

*Legacy*

# Coderetreat

**Matteo Baglini**  
**@matteobaglini**  
**#lcrb14**



# Become a Better Developer

Ultimately, *greatness* comes from *practicing*; applying the *theory over and over* again, using *feedback* to get *better every time*.

In software we do our ***practicing on the job***, and that's why we make ***mistakes on the job***.

We need to find ways of ***splitting*** the ***practice*** from the ***profession***.

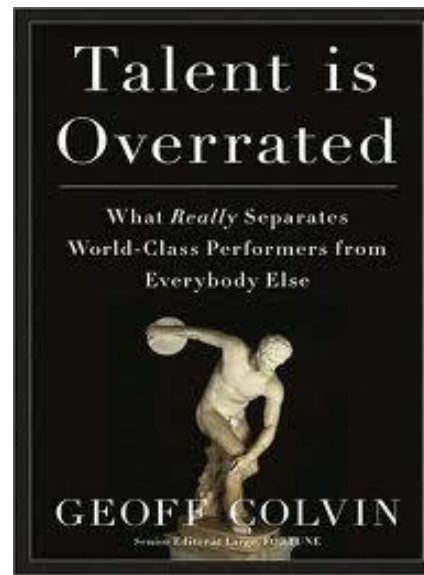
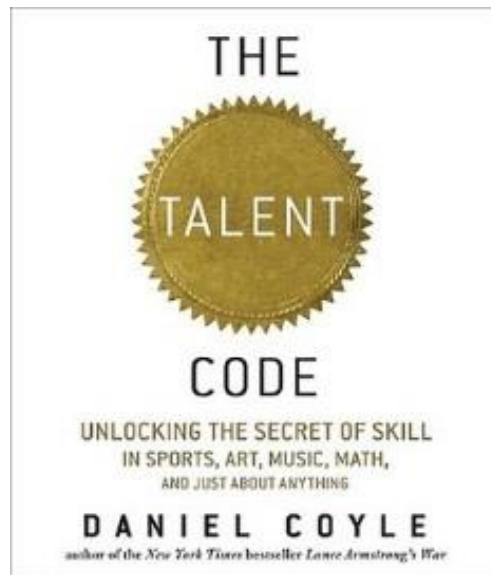
***We need practice sessions.***



Dave Thomas,  
Pragmatic Programmers

# Deliberate Practice

Contrary to what you might believe, merely *doing your job* every day *doesn't qualify as real practice*.



The secret is life-long period of deliberate effort to improve performance in a specific domain. *The secret is what researcher calls Deliberate Practice.*

