Software Development and Refactoring

CSCI3081W

Program Design and Development

The Parts and the Process

- **Specification** (**Requirements)** : What does the client want?
- **Design (Architecture)** : The foundation to which all code must attach.
- **Implementation**.*the fun part*. Who does what? Any standards?
- **Validate (Testing)**. Who? When? How?
- **Evolution (Maintenance)**. Bugs or new features?
  - Releases. How many and when

Terminology

Software :

ALL components including code, design, documentation, and user manuals.

Software Development Process :

Set of activities that produce software.

Process Model or Paradigm :

Simplified description of a process (a specific approach or method for the software development process).

Upstream Activities (Prerequisites) :

Planning and design PRIOR to coding.

Software Engineering :

A discipline concerned with ALL aspects of software development.

What would Berry say …

The Inevitable Pain of Software Development

by Daniel Berry

- **"The hardest single part of building a software system is deciding precisely what to build …." (via Fred Brooks[13])**

Berry (and many others) claim that

it is the **requirements** that are the most difficult aspect.

Build the right software. This is the hard part.

Build the software right. This is still hard, but pales in comparison.

What would Berry say …

The Inevitable Pain of Software Development

by Daniel Berry

- "The hardest single part of building a software system is deciding precisely what to build …." (via Fred Brooks[13])

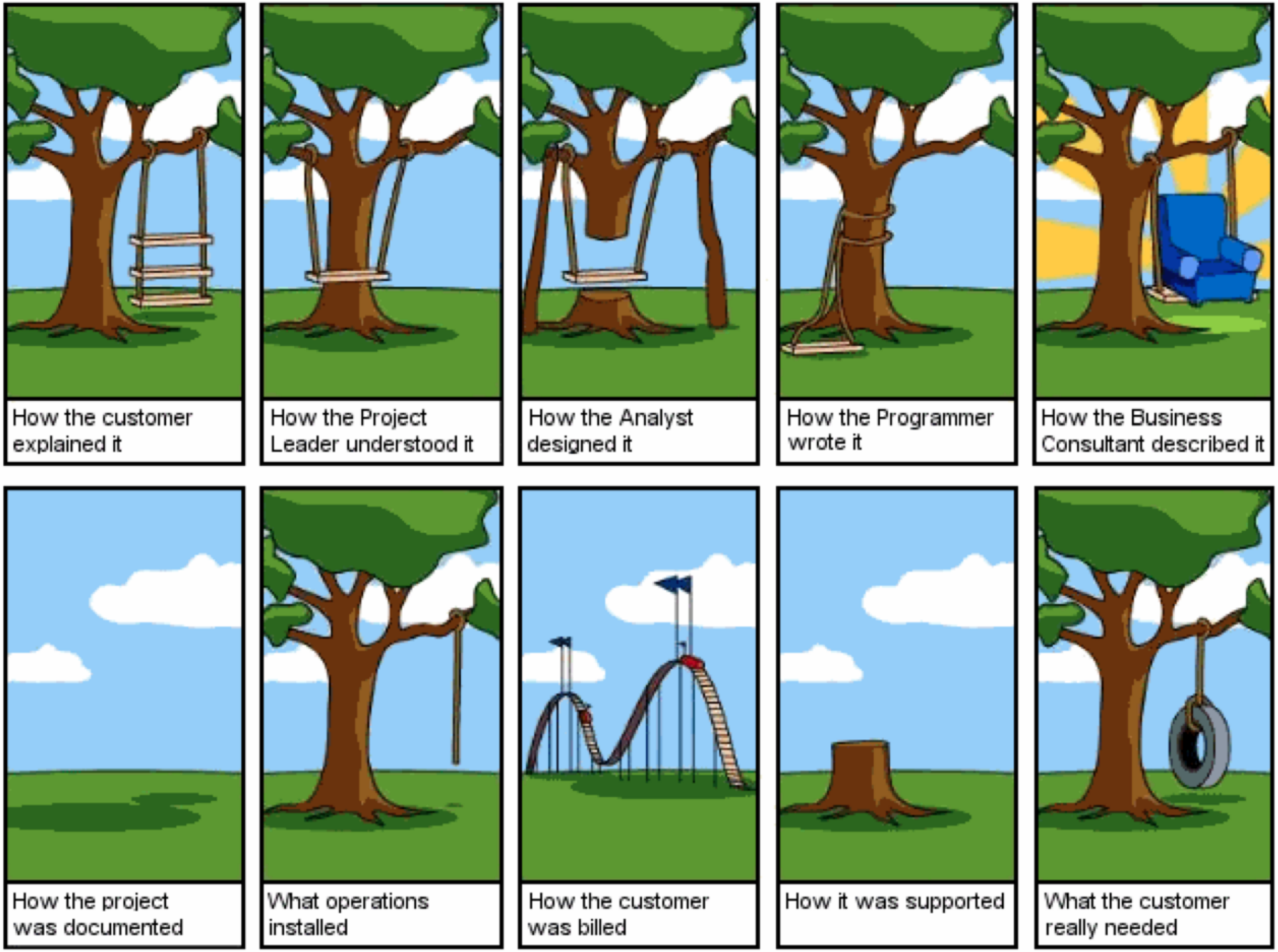
- **"But, what is so difficult about understanding requirements?"**

