

# Software development

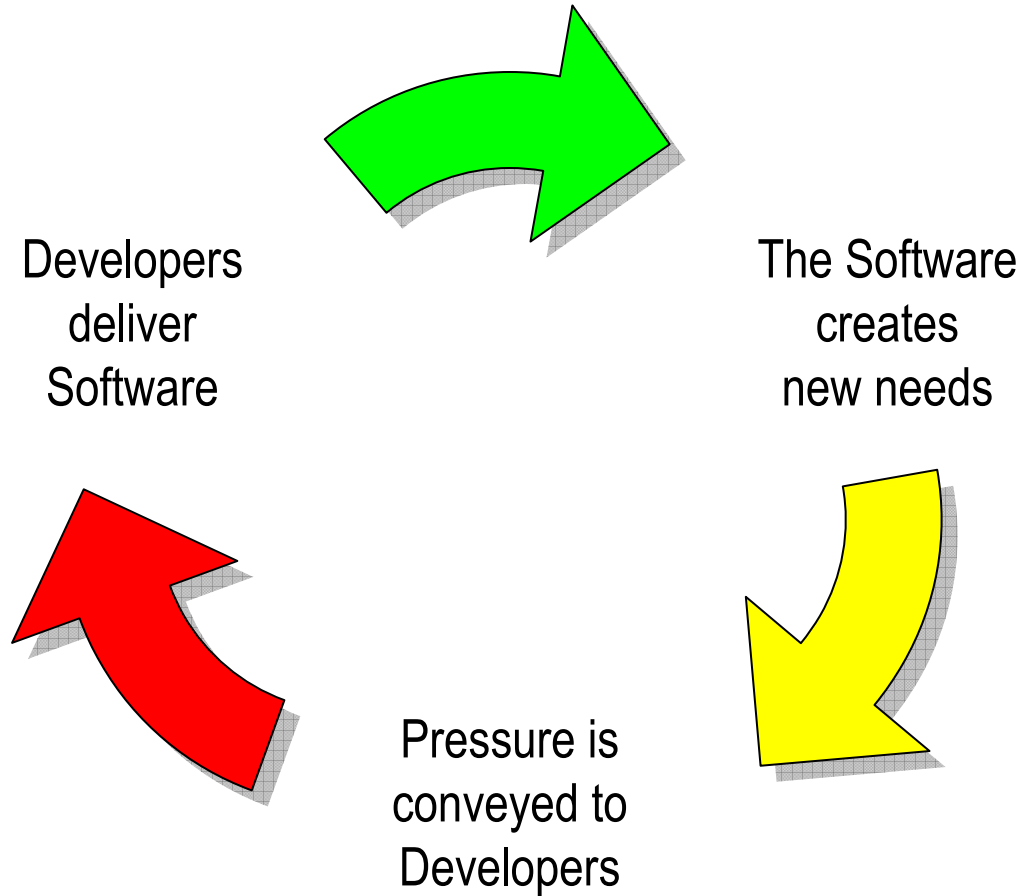
(Based on 14 years of experience leading Smalltalk projects)

Leandro Caniglia

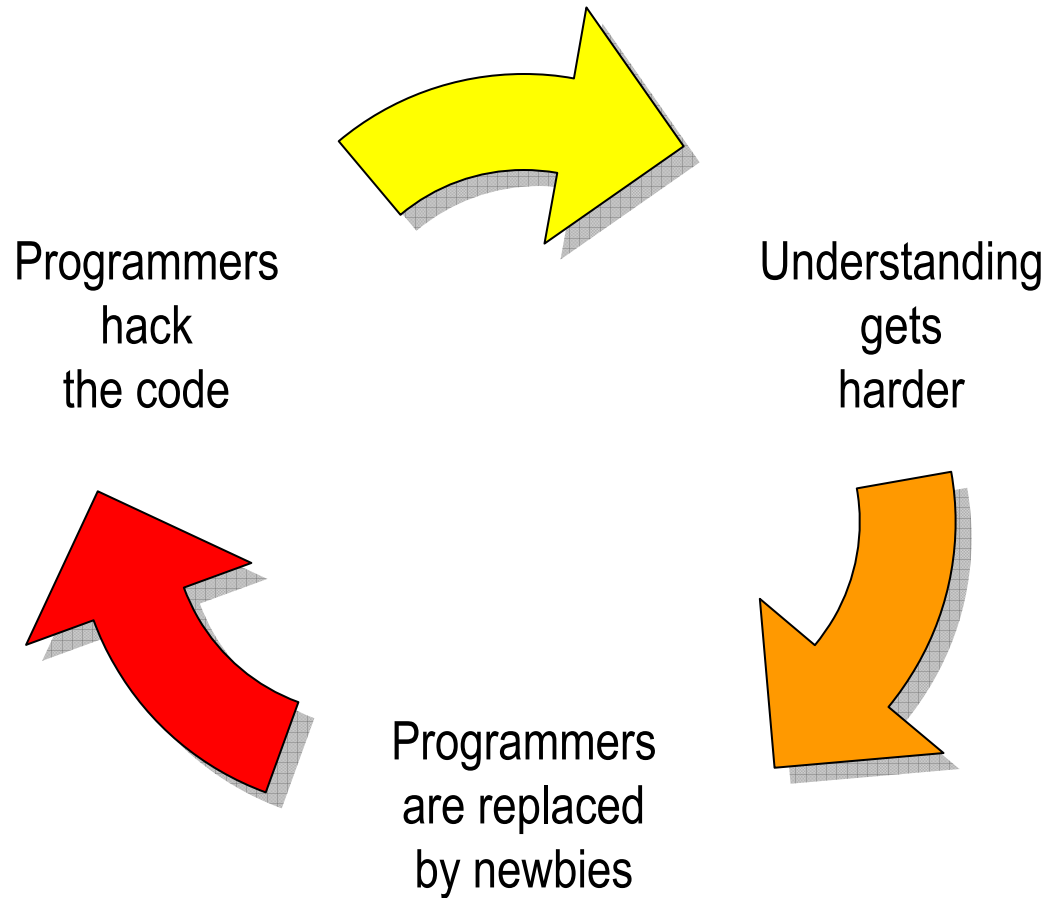
**Caesar Systems**

(Houston & Buenos Aires)

