UP AND RUNNING
SOFTWARE DEVELOPMENT

Agile Methodology

Presented to Jivko Sinapov's Class on 4/25/14 (HCI 573X: Web Applications)

Peter J. Hanson, CTO

Plan

About Us

We Build Software

Est. 1995

Virtual

Growing

Our engineers trying to design

Clients (some)

Tech (some)


Ruby


MySQL


CouchDB


Drupal


HTML
CSS
5
3


RAILS


mongoDB


Symfony


JavaScript


python


Magento
Open Source eCommerce


ORACLE


Microsoft
SQL Server


django


amazon
web services


php


Microsoft
.NET


ANGULARJS
by Google


WordPress


jQuery
write less, do more.

Tech (some)


Ruby

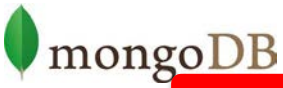

MySQL


CouchDB

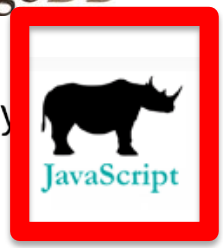

Drupal


HTML
CSS


RAILS


mongoDB


Symfony


JavaScript


python


Magento
Open Source eCommerce


ORACLE


Microsoft
SQL Server


django


amazon
web services


php


Microsoft
.NET


ANGULARJS
by Google


WordPress


jQuery
write less, do more.

What Is Agile?

- **Methodology**
  - Results-focused Approach
- **Impacts**
  - Individuals/Interactions
  - Working Software
  - Customer Collaboration
  - Handling Change

Why Agile?

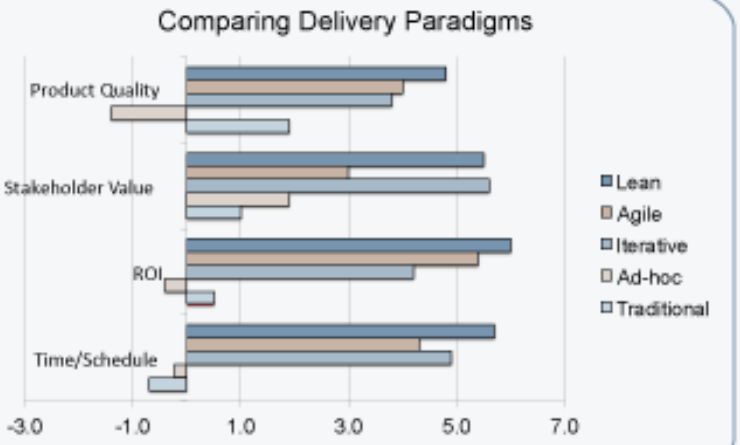
How do software development paradigms compare?

Product Quality. When it comes to the quality of the system deliver ed, what is your experience regarding the effectiveness of [paradigm] software development teams?

Stakeholder Value. When it comes to ability to deliver a solution which meets the actual needs of it's stakeholders, what is your experience regarding the effectiveness of [paradigm] software development teams?

ROI. When it comes to effective use of return on investment (ROI), what is your experience regarding the effectiveness of [paradigm] software development teams?

Time/Schedule. When it comes to time/schedule, what is your experience regarding the effectiveness of [paradigm] software development teams?



Source: 2013 IT Project Success Rates Survey, Ambysoft.com/surveys/success2013.html

SCOTT AMBLER + Associates

Agile Manifesto

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over Processes and tools
Working software over Comprehensive documentation
Customer collaboration over Contract negotiation
Responding to change over Following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas"

Sources: http://en.wikipedia.org/wiki/Agile_software_development C/O http://agilemanifesto.org/principles.html

Agile Principles

1. **"Customer satisfaction** by rapid delivery of useful software
2. Welcome **changing requirements**, even late in development
3. Working software is **delivered frequently** (weeks rather than months)
4. **Working software** is the principal measure of progress
5. **Sustainable development**, able to maintain a constant pace
6. **Close, daily cooperation** between business people and developers
7. Face-to-face **conversation** is the best form of communication (co-location)
8. Projects are built around **motivated individuals, who should be trusted**
9. **Continuous attention** to technical excellence and good design
10. **Simplicity**—the art of maximizing the amount of work not done—is essential
11. **Self-organizing** teams
12. **Regular adaptation** to changing circumstances"

Sources: http://en.wikipedia.org/wiki/Agile_software_development C/O http://agilemanifesto.org/principles.html

Customers/Change

"No plan survives contact with the enemy"

- Helmuth von Moltke

"In preparing for battle I have always found that plans are useless, but planning is indispensable.""