"Michael Jackson, in his Keynote address at the 1994 International Conference on Requirements Engineering [22] said that two things are known about requirements:

They will change.

They will be misunderstood."

Which brings to mind this cartoon Alex Gorbatchev posted a few days ago:



http://www.codinghorror.com/blog/2005/03/on-software-engineering.html

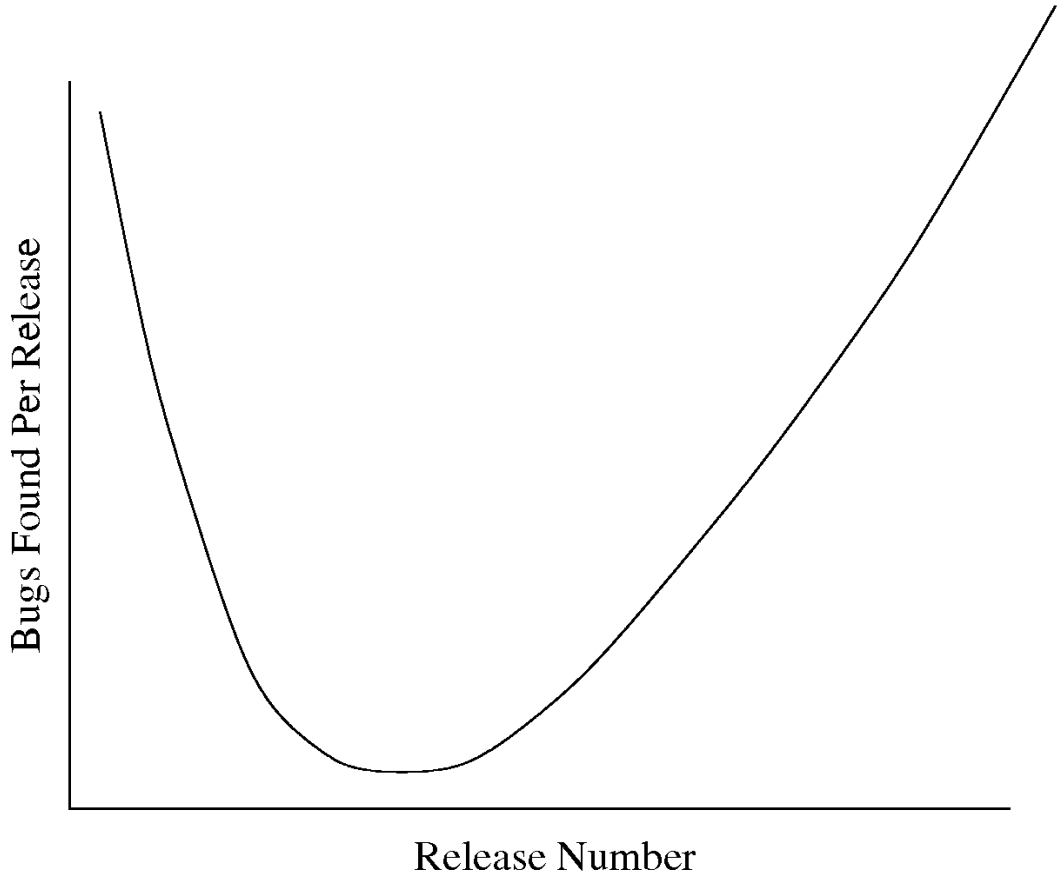
What would Berry say …

The Inevitable Pain of Software Development

by Daniel Berry

- "The hardest single part of building a software system is deciding precisely what to build …." (via Fred Brooks[13])
- "But, what is so difficult about understanding requirements?"
- **"Heretofore, no single method has put a dent into this essential problem, although all the discovered methods have combined to improve programming."**

