software assessment
The B-29 was the main bomber of US Air forces and it provided the strategic advantage of reaching over the Pacific Ocean.

This three billion dollar project was the largest government commitment ever to a single project, including the Atomic Bomb.

http://en.wikipedia.org/wiki/B-29
http://en.wikipedia.org/wiki/Tupolev_Tu-4
http://www.rb-29.net/HTML/03RelatedStories/03.03shortstories/03.03.10contss.htm
During 1944, 3 bombers had to land in Russia after bombing missions in Japan. The Russians refused to return them. The B-29 was not a legacy system, but:
- it was tremendously valuable
- it was unknown to the Russians
- it was estimated that to build one from scratch would take about 5 years

http://en.wikipedia.org/wiki/B-29
http://en.wikipedia.org/wiki/Tupolev_Tu-4
The challenge was to understand the planes well enough to be able to build a factory that would build them. This had to go beyond just the structure.
They approached the problem from several directions:
- one plane was disassembled into pieces,
- one plane was used for flying, and
- one plane was used as a comparison model and for training pilots.
They approached the problem from several directions:
- one plane was disassembled into pieces,
- one plane was used for flying, and
- one plane was used as a comparison model and for training pilots.
They approached the problem from several directions:
- one plane was disassembled into pieces,
- one plane was used for flying, and
- one plane was used as a comparison model and for training pilots.
Tuesday, October 25, 11

They approached the problem from several directions:
- one plane was disassembled into pieces,
- one plane was used for flying, and
- one plane was used as a comparison model and for training pilots.
Tuesday, October 25, 11

They approached the problem from several directions:
- one plane was disassembled into pieces,
- one plane was used for flying, and
- one plane was used as a comparison model and for training pilots.
They approached the problem from several directions:
- one plane was disassembled into pieces,
- one plane was used for flying, and
- one plane was used as a comparison model and for training pilots.
They eventually managed to build their own plans.
The Russians reverse engineered the plans in 1 year and produced the first piece 1 year later: 105,000 pieces assembled in 2 years. Tu-4 first flew on May 19, 1947. Serial production started immediately, and the type entered large scale service in 1949.

It is said that they copied even the flaws, as the engines were as unreliable as in the American version.
software assessment

tudorgirba.com
When we think of software development, we think of building something. That is we do not have the system, then we develop it, and then we have it.
However, multiple studies show that up to 50% of the time developers do not develop, they read code as a means to understand it.

assessment
Systems are large. Supposing that you read one line in 2 seconds, it would take about one man–month of work to read a quarter million lines of code:

\[
250'000 \text{ lines of code} \\
\times 2 = 500'000 \text{ seconds} \\
/ 3600 = 140 \text{ hours} \\
/ 8 = 17.5 \text{ days}
\]
data \rightarrow \text{analyses} \rightarrow \text{knowledge}
reverse engineering is analyzing a subject system to:
- identify components and their relationships, and
- create more abstract representations

http://dx.doi.org/10.1109/52.43044
reverse engineering is analyzing a subject system to:
- identify components and their relationships, and
- create more abstract representations

data ➔ analyses ➔ models ➔ knowledge
data → analyses → models → knowledge → assessment
Tuesday, October 25, 11

This book is used as inspiration for the course. The book is now open source and can be found at:
http://www.iam.unibe.ch/~scg/OORP/
Weinberg was among the first to point out that programming is a human activity. In one of his stories, he points out how chatting around a vending machine helped solving problems. You can read about the vending machine story here: http://www.crosstalkonline.org/storage/issue-archives/2008/200808/200808-0-Issue.pdf
Read all code in one hour

Problem: How to get a first impression of the code?
Solution: Scan all code in one short session.

Issues:
- limit your time, and isolate from interruptions.
- use a checklist.
- look for root and abstract classes.
- beware of misleading comments.
- log your questions and findings.
/**
 * We hang our heads in shame. There are still bugs in ArgoUML and/or GEF that cause corruptions in the model.
 * Before a save takes place we repair the model in order to be as certain as possible that the saved file will reload.
 * TODO: Split into small inner classes for each fix.
 *
 * @return A text that explains what is repaired.
 */
Snippet from an old Moose

```perl
updateTypeAccordingToEntities

"-- ugly code, will change once we move to CollectiveBehavior --"

<table>
<thead>
<tr>
<th>common wantedType class</th>
</tr>
</thead>
</table>
common := self commonMetaDescription.

wantedType := (common name, 'Group') asSymbol.

self metaDescription name == wantedType ifTrue: [ ^self ].

class := AbstractGroup allSubclasses

detect: [ :each | each asMetaDescription name == wantedType ]

ifNone: [ ^self changeTypeToDefaultType ].

self changeTypeTo: class.
```
I took a course in speed reading and read “War and Peace” in twenty minutes. It's about Russia. - Woody Allen

Why read all code in 1 hour? Because we have a built-in mechanism to think fast.

Get a first impression, but do not rely on it.
Use it for guiding your future investigations.
What is an analysis in general?