- Dwight D. Eisenhower

Customers /Change

Iterations

- Time boxed
- Allows adaptation

Customers/Change

- Customers are part of the team
- Frequent feedback
- Know what to decide

Customers/Change

Design

- Guidelines
- Not dictatorial

Customers/Change

- Keep it releasable
  - Stable
  - Release at the drop of a hat
- Integrate regularly
  - Don’t drift

Customers/Change

Set up your processes early

- Testing harness
- Build process
- Continuous integration

Communication

- Listen to the users
  - Get their feedback
  - Take complaints seriously
- “I am not my user”

Communication

Standups

- Keep them quick, <= 15 minutes
- Stand up, chairs are comfy
- Agenda:
  - What I did yesterday
  - What I'm doing today
  - Any blockers
- Pigs only, no chickens

Communication

Code Reviews

- Can help catch a number of issues
- Verify design
- Coding style/conventions

Don’t be this guy

Working Code

Clean/Organized Code

- Easier to read
- Less brittle
- Example

Working Code

Clean/Organized Code

```
#:: ::-| ::-| .-. :||-:: 0-| .-| ::||-| .:|-. :||
open(Q,$0);while(<Q>){if(/^#(.*)$/){for(split('-',$1)){$q=0;for(split){s/\|/:.:/xg;s/:/../g;
$Q=$_?length:$_;$q+=$q?$Q:$Q*20;}print chr($q);}}}print"\n";
#.: ::||-| .||-| :|||-| ::||-| ||-:: :|||-| .:|
```

Working Code

Clean/Organized Code

```
peteh@macbookpro-4 ~ $ cat test.pl
#:: ::-| ::-| .-. :||-:: 0-| .-| ::||-| .:|-. :||
open(Q,$0);while(<Q>){if(/^#(.*)$/){for(split('-',$1)){$q=0;for(split){s/\|/:.:/xg;s/:/../g;
$Q=$_?length:$_;$q+=$q?$Q:$Q*20;}print chr($q);}}}print"\n";
#.: ::||-| .||-| :|||-| ::||-| ||-:: :|||-| .:|
peteh@macbookpro-4 ~ $ perl test.pl
The Perl Journal
peteh@macbookpro-4 ~ $
```

Working Code

Clean/Organized Code

This easier?

```
print "The Perl Journal";
```

Working Code

Names

- Important
  - Ease of reading
  - Self documents
  - No chasing down code
- Example

Working Code

Names

```
$temp = foobar(4);

$a = $b . " " . $c;
```

Working Code

Names

```
$root_number  = square_root(4);

$full_name = $first_name . " " .
$last_name;
```

Working Code

Names

Don’t abbreviate names

What’s easier? $report or $rprt

Or

LibraryStorageSystem or LbryStrSys

“Why, Eckhardt, you oughta think about the future.” – The Joker, Batman

Working Code

Documentation

- Code should document itself
- Supplement with comments
- Maintain comments or remove
- Code is the truth

Working Code

Design Principles

Be SOLID, not STUPID

Working Code

SOLID

- Single Responsibility Principle
- Open/Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle

Working Code

STUPID

- Singleton
- Tight Coupling
- Untestability
- Premature Optimization
- Indescriptive Naming
- Duplication

Working Code

Singleton

- Anti-pattern
- Global scope hack
- Hard to test

Working Code

Singleton

```php
/* Singleton Example */
class person {
    public function __construct() {

        $log = log::getInstance();  // just coupled person to the log class here
        $log->record("instantiated person");
    }
}
```

```php
/* Dependency Injection Example */

class person {
    public function __construct(log $log) {
        $log->record("instantiated person");
    }
}
```

Working Code

Tight Coupling

Concrete implementations

```php
/* coupling example */
class person {
    public function __construct() {
        $this->db = new Database();
        $this->log = log::getInstance();
        $this->address = new Address();
    }

}
```

Working Code

Untestability

- Hard to test or no tests
- Almost always a symptom of tight coupling

Working Code

Premature Optimization

- Don’t do it
- If you are an expert, don’t do it yet
- Cost with no benefit

“Premature optimization is the root of all evil.”