Belady-Lehman Graph



1. What does the minimum point signify?
2. What does the downward slope signify?
3. What does the upward slope signify?
4. What would an ideal graph look like (that is somewhat realistic)?

Belady-Lehman Graph

"The time at which the minimum point comes and the slopes of the curves before and after the minimum point vary from development to development. The more complex the CBS is, the steeper the curve tends to be."

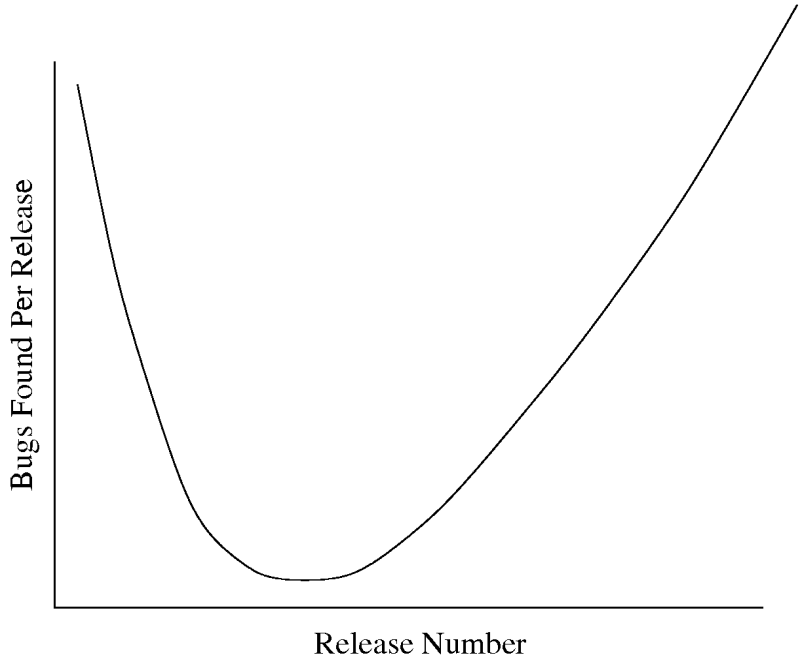


Figure 1: Belady-Lehman Graph

What does this have to do with software development methods and tools?

Barry is asking

"Why is there no silver bullet, and why can there not be a silver bullet?"

