Coupling & Cohesion

Pillars of Software Development

Steven Teleki
Managing Director of Engineering, The Advisory Board Company
Past Chair, IEEE Computer Society, Austin Chapter

Twitter Version

Composing a software system from ***weakly coupled*** and ***highly cohesive components*** will ***increase code quality*** and ***developer productivity***.

Agenda

Composing a software system from ***weakly coupled*** and ***highly cohesive components*** will ***increase code quality*** and ***developer productivity***.

code examples

Coupling

Composing a software system from ***weakly coupled*** and ***highly cohesive components*** will ***increase code quality*** and ***developer productivity***.

Dictionary: Coupling

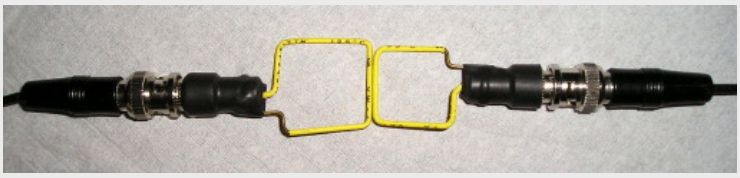
coupling (ˈkʌplɪŋ)

— n

1. a mechanical device that connects two things
2. a device for connecting railway cars or trucks together
3. ...

coupling. Dictionary.com. *Collins English Dictionary - Complete & Unabridged 10th Edition.* HarperCollins Publishers. http://dictionary.reference.com/browse/coupling (accessed: December 28, 2011).

Parasitic Coupling



Loops for Coupling Measurement

Smith, Douglas C. *The Square Shielded Loop*. http://emcesd.com/tt2008/tt070508.htm Web. 26 Dec 2011.

Myers on Coupling

"Coupling is a measure of the relationship among modules."

Myers, Glenford. *Reliable Software Through Composite Design*. New York: Petrocelli/Charter, 1975. Print.

Kernighan and Plaguer on Coupling

"... the modules are kept as uncoupled as possible, and the coupling that exists is kept visible."

Kernighan, Brian W. and P.J. Plauger. *Software Tools*. Reading: Addison-Wesley, 1976. Print.

Meyer on Coupling

Two rules of modularity:

"Few Interfaces: Every module should communicate with as few others as possible."

"Small Interfaces or Weak Coupling: If two modules communicate, they should exchange as little information as possible."

Meyer, Bertrand. *Object-Oriented Software Construction, 2nd Ed.* Upper Saddle River: Prentice Hall PTR, 1997. Print.

Lakos on Coupling

"Physical" vs. "logical" coupling

"Insulation is the process of avoiding or removing unnecessary compile-time coupling."

Lakos, John. *Large-Scale C++ Software Design*. Reading: Addison-Wesley, 1996. Print.

McConnell on Coupling

"Coupling describes how tightly a class or routine is related to other classes or routines."

McConnell, Steve. *Code Complete 2*. Redmond: Microsoft Press, 2004. Print.

Kinds of Coupling

- Simple-data-parameter coupling: all data are primitive data types.
- Simple-object coupling: "has-a"
- Object-parameter coupling: parameter is a non-primitive object
- Semantic coupling: relying on some knowledge non-deducible from code

McConnell, Steve. *Code Complete 2*. Redmond: Microsoft Press, 2004. Print.

Larman on Coupling

"Coupling is a measure of how strongly one element is connected to, has knowledge of, or relies on other elements."

Larman, Craig. *Applying UML and Patterns 3rd Ed.* Upper Saddle River: Prentice Hall PTR, 2005. Print.

Dependencies

Dependency is used as a synonym for coupling.

M *depends-on* N

```
class M extends N {                 is-a
class M implements N {              implements-a
    N n1;                           has-a
    N f(N n2) {                     parameter coupling
        N n3;                       local coupling
    }
}
```

Cohesion

Composing a software system from ***weakly coupled*** and ***highly cohesive components*** will ***increase code quality*** and ***developer productivity***.