- Donald Knuth

Working Code

- Indescriptive Naming
  - Covered this earlier
- Duplication
  - Don’t Repeat Yourself (DRY)
  - One source for things

Working Code

Single Responsibility Principle

> A class should never have more than one reason to change

Working Code

```php
<?php
namespace Model;

interface UserInterface
{
    public function setId($id);
    public function getId();

    public function setName($name);
    public function getName();

    public function setEmail($email);
    public function getEmail();
    public function getGravatar();

    public function findById($id);
    public function insert();
    public function update();
    public function delete();
}
```

Working Code

```php
<?php
namespace Mapper;
use ModelUserInterface;

interface UserMapperInterface
{
    public function findById($id);
    public function insert(UserInterface $user);
    public function update(UserInterface $user);
    public function delete($id);
}
```

Working Code

```php
<?php
namespace Model;

interface UserInterface
{
    public function setId($id);
    public function getId();

    public function setName($name);
    public function getName();

    public function setEmail($email);
    public function getEmail();
    public function getGravatar();
}
```

http://www.sitepoint.com/the-single-responsibility-principle/

Working Code

Open/Closed Principle

Open for extension, closed for modification

Working Code

```php
class Image
{
  private $fPath = null;
  public function __construct($fPath)
  {
    $this->fPath = $fPath;
  }
  private function GetFileExtention()
  {
    $strFileName = basename($this->fPath);
    $strExtension = array_pop(explode(".", $strFileName));
    return $strExtension;
  }
  public function SendToBrowser()
  {
    $strFileExtention = $this->GetFileExtention();
    switch ( $strFileExtention )
    {
      case 'gif' :
        header("Content-type: image/gif");
        break;
      case 'jpg' :
        header("Content-type: image/jpg");
        break;
    }
    $strFileContent = file_get_contents($this->fPath);
    echo $strFileContent;
    die();
  }
}
```

Working Code

```
1.  abstract class AnImage
2.  {
3.    public function __construct($fPath)
4.    {
5.      $this->fPath = $fPath;
6.    }
7.    public function SendToBrowser()
8.    {
9.      $this->SendFileHeader();
10.     $strFileContent = file_get_contents($this->fPath);
11.     echo $strFileContent;
12.     die();
13.   }
14.   public abstract function SendFileHeader();
15. }
```

Working Code

```php
class JpgImage extends AnImage
{
  public function SendFileHeader()
  {
    header("Content-type: image/jpg");
  }
}
```

```php
class GifImage extends AnImage
{
  public function SendFileHeader()
  {
    header("Content-type: image/gif");
  }
}
```

http://binary.freeperspective.net/countzero/2009/02/13/open-close-principle-object-orientated-design-in-php/

Working Code

Liskov Substitution Principle

> Objects can be replaced with sub classes without altering the correctness of that program

Working Code

```php
/* LSP example */

class Person {
    protected $addresses;

    public function __construct() {
        $this->addresses = array();
    }

    public function getAddresses() {
        return $this->addresses;
    }
}

class Employee extends Person {

    public function __construct() {
        $this->addresses = new AddressList();
    }
}


class User {
   protected $person;
   public function __construct(Person $person) {
       $this->person = $person;
   }

   public function getFirstAddress() {
      $addresses = $this->person->getAddresses();
      return $addresses[0];
   }
}
```

