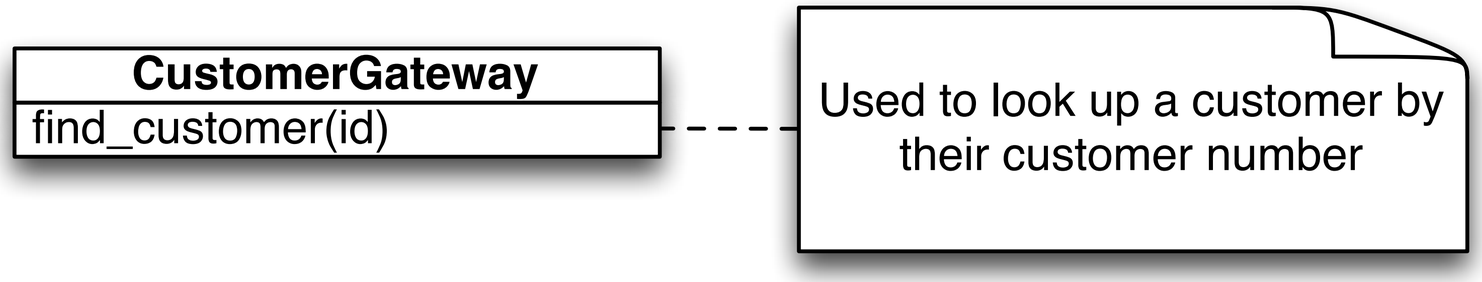
Clear Documentation

Designing and Maintaining Software (DAMS)

Louis Rose

Bad documentation

Misleading or contradictory



Bad documentation

Redundant

```
// Utility method that returns if
// this.closed is true. Throws an
// exception if the timeout is
// reached and this.closed is
// still not true
public void waitForClose(long timeoutMillis)
throws Exception {
  if (!closed) {
    wait(timeoutMillis);
    if (!closed) throw new Exception;
  }
}
```

Bad documentation

Mandated **and** Journaling

All official University documents must have a 'Document History and Version Control' table on the final page separate from the rest of the document, as shown below.

Document History and Version Control Table Example

Version Number	**Date Approved**	**Approved By**	**Brief Description**
Start from 1.00 (minor amendments are .01, .02 etc. reviews are 1.00, 2.00, 3.00 etc.)	Written as DD/MMM/YYYY (the date the last amendment or review was made)	Which approval authority in the University approved the amendment or review (e.g. Vice-Chancellor, Council or Academic Board)	Include any position title changes, amendments/additions to document and details. Give reason for amendment: For example, "*Deputy Vice-Chancellor Research changed to Deputy Vice-Chancellor Research and International to reflect change in title".*

Bad documentation

Missed opportunities

```
def progressed?
  # automatic progression: all grades are 40 or greater,
  # compensated progression: grades are 40 + no more
  # than 2 grades require compensation
  grades.all? { |g| g >= 40 } or
  (compensated(grades).average >= 40 and
  grades.select { |g| g < 40 }.size <= 2)
end
```

Better documentation

Prefer expressive artefacts over annotations

```
def progressed?
  progressed_automatically? or
  progressed_via_compensation?
end

def progressed_automatically?
  grades.all? { |g| g >= 40 }
end

def progressed_via_compensation?
  compensated(grades).average >= 40 and
  grades.select { |g| g < 40 }.size <= 2
end
```

Bad comments

All of the above and…

Bad comments

Turning off code

```
def update(name, address, email)
  user.name = name
  user.address = Address.new(address)
  user.email = email

  user.save # if user.valid
end
```

Bad comments

End of scope markers

```
    end # inner if
  end # while
end # block
end #outer if
end # method
end # class
end # module
```

Literate Programming

“I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature.”

- Donald E. Knuth
Literate Programming
Comput. J. 27(2): 97-111 (1984)

Literate Programming

Syntax Highlight Code

We syntax-highlight blocks of code with the nifty **highlight** package that includes heuristics for auto-language detection, so you don't have to specify what you're coding in.

```
{Highlight} = require 'highlight'

marked.setOptions
  highlight: (code, lang) ->
    Highlight code
```

https://github.com/jashkenas/journo/

Summary

Good documentation complements
and elaborates on the design artefact

Bad documentation can be stale,
misleading, contradictory, redundant,
or trying to compensate for a weakness
in the artefact

Literate programming could be
the future of clear code?