Process Models

Requirements • Design • Implement • Test • Evolve

- **Waterfall Model** (ordered by activity, predictive, heavyweight)
- **Evolutionary Development** (ordered by function, adaptive, lightweight).
- Formal Systems Development (mathematical statements, automatic code generation)
- Reuse-Based Development (integration of existing components)

- Build-and-Fix (the no-formal-model model)

Waterfall vs Iterative

Waterfall – 1 Year	**Iterative – 1 Year**
• 2-month analysis phase, • 4-month design phase, • 3-month coding phase, and • 3-month testing phase. Release is at the end of the year.	• Each iteration is 3 months. • Each iteration produces a release with ¼ of the total functionality. • Each iteration has analysis, design, code, and test.
Activity Based	Functionality or Feature Based

The Waterfall Philosophy

The Inevitable Pain of Software Development by Daniel Berry

"The hardest single part of building a software system is deciding precisely what to build ...." (via Fred Brooks[13])

Berry (and many others) claim that

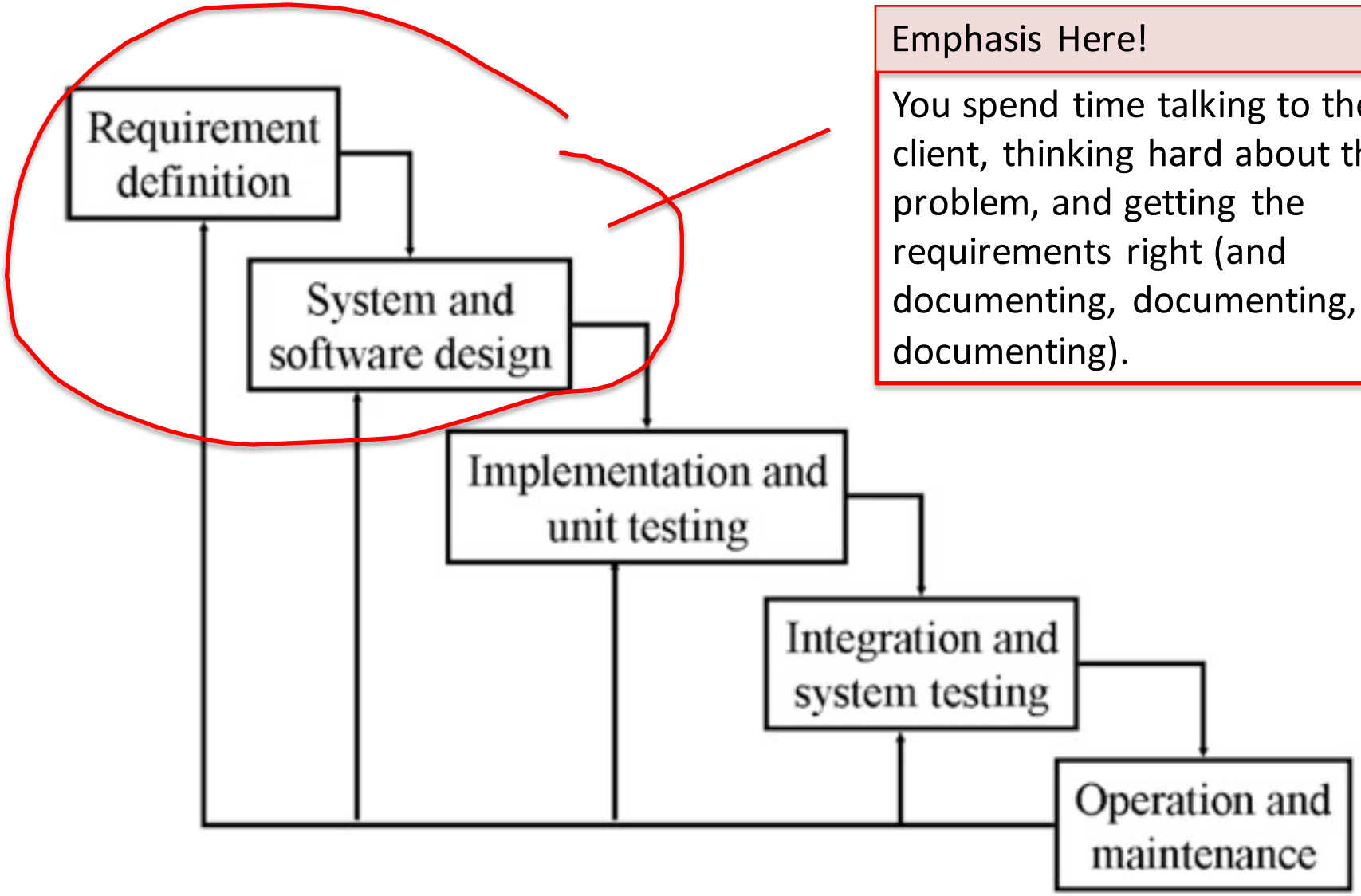
it is the **requirements** that are the most difficult aspect.