Working Code

Interface Segregation Principle

> No client should be forced to depend on methods it does not use

Working Code

```php
/* ISP Example */

interface Vehicle {
    public function startEngine();
    public function accelerate();
    public function brake();
    public function lightsOn();
    public function signalLeft();
    public function signalRight();
    public function changeGear($gear);
    public function stopRadio();
    public function ejectCD();
}
```

Working Code

```php
interface  SpeedControl{
    public function startEngine();
    public function accelerate();
    public function brake();
    public function changeGear($gear);
}

interface Signaling {
    public function lightsOn();
    public function signalLeft();
    public function signalRight();
}

interface RadioCD {
    public function stopRadio();
    public function ejectCD();
}
```

http://code.tutsplus.com/tutorials/solid-part-3-liskov-substitution-interface-segregation-principles--net-36710

Working Code

Dependency Inversion Principle

- High-level modules should not depend on low-level modules. Both should depend on abstractions.
- Abstractions should not depend upon details. Details should depend upon abstractions.

Working Code

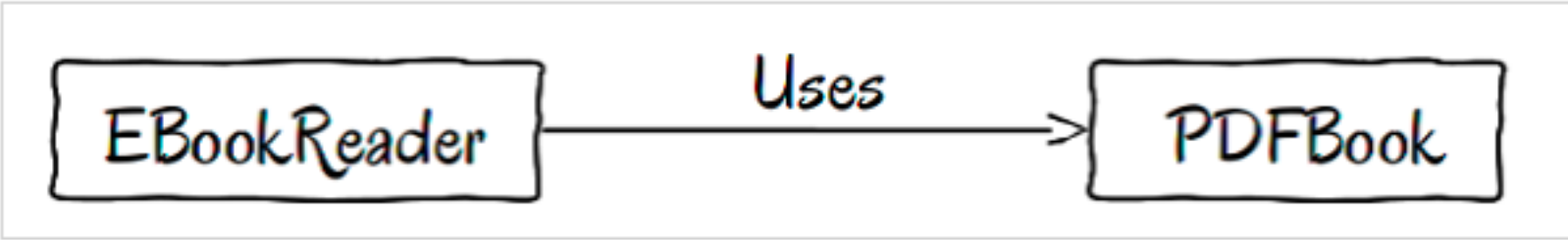
```php
class Test extends PHPUnit_Framework_TestCase {

    function testItCanReadAPDFBook() {
        $b = new PDFBook();
        $r = new PDFReader($b);

        $this->assertRegExp('/pdf book/', $r->read());
    }

}

class PDFReader {

    private $book;

    function __construct(PDFBook $book) {
        $this->book = $book;
    }

    function read() {
        return $this->book->read();
    }

}

class PDFBook {

    function read() {
        return "reading a pdf book.";
    }
}
```

