Lightning Talks

Bernhard Merkle – Build Your Software with b3

Dominic Robinson – Message Passing Concurrency in C++

Jason McGuiness – Anti-Parallelism

James Coplien – Let's Discuss

Roy Osherove – Conversational Patterns

Kevlin Henney – What Motivates Programmers

Dirk Haun – Google Summer of Code

Koen Deforche – Wt::Dbo: a C++ ORM Library modern style

Bonus:

Tom Gilb – Love Quantification

Lightning Talks

Bernhard Merkle – Build Your Software with b3

Dominic Robinson – Message Passing Concurrency in C++

Jason McGuiness – Anti-Parallelism

James Coplien – Let's Discuss

Roy Osherove – Conversational Patterns

Kevlin Henney – What Motivates Programmers

Dirk Haun – Google Summer of Code

Koen Deforche – Wt::Dbo: a C++ ORM Library modern style

Build your software with b3

ACCU conf, Oxford, Lightning talk
Bernhard Merkle, SICK AG
April 15, 2010

yet another buildsystem ?

b3

a new buildsystem

b3

a new buildsystem

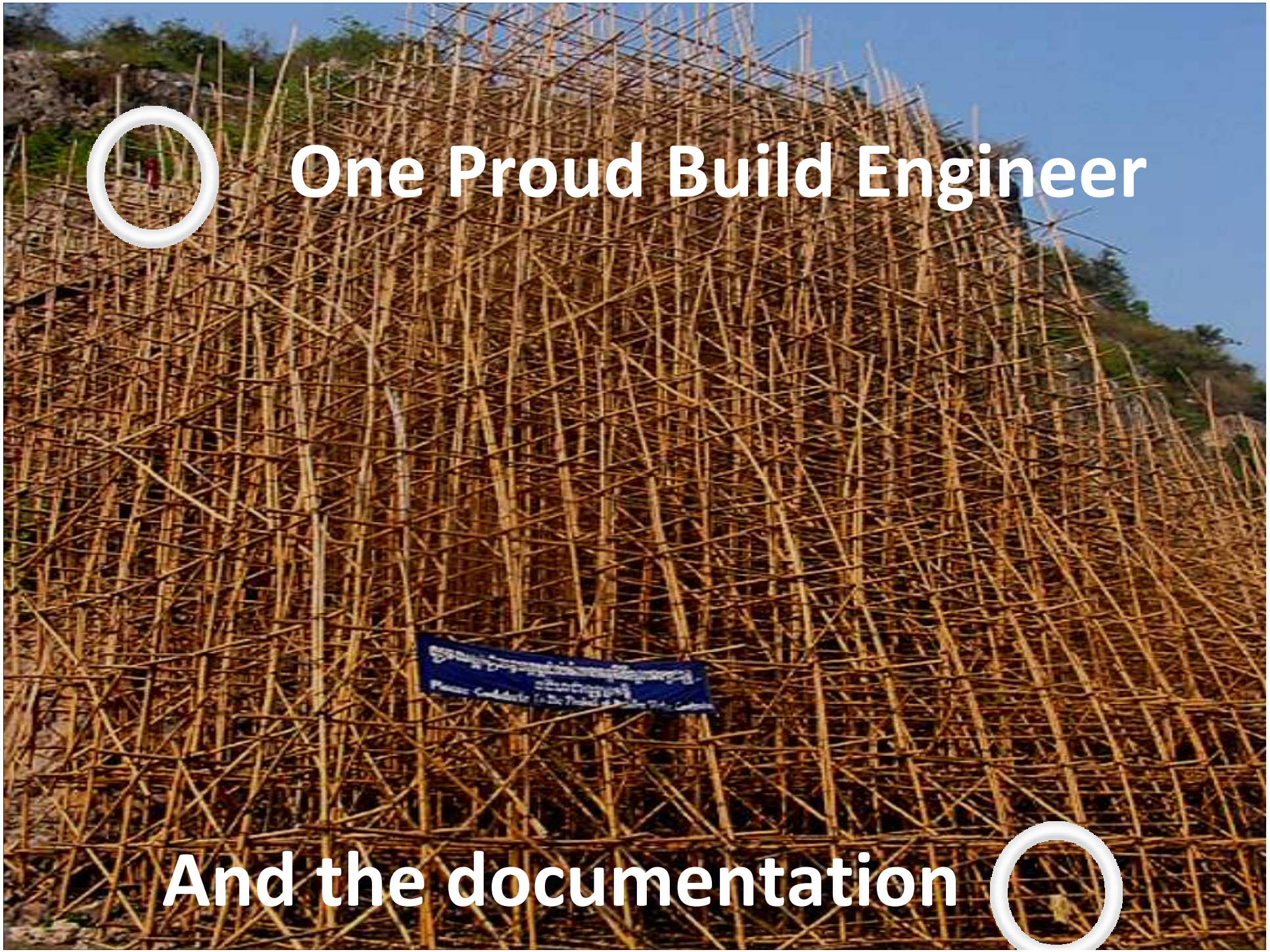
- make build processes **simpler**, repeatable, reproducible and **adaptable**
- **bridge discontinuities** between **building** and **provisioning** software
- **align overlapping** build technologies and **hide/abstract** them

b3

“be three”

“bee three”


One Proud Build Engineer


One Proud Build Engineer
And the documentation


[nc]make
[bla...]make
ANT-files...

The solution is really very simple

What we need is a formal description of

- the components
- their dependencies
- the processing
- the result

The solution is really very simple

What we need is a formal description of

- +

Modeling

The solution is really very simple

…and a concrete syntax

with syntax coloring
and code completion
and diff & merge
and quick fix
and …

The solution is really very simple

…and a concrete syntax

DSL aware Editor

Eclipse TMF, e.g. Xtext

The solution is really very simple

…and a concrete syntax

(f…) Eclipse

Emacs

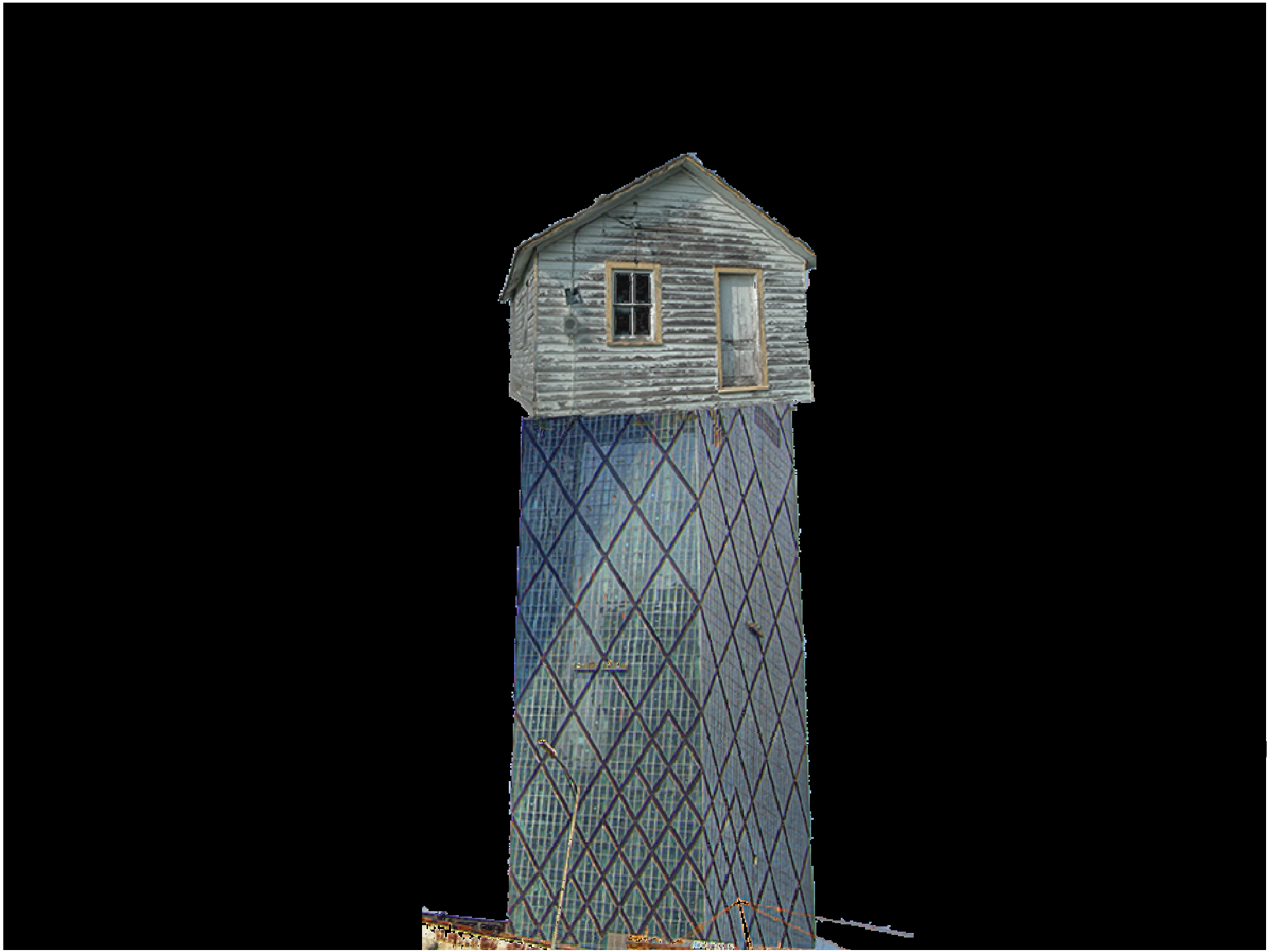
build file example

```
unit myProduct version 1.0.0 {
    requires {
        eclipse.feature/org.myorg.myproj/1.0;
    }

repositories {
      repository p2 { this.location = ”http://www.eclipse.org/updates-3.5”; }
      repository svn { this.location = ”svn+ssh://org.my.org/productx”; }
    }

builder runnableProduct {
      input { org.myorg.myproj#p2Repository; }
      output {  myProduct.zip; }

      PDE.runnableProduct(input, output);
    }
}
```

