Clear Names

Designing and Maintaining Software (DAMS)

Louis Rose

Naming is hard

“There are only two hard things in Computer Science: cache invalidation and naming things.”

- Phil Karlton

http://martinfowler.com/bliki/TwoHardThings.html

Integrity

Reveals intent	d	elapsed_time_in_days
Honest	accounts_list // no dups	accounts_set accounts unique_accounts
Consistent	FileProcessor JobManager UserService	FileMessageRouter JobMessageRouter UserMessageRouter
Meaningfully Distinct	copy(a1, a2)	copy(source, destination)

Utility

Searchable	7 e exams.each { lel … }	MAX_MODULES_PER_STUDENT exam_number exams.each { lel … }
Pronouncable	genymdhms	generationTimestamp
Avoid encodings	name_string int_exam_number Exam implements IExam	name exam_number ExamImpl implements Exam

Tactics

Use noun phrases for classes and verb phrases for methods: **ExamRoom#start_timer**

Avoid puns: **kill** is clearer than **whack** or **eat_my_shorts**

Use solution domain names: **AccountVisitor, JobQueue**

Use problem domain names **ExamRoom, Examiner**

Ubiquitous Language

“Design calls for a versatile, shared team language, and a lively experimentation with language that seldom happens on software projects.”

- Eric Evans
Domain Driven Design

Emergence

When refactoring, extract methods with complete nonsensical names: **foo, bar, baz**

Begin to understand what the new methods do, so rename them: **calculate, process, execute**

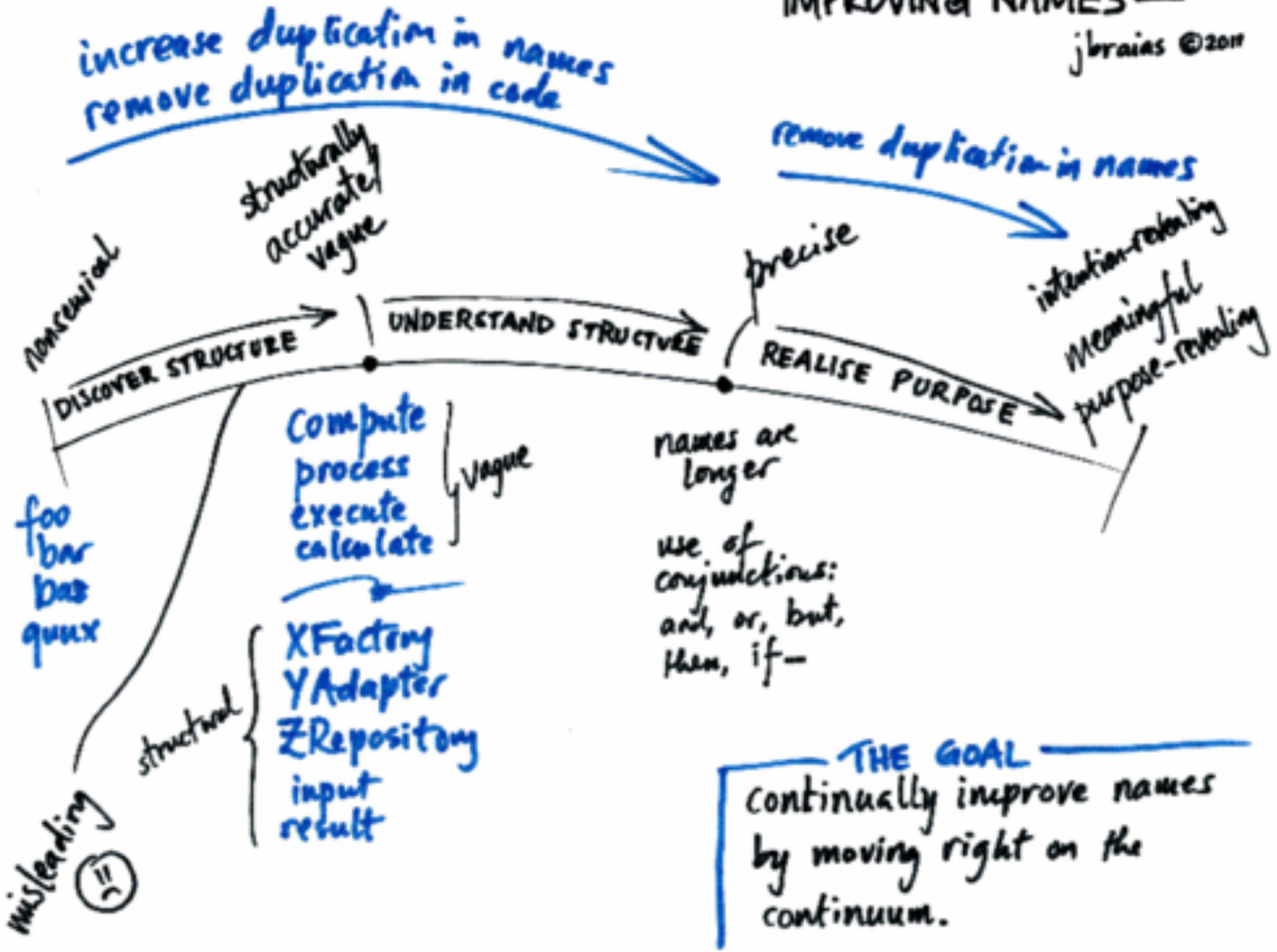
Discover that the new names are not precise, so rename them: **process_cheque_and_notify_receipient**

Discover multiple responsibilities, so separate and attach new (possibly nonsensical names)

http://blog.thecodewhisperer.com/2011/06/15/a-model-for-improving-names/
http://www.jbrains.ca/permalink/the-four-elements-of-simple-design

— IMPROVING NAMES —

jbrains ©2011



Summary

Aim for names that have integrity and utility

Use names from the solution and problem domains

Names and their abstractions can (should?)
co-evolve during design and maintenance