Working Code

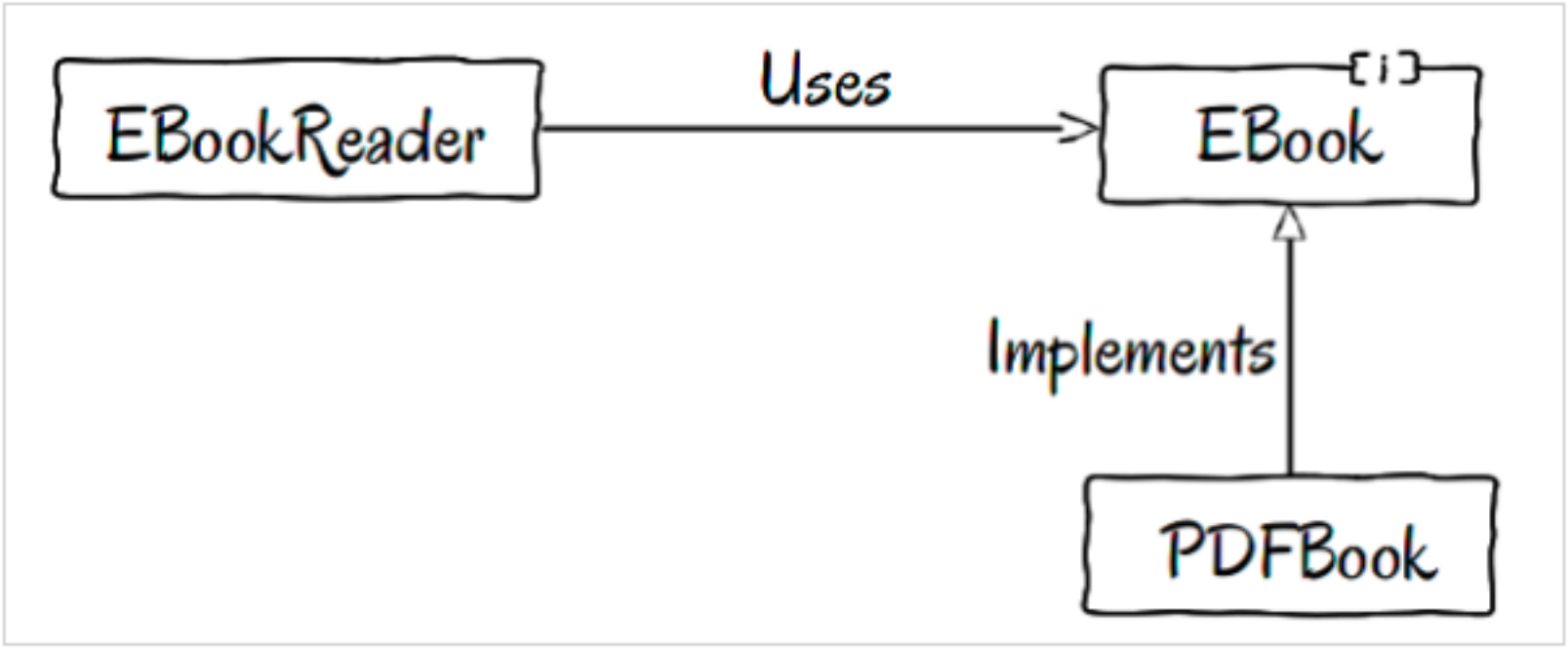
Working Code

```
01  class Test extends PHPUnit_Framework_TestCase {
02
03      function testItCanReadAPDFBook() {
04          $b = new PDFBook();
05          $r = new EBookReader($b);
06
07          $this->assertRegExp('/pdf book/', $r->read());
08      }
09
10  }
11
12  class EBookReader {
13
14      private $book;
15
16      function __construct(PDFBook $book) {
17          $this->book = $book;
18      }
19
20      function read() {
21          return $this->book->read();
22      }
23
24  }
25
26  class PDFBook {
27
28      function read() {
29          return "reading a pdf book.";
30      }
31  }
```

Working Code



Working Code



Working Code

```php
interface EBook {
    function read();
}

class EBookReader {

    private $book;

    function __construct(EBook $book) {
        $this->book = $book;
    }

    function read() {
        return $this->book->read();
    }

}

class PDFBook implements EBook {

    function read() {
        return "reading a pdf book.";
    }
}

class MobiBook implements EBook {

    function read() {
        return "reading a mobi book.";
    }
}
```