# Corey Haines



# **J. B. Rainsberger**



# Structure of the Day

10:00 - 11:00 Session #1

11:00 - 12:00 Session #2

12:00 - 13:00 Session #3

14:00 - 15:00 Session #4

15:00 - 16:00 Session #5

16:00 - 17:00 Session #6



13:00 - 14:00  
Lunch



17:00 - 18:00  
Retrospective

# Structure of a Session

45' Coding

10' Retrospective

5' Break

# RULES!

---

1. You SHALL!

2. You WILL!

3. You MUST!





# Experiment

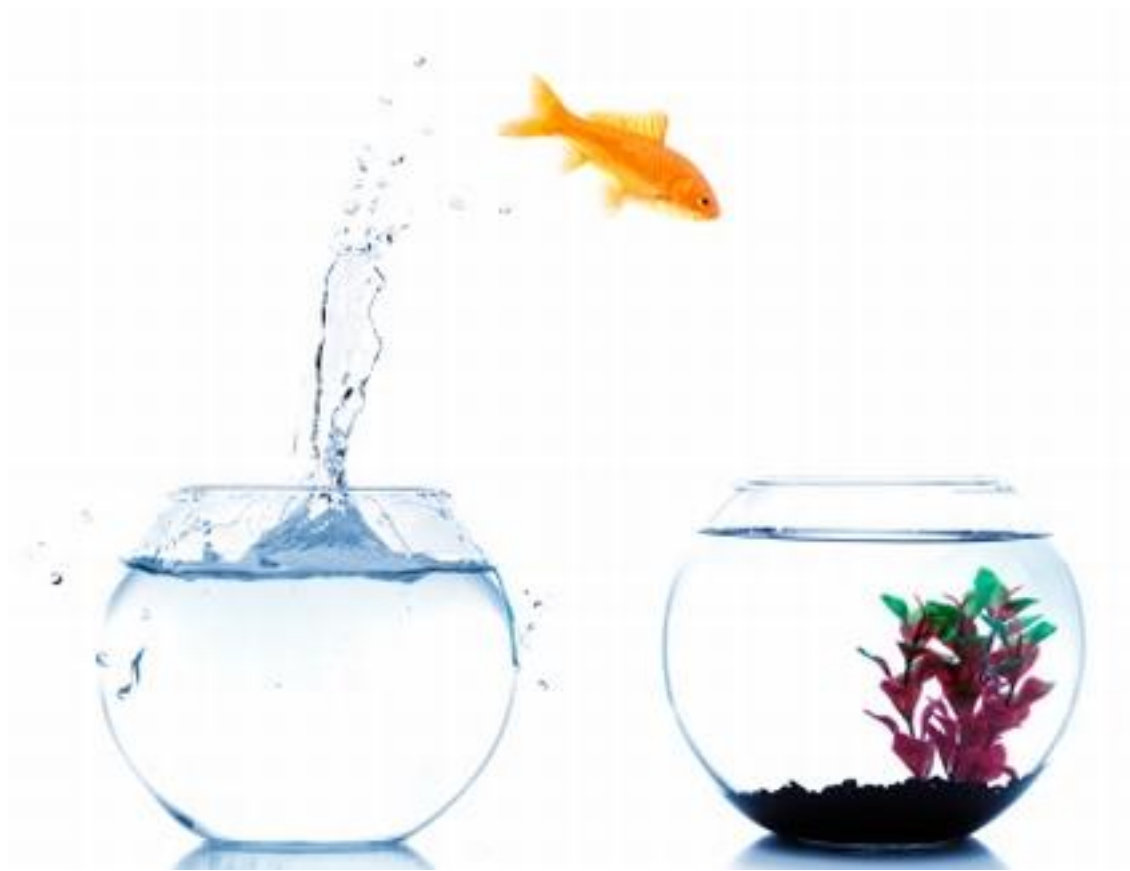




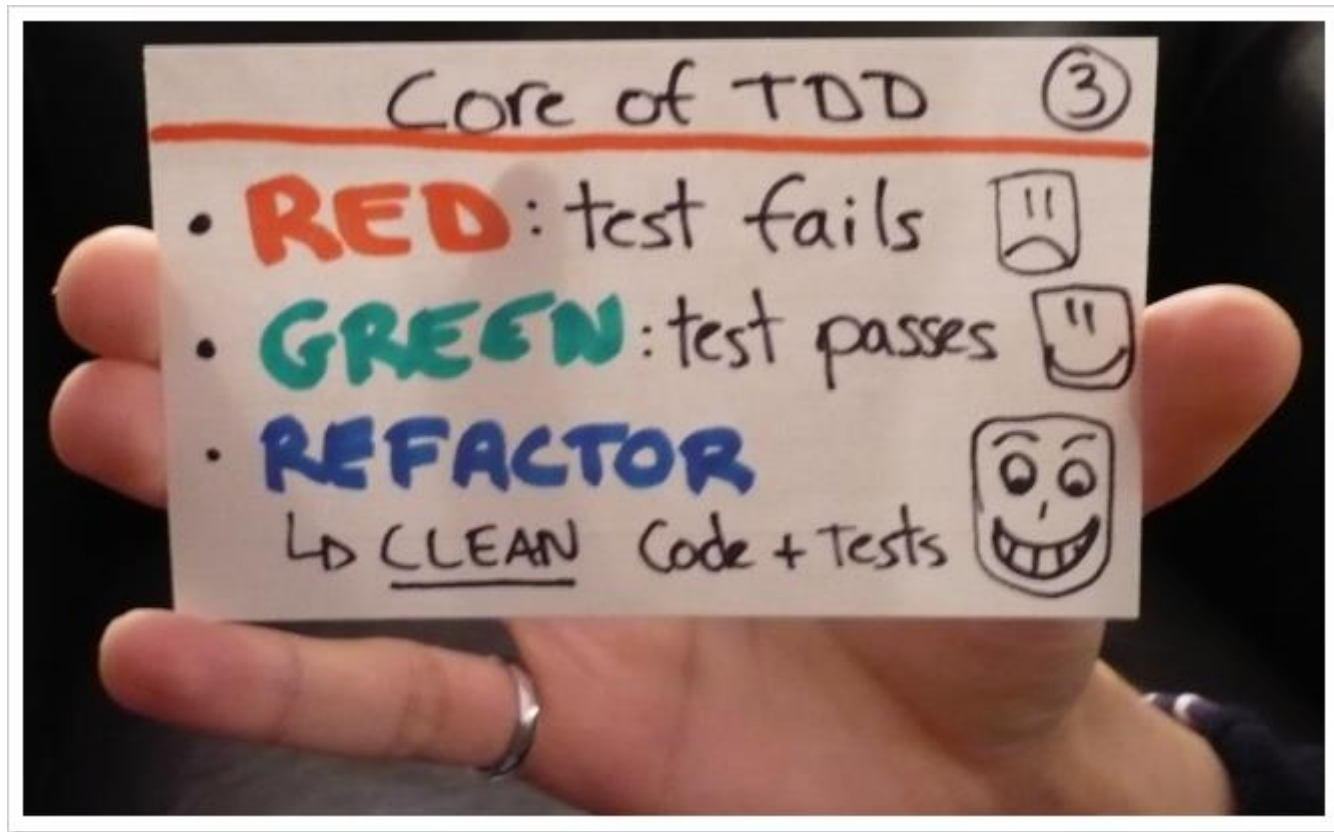
# ***Pair Programming***



# ***Change Pair***



# Test-Driven Development



# **Delete Your Code**





Focus  
ON  
WHAT  
matters

A handwritten-style graphic featuring the text "Focus ON WHAT matters". The word "Focus" is in a large, elegant script font, with several short, straight lines