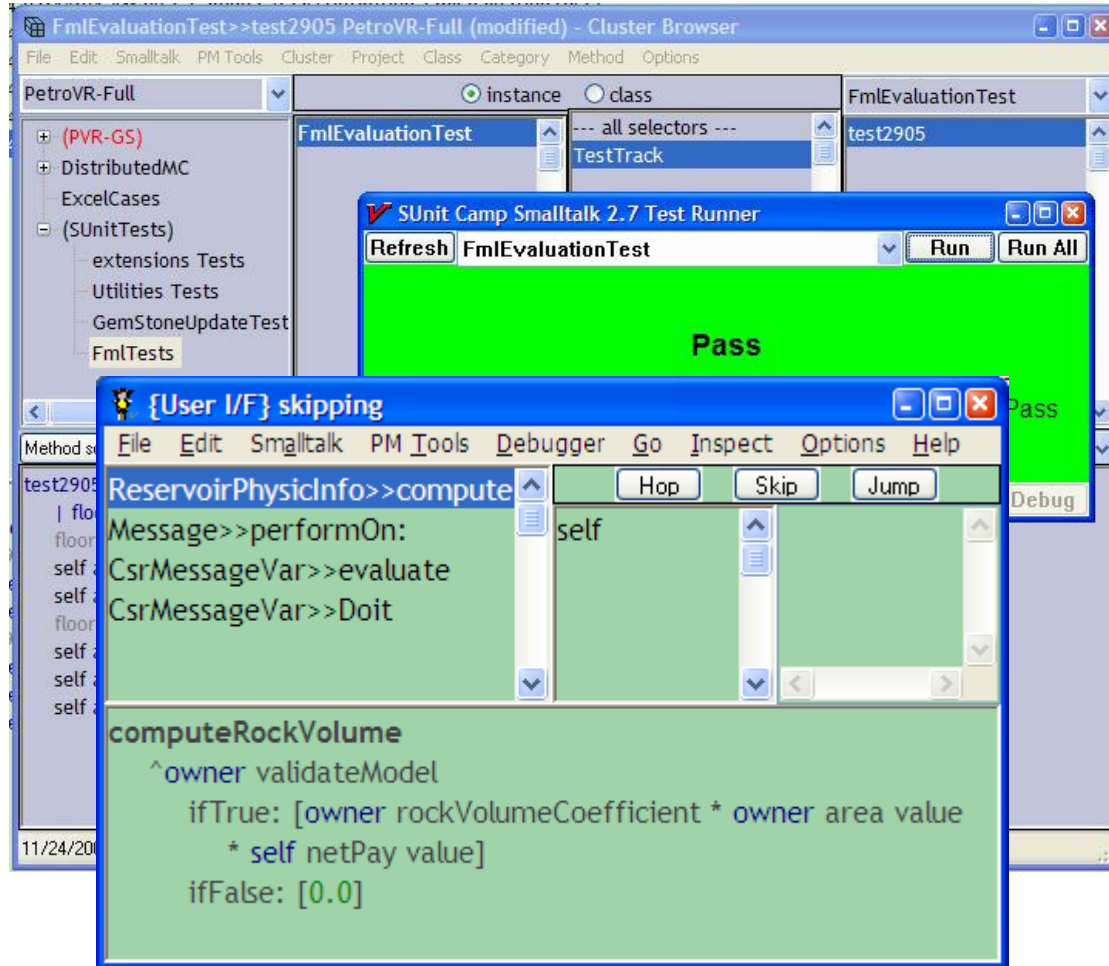
# The vicious cycles (I)



# The vicious cycles (II)



# Coding Culture

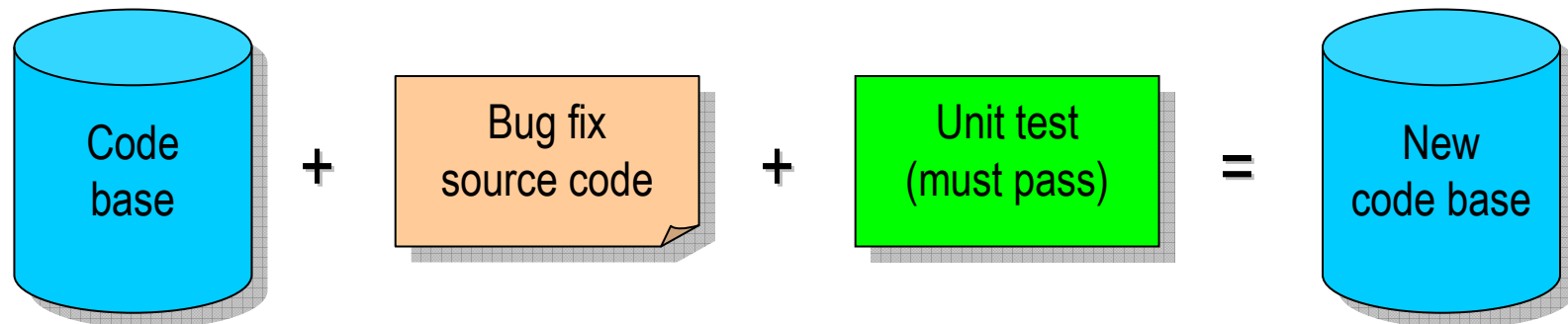
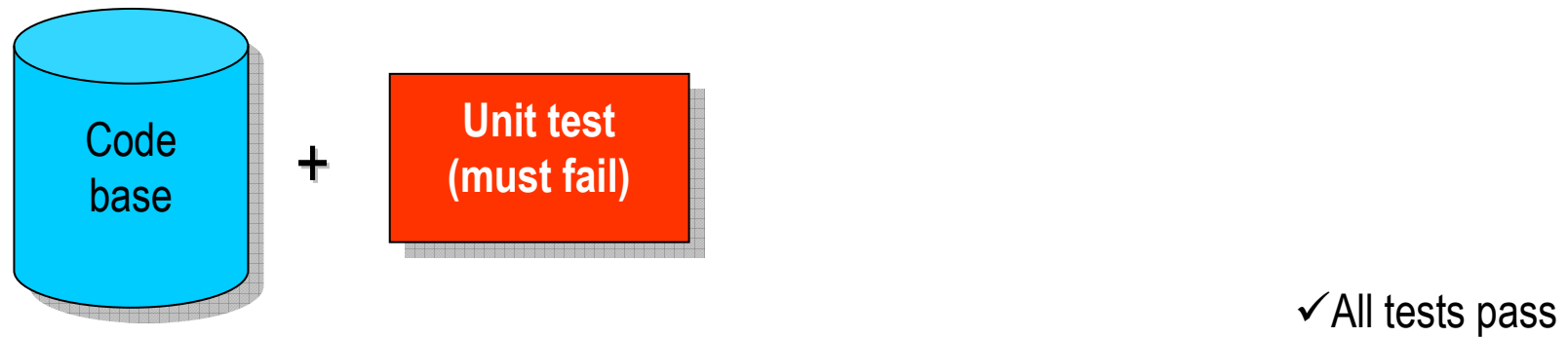
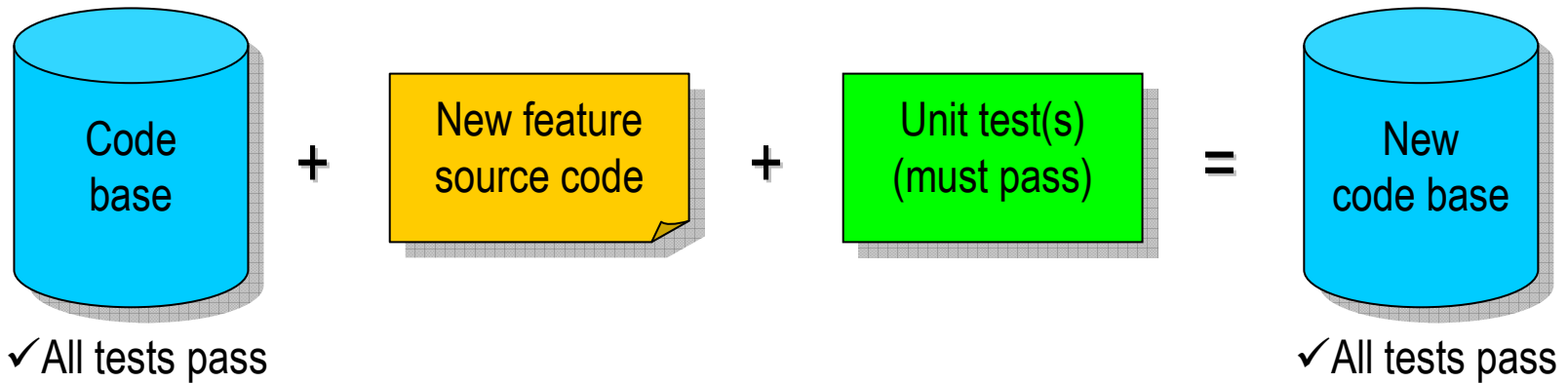


