Ending Legacy Code in Our Lifetime

Stuart Halloway
CEO
Relevance, Inc.
What is Legacy Code?

http://www.oyez.org/justices/potter_stewart/
A New Characterization

- “Legacy” is the degree to which code
  - fails to capture intent
  - fails to communicate intent
  - captures irrelevant detail (ceremony)

- Other characterizations of legacy are implications or symptoms of the above
The Last Big Thing

https://duke.dev.java.net/
The Current Big Thing

http://www.rubyonrails.org/
The Next Big Thing
The IT Buyer’s View of “Big Things”

http://en.wikipedia.org/wiki/Sisyphus
Our Goal: Build so that the next big thing is not a “do over”
A More Modest Goal:
Get a little bit of reuse and maintainability before the next “do over”
Code Today (Death Star)
The Future (Golden Egg)

Essence
Ceremony: Enemy of Agility

http://pragdave.pragprog.com/pragdave/2006/06/glue_that_doesn.html
Ceremony: Code’s Worst Enemy

Photo courtesy John Lam (http://iunknown.com)

http://www.catchysoft.com/fbricks.html
Ceremony: Enemy of Security

http://www.schneier.com/blog/
Ceremony in the Java Language: Getting an Object Reference

```java
Document d = new Document();


@Autowired
public void setDocument(Document document) {
    this.document = document;
}
Ceremony in the Ruby Language: Asymmetry of Ivar and Method

http://www.rubyonrails.org/deprecation

@params, @session, @flash, @request, @cookies, @headers, @response

Use the params, session, flash, etc methods instead of working with the instance variables directly.
Ceremony in Libraries: Overweight XML Configuration

It’s prettier at this size, I assure you...
Ceremony in Application Code: Poorly Composed Methods

```ruby
def list
    self.crud_context = :list
    @options = {}
    unless exporting_a_full_download
        @options.smart_merge!(pagination_options)
    end
    @options.smart_merge!(order_options)
    @options.smart_merge!(filter_options)
    merge_count_or_find_options(@options)

    if pagination && !exporting_a_full_download
        if @options[:non_ar_column]
            col = @options[:non_ar_column]
            dir = @options[:dir]
            @options.delete :non_ar_column
            @options.delete :dir
            model_pages, models = paginate model_name.downcase.pluralize, @options
            if model_ui.has_sortable_column?(col)
                model_ui.sort_models(models, col)
            else
                logger.warn("Possible intrusion attempt: Invalid sort column #{col}"
            end
        else
            models.reverse! if dir == 'DESC'
        end
    else
        model_pages = []
        models = model.find(:all, @options)
    end

    From http://streamlinedframework.org/
```
Current Practices

- Rails
- Ruby
- Continuous Integration
- Real Unit Tests
- 100% Code Coverage
- Relentless Review and Audit
Where are We?

http://en.wikipedia.org/wiki/Home_plate#First_base
The Blub Paradox

http://www.paulgraham.com/avg.html
Generalizing Blub

- It isn’t just about language selection
- Abstractions are everywhere
- Wrong abstractions create ceremony
Wherefore the stable layer?

Domain Layer
DSLs

Dynamic Layer
Ruby, Groovy

Stable Layer
Java (Scala?)

http://ola-bini.blogspot.com/2008/01/language-explorations.html
A Warning

Questions?

Stuart Halloway
CEO
Relevance, Inc.