radiating upwards from its top, resembling a sunburst. Below "Focus" is a dark, horizontal banner with the word "ON" in a small, white, sans-serif font. Underneath the banner, the word "WHAT" is written in a large, bold, serif font. Finally, the word "matters" is written in a cursive script font, positioned below "WHAT". The entire graphic is rendered in black on a white background.



# **Put Your Code Under Test**





# Find Bad Code Smell



# ***Find Bad Code Smell***

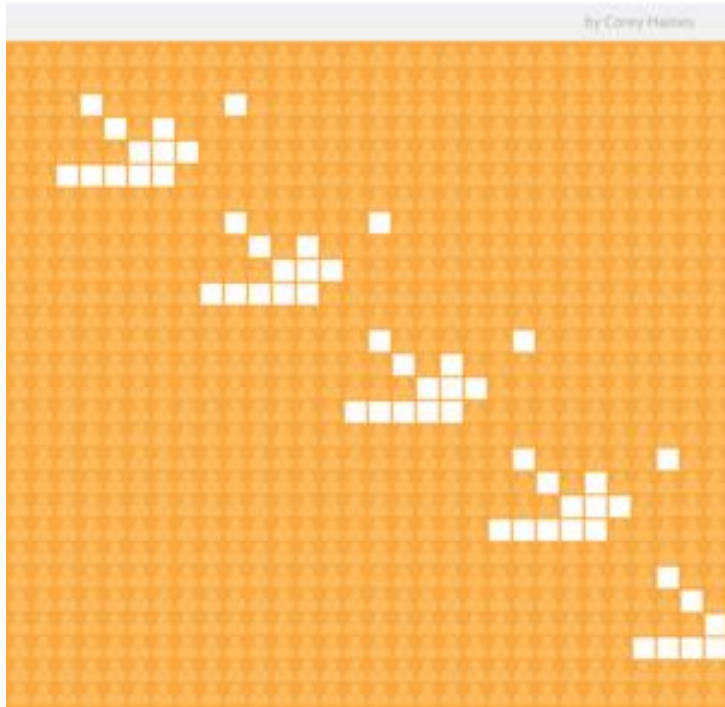
- Duplicated Code
- Long Method
- Large Class
- Long Parameter List
- Divergent Change
- Shotgun Surgery
- Feature Envy
- Data Clumps
- Lazy Class
- Primitive Obsession
- Switch Statements
- Temporary Field
- Message Chains
- Middle Man
- Inappropriate Intimacy
- Data Class
- Comments