✓ All dev team members should write code every day

✓ All dev team members should debug every day

✓ All dev team members should write SUnit tests every day

# The Testing Culture: Part 1



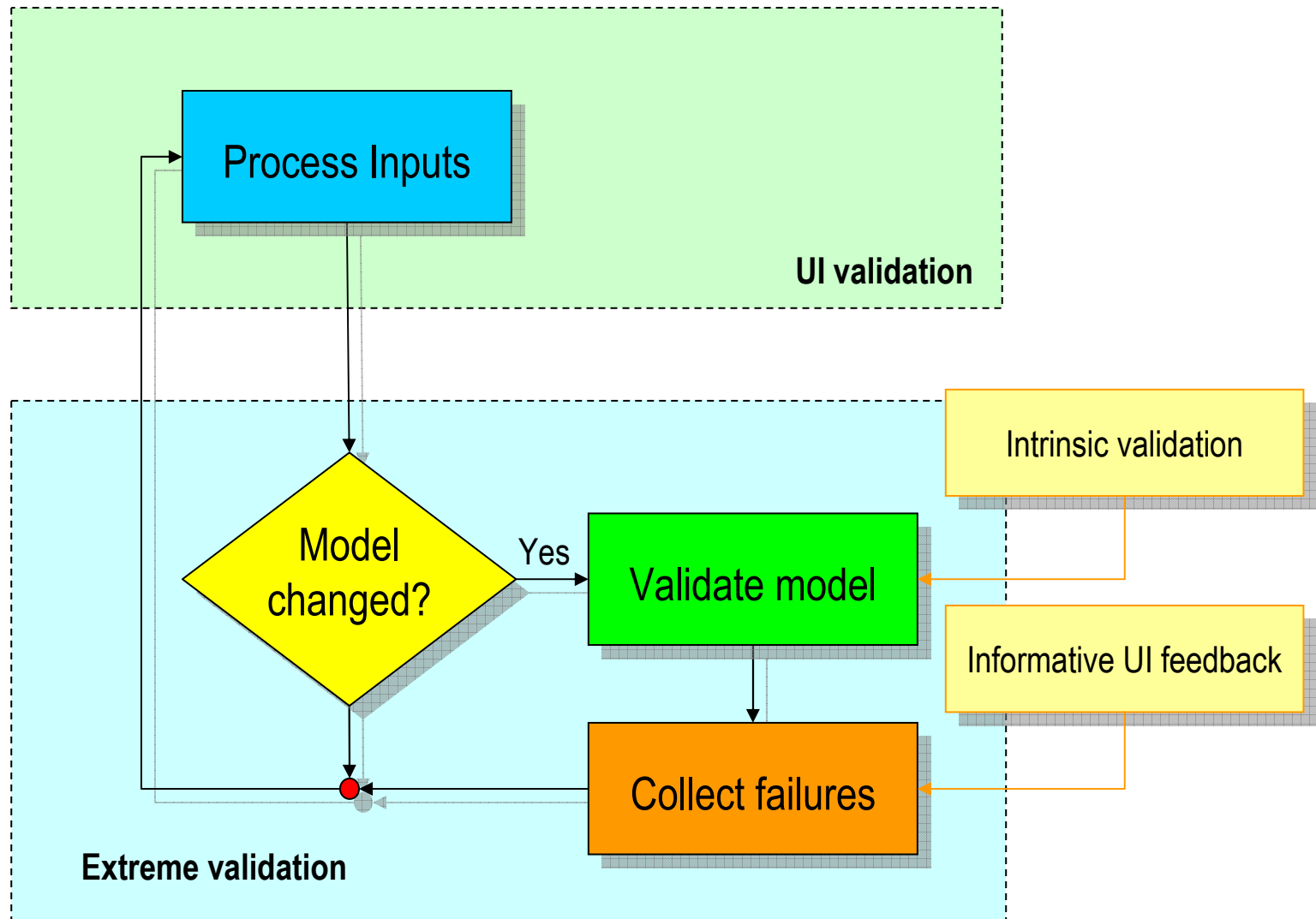
# Three properties of tests

✓ *Unit tests must be simple, not trivial*  
(do not test setters & getters, etc.)

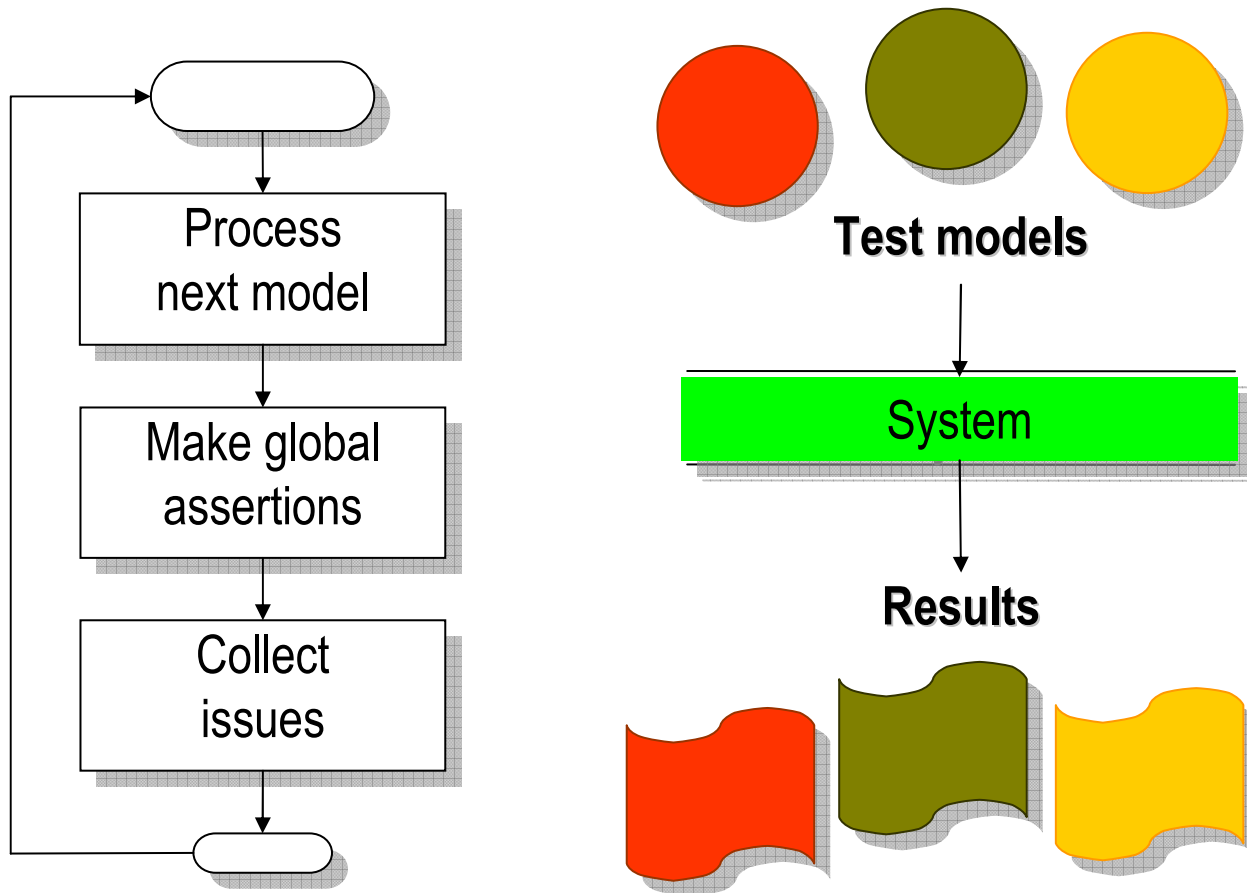
✓ *Unit tests should not depend on implementation details*