Dictionary: Cohesion

cohesion (kəʊˈhiːʒən)

— n

1. the act or state of cohering; tendency to unite

cohesion. Dictionary.com. *Collins English Dictionary - Complete & Unabridged 10th Edition*. HarperCollins Publishers. http://dictionary.reference.com/browse/coupling (accessed: December 28, 2011).

Dictionary: Cohere

cohere (kəʊˈhɪə)

— vb

1. to hold or stick firmly together
2. to be connected logically; be consistent
3. physics to be held together by the action of molecular forces

cohere. Dictionary.com. *Collins English Dictionary - Complete & Unabridged 10th Edition*. HarperCollins Publishers. http://dictionary.reference.com/browse/coupling (accessed: December 28, 2011).

Cohesion in Military

"The bonding together of members of an organization in such a way as to sustain their will and commitment to each other, their unit and the mission."

- Earnest G. Cunningham

Powell, Kenneth, et.al. *Unit Cohesion, Cross Leveling and Readiness*. BCP International Limited. 2006.

Myers on Cohesion

"Module strength is a measure of the relationship among elements in individual modules."

Myers, Glenford. *Reliable Software Through Composite Design*. New York: Petrocelli/Charter, 1975. Print.

Kernighan and Plaguer on Cohesion

"Each module is also cohesive*: it has good reasons for being a separate entity. It is not a tangle of multiple functions lumped arbitrarily, nor is it a displaced fragment of some other module."*

Kernighan, Brian W. and P.J. Plauger. *Software Tools*. Reading: Addison-Wesley, 1976. Print.

Constantine & Yourdon on Cohesion

"Cohesion of each module -- how tightly bound or related its internal elements are to one another."

- Coincidental
- Logical
- Temporal
- Procedural
- Communicational
- Sequential
- Functional

Constantine, Larry and Ed Yourdon. *Structured Design*. Englewood Cliffs: Prentice Hall, 1979. Print.

McConnell on Cohesion

"Cohesion refers to how closely all the routines in a class or all the code in a routine support a central purpose-how focused the class is."

McConnell, Steve. *Code Complete* 2. Redmond: Microsoft Press, 2004. Print.

Larman on Cohesion

"Cohesion (or more specifically functional cohesion) is a measure of how strongly related and focused the responsibilities of an element are."

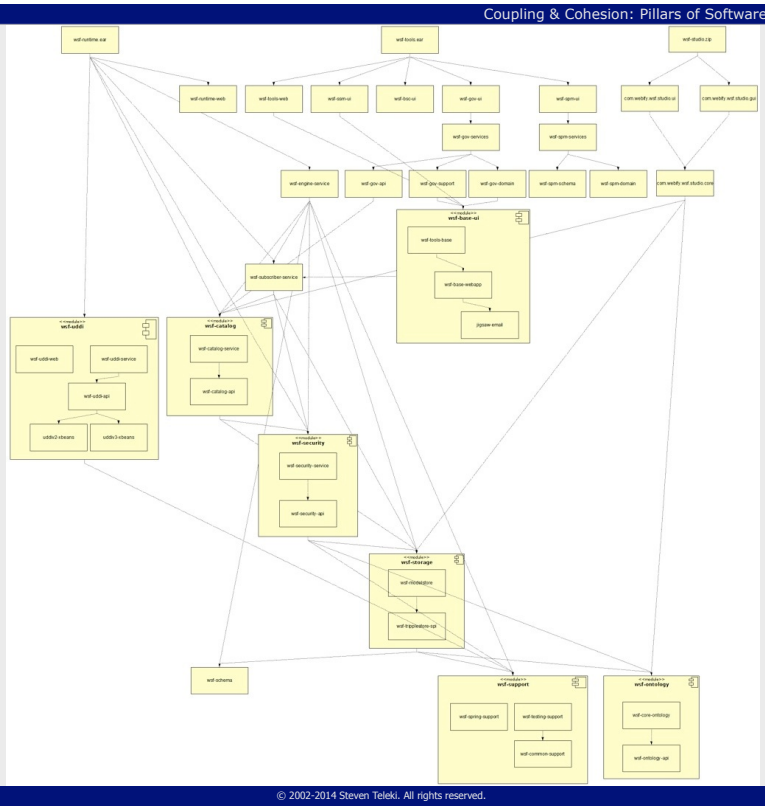
Larman, Craig. *Applying UML and Patterns 3rd Ed.* Upper Saddle River: Prentice Hall PTR, 2005. Print.

