


The ultimate goal of all computer science is the program.

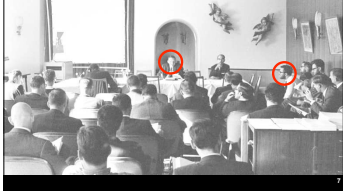
Designers, programmers and engineers must once again come to know and comprehend the composite character of a program, both as an entity and in terms of its various parts.



The Lego Hypothesis

James Noble
Victoria University of Wellington
New Zealand

1968



1968

The **general admission** of the existence of the **software failure** in this group of responsible people is the *most refreshing experience* I have had in a number of years

Dijkstra

The Dream

In the beginning,

so our myths and stories tell us,

the programmer created the program

from the eternal nothingness of the void.

The Dream

In the future

Programs will be built out of reusable parts

Software parts will be available worldwide

Software engineering will be set free from the mundane necessity of programming

1968

The market would consist of specialists in system building, who would be able to use tried **parts** for all the more commonplace parts of their systems. ... The ultimate consumer of systems based on components ought to see considerably improved **reliability and performance**, ... and also to **avoid the now prevalent failings** of the more mundane parts of systems, which have been specified by **experts**, and have then been written by **hacks**.

--- M. Douglas McIlroy, [4 Mass Produced Software Components]

Robinson, Hovenden, Hall, Rachel

Fordism ... has four basic principles:


- standardised products
- repeated tasks having potential for automation
- unautomated tasks analysed using work study methods, known as Taylorism.
- production lines with the work moving to the workers.

--- Hugh Robinson, Fiona Hovenden, Pat Hall and Janet Rachel, [4 Postmodern Software Development]

The Lego Hypothesis

Programs can be built out of many small independent components

In the same way that a model house can be built up out of many small independent Lego blocks

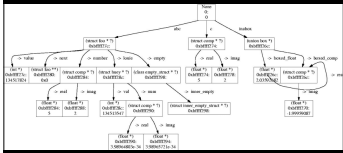


The Lego Hypothesis

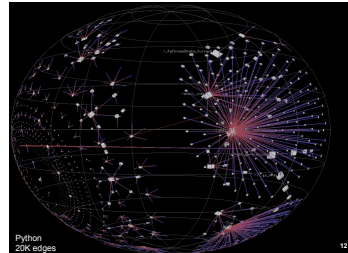
- Components are *atoms*
 - Components are small
 - Components are indivisible
 - Components are substitutable
 - Components are more similar than different
 - Components are coupled to only a few, neighboring components
 - Components are *abstract encapsulations*
- All components are equal
- A system can be explained reductively from its components and their composition

An Experiment

- Look at the structures of object graphs



Pictures by Andreas Zeller & Thomas Zimmermann
<http://www.st.cs.uni-sb.de/memograph/>



Python
20K edges

The Problem of Prime Numbers

"The task is to instruct a computer to print a table of the first thousand prime numbers, 2 being considered the first prime number."

E.W.Dijkstra, *Notes on Structured Programming*

The Problem of Prime Numbers

How do we do this when we are **not** writing the first program in the world?



Postmodernity

... that **condition** in which, for the first time, and **as a result of technologies** which allow the large-scale storage, access, and re-production of records of the past, the **past** appears to be included in the **present**.

Steven Connor, *Cambridge Companion to Postmodernism*

Lyotard

vspace*{5mm} Recourse to **grand narratives** is **forbidden**; But, as we have seen, the "**small narrative**" is a form which superbly allows imaginative invention, and most of all in science.

Grand Narratives

- Algorithmic computation — correctness
- Algorithmic efficiency

ACM 2001 Computing Curriculum

- Software that fails to meet these goals is evidence of the moral weakness of its programmers

Martin Rinard, OOPSLA Onward 2005

Little Narratives

- How then do you make decisions?
 - Negotiation between various narratives
 - Open to the environment
 - Contingent, context dependent
 - No single correct answer
- Negotiation
- Local correctness

Eco

The job ... is a **trial and error process**, very similar to what happens in an **Oriental bazaar** when you are buying a carpet. The merchant asks 100, you offer 10 and after an hour of **bargaining** you agree on 50.

http://www.themodernword.com/eccoeco_guardian54.html



Extreme Programming

- Customer and developers negotiate over **user stories**
- Iterate and Refactor
- Build a system, not components

Design Patterns

- Solution to a problem in a context
- Small stories, about bits of designs
- Patterns don't build complete, whole programs

Aspect-Oriented Design

```

class Lot {
    String designation;
    Section locality;
    MailDeliveryPostcode postcode;
    int incoming_mail_volume;
    RefuseCollection route;
    double refuse_load;
    double recycle;
}
    
```

Scrap-Heap Programming

- Programmers work bottom up...
- From a Scrap-Heap
- Starting with whatever they can
 - Find
 - Scavenge
 - Steal
 - Google
- Then work out what they can build
- Then negotiate about requirements

Glue Programming

- ✓ What's at the end of the power-law?
- ✓ Lots of very small components
- ✓ Large components cannot be changed
- ✓ "Glue code" ties them together

Aspect-Oriented Design

```

aspect MailDelivery {
    MailDeliveryPostcode Lot.postcode;
    int Lot.incoming_mail_volume;
}

aspect RefuseCollection {
    RefuseCollection Lot.route;
    double Lot.refuse_load;
    double Lot.recycle;
}

class Lot {
    String designation;
}
    
```

Good Enough Software

- "Why do you call me good?"
 - Good enough for what?
 - For whom?
- Software neither correct nor efficient
 - But "good enough" for context of use
- XP - explicitly balances variables
- Acceptability-Oriented
 - End-to-end arguments

MASHUP

<http://www.yes2wind.co.nz>

REMIX • REUSE • RESAMPLE

radio.achoditto.com

Perl

When I started designing Perl I lovingly reused features from many languages ... from C, sh, csh, grep, sed, awk, Fortran, COBOL, PL/I, BASIC-PLUS, SNOBOL, Lisp, Ada, C++, and Python. To the extent that Perl rules rather than sucks, it's because the various features of these languages ruled rather than sucked.

Larry Wall

KLF

item[Number One:]
 Every Number One song ever written is only made up from bits from other songs. There is no lost chord. No changes untried. No extra notes to the scale or hidden beats to the bar. There is no point in searching for originality.

\\ -- Jimmy Gaulty and Bill Drummond, text@The Manual

KLF

So why don't all songs sound the same?

Well, it's because although the chords, notes, harmonies, beats and words have all been used before their own soul shines through; their personality demands attention.

\\ -- Jimmy Gaulty and Bill Drummond, (it The Manual)

The Lego Hypothesis Revisited

- Programs are built out of components
 - But **not** Lego components
 - Concrete stuff, not abstractions
- Old, new, borrowed, blue...
- Components are not all equal
- Interactions interdependencies are highly complex



The Lego Hypothesis

In the beginning,
so our myths and stories tell us,
the programmer created the program
from the eternal nothingness of the void.

80

The Lego Hypothesis

Today, we have a wide world of software
Programs are built out of other programs
Software Engineering is programming
(in the widest sense)
and much more besides

81



Credits

Co-conspirator
Robert Biddle
Java Object Graphs Wrangler
Alex Potanin & Hayden Melton
Software Structure Wranglers
Jerome Doleman & Nick Chapman
Maths Wranglers
Matt Visser & Marcus Frean
Gareth Baxter

83