✓ *Unit tests should be fast*  
(number of tests \* duration = total time)

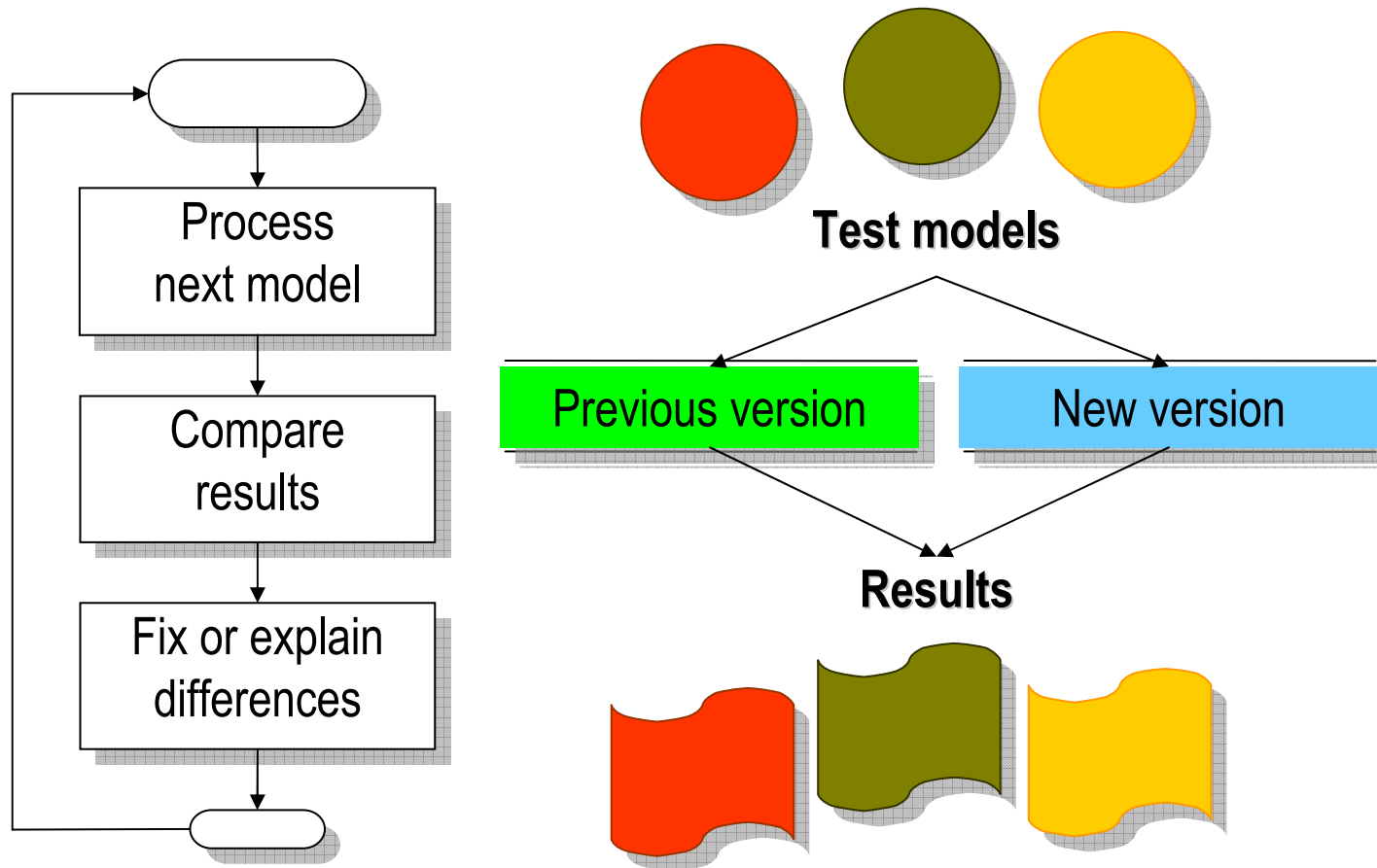
# The Testing Culture: Part 2



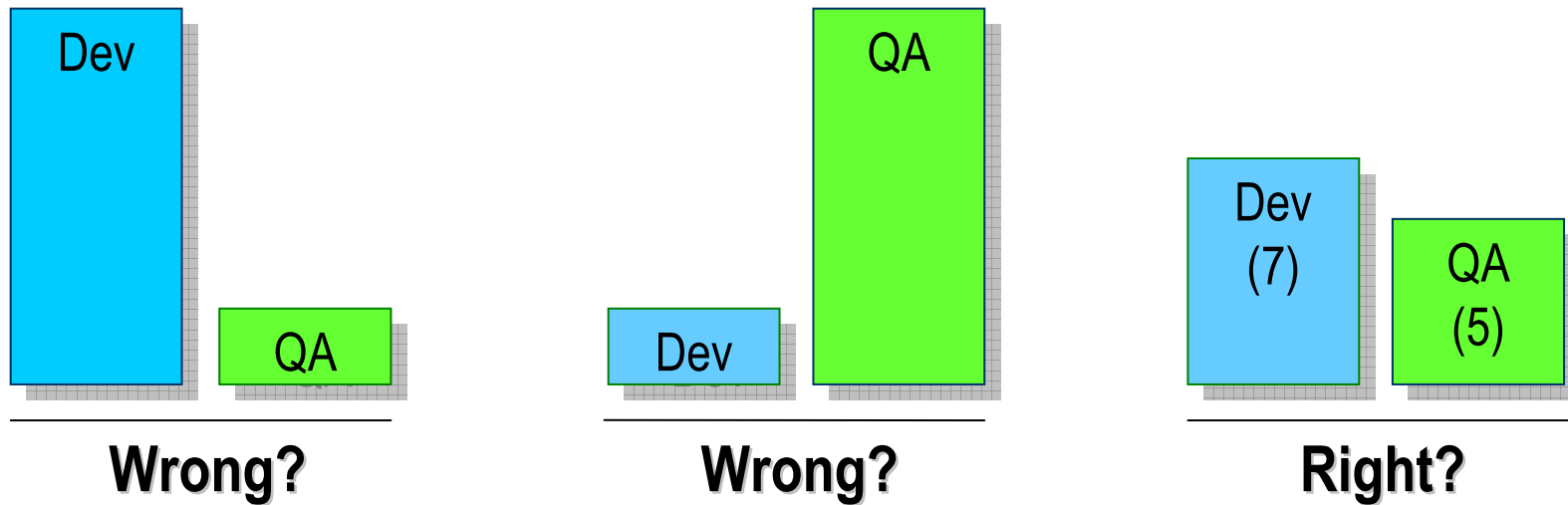
# The Testing Culture: Part 3



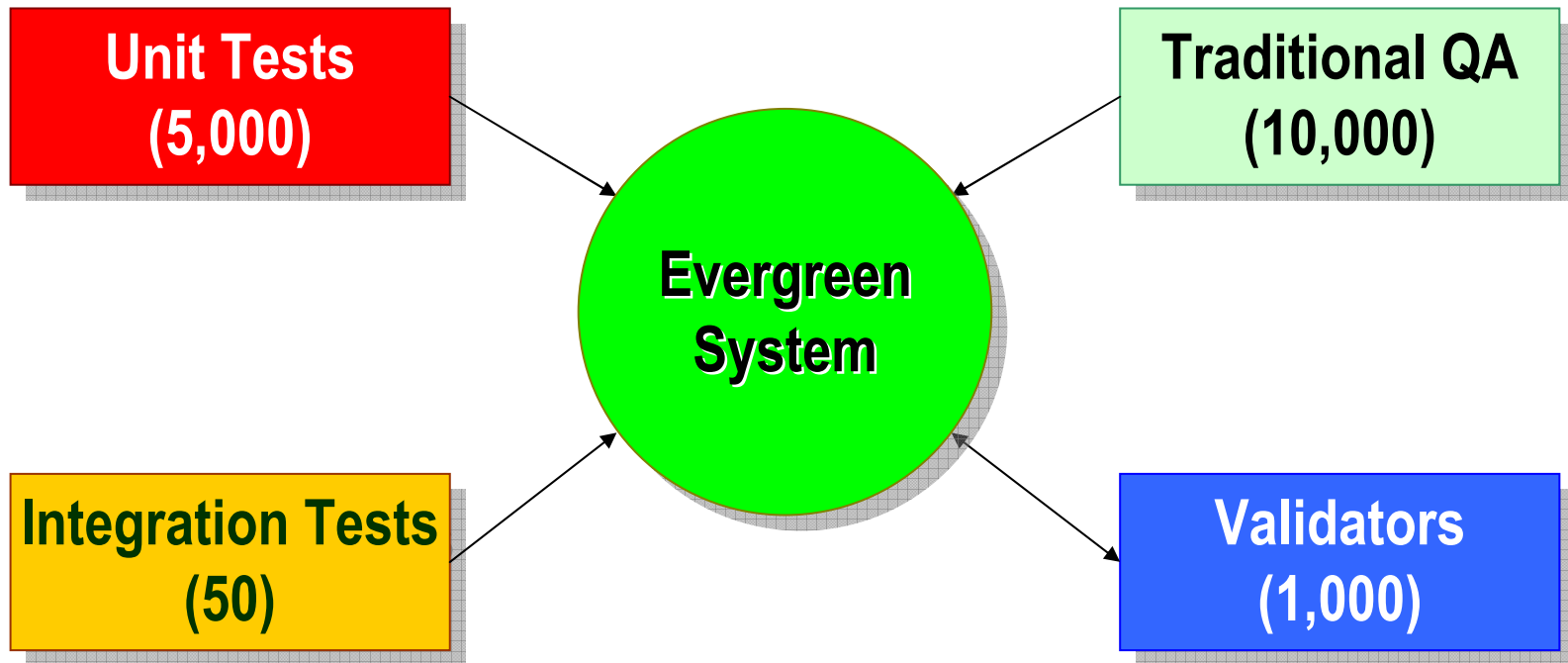
# The Testing Culture: Part 3



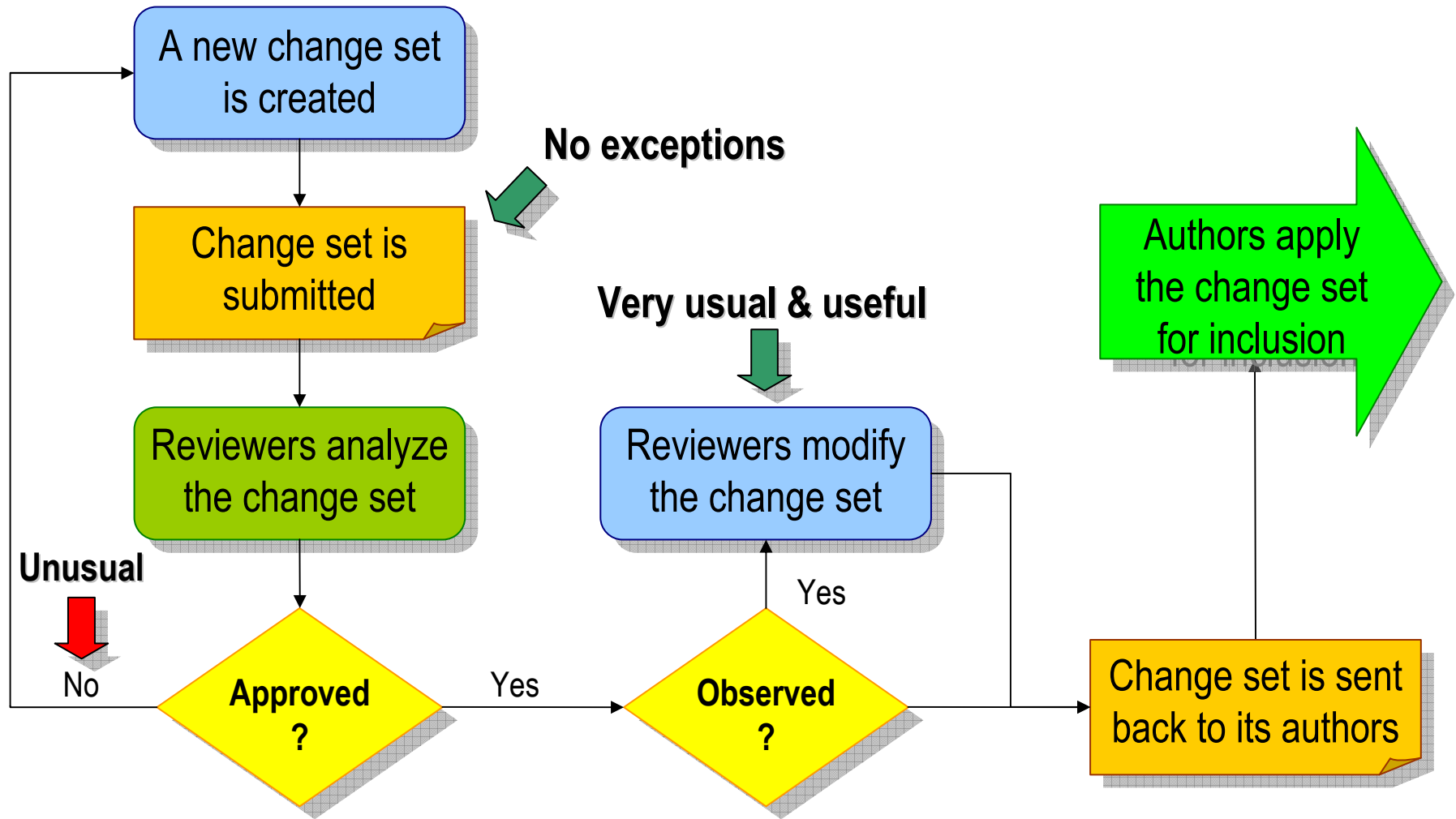
# The Testing Culture: Part 4



# The Evergreen System



# Internal QA: code revision



# Complexity metrics

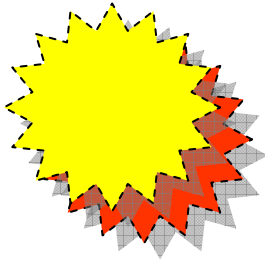
Size
# classes
# methods
LOC

Structure
# arguments
# slots
#msgs/method
class depth

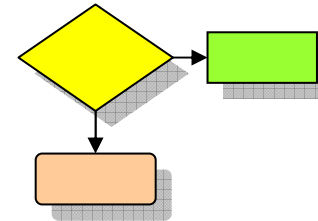
Linguistic
words
incursion
polymorphism
# identifiers