Coupling & Cohesion

Practical definitions:

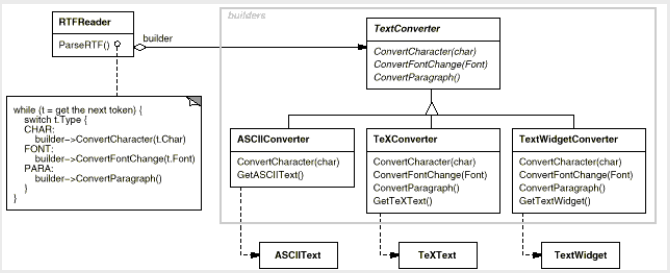
Coupling is any relationship between two software parts.

Cohesion is the degree to which the responsibilities of a software part form a meaningful unit.



Design Patterns are Patterns of Coupling & Cohesion

Builder



Gamma, Erich et. al. *Design Patterns*. Reading: Addison Wesley, 1995. Print.

Components

Composing a software system from ***weakly coupled*** and ***highly cohesive*** ***components*** will ***increase code quality*** and ***developer productivity***.

Modular Design

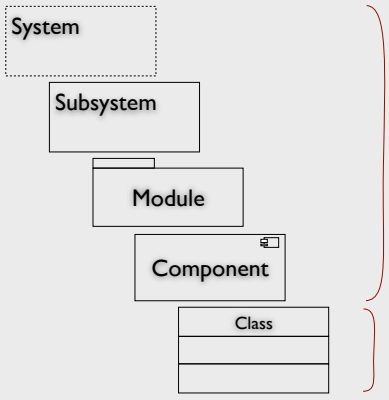
Build the system from cooperating components.

"Language shapes the way we think, and determines what we can think about."

— B.L. Whorf

Stroustrup, Bjarne. *The C++ Programming Language*. Reading: Addison-Wesley, 1987. Print.

