td>14/10/2011 8:52 PM</td>
<td>DVS</td>
<td>26/01/2012 1:05 PM</td>
<td>IK</td>
</tr>
<tr>
<td>Hypothesis 4</td>
<td>0</td>
<td>0</td>
<td>14/10/2011 8:54 PM</td>
<td>DVS</td>
<td>26/01/2012 1:05 PM</td>
<td>IK</td>
</tr>
<tr>
<td>Hypothesis 5</td>
<td>0</td>
<td>0</td>
<td>16/03/2012 2:00 PM</td>
<td>DVS</td>
<td>16/03/2012 7:32 PM</td>
<td>DVS</td>
</tr>
<tr>
<td>Hypothesis 6</td>
<td>0</td>
<td>0</td>
<td>01/05/2012 1:35 PM</td>
<td>DVS</td>
<td>06/05/2012 2:42 PM</td>
<td>IK</td>
</tr>
</tbody>
</table>

#### References 1-2

- 10.57% Coverage

---

**Why You Should Read This**

Simply put, the more effectively you report a bug, the more likely an engineer will actually fix it. These bug writing guidelines are an attempt at a general tutorial on writing effective bug reports for novice bug writers; not every sentence may precisely apply to your software project.

**How to Write a Useful Bug Report**

Useful bug reports are ones that get bugs fixed. A useful bug report normally has two qualities:

1. Reproducible. If an engineer can't see it or conclusively prove that it exists, the engineer will probably stamp it "WORKSFORME" or "INVALID", and move on to the next bug. Every detail you can provide helps.
2. Specific. The quicker the engineer can isolate the issue to a specific problem, the more likely it'll be expediently fixed. (If a programmer or tester has to decipher a bug, they spend more time cursing the submitter than fixing or testing the problem.)

---

#### References

- 9.49% Coverage

Reference 1 - 17.79% Coverage

From: Mark Doliner <mark@bl...ngent.net>
Gaim 2.0.0
2005-11-20 19:29
Me again!

Gaim is looking good. There are a few things we should talk about before we can release a beta.

1. Voice/video
   It seems like the general consensus is that we won't delay Gaim 2.0 for anything related to voice/video, and we probably won't release Gaim 2.0 with support for Gtalk or with support for voice/video for any protocol other than Google Talk. And if our Gtalk support in 3.1 is causing problems then we'll #2 or #3 it or something. The IBC license incompatibility is currently a major concern to me. Does anyone disagree with any of this?

2. Ability to specify the idle-away status
   We still need a UI for selecting the idle-away state (all the backend code should be working fine). I'm fine with having a dropdown in prefs to select one of the saved statuses. Does anyone have a better idea?

3. Per-account status bars
   I personally don't like them. I think they hurt more than they help. They clutter the bottom of the buddy list, they're distracting, they take up a lot of room, and they're a little confusing. I don't think I've ever set a different status for an individual account. I don't think the functionality will be used very often, and I think it's unnecessary to have it at the bottom of the buddy list.

   Please vote:
   a. keep them! (and make sure everything works perfectly)
   b. remove them! (and figure out a better way to display account disconnected messages)
   c. keep them! (and add a 'Save' button)

   I'll be out of town Wednesday through Saturdayish. You guys have done an awesome job of not falling too far behind in my totally unreasonable schedule. If we deal with everything above, do you think we can release a beta on Monday the 28th?

   We should have a 4 week string freeze before we release. If we freeze on December 2nd we can release by the end of the year.

   -Mark
Preconditions for applying content analysis
Preconditions for content analysis
Conclusion

Content analysis is a data analysis technique for finding evidence of concepts of interest from various texts.

Lots of variations on content analysis:
- Deductive vs. inductive vs. mixed coding
- Manifest vs. pattern vs. latent codes
- Different units of coding
- Different overall research strategies

Content analysts face issues of reliability and validity.
Sources for further study


http://writing.colostate.edu/guides/research/content/index.cfm