KISS – typical cases by default

```
import org.myorg.myproj.repositories;

unit myProduct version 1.0.0 implements RCPApp {
    builder repository {
        input { eclipse.feature/org.myproj/1.0#p2Repository; }
    }
}
```


光华桥
GUANGHUA Bridge




… here is the perfect spot for our
Social Media Integration…

flexibility

maximal simplicity

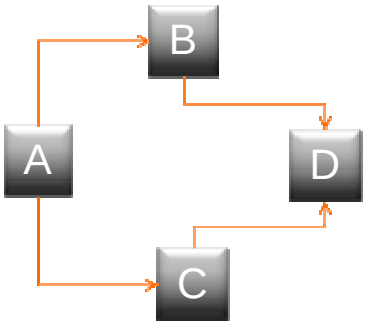
maximal simplicity

formal, simple, and flexible

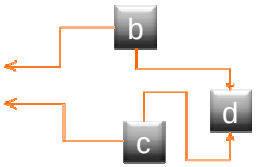
```
concern ReplaceUnwantedBundles {
   context unit requires osgi.bundle/org.myorg.unwanted {
     -  requires osgi.bundle/org.myorg.unwanted;
     + requires osgi.bundle/org.myorg.wanted/2.0;
   }

builder runnableProduct {
   input { with ReplaceUnwantedBundles : org.myorg.myproj#make; }
   }
}
```

b3 functional overview

build model



advice



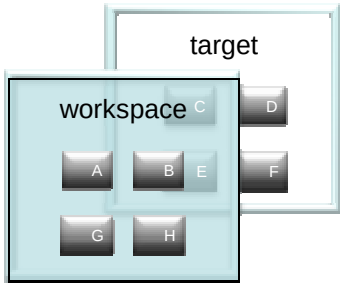
engine

actors

for(;;) …javac
ant …
gcc …

import
(materializers)

resolvers
meta data translators
connectors



b3 offers

a generic engine running

with build model
and expression model
and evaluation
and run builders
and junit testing
and p2 site aggregator

b3 supports

generic

debugging
packaging
dialogs/wizardry
Xtext candy
discovery

technology support

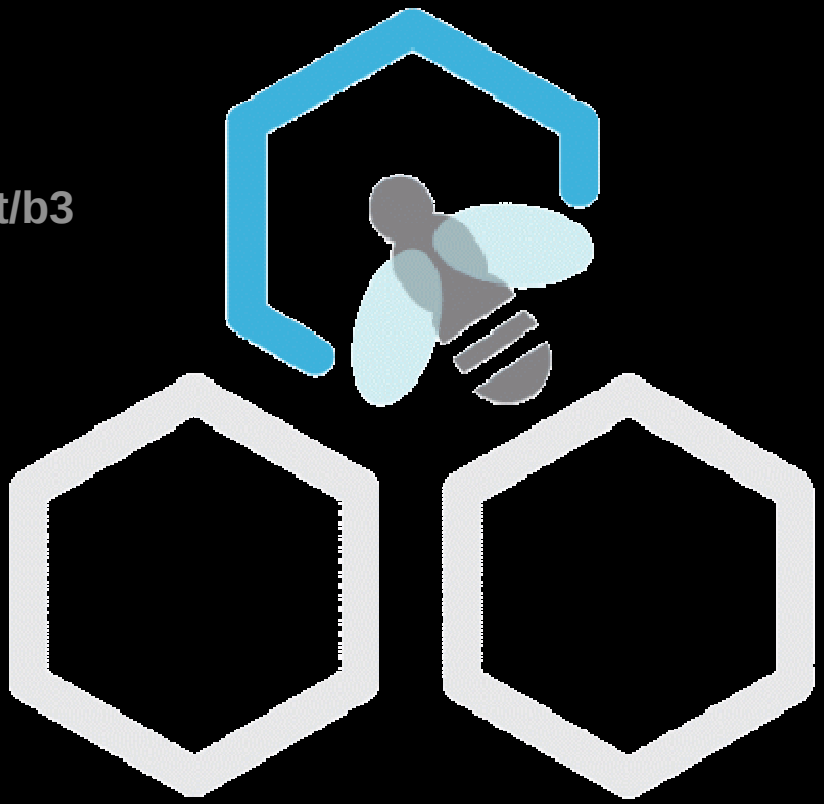
git, cvs, svn…
building X, Y, Z…
p2 publishing


Intrigued?
Get involved!

b3

Visit us at:
http://www.eclipse.org/modeling/emft/b3

Get the documentation



Legal…

b3

Visit us at:
http://www.eclipse.org/modeling/emft/b3

Get the documentation

Lightning Talks

Bernhard Merkle – Build Your Software with b3

Dominic Robinson – Message Passing Concurrency in C++

Jason McGuiness – Anti-Parallelism

James Coplien – Let's Discuss

Roy Osherove – Conversational Patterns

Kevlin Henney – What Motivates Programmers

Dirk Haun – Google Summer of Code

Koen Deforche – Wt::Dbo: a C++ ORM Library modern style

erlesque

message passing concurrency in C++

Dominic Robinson (dominic@snsys.com)

erlesque – the good stuff* from *erlang*

- No shared state, message passing
  - no locks
- Transparent distribution
  - “procs” distributed across many processes on many hosts
- Selective receive
  - essential to simplify complex state machines
- Pattern matching
  - succinct, type safe message matching
- Remote crash recovery
  - linked processes
  - exception propagation

**not all of it*

etuples

- messages are encoded as etuples
- etuples
  - can contain anything, including other etuples
    - can represent arbitrary DAGs
  - dynamically typed
    - not themselves templates
    - have template ctors
  - immutable (single assignment)
- extract values by pattern matching

message pattern matching

- We can match on types and specific values
- Match clauses establish local vars bound to matched values
- Bridges the dynamic / static type divide
- Very succinct code

```
receive()
{
        arity3( ( “accu” ), int year, ( true ) )
        {
            std::cout <<  “ACCU  “ << year << std::endl;
        }

        arity3( std::string name, int year, bool worthwhile )
        {
            std::cout << name << “ “ << year << “ “ << worthwhile << std::endl;
        }
}
```

```
void increment_server( etuple args )
{
    int sum = 0;
    for (;;)
    {
        receive()
       {
            arity4( ( atoms::call ), epid from, eref ref, int increment )
            {
                sum += increment;
                e::send( from, etuple( ref, sum ) );
            }

           any( etuple anything )
           {
                e::log( “unexpected“, anything );
           }
       }
    }
}
```

```
int call_increment_server( epid server, int increment )
{
    eref ref = e::monitor( server );
    e::send( server, etuple( atoms::call, self(), ref, increment ) );
    receive( )
    {
        arity2( ( ref ), int result )
        {
            return result;
        }

        arity5( ( atoms::down ), ( ref ), epid from, atoms::atom reason, etuple info )
        {
            throw e::exception( "server down", reason, info );
        }
    }
    after( 5000 )
    {
       throw e::exception( "timeout" );
    }
}
```

```
epid start_inc_server( etuple args )
{
   return e::spawn_link_proc( increment_server, args );
}



int main()
{
    epid server = start_inc_server();
    for ( int i = 0; i < 10; ++i )
    {
        std::cout << call_inc_server( server, 1 ) << std::endl;
   }
}
```

Lightning Talks

Bernhard Merkle – Build Your Software with b3

Dominic Robinson – Message Passing Concurrency in C++

Jason McGuiness – Anti-Parallelism

James Coplien – Let's Discuss

Roy Osherove – Conversational Patterns

Kevlin Henney – What Motivates Programmers

Dirk Haun – Google Summer of Code

Koen Deforche – Wt::Dbo: a C++ ORM Library modern style

Lightning Talks

Bernhard Merkle – Build Your Software with b3

Dominic Robinson – Message Passing Concurrency in C++

Jason McGuiness – Anti-Parallelism

James Coplien – Let's Discuss

Roy Osherove – Conversational Patterns

Kevlin Henney – What Motivates Programmers

Dirk Haun – Google Summer of Code

Koen Deforche – Wt::Dbo: a C++ ORM Library modern style

acc 2010 Lightning Talks

Bernhard Merkle – Build Your Software with b3

Dominic Robinson – Message Passing Concurrency in C++

Jason McGuiness – Anti-Parallelism

James Coplien – Let's Discuss

Roy Osherove – Conversational Patterns

Kevlin Henney – What Motivates Programmers

Dirk Haun – Google Summer of Code

Koen Deforche – Wt::Dbo: a C++ ORM Library modern style

Lightning Talks

Bernhard Merkle – Build Your Software with b3

Dominic Robinson – Message Passing Concurrency in C++

Jason McGuiness – Anti-Parallelism

James Coplien – Let's Discuss

Roy Osherove – Conversational Patterns

Kevlin Henney – What Motivates Programmers

Dirk Haun – Google Summer of Code

Koen Deforche – Wt::Dbo: a C++ ORM Library modern style

What Motivates Programmers

Kevlin Henney

kevlin@curbralan.com

@KevlinHenney

When Kenneth Feinberg [...] ordered bailed-out firms to cut exec pay and restructure (that is, downgrade) bonuses, many Americans greeted the news with unqualified approval. Joy, even. This new policy fits well with our human instincts for justice and revenge. Given how they've performed, bankers' pay just seems so patently unfair.