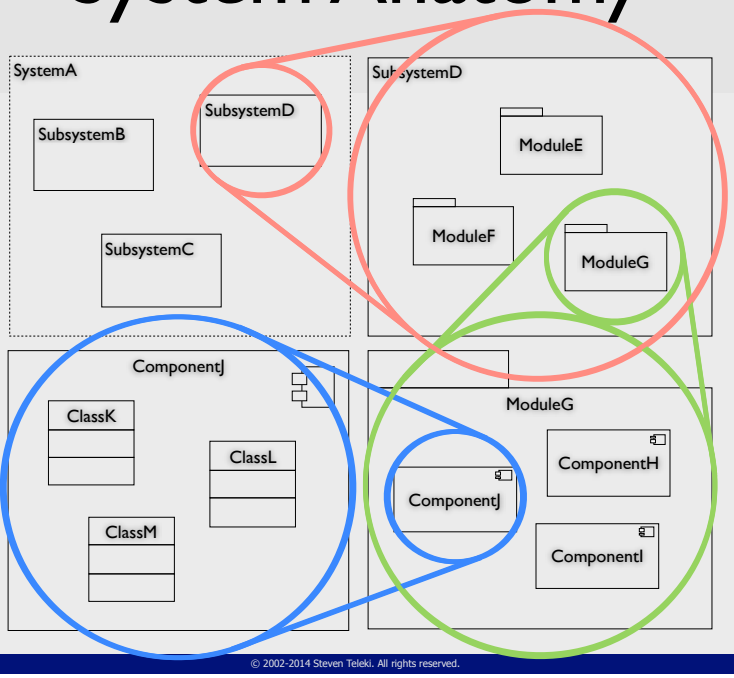
Terminology



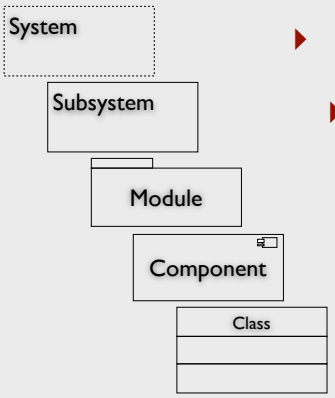
Represent Entities from the Problem Domain

Abstract Data Type (ADT)

System Anatomy



Typical Sizes



- 250 (100 - 400) KLOC
  - 50 (20 -100) KLOC
    - 20 (10 - 40) KLOC
      - 2 (1 - 4) KLOC
        - 200 (100 - 1K) LOC

Version, Version, Version

- Version at the Module or Subsystem level
- Major, Minor, Revision, Build
- Pretend that each versioned part is an independent project, like an "open source" project (even if only used "in-house")

Quality & Productivity

Composing a software system from ***weakly coupled*** and ***highly cohesive components*** will ***increase code quality*** and ***developer productivity***.

Quality & productivity are tightly connected.

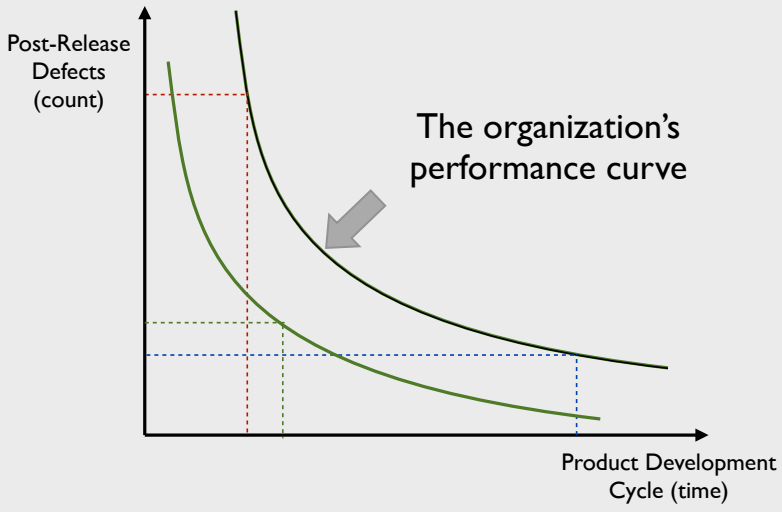
"To produce a high-quality software system, each of the system's parts must also be of high quality."

— Watts S. Humphrey

The Challenge

Shorten the product lifecycle while ***at the same time*** reduce the number of post-release defects.

Find the Optimum



Some Data

- System Size: 70 - 350 KLOC
- Defect Density: 340 - 24,000 def/MLOC
- Average Defect Fix Time: 1 - 32 hours

Code Example

An attempt to show these concepts
on a small-scale program

The Problem

- Generate the license plate to be issued
- Follow a pattern, like: LDD-LLL
- Save/Load license plate is provided