# Simple Design

## UNDERSTANDING THE 4 RULES OF SIMPLE DESIGN

*And other lessons from watching 1000's of pairs work on Conway's Game of Life*

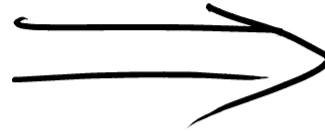
by Corey Haines



- Passes all the tests
- Contains no duplication
- Express developer intent
- Contains as little code as possible

# Simple Design

remove  
duplication



structure  
emerges

improve  
names



redistribute  
responsibilities

# Simple Design

remove  
duplication



improve  
names



structure  
emerges



new structures need names



~~redistribute  
responsibilities~~  
abstractions  
emerge!

# Refactor

<pre>import logging import string from random import Random from django import http from django.core.mail import send_mail from django.core.urlresolvers import reverse from django.http import HttpResponseRedirect from django.shortcuts import get_object_or_404 from django.contrib import auth from django.contrib.auth.decorators import login_required from django.contrib import messages from django.template import Context, loader  from l10n import ugettext as _  import jingo  import amo from bandwagon.models import Collection from manage import settings  from .models import UserProfile from .signals import logged_out from .users import forms from users.utils import EmailResetCode  log = logging.getLogger('z.users')  def confirm(request, user_id, token):     user = get_object_or_404(UserProfile, id=user_id)      if not user.confirmationcode:         return http.HttpResponseRedirect(reverse('users.login'))      if user.confirmationcode != token:         log.info("Account confirmation failed for user (%s)", user)         return http.HttpResponse(status=400)      user.confirmationcode = ''     user.save()     messages.success(request, _('Successfully verified'))     log.info("Account confirmed for user (%s)", user)     return http.HttpResponseRedirect(reverse('users.login'))  def confirm_resend(request, user_id):     user = get_object_or_404(UserProfile, id=user_id)      if not user.confirmationcode:         return http.HttpResponseRedirect(reverse('users.login'))      # Potential for flood here if someone requests a confirmationcode and then     # re-requests confirmations. We may need to track requests in the future.     log.info("Account confirm re-requested for user (%s)", user)     user_email_confirmation_code()      msg = _((("An email has been sent to your address (%s) to confirm "         "your account. Before you can log in, you have to activate "         "your account by clicking on the link provided in this "         "email.")), format(user_email))     messages.info(request, msg)      return http.HttpResponseRedirect(reverse('users.login'))  @login_required def delete(request):     amouser = request.user.get_profile()     if request.method == 'POST':         form = forms.UserDeleteForm(request.POST, request=request)         if form.is_valid():             messages.success(request, _('Profile Deleted'))             amouser.delete()             logout(request)             form = None         else:             form = forms.UserDeleteForm()      return jingo.render(request, 'users/delete.html',         {'form': form, 'amouser': amouser})  @login_required def edit(request):     amouser = request.user.get_profile()     if request.method == 'POST':</pre>	<pre>self.nickname = None self.homepage = '' self.deleted = True self.picture_type = '' self.save()  def save(self, force_insert=False, force_update=False, using=None):     # we have to fix stupid things that we defined poorly in reworka     if self.resetcode_expires is None:         self.resetcode_expires = datetime.now()      super(UserProfile, self).save(force_insert, force_update, using)  def check_password(self, raw_password):     if 'S' not in self.password:         valid = (get_hexdigest('md5', '', raw_password) == self.password)         if valid:             # Upgrade an old password.             