QA
% coverage
% recidivism
age

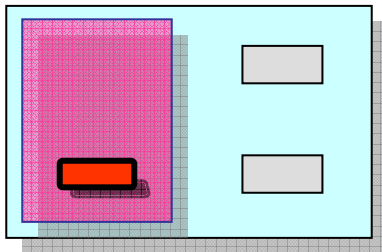
# Ad hoc GUI programming



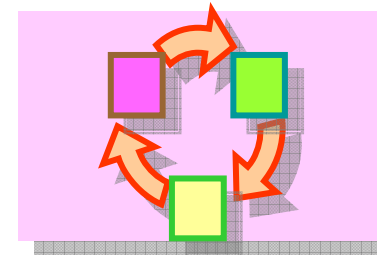
**Fuzzy model**



**Model logic**

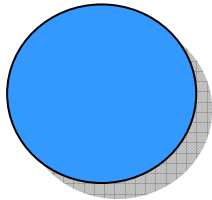


**Partial refresh**

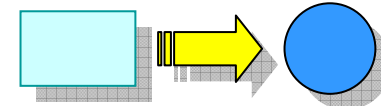


**Circular update**

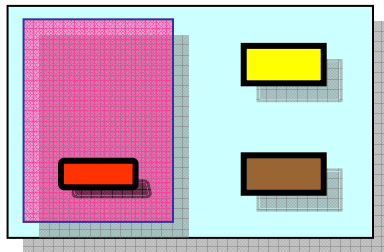
# Predictable GUI programming



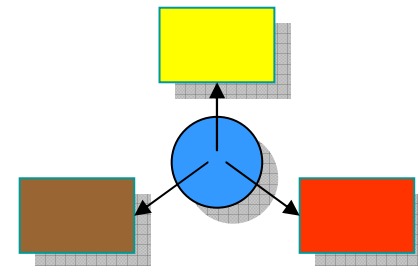
**Single model**



**No model logic**



**Extreme refresh**

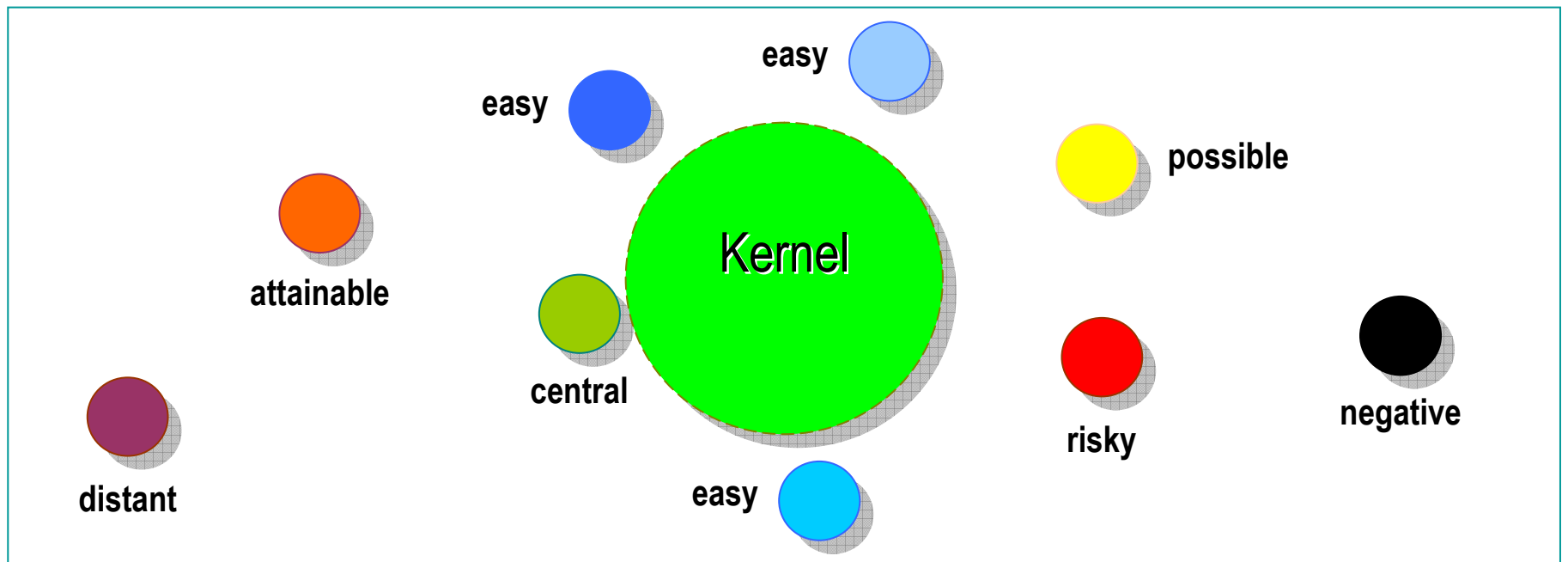


**Unidirectional update**

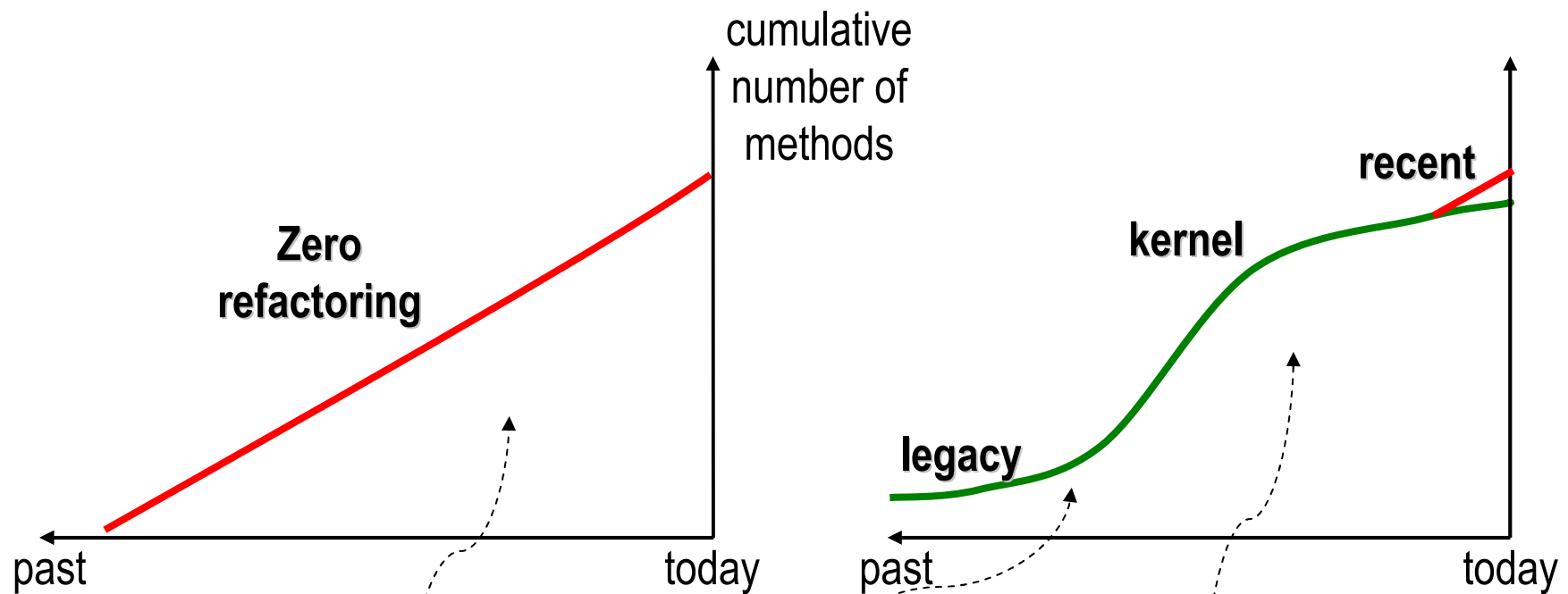
# Questions

- What's the overhead of writing tests and validators?
- Is Extreme Validation fast enough?
- What's the overhead of code reviewing?
- What's the impact of databases?
- What if extreme refreshing generates flicking?