1. TODO: Initialize the system
2. TODO: Generate next plate

Example

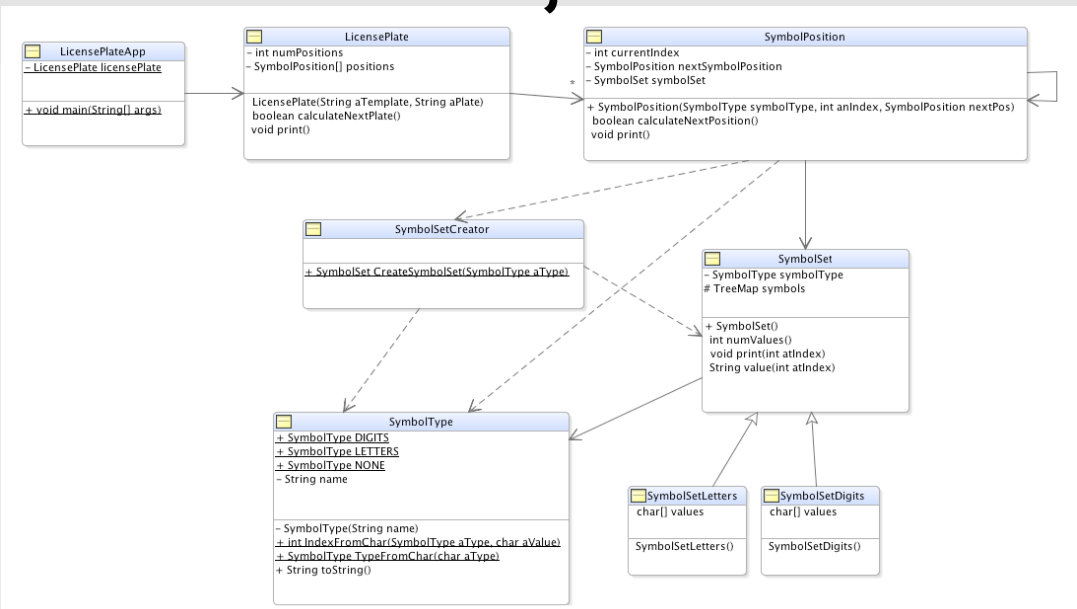
- Current pattern: `LDD-LLL`
- Last plate issued: `M59-ZZZ`
- Next plate will be: `M60-AAA`

L - letters [A..Z]
D - digits [0..9]

Solution 1: One Routine

```
private static void solution1() {
  String plateTemplate  = "LDDLLL";
  String plateValue     = "M59ZZZ";
  String nextPlateValue = "";

  System.out.println(plateValue);
  char c;
  boolean carry = true;

  int numPositions = plateTemplate.length();
  for (int i = numPositions - 1; i >= 0; i--) {
      if (carry) {
          carry = false;
          c = plateValue.charAt(i);
          c = ++c;
          switch (plateTemplate.charAt(i)) {
          case 'L':
              if (c > 'Z') {
                  c = 'A';
                  carry = true;
              }
              break;
          case 'D':
              if (c > '9') {
                  c = '0';
                  carry = true;
              }
              break;
          }
      } else
          c = plateValue.charAt(i);
      nextPlateValue = c + nextPlateValue;
  }
  System.out.println(nextPlateValue);
}
```


Solution 2: Object-Oriented



Additional Requirements for License Plates

1. Support changing patterns
2. Support graphics
3. Support additional characters

Requirements Evolve

- Current pattern: `GLL-LDDD`
- Last plate issued: `kRM-Z999`
- Next plate will be: `kRN-A000`