self.set_password(raw_password)             self.save()             return valid      algo, salt, hsh = self.password.split('\$')     return hsh == get_hexdigest(algo, salt, raw_password)  def set_password(self, raw_password, algorithm='sha512'):     self.password = create_password(algorithm, raw_password)  def email_confirmation_code(self):     log.debug("Sending account confirmation code for user (%s)", self)      url = "%s%s" % (settings.SITE_URL,         reverse('users/confirm',             args=(self.id, self.confirmationcode)))     domain = settings.DOMAIN     t = loader.get_template('users/email/confirm.txt')     c = {'domain': domain, 'url': url, }     send_mail(_('Please confirm your email address'),         t.render(Context(c)), None, [self.email])  def create_django_user(self):     """Make a django.contrib.auth.User for this UserProfile."""     # Reusing the id will make our life easier, because we can use the     # OneToOneField as pk for Profile linked back to the authuser     # in the future.     self.user = DjangoUser(id=self.pk)     self.user.first_name = self.first_name     self.user.last_name = self.last_name     self.user.username = self.email     self.user.email = self.email     self.user.password = self.password     self.user.date_joined = self.created      if self.group_set.filter(rules='*').count():         self.user.is_superuser = self.user.is_staff = True      self.user.save()     self.save()     return self.user  apps/users/models.py 171.13 Bot</pre>	<pre>+ 35 lines: def _get_reset_url(self):     # URLs for a single user.     class TestPasswordResetForm(UserFormBase):         def test_request_fail(self):             r = self.client.post('/en-US/firefox/users/pwreset',                 {'email': 'someemail@someomain.com'})              eq(len(mail.outbox), 0)             self.assertFormError(r, 'form', 'email',                 ("That e-mail address doesn't have an "                  "associated user account. Are you sure "                  "you've registered?"))          def test_request_success(self):             self.client.post('/en-US/firefox/users/pwreset',                 {'email': self.user.email})              eq(len(mail.outbox), 1)             assert mail.outbox[0].subject.find('Password reset') == 0             assert mail.outbox[0].body.find('pwreset/%s' % self.uid36) &gt; 0      class TestUserDeleteForm(UserFormBase):         + 22 lines: def test_bad_password(self):  apps/users/tests/test_forms.py 73.1 122</pre>	<pre>urlpatterns = patterns('',     # URLs for a single user.     ('^user/(?P&lt;user_id&gt;)/\$', include(detail_patterns)),      url('^(users/delete\$)', views.delete, name='users.delete'),     url('^(users/edit\$)', views.edit, name='users.edit'),     url('^(users/login\$)', views.login, name='users.login'),     url('^(users/logout\$)', views.logout, name='users.logout'),     url('^(users/register\$)', views.register, name='users.register'),      # Password reset stuff     url('^(users/pwreset/?\$)', auth_views.password_reset,         {'template_name': 'users/pwreset_request.html',          'email_template_name': 'users/email/pwreset.txt',          'password_reset_form': forms.PasswordResetForm,          'name': 'users.pwreset'},         name='users.pwreset'),     url('^(users/pwreset/\$)', auth_views.password_reset_done,         {'template_name': 'users/pwreset_sent.html'}),     url('^(users/pwreset/(?P&lt;uid36&gt;[-\w]+)/?P(&lt;token&gt;[-\w]+)/\$)',         auth_views.password_reset_confirm,         {'template_name': 'users/pwreset_confirm.html',          'set_password_form': forms.SetPasswordForm,          },         name='users.pwreset_confirm'),     url('^(users/pwreset/complete\$)', auth_views.password_reset_complete,         {'template_name': 'users/pwreset_complete.html'}) )</pre>	<pre>apps/users/urls.py 30.1 Bot self.client = Client() self.user = User.objects.get(id='4043307') self.user_profile = self.user.get_profile()  class TestEdit(UserFormBase):     def test_email_change_mail_sent(self):         self.client.login(username='jbalogh@mozilla.com', password='foo')          data = {'nickname': 'jbalogh',             'email': 'jbalogh.changed@mozilla.com',             'first_name': 'DJ SurFTurf',             'last_name': 'jbalogh', }          r = self.client.post('/en-US/firefox/users/edit', data)         self.assertContains(r, "An email has been sent to %s" % data['email'])      # The email shouldn't change until they confirm, but the name should     u = User.objects.get(id='4043307').get_profile()     self.assertEqual(u.first_name, 'DJ SurFTurf')     self.assertEqual(u.email, 'jbalogh@mozilla.com')      eq(len(mail.outbox), 1)     assert mail.outbox[0].subject.find('Please confirm your email') == 0     assert mail.outbox[0].body.find('id.%s@malchange' % self.user.id) &gt; 0</pre>