Webster’s definition of analysis:
- Detailed examination of the elements or structure of something, typically as a basis for discussion or interpretation.
- The process of separating something into its constituent elements. Often contrasted with synthesis.
Here is a small example: How many methods are there in this class?
public class Library {

    List books;

    public Library() { … }

    public void addBook(Book b) { … }

    public void removeBook(Book b) { … }

    private boolean hasBook(Book b) { … }

    protected List getBooks() { … }

    protected void setBooks(List books) { … }

    public boolean equals(…) { … }

}
public class Library {
    List books;
    public Library() {…}
    public void addBook(Book b) {…}
    public void removeBook(Book b) {…}
    private boolean hasBook(Book b) {…}
    protected List getBooks() {…}
    protected void setBooks(List books) {…}
    public boolean equals(…) {…}
}

NOM = 7  6

But, is a constructor a method? If the metric computation does not consider it as a method, we get 6 instead of 7.
public class Library {
    List books;
    public Library() {…}
    public void addBook(Book b) {…}
    public void removeBook(Book b) {…}
    private boolean hasBook(Book b) {…}
    protected List getBooks() {…}
    protected void setBooks(List books) {…}
    public boolean equals(…) {…}
}

What about setters and getters? Are they to be considered as methods? If no, we have only 4.
public class Library {
    List books;
    public Library() {...}
    public void addBook(Book b) {...}
    public void removeBook(Book b) {...}
    private boolean hasBook(Book b) {...}
    protected List getBooks() {...}
    protected void setBooks(List books) {...}
    public boolean equals(...) {...}
}
public class Library {
    List books;
    public Library() {...}
    public void addBook(Book b) {...}
    public void removeBook(Book b) {...}
    private boolean hasBook(Book b) {...}
    protected List getBooks() {...}
    protected void setBooks(List books) {...}
    public boolean equals(…) {...}
}

NOM = 7  6  4  3  2

equals() is a method expected by Java, so we might as well not consider it a real method.
public class Library {
    List books;
    public Library() {…}
    public void addBook(Book b) {…}
    public void removeBook(Book b) {…}
    private boolean hasBook(Book b) {…}
    protected List getBooks() {…}
    protected void setBooks(List books) {…}
    public boolean equals(…) {…}
}

NOM = 7, 6, 4, 3, 2 ?

So how many methods are there? All these are valid answers depending on what we understand by the question.

Now, if we turn the situation around, and you get a report that says a class has 70 methods. What does it mean? You have to know what the actual computation does.

And this is a simple metric.
public class Library {
    List books;
    public Library() {...}
    public void addBook(Book b) {...}
    public void removeBook(Book b) {...}
    private boolean hasBook(Book b) {...}
    protected List getBooks() {...}
    protected void setBooks(List books) {...}
    public boolean equals(...) {...}
}

NOM = 7, 6, 4, 3, 2 ?

So how many methods are there? All these are valid answers depending on what we understand by the question.

Now, if we turn the situation around, and you get a report that says a class has 70 methods. What does it mean? You have to know what the actual computation does.

And this is a simple metric.
importers
models
analyses

data
classes select: #isAnnotated

McCabe = 21

LOC = 753,000
classes select: #isGod

McCabe = 21
LOC = 753,000
classes select: #isGod

McCabe = 21

LOC = 753,000

Tuesday, October 25, 11
classes select: #isGod

McCabe = 21

LOC = 753,000
classes select: #isGod

McCabe = 21

LOC = 753,000

Tuesday, October 25, 11
classes select: #isGod

McCabe = 21

LOC = 753,000
data → importers → models → analyses
importers
models
analyses
What is this made of?
view interaction menu: #mooseMenu.
view shape rectangle
  height: #numberOfMethods;
  width: #numberOfAttributes;
  linearFillColor: #numberOfLinesOfCode within: classGroup.
view nodes: classGroup.
view edgesFrom: #superclass.
view treeLayout
What is this made of?
composer tabulator with: [:t |
  t row: [:r | r column: #namespaces;
    column: #classes; column: #methods];
  row: #details.
  t transmit to: #namespaces; andShow: [:a |
    a tree
    title: 'Namespaces';
    display: [:m | m allNamespaces select: #isRoot ];
    children: #childScopes;
    format: #name ].
  t transmit from: #namespaces; to: #classes; andShow: [:a |
    a list
    title: 'Classes';
    display: [:n | n classes ];
    format: #name].
  t transmit from: #classes; to: #methods; andShow: [:a |
    a list
    title: 'Methods';
    display: [:c | c methods ];
    format: #name].
  t transmit from: #methods; to: #details; andShow: [:a |
    a text
    display: #formattedSourceText ]
].
composer openOn: model
Moose is a platform for software and data analysis.

It is an open source project since 1996. It is supported by several research groups around the world, and it is increasingly adopted in industrial projects.

Download 4.6

The Moose Book

moosetechnology.org

Tuesday, October 25, 11
Moose version 4

the book that shows the outside the inside and the philosophy of the Moose platform

by Tudor Girba

About
This book offers an overview of the Moose & platform for software and data analysis. More specifically it covers version 4.

Currently, the book is in a preliminary shape, with a number of parts still under writing.

Feedback is more than appreciated. Please contact me or leave a comment on this site.

themoosebook.org
Data \rightarrow \text{importers} \rightarrow \text{models} \rightarrow \text{analyses}

\text{data} \rightarrow \text{importers} \rightarrow \text{models} \rightarrow \text{analyses}

\text{moosetechnology.org}
development  assessment
development

assessment

educated
tailored
explicit
development

assessment

- explicit
- tailored
- educated

humane-assesment.com
software assessment

tudorgirba.com
humane assessment

tudorgirba.com

Tuesday, October 25, 11
More details at: http://humane-assessment.com
More details at: http://humane-assessment.com