Bankers, of course, argue that fairness has nothing to do with it. And they're right. [...] Pay should be structured to recruit certain kinds of talent — yes, how you pay affects who applies — and it should help align individual goals with company goals.

More than anything, argue the bankers, pay should motivate: huge bonus cheques are to ensure superior performance from superior talent.

On this point, the bankers are wrong. We've recently gathered evidence suggesting that dangling exorbitant sums of money in front of workers doesn't improve performance. If anything, it negatively affects it.

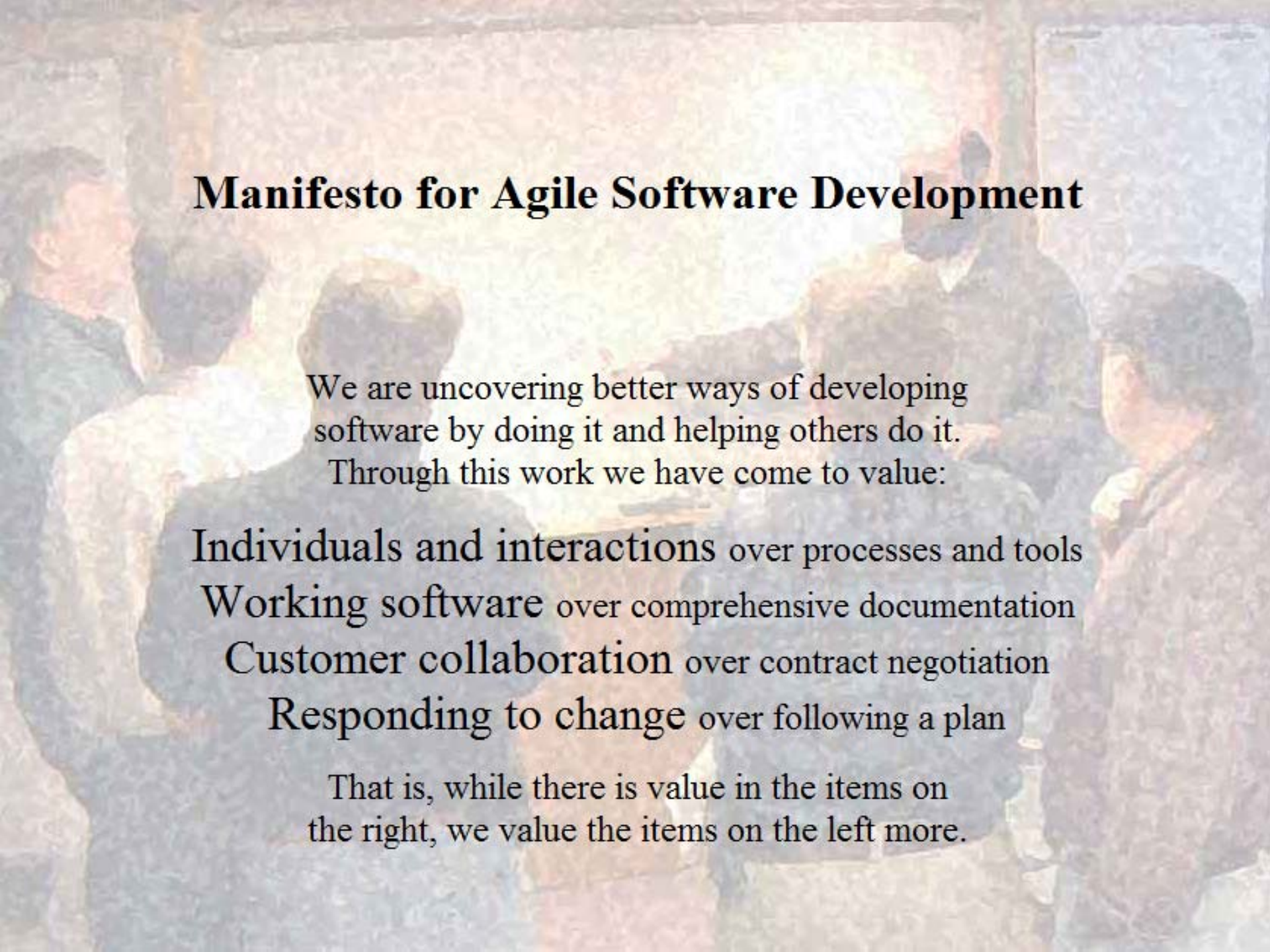
http://www.wired.co.uk/wired-magazine/archive/2010/03/start/dan-ariely-bonuses-boost-activity,-but-not-quality.aspx

If you want people to perform better, you reward them, right? [...]

But that's not happening here.

You've got an incentive designed to sharpen thinking and accelerate creativity and it does just the opposite. It dulls thinking and blocks creativity.

http://www.ted.com/talks/lang/eng/dan_pink_on_motivation.html

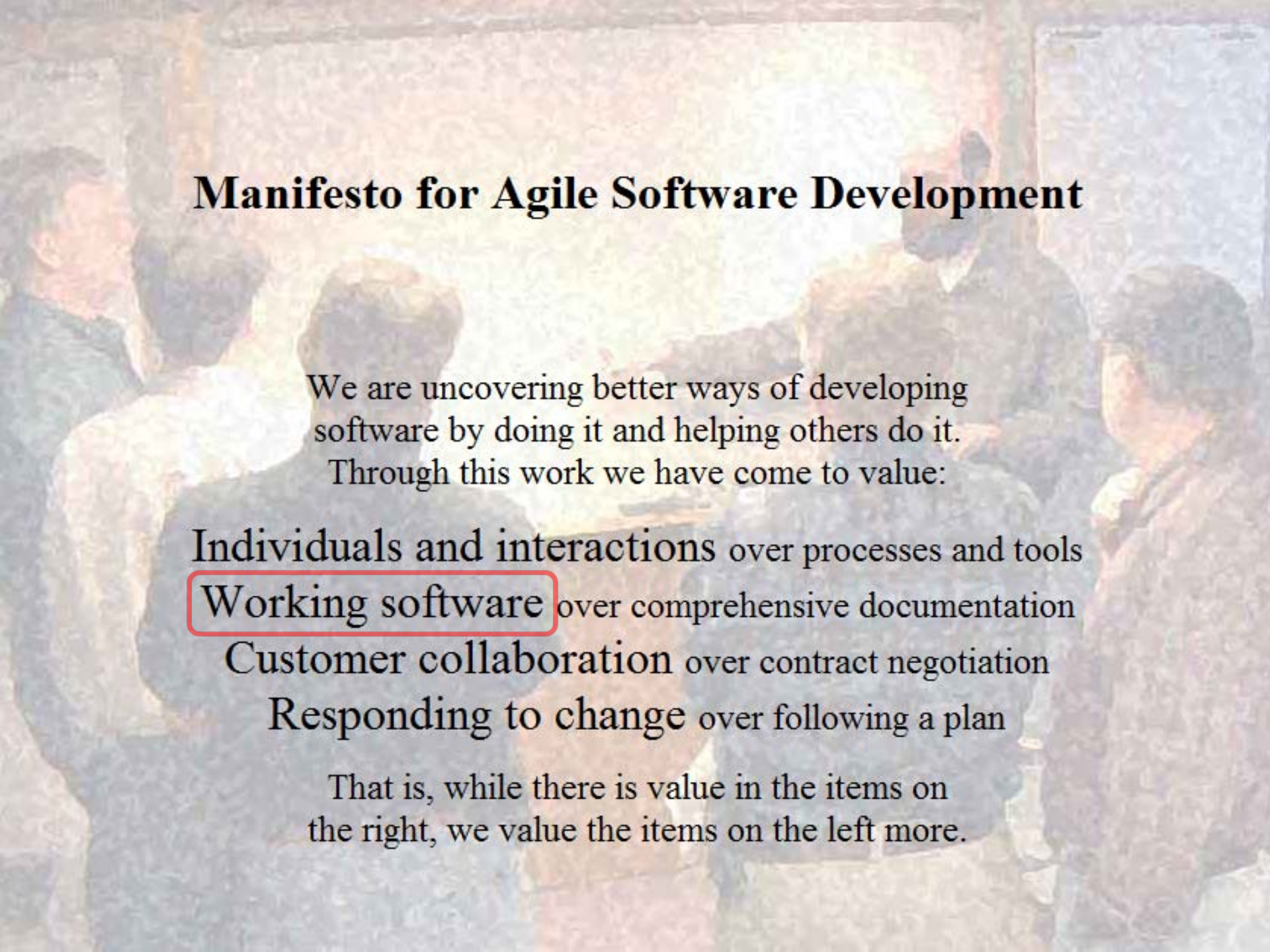


Manifesto for Agile Software Development

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.