<pre>msg = _((("An email has been sent to your address (%s) to confirm "     "your account. Before you can log in, you have to activate "     "your account by clicking on the link provided in this "     "email.")), format(user_email)) messages.info(request, msg)  return http.HttpResponseRedirect(reverse('users.login'))  @login_required def delete(request):     amouser = request.user.get_profile()     if request.method == 'POST':         form = forms.UserDeleteForm(request.POST, request=request)         if form.is_valid():             messages.success(request, _('Profile Deleted'))             amouser.delete()             logout(request)             form = None         else:             form = forms.UserDeleteForm()      return jingo.render(request, 'users/delete.html',         {'form': form, 'amouser': amouser})  @login_required def edit(request):     amouser = request.user.get_profile()     if request.method == 'POST':</pre>	<pre>apps/users/models.py 171.13 Bot @load l10n % @l10n: This is an email. Whitespace matters! % @blocktrans % Welcome to {{ domain }}.  Before you can use your new account you must activate it - this ensures the e-mail address you used is valid and belongs to you. To activate your account, click the link below or copy and paste the whole thing into your browser's location bar:  {{ url }}  Once you've successfully activated your account, you can throw away this e-mail  Thanks!  The {{ domain }} staff % @blocktrans %  Before you can use your new account you must activate it - this ensures the e-mail address you used is valid and belongs to you. To activate your account, click the link below or copy and paste the whole thing into your browser's location bar:  {{ url }}  Once you've successfully activated your account, you can throw away this e-mail  Thanks!  The {{ domain }} staff % @blocktrans %</pre>	<pre>%/contrib/auth/forms.py 180.9 60x % block title %{{ page_title(_('Password Reset')) }}% endblock %  % block content % <div class="primary role=" main"=""> <h2>{{ _('Password Reset') }}</h2> <div class="primary features"> <p>&lt;form method="post" action="" class="featured-inner object-leads user-input"&gt;     {{ csrf() }}     {{ fields }} <p class="instruction">         (X trans X) Forgotten your password? Enter your e-mail address below,         and we'll e-mail instructions for setting a new one.     </p> <p>&lt;/p&gt; &lt;/ul&gt; &lt;/div&gt; <div class="form-email"> {{ _('Email Address') }} &lt;abbr class="req" title="{{ _('required') }}"&gt;{{ _('abbr') }} {{ form.emailsafe }} {{ form.email.errors.safe }} &lt;/li&gt; &lt;/ul&gt; &lt;/div&gt; &lt;/div class="form-control"&gt;</div></p></p></div></div></pre>	<pre>apps/users/tests/test_views.py 17.17 14x # "Can be used while we're transitioning from separate first/last name # to a single field. Bug 546818" return "%s %s" % (self.first_name, self.last_name).strip()  @property def picture_url(self):     split_id = re.match(r"(^d+P)(d(0,3)?)(d(1,3)?)", str(self.id))     if not self.picture_type:         return settings.MEDIA_URL + '/img/zamboni/anon-user.png'     else:         return settings.USER_PIC_URL % (             split_id.group(2) or 0, split_id.group(1) or 0, self.id,             int(time.mktime(self.modifieddatetime)))  @cached_property def is_developer(self):     return bool(self.addons.filter(authors=self,         addonus__listed=True)[1:])  @property def display_name(self):     if not self.nickname:         return "%s %s" % (self.first_name, self.last_name)     else:         return self.nickname  @property def welcome_name(self):</pre>	
apps/users/views.py 50.0-1	Top apps/users/templates/users/email/confirm.txt 9.0-1	All apps/users/templates/users/pwreset_request.html 1.1	Top apps/users/models.py 86.9 51x	



Let's code  
together