Waterfall Philosophy
Prevent change through careful planning.

Build the right software. **This is the hard part.**

Build the software right. This is still hard, but pales in comparison.

Waterfall Model (with feedback)



Emphasis Here!

You spend time talking to the client, thinking hard about the problem, and getting the requirements right (and documenting, documenting, documenting).

Software Engineering. 6th Edition. Ian Sommerville. Fig. 3.1 p.45

Prerequisites : Requirements

specification • requirements engineering • functional specs • spec • analysis

does your client

What exactly ~~do you~~ want the software to do ?

Waterfall vs Agile

For whom are the requirements written?

How are they communicated?

You have to understand them to construct the code!

Prerequisites : Requirements

"Requirements analysis involves figuring out what the users and customers of a software effort want the system to do."

"The most important thing is communication with your users and customers."

The Fowler Perspective
(UML – Unified Modeling Language)

- Use Cases : how people will interact with the system. (Use domain knowledge.)
- Class Diagram : Conceptual Perspective.

UML might help to
Visualize, Share, Clarify, and/or Document Requirements.

You have to know something about the business – behavioral science, restaurants, clothing design, robots, coffee pots, ...

Adapt to Change

Evolutionary / Agile / Iterative Development

Iterative Philosophy
Embrace change by evolving code and requirements.

- Embrace change – plan for it.
- Reevaluate requirements constantly.
- Break process down by functionality (not activity).

- Refactoring
  Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior. (Fowler. www.refactoring.com)
- Unit / Automated Regression Testing
- Continuous Integration

Code Complete – Refactoring Advice

REFACTORING DOES NOT CHANGE CODE BEHAVIOR

- SAVE THE CODE you start with
- Small changes, one at a time
- Make a list
- Frequent checkpoints
- Use your compiler warnings ☺
- Add tests
- Retest

Warning Signs – Need to Refactor

- Code is duplicated
- Routine is too long
- Loops too deeply nested
- Abstraction not consistent (weak cohesion)
- Parallel class modifications needed
- Related data not managed together
- Class data too intermixed (strong coupling)
- Bad names

What did Berry say ...

The Waterfall model would work if the programmers could understand a stage thoroughly and document it fully before going on [29].

The Pain :

- Understanding is difficult and elusive, and in particular, documentation is a pain.
- It is a pain to some because it obstructs programmers from coding in favor of seemingly endless, useless documentation.
- It becomes a pain when full requirements are learned only after significant implementation (The Wicked Problem).
- It becomes a pain when new requirements are continually discovered.

"Michael Jackson said that two things are known about requirements[22]:

They will change.
They will be misunderstood."

What did Berry say ...

The PAIN of Agile

- Refactoring, itself, is painful [18]. You may have to throw out perfectly good code.
- Test cases have to be written PRIOR to writing code.
- Client has to be present or available at all times.

How do you reconcile these ideas?

The philosophy behind Waterfall is to first fully understand the software requirements, then plan completely and accordingly.

"Requirements will change.

Requirements will be misunderstood."

Evolutionary Methods embrace change. "Design is part of the programming process and as the program evolves the design changes."

Changes to software incur costs and complications. The further along in the process, the more difficult and expensive it gets.

To maximize the usefulness of any method/tool, you must be disciplined.