# The system kernel

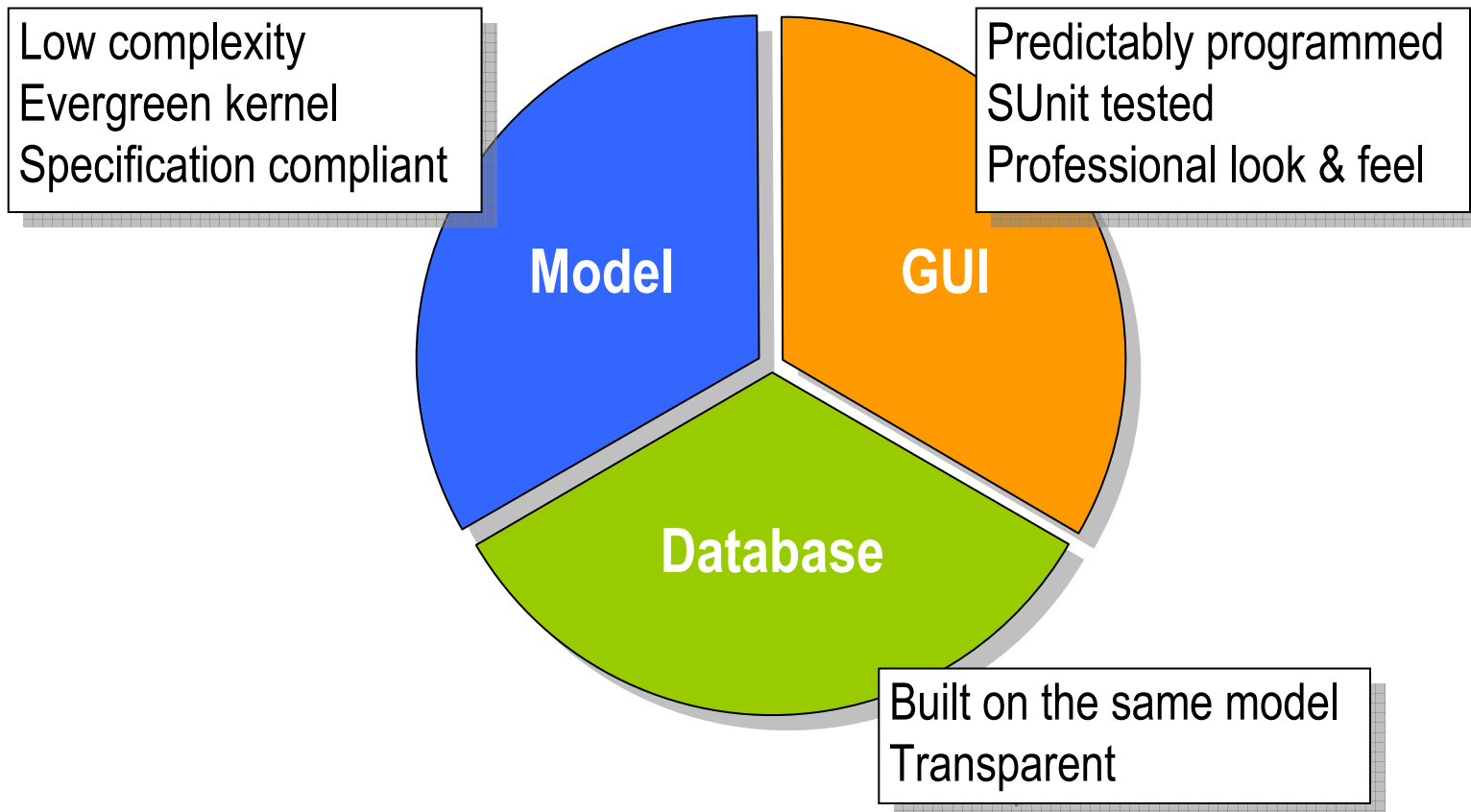


# Controlling the complexity

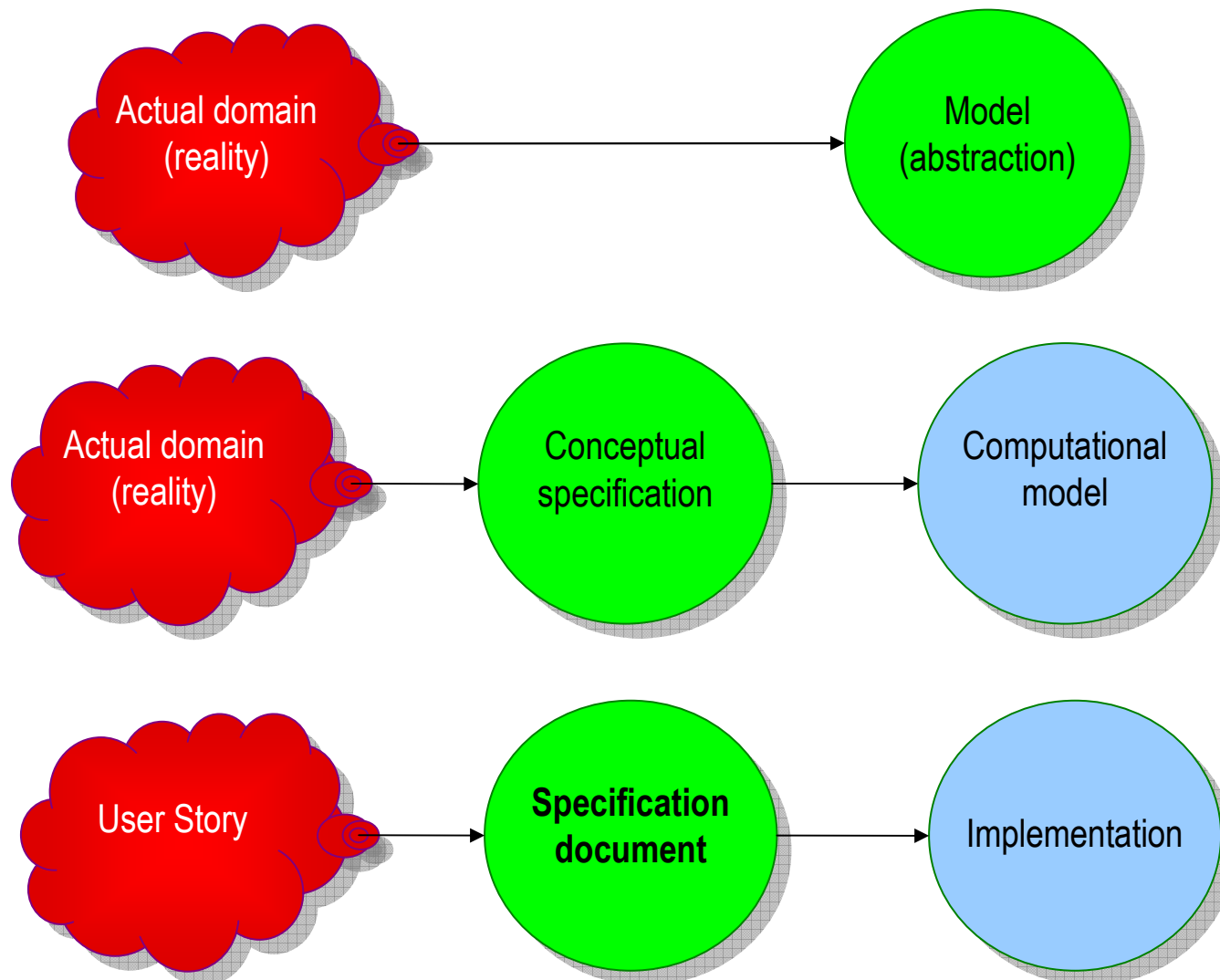


Concavity = 0	Concavity > 0	Concavity < 0
No refactoring	Refactoring	Kernel reuse