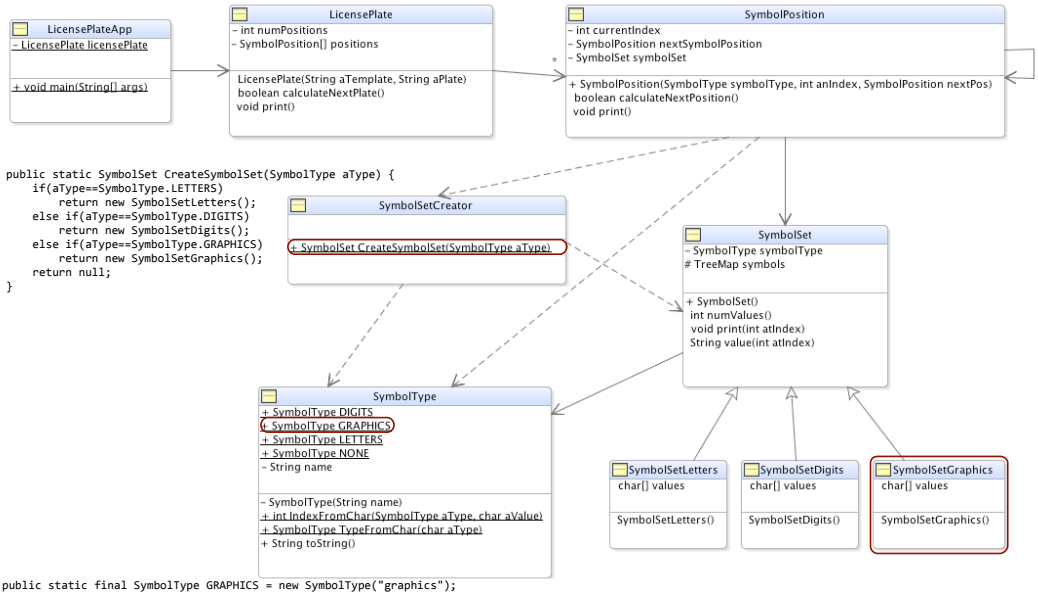
L - letters [A..Z]
D - digits [0..9]
G - graphics [a..z]

Solution 1 Revisited

```
private static void solution1() {
  String plateTemplate  = "LDDLLL";
  String plateValue     = "M59ZZZ";
  String nextPlateValue = "";

  System.out.println(plateValue);
  char c;
  boolean carry = true;

  int numPositions = plateTemplate.length();
  for (int i = numPositions - 1; i >= 0; i--) {
      if (carry) {
          carry = false;
          c = plateValue.charAt(i);
          c = ++c;
          switch (plateTemplate.charAt(i)) {
          case 'L':
              if (c > 'Z') {
                  c = 'A';
                  carry = true;
              }
              break;
          case 'D':
              if (c > '9') {
                  c = '0';
                  carry = true;
              }
              break;
          }
      } else
          c = plateValue.charAt(i);
      nextPlateValue = c + nextPlateValue;
  }
  System.out.println(nextPlateValue);
}
```

```
private static void solution1() {
  String plateTemplate  = "GLLLDDD";
  String plateValue     = "kRMZ999";
  String nextPlateValue = "";

  System.out.println(plateValue);
  char c;
  boolean carry = true;

  int numPositions = plateTemplate.length();
  for (int i = numPositions - 1; i >= 0; i--) {
      if (carry) {
          carry = false;
          c = plateValue.charAt(i);
          c = ++c;
          switch (plateTemplate.charAt(i)) {
          case 'L':
              if (c > 'Z') {
                  c = 'A';
                  carry = true;
              }
              break;
          case 'D':
              if (c > '9') {
                  c = '0';
                  carry = true;
              }
              break;
          case 'G':
              if (c > 'z') {
                  c = 'a';
                  carry = true;
              }
              break;
          }
      } else
          c = plateValue.charAt(i);
      nextPlateValue = c + nextPlateValue;
```


Solution 2 Revisited



Conclusion

Coupling & Cohesion

Practical definitions:

Coupling is any relationship between two software parts.

Cohesion is the degree to which the responsibilities of a software part form a meaningful unit.

First Order Principle of Software Development: **Increase Cohesion & Reduce Coupling**

"To achieve quality, there is no substitute for knowledge."

— W. Edwards Deming

The **only** *source of agility is* **knowledge.**

Your Letters & Comments are Welcome!

Steven Teleki
steve@teleki.net

Visit: http://steven.teleki.net/
- Software Development Reading List
- Slides from this talk and previous talks

Connect on LinkedIn and Twitter (SteveTeleki)