Manifesto for Agile Software Development

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.

What is not true is the claim that stakeholder value is the only thing that is important. [...] Agile methods balance two things. One is the maximizing of value creation. The other thing is the maximizing of the chances of actually delivering something. ***These two goals are sometimes in conflict!*** *Projects that dogmatically focus on stakeholder value are working on the right things but still risk failing completely. The simple reason agile focuses on "working software" is that this is one of the primary ways of insuring that the system being worked on will actually work.*

Niklas Bjørnerstedt

http://www.leanway.no/?p=280

Ask leaders what they think makes employees enthusiastic about work, and they'll tell you in no uncertain terms. In a recent survey we invited more than 600 managers from dozens of companies to rank the impact on employee motivation and emotions of five workplace factors commonly considered significant: recognition, incentives, interpersonal support, support for making progress, and clear goals. "Recognition for good work (either public or private)" came out number one.

Unfortunately, those managers are wrong.

Having just completed a multiyear study tracking the day-to-day activities, emotions, and motivation levels of hundreds of knowledge workers in a wide variety of settings, we now know what the top motivator of performance is—and, amazingly, it's the factor those survey participants ranked dead last. It's *progress*.

http://hbr.org/2010/01/the-hbr-list-breakthrough-ideas-for-2010/ar/1

Lightning Talks

Bernhard Merkle – Build Your Software with b3

Dominic Robinson – Message Passing Concurrency in C++

Jason McGuiness – Anti-Parallelism

James Coplien – Let's Discuss

Roy Osherove – Conversational Patterns

Kevlin Henney – What Motivates Programmers

Dirk Haun – Google Summer of Code

Koen Deforche – Wt::Dbo: a C++ ORM Library modern style

Lightning Talks

Bernhard Merkle – Build Your Software with b3

Dominic Robinson – Message Passing Concurrency in C++

Jason McGuiness – Anti-Parallelism

James Coplien – Let's Discuss

Roy Osherove – Conversational Patterns

Kevlin Henney – What Motivates Programmers

Dirk Haun – Google Summer of Code

Koen Deforche – Wt::Dbo: a C++ ORM Library modern style

Wt::Dbo: a C++ ORM library

Koen Deforche

April 15, 2010

An Object Relational Mapper for C++ ?

Sole raison d'être: simplify development !

- Open source.
- No code generation.
- No XML mapping definitions.
- No macro hacks.
- Concise.

Started September '09.
First release December '09 (Wt 3.1.0).

`http://www.webtoolkit.eu/wt/doc/tutorial/dbo/tutorial.html`

Mapping fields

```
namespace dbo = Wt::Dbo;

class User {
public:
  enum Role { Visitor = 0, Admin = 1 };

  std::string name;
  Role        role;
  int         karma;

  template<class Action>
  void persist(Action& a)
  {
    dbo::field(a, name,  "name");
    dbo::field(a, role,  "role");
    dbo::field(a, karma, "karma");
  }
};
```

Mapping a class

```
// Create a backend
dbo::backend::Sqlite3 sqlite3("blog.db");

// Create a session
dbo::Session session;
session.setConnection(sqlite3);

// Map class(es)
session.mapClass<User>("user");

// Create the schema
session.createTables();
```

```
=> create table user(
     id integer primary key autoincrement,
     version integer not null,
     name text not null,
     role integer not null,
     karma integer not null
   )
```

Persisting a new object

```
dbo::Transaction transaction(session);

User *user = new User();
user->name = "Joe";
user->role = User::Visitor;
user->karma = 13;

dbo::ptr<User> userPtr = session.add(user);

transaction.commit();

std::cout << "User created with id: " << userPtr.id() << std::endl;
```

```
=> insert into user(version, name, role, karma)
   values(?, ?, ?, ?)
```

Updating an object

```
dbo::Transaction transaction(session);

dbo::ptr<User> user = ...;

if (user->role == User::Visitor)
  user.modify()->karma++;

transaction.commit();
```

```
=> update user
   set version = ?, name = ?, role = ?, karma = ?
   where id = ? and version = ?
```

ptr<*T*>, collection< ptr<*T*> >

- **ptr<T>**: a smart pointer class for objects
  - Memory management (reference counting)
  - Dirty tracking (write access with `modify()`)
  - Persistence methods:
    - `flush()`, `remove()`, `reread()`, `id()`
- **collection< ptr<T> >**: STL-like container for query results
  - Iterators implement *InputIterator* requirements

Mapping Many-to-One relations

```
class Post {
public:
  dbo::ptr<User> user;

  template<class Action>
  void persist(Action& a) {
    dbo::belongsTo(a, user, "user");
  }
};

class User {
public:
  dbo::collection< dbo::ptr<Post> > posts;

  template<class Action>
  void persist(Action& a) {
    dbo::hasMany(a, posts, dbo::ManyToOne, "user");
  }
};
```

Mapping Many-to-Many relations

```
class Post {
public:
  dbo::collection< dbo::ptr<Tag> > tags;

  template<class Action>
  void persist(Action& a) {
    dbo::hasMany(a, tags, dbo::ManyToMany, "post_tags");
  }
};

class Tag {
public:
  dbo::collection< dbo::ptr<Post> > posts;

  template<class Action>
  void persist(Action& a) {
    dbo::hasMany(a, posts, dbo::ManyToMany, "post_tags");
  }
};
```

Queries

```
dbo::Transaction transaction(session);

typedef boost::tuple< dbo::ptr<Post>, dbo::ptr<User> > PostUser;
typedef dbo::collection<PostUser> PostsUsers;

// Find all posts of users with karma < 5
PostsUsers posts = session_->query<PostUser>
  ("select p, u from post p join user u on p.user_id = u.id "
   "where u.karma < ?").bind(5);

foreach (PostUser postUser, posts) {
  dbo::ptr<Post> post;  dbo::ptr<User> user;
  boost::tie(post, user) = *i;

  std::cerr << "Post " << user->title << " posted by " << user->name
            << std::endl;
}

transaction.commit();
```

How it works

- The library defines several *actions*
  - loading, saving, deleting objects
  - propagating transaction outcome
  - creating, dropping schema
  - ...
- Database objects, their fields and relations are visited with `persist()`
- *Traits* classes for field types
- Prepared statements, lazy loading

Love Quantification

Quality quantification for sceptics

tom@gilb.com

www.gilb.com

Subset April 16 2010

The Word from the Lord!

Gilb's PRINCIPLE OF 'QUALITY QUANTIFICATION'

"All qualities *can* be expressed quantitatively",
***'qualitative'* does *not* mean unquantifiable.**
And.. Quantification is NOT the same as MEASUREMENT!

"In physical science the first essential step in the direction of learning any subject is to find principles of numerical reckoning and practicable methods for measuring some quality connected with it.

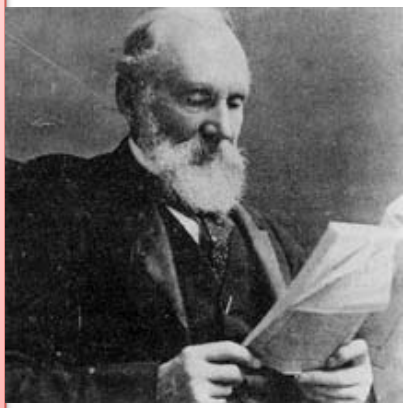
I often say that when you can **measure** what you are speaking about, **and express it in numbers**, you know something about it;

but when you cannot **measure** it, when you cannot **express it in numbers**, your knowledge is of a meagre and unsatisfactory kind;

it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of Science, whatever the matter may be."

Lord Kelvin, 1893

From http://zapatopi.net/kelvin/quotes.html



"All Qualities *can* be expressed Quantitatively"

Means

- It seems to be always possible to do so, ***technically*** – i.e. to ***define*** a 'scale of measure'
  - And
- Seems possible to always put useful numbers in terms of that scale, for a real world
  - In order to express present, past, and future states of that quality
  - In order to 'analyze' and to specify 'requirements ' levels

Does NOT imply

- You *always* should quantify
  - It should 'pay off' to do so
- Nor imply, "Context free numbers"
  - We define context completely and unambiguously, in Planguage
  - Where, when, if
  - Assumptions
  - Issues
  - Terms: defined as:…….
- And is NOT related to the measuring tool you may or may not apply!
  - You can usefully quantify without ever intending or really measuring!
  - Quantification is 'language', Measurement is 'Process'

But many people (you?) are quite sceptical of this idea of '**quality quantification'**

Belief

- Cannot be done
- Nobody ever did it
- Too difficult to do
- Too difficult to 'measure'
  - accurately
  - reliably

But

- Just because you cannot quantify a quality concept, does not mean it cannot be done!
- I did not say "*qualitative* concept" I said a **'quality'**
  - Which I define as "a dimension of **how well** a function is performed"
  - Aka 'ilities'
  - NOT How MUCH, (work capacity)
  - NOT 'HOW' (design)
- Try: Google-ing it
  - "Intuitiveness Metric (Gilb)"

Eindhoven Holland
Real Case Exercise: *Aspects of Love, or Love is a many splendored thing!*

The 'Challenge': **"You can't quantify 'Love, Tom!"**

- **Teachable Method:**
  - Make an inventory of love's many aspects
    - Duration: 6 minutes
    - Same method as Descartes, 'Analysis'
  - Quantify one requirements for love
    - Descartes Method: mastering each decomposed detail
  - The concept (Love) is the SET of aspects
    - Cartesian 'Synthesis'

See note for Sutra

Descartes On Small

- "We should bring the whole force of our minds to bear
  - upon the most minute and simple details
  - and to dwell upon them for a long time
  - so that we become accustomed to perceive the truth
  - clearly and distinctly."
- Rene Descartes,' Rules for the Direction of the Mind', 1628

Love Attributes:

Brainstormed By Dutch Male Engineers

(French *Women* might have a different 'model')

- *Kissed-ness*
- *Care*
- *Sharing*
- *Respect*
- *Comfort*
- *Friendship*
- *Sex*
- *Understanding*
- *Support*
- *Attention*
- *Passion*
- *Satisfaction*

Notice that the ‘Arts’ have long understood that ‘Love’ has multidimensional attributes!