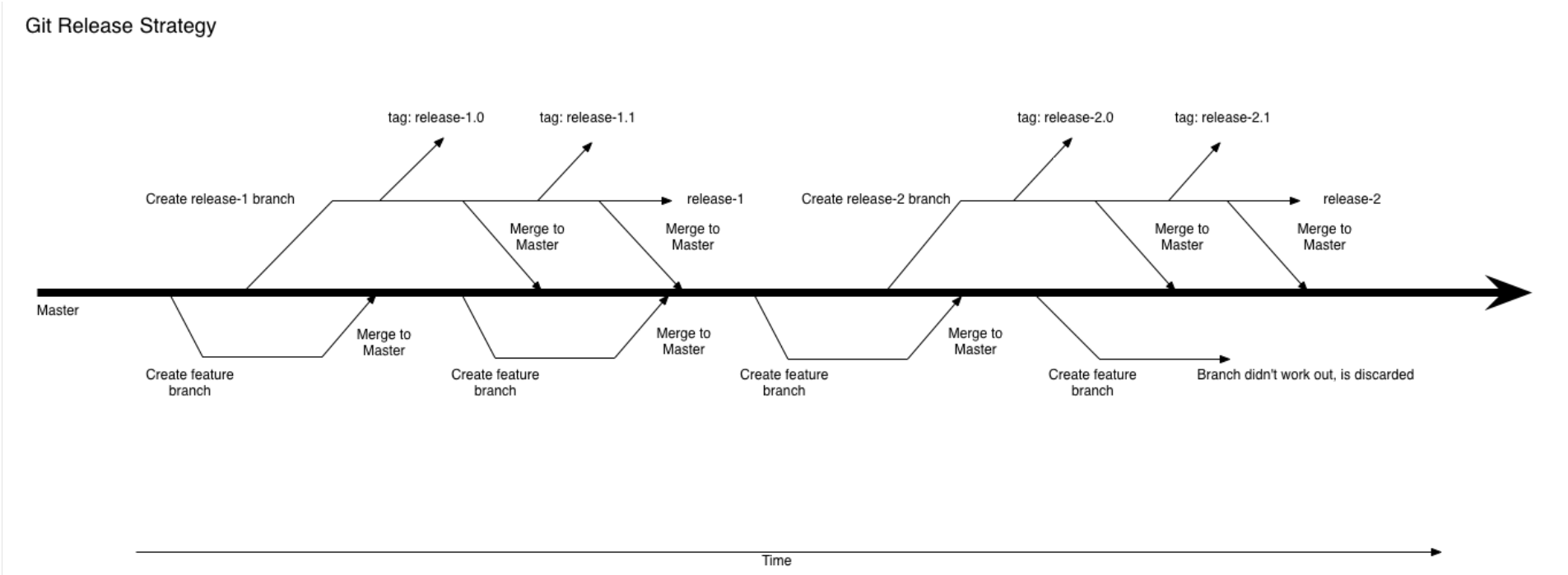
http://code.tutsplus.com/tutorials/solid-part-4-the-dependency-inversion-principle--net-36872

Tools/Processes

Utilize Version Control

- Git/Subversion
- Usage Strategy

Tools/Processes



Source: Peter Hanson & Associates – Share with the World Licensing :)

Tools/Processes

Centralize Knowledge/Ticketing

- Tracks information
- Common point of exchange
- JIRA/Redmine/Trac/Basecamp (maybe)

Tools/Processes

- Continuous Integration
- Automation
  - Testing
  - Builds

Summary

- There is a lot here
- Use what you'd like
- Use at the time of need based on what will help you the most
- Keep learning: read/apply, read/apply, …

Recommended Reading

Books

- **The Pragmatic Programmer: From Journeyman to Master**, Andrew Hunt & David Thomas, ISBN-13: 978-0201616224
- **Practices of an Agile Developer: Working in the Real World**, Venkat Subramaniam & Andrew Hunt, ISBN-13: 978-0974514086
- **Clean Code: A Handbook of Agile Software Craftsmanship**, Robert C Martin, ISBN-13: 978-0132350884
- **The Clean Coder: A Code of Conduct for Professional Programmers**, Robert C Martin, ISBN-13: 978-0137081073

Recommended Reading

Sites

- http://www.phptherightway.com/
- Google: “Scott Ambler and agile”
- Google: “how to communicate effectively at work”
- There are some good blogs, and I think they are helpful. To optimize your learning, I’d recommend the books and applying though if possible. The blogs may feel good (like you’re doing something) and are useful reminders, but they are not as effective as putting your time in with the books and at the keyboard.

Take Action

- Pick one book. Read it, apply it.
- Build something that you care about (create a business, help a non-profit, make a family site, etc.).