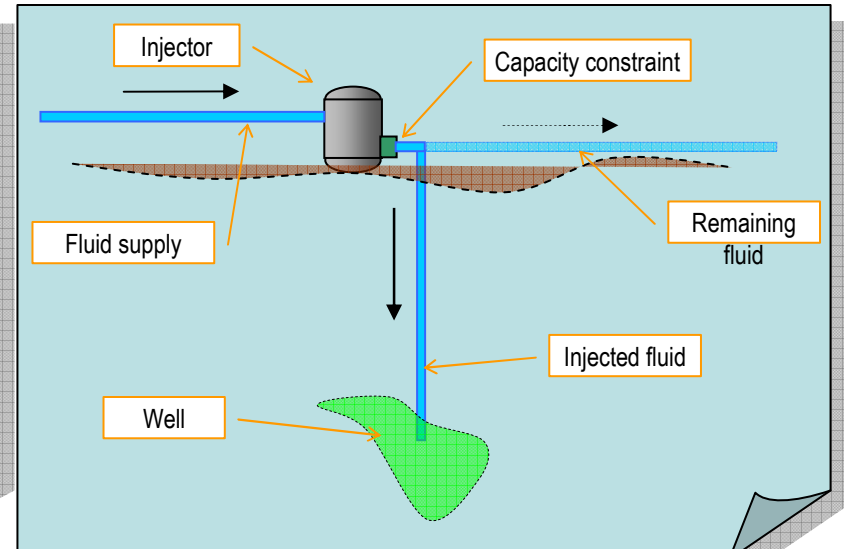
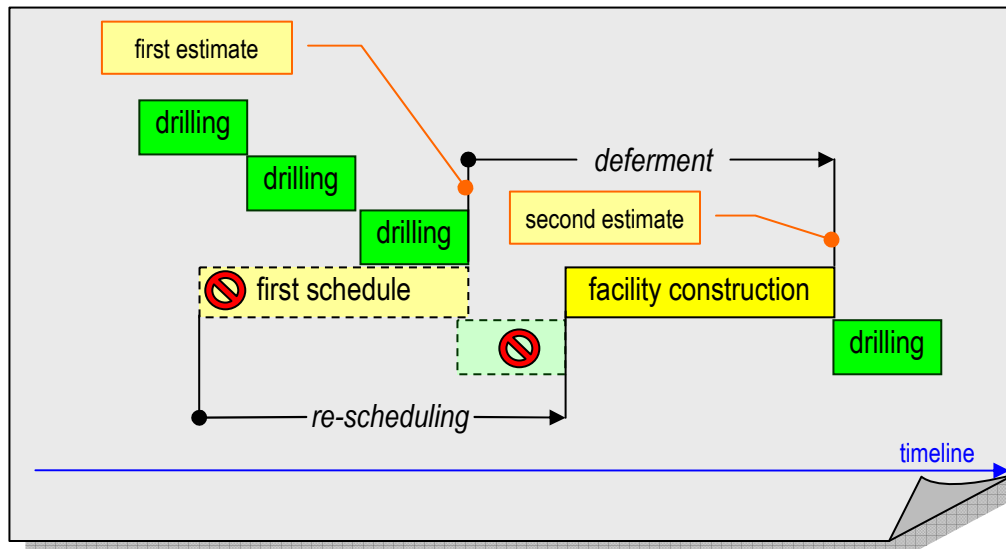
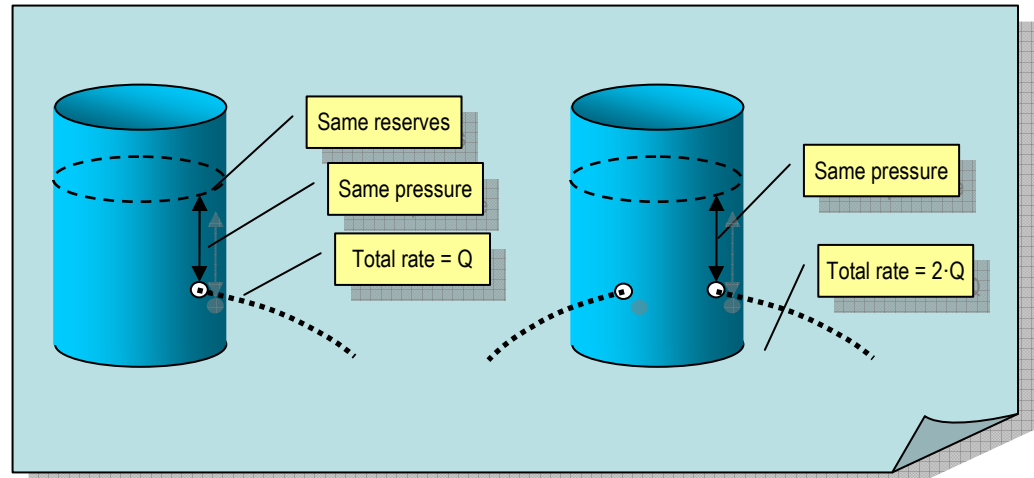
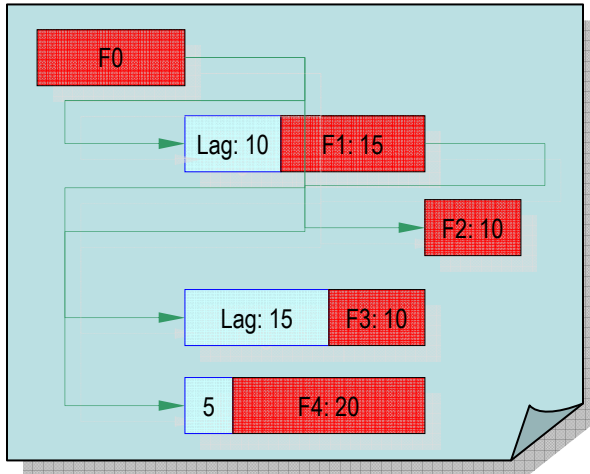
# Managing the triad



# Documentation



# Specification documents



# The usual hierarchy