- **Kissed-ness**
- **Care**
- **Sharing**
- **Respect**
- **Comfort**
- **Friendship**
- **Sex**
- **Understanding**
- **Trust**

Support

Attention

Passion

Satisfaction

…

…

Which aspect do you guess they chose to quantify first?

Part of their list

- Kissed-ness
- Care
- Sharing
- Respect
- Comfort
- Friendship
- Sex
- Understanding
- Trust

Which aspect do you guess they chose to quantify first?

Part of their list

- Kissed-ness
- Care
- Sharing
- Respect
- Comfort
- Friendship
- Sex
- Understanding
- Trust

NO! Not THAT one!

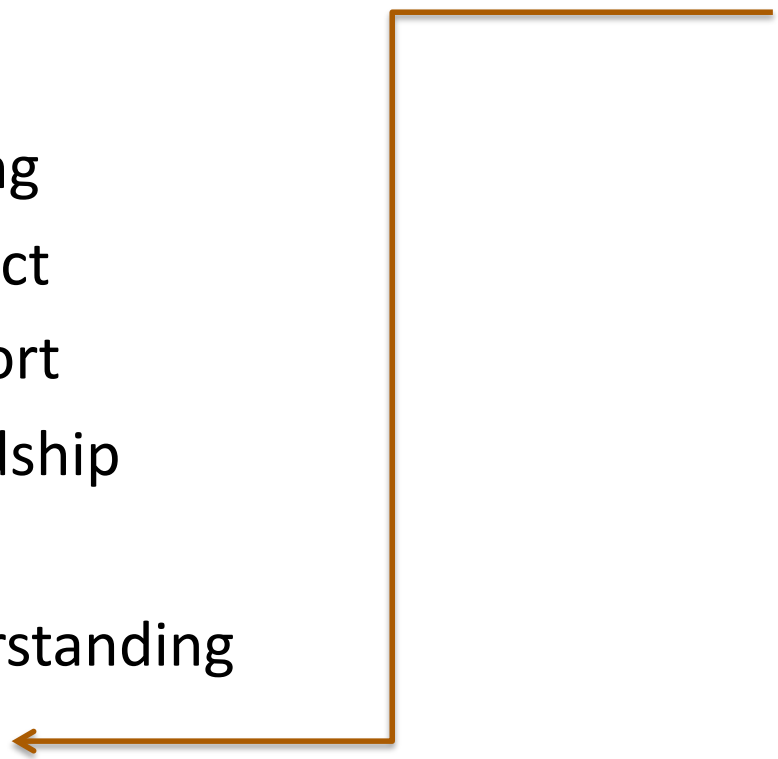
Which aspect do you guess they chose to quantify first?

Part of their list

- Kissed-ness
- Care
- Sharing
- Respect
- Comfort
- Friendship
- Sex
- Understanding
- Trust

NO! Not THAT one!

- This one



Trust [Caroline]

- **Love.Trust.Truthfulness**

  Ambition: No lies.

  Scale:

  Average **Black** lies/month from [defined sources].

  Meter:

  independent confidential log from sample of the defined sources.

  Past Lie Level:

  Past [My Old Mate, 2004] 42 <- Bart

  Goal

  [My Current Mate, Year = 2005] Past Lie Level/2

  Black: Defined: Non White Lies

- Other aspects of Trust:
- 1. 'Truthfulness'

  2. Broken Agreements

  3. Late Appointments

  4. Late delivery

  5. Gossiping to Others

The British are too shy to confront ideas like ‘love’ and ‘sex’ directly

- They use euphanisms
  - Like ‘Camararderie’

Camaraderie (Real Case UK)

Ambition: *to maintain an exceptionally high sense of good personal feelings and co-operation amongst all staff: family atmosphere, corporate patriotism. In spite of business change and pressures.*

Scale: **probability that individuals enjoy the working atmosphere so much that they would not move to another company for less than 50% pay rise**.

Meter: Apparently real offer via CD-S

Past [September 2001] 60+ % <- R & CD

Goal [Mid 2002] 10%, [End 2002] **<1%** <- R & CD

Rationale:

maintain staff number, and morale as core of business and business predictability for customers.

My ‘Christian’ Friend

- Lawrence Day. Seattle Washington
  - Divinity Doctor (hobby)
  - Lay Preacher
  - President <Christian Fellowship Association> (USA)
  - Web business processes, Boeing
- “Love (a central Christian value) is not quantifiable”
  - Not in Bible
  - Little guidance from God and Jesus
    - About Love Engineering

Silence for 6 weeks

- But then an email appeared from Lawrence
- “Humble apologies Tom
  - But, you were right……

Love: Biblical Dimensions
<- Lawrence Day, Boeing

The biblical citation (Book of First Corinthians) I included gives the quantification of the term "love" (agape in Greek). The 'quantification' for love would be as follows:

---------->

A person who loves acts the following way toward the person being loved:

1. **suffereth long**
2. **is kind**
3. **envieth not**
4. **vaunteth not itself, vaunteth...:**
   or, is not rash (Vaunt = extravagant self praise)
5. **is not puffed up**
6. **Doth not behave itself unseemly**
7. **seeketh not her own**
8. **is not easily provoked**
9. **thinketh no evil**
10. **Rejoiceth not in iniquity (=an unjust act)**
11. **rejoiceth in the truth**
12. **Beareth all things**
13. **believeth all things**
14. **hopeth all things**
15. **endureth all things**
16. **never faileth**

A Paper on ‘Love Quantified’

http://www.gilb.com/tiki-download_file.php?fileId=335

-download_file.php?

Love Quantified

By:

Lawrence E. Day

for

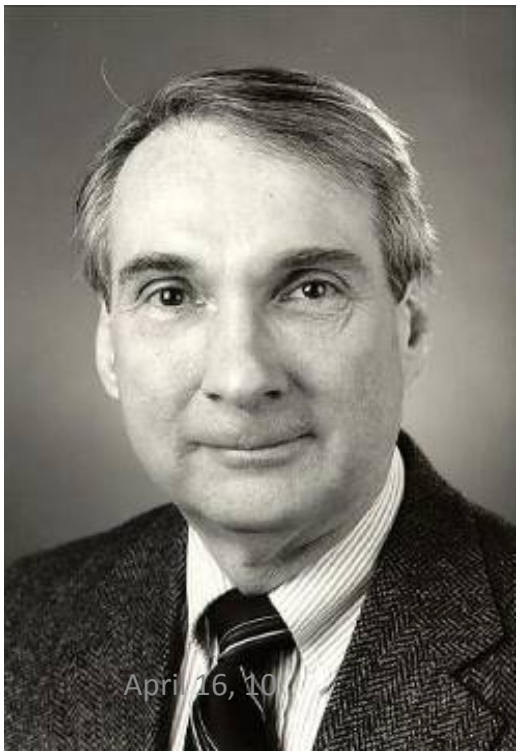
Dr. Larry Beebe

And

Dr. Raghu Korrapati

Table of Content

Mathematical Models of Love & Happiness



J. C. Sprott

Department of Physics

University of Wisconsin - Madison

Presented to the

Chaos and Complex Systems Seminar

Outline

- **Love model** - Inspired by Steve Strogatz (Cornell University)
- **Happiness model** - In collaboration with Keith Warren (Ohio State Univ)

Simple Linear Model

- $dR/dt = aR + bJ$
- $dJ/dt = cR + dJ$
- where
  - *R* is Romeo's love for Juliet
  - *J* is Juliet's love for Romeo
  - (or hate if negative)
  - *a, b, c, d* are constants that determine the "Romantic styles"

Limitations of Model

- It's difficult to quantify and measure love and hate.
- Love is not a scalar (different types).
- Parameters change in time and with the situation.
- Parameters may be different for love and hate.
- There are always other variables

Some “Romantic Styles”

$dR/dt = aR + bJ$

- $a=0$ (out of touch with own feelings)
- $b=0$ (oblivious to other’s feelings)
- $a>0$, $b>0$ (eager beaver)
- $a>0$, $b<0$ (narcissistic nerd)
- $a<0$, $b>0$ (cautious lover)
- $a<0$, $b<0$ (hermit)

Number of Pairings

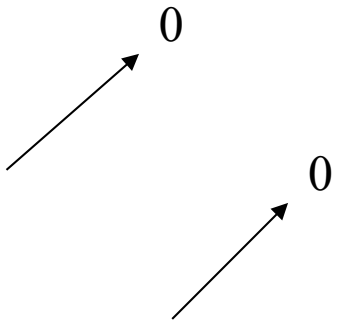
- 6 styles for Romeo X 6 styles for Juliet = 36 different pairings.
- Only 21 give unique dynamics (because of R/J symmetry)

but… It's actually worse than that:

- 4 parameters with 3 choices (-,0,+) for each gives $3^4 = 81$ combinations of which 45 are unique
- And there are subclasses depending on values and initial conditions.