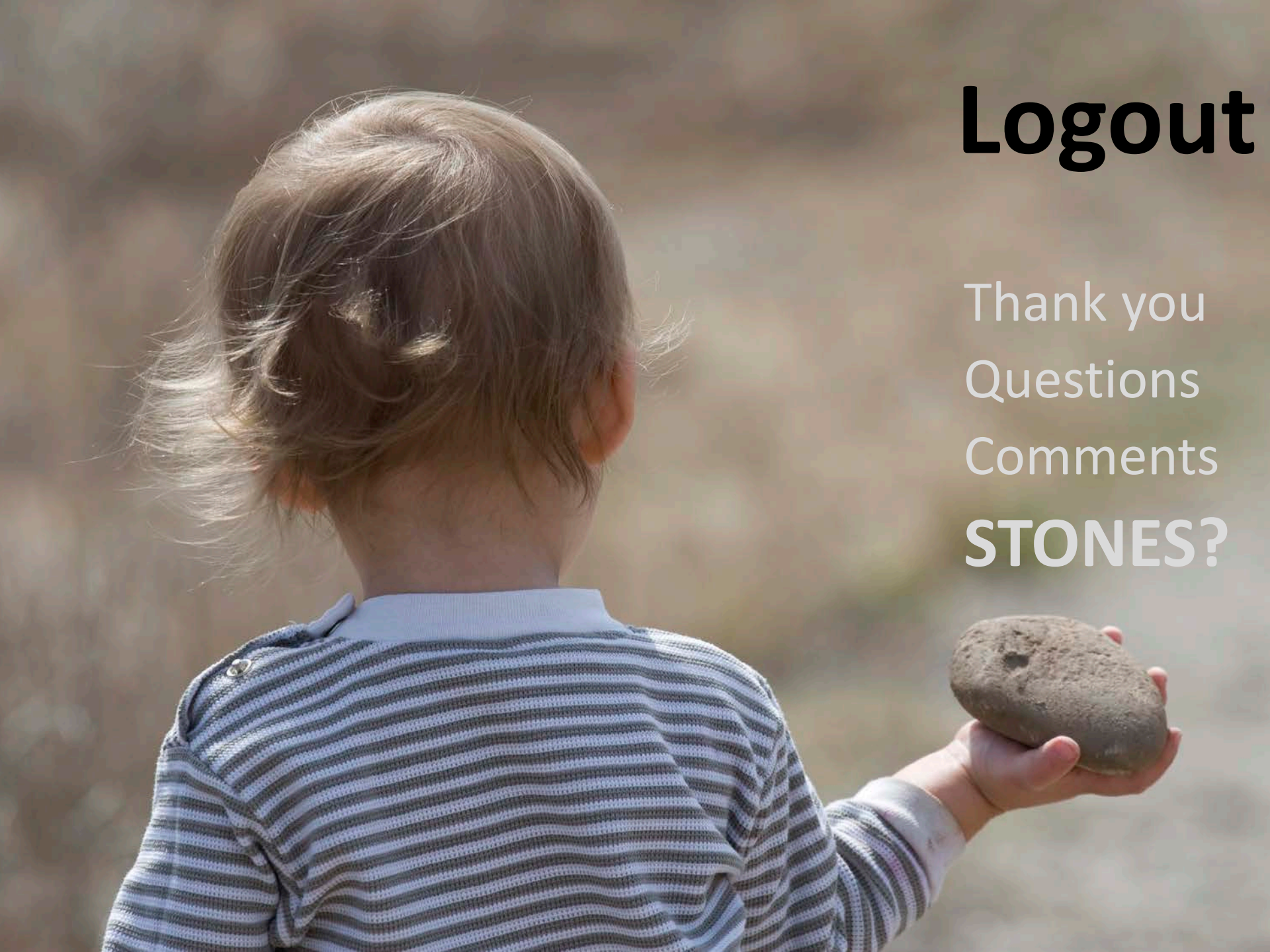
Learn with a purpose!

- Assess your current projects. Are there specific pain points? What practices here could help those? Apply the concept.

My Contact Information

- Skype (preferred): uar_pete.hanson
- Phone: 906-281-1178
- Email: Peter@upandrunning.com

I will not write any code for you because that could negatively affect your learning. If you want a sounding board though, I'd like to help and let's have a Skype call. Open door.


Logout
Thank you
Questions
Comments
STONES?