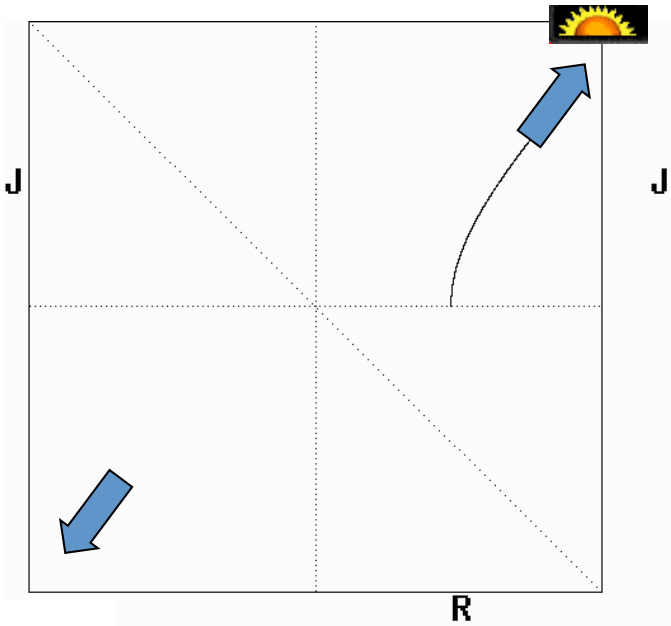
Both out of touch with their own feelings

- $dR/dt = aR + bJ$
- $dJ/dt = cR + dJ$
- Four subclasses:
  - $b > 0$, $c > 0$ (mutual love fest or war)
  - $b > 0$, $c < 0$ (never-ending cycle)
  - $b < 0$, $c > 0$ (never-ending cycle)
  - $b < 0$, $c < 0$ (unrequited love)



Out of touch with their own feelings (continued)

$b > 0,\ c > 0$

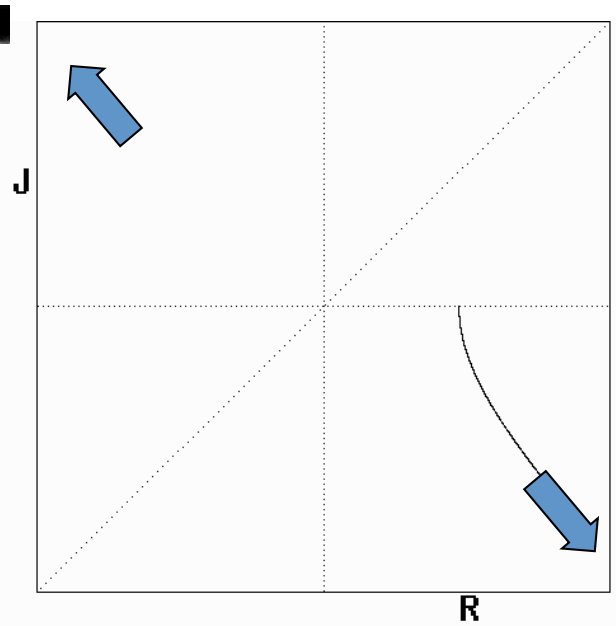


War

Two lovers

Love fest (or war)

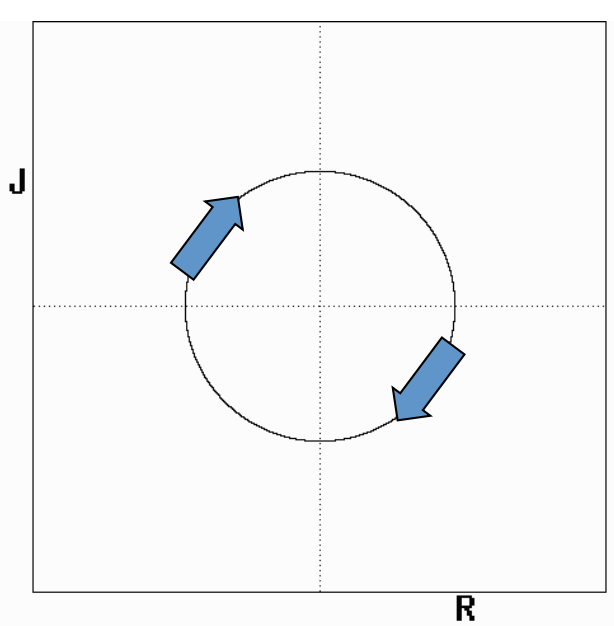
$b < 0,\ c < 0$



Two nerds

Unrequited love

$b > 0,\ c < 0$

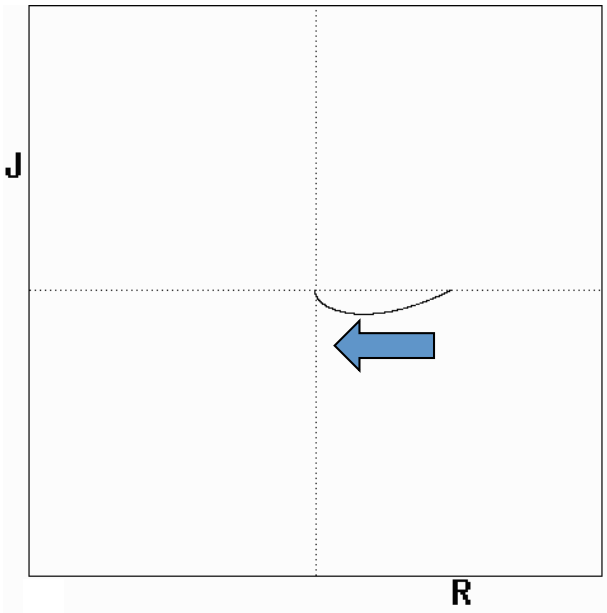


Nerd + lover

Never-ending cycle

With Self-Awareness and $bc < 0$ (nerd + lover)

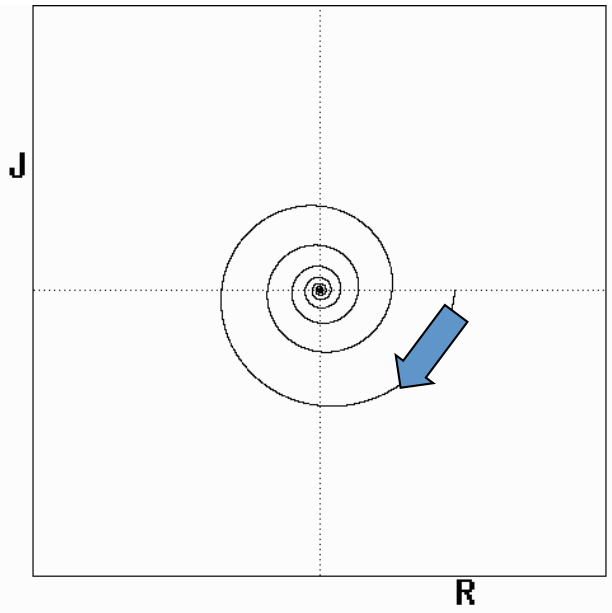
$a + d < -2|bc|^{1/2}$



Extremely cautious

Rapid apathy

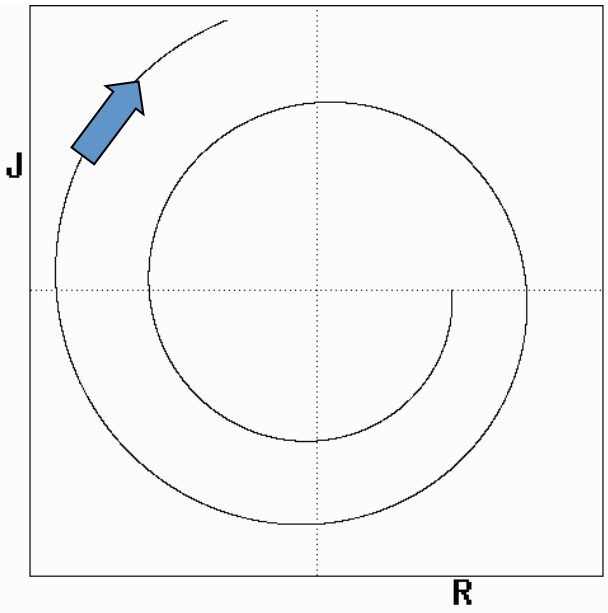
$a + d < 0$



Somewhat cautious

Eventual apathy

$a + d > 0$



Overly eager

Growing volatility

(The only equilibrium is apathy)

Fire and Water
(Do opposites attract?)

- Take $c = -b$ and $d = -a$
- Result depends on a, c, and the initial conditions
- Can end up in any quadrant
- Or with a steady oscillation
- But never apathy

Peas in a Pod
(Are clones bored or blissful?)

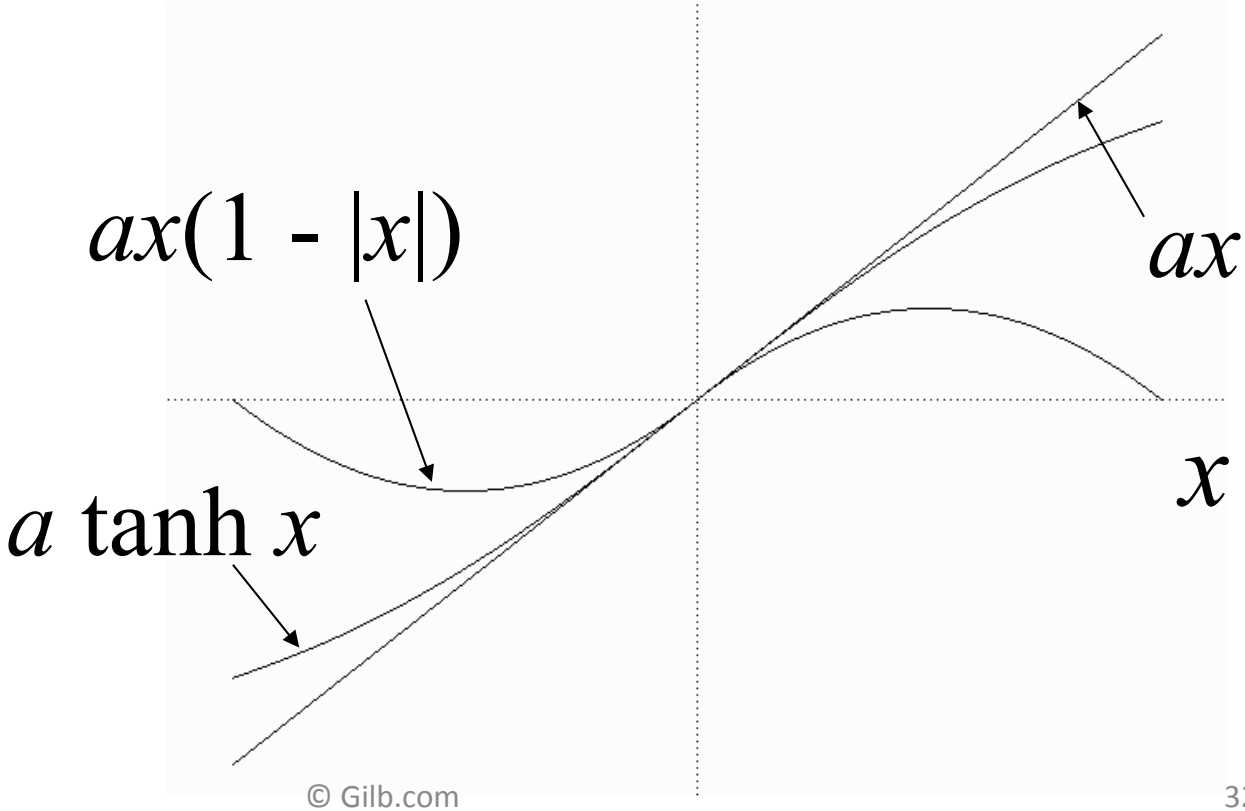
- Take $c = b$ and $d = a$
- Result depends on a, b, and the initial conditions
- Can end up in any quadrant
- Or at the origin (boredom)
- But no oscillations

Romeo the Robot (How does Juliet react?)

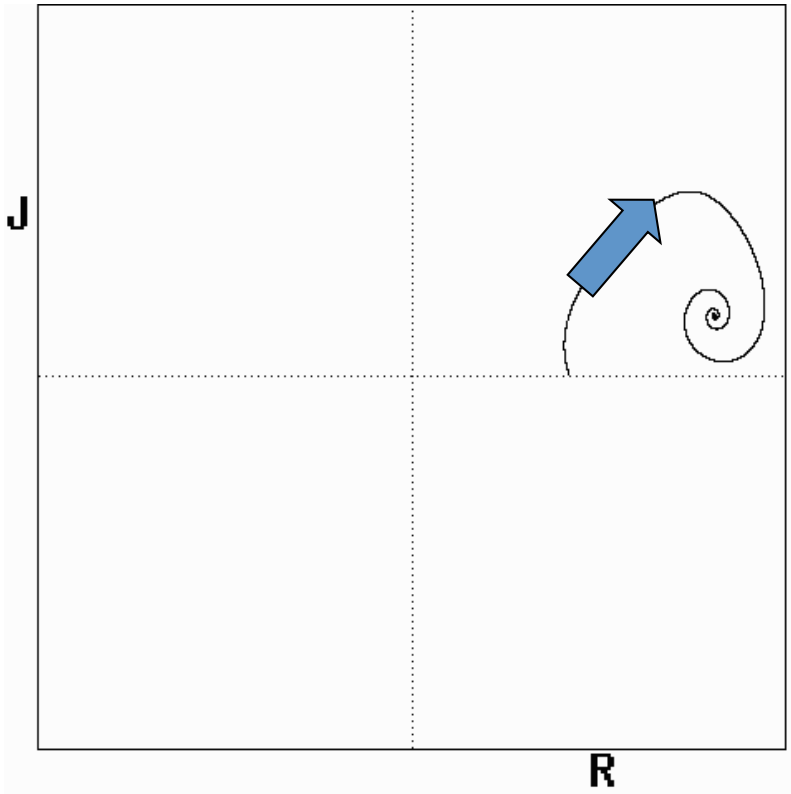
- Take $a = b = 0$ ($dR/dt = 0$)
- $dJ/dt = cR + dJ$
- There is an equilibrium at $J = -cR/d$
- Can be either love or hate depending on signs of R, c, and d
- Stable if $d < 0$, unstable if $d > 0$
- Her feelings never die
- No oscillations are possible

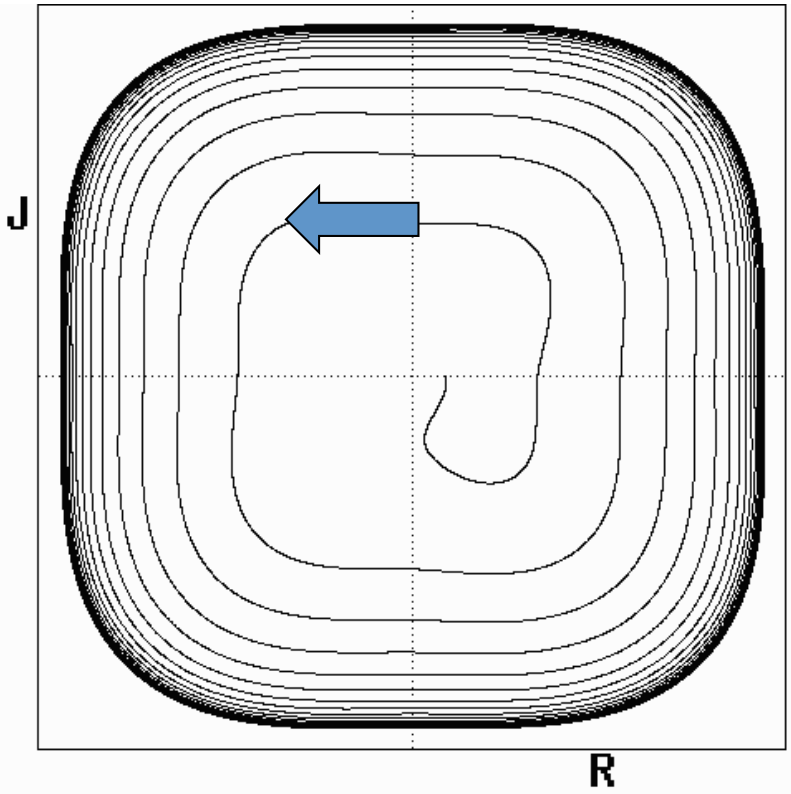
Effect of Nonlinearities

Replace ax with $ax(1-|x|)$, etc.
(logistic function)



New kinds of Dynamics





New equilibrium points

Limit cycles

(but no chaos in 2D)

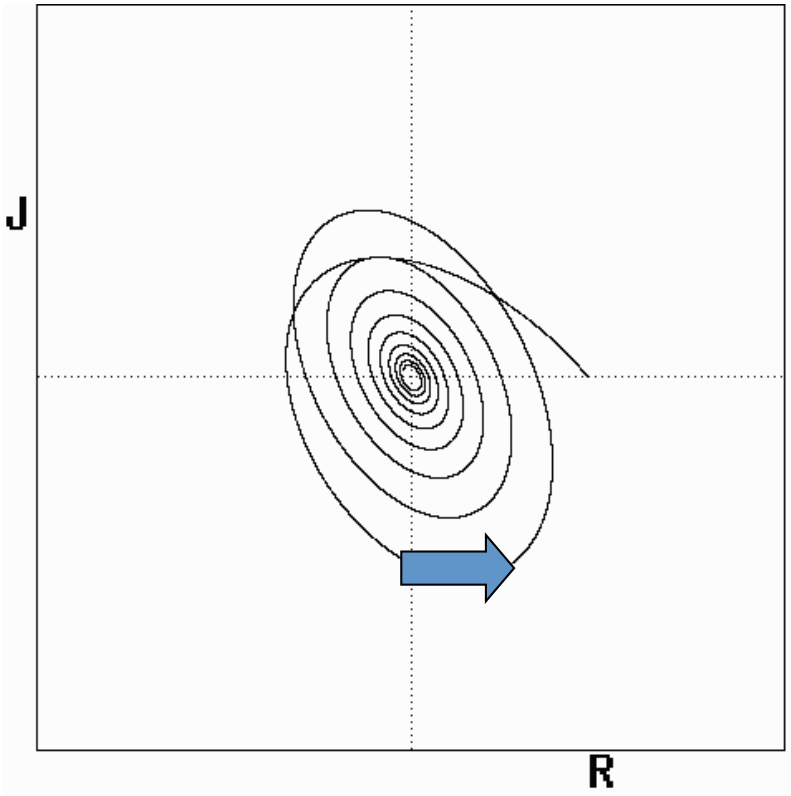
A Love Triangle

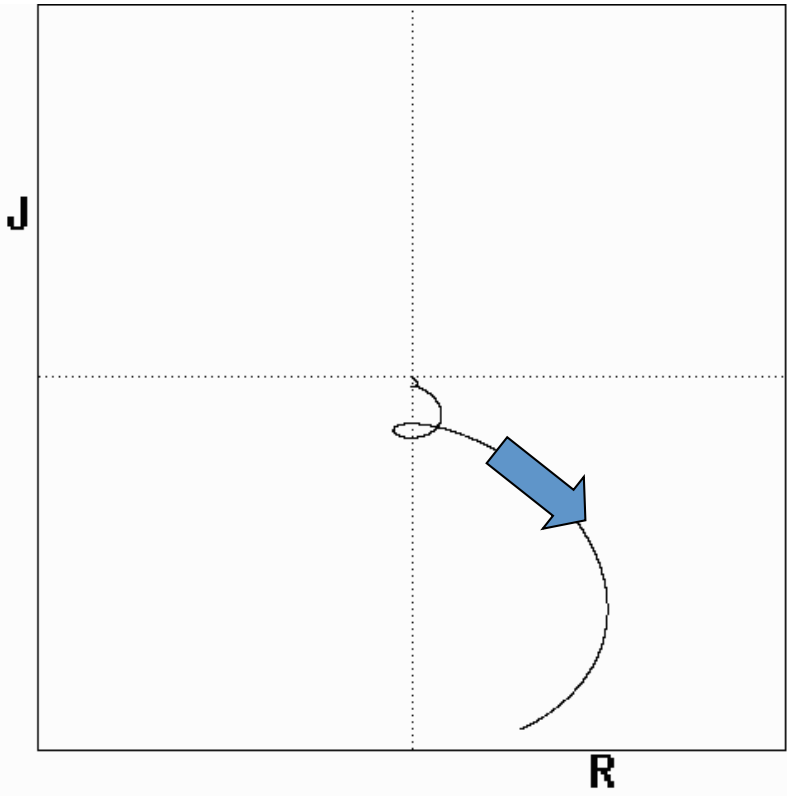
- Romeo has a mistress, Guinevere
- Guinevere and Juliet don't know about one another
- Romeo responds to each with the same romantic style (same *a* and *b*)
- Guinevere's hate has the same effect on his feelings for Juliet as does Juliet's love, and vice versa

Love Triangle Equations

- $dR_J/dt = aR_J + b(J - G)$
- $dJ/dt = cR_J + dJ$
- $dR_G/dt = aR_G + b(G - J)$
- $dG/dt = eR_G + fG$
- System is 4D (4 variables)
- There are 6 parameters
- System is linear (no chaos)

Linear Love Triangle Examples



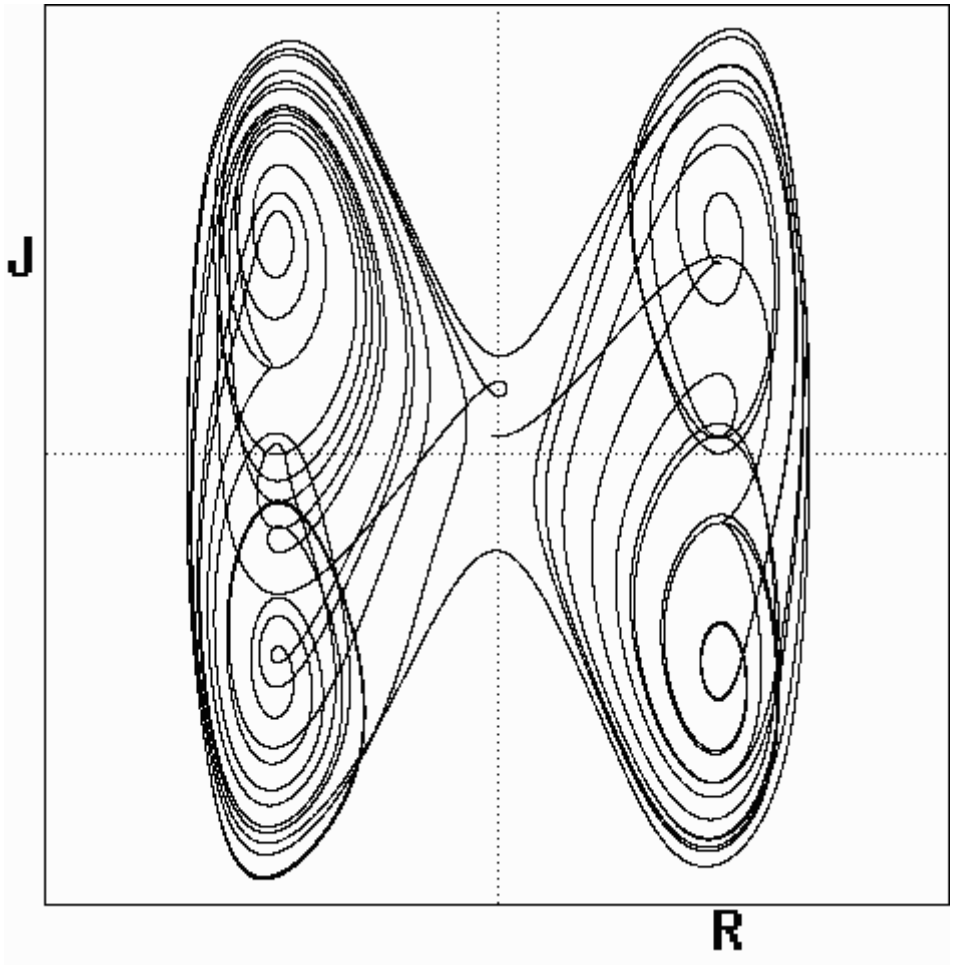


Romeo's Fate

- Averaged over all romantic styles (combinations of parameters) and initial conditions:
  - 37% loves Juliet & hates Guinevere
  - 37% loves Guinevere & hates Juliet
  - 6% loves both (2% everyone in love)
  - 6% hates both (2% everyone in hate)
  - 14% apathy (10% everyone apathetic)
- Anything can happen!

One Chaotic Solution of Nonlinear Love Triangle

"Strange attractor of love"



$a,b,c,f > 0$; $d,e < 0$

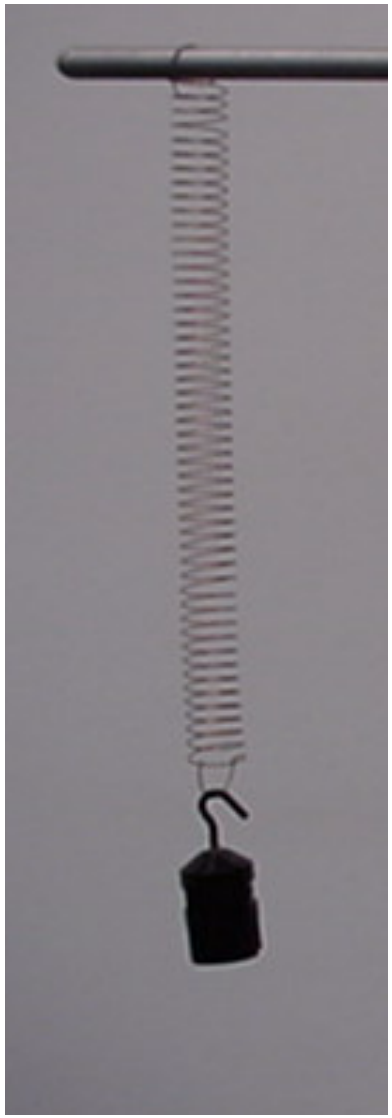
(Romeo is an "eager beaver")

Possible Further Studies

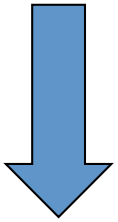
- What happens if Guinevere and Juliet know about one another? (6D system)
- What happens if only Guinevere knows about Juliet? (5D system, asymmetric)
- What happens if Juliet and/or Guinevere has another lover? (6D or 8D system)
- What are the dynamics of a free-love commune? (large-D system)
- Is there an optimum pairing of romantic styles that encourages success or portends failure?

If such problems interest you, let's collaborate!

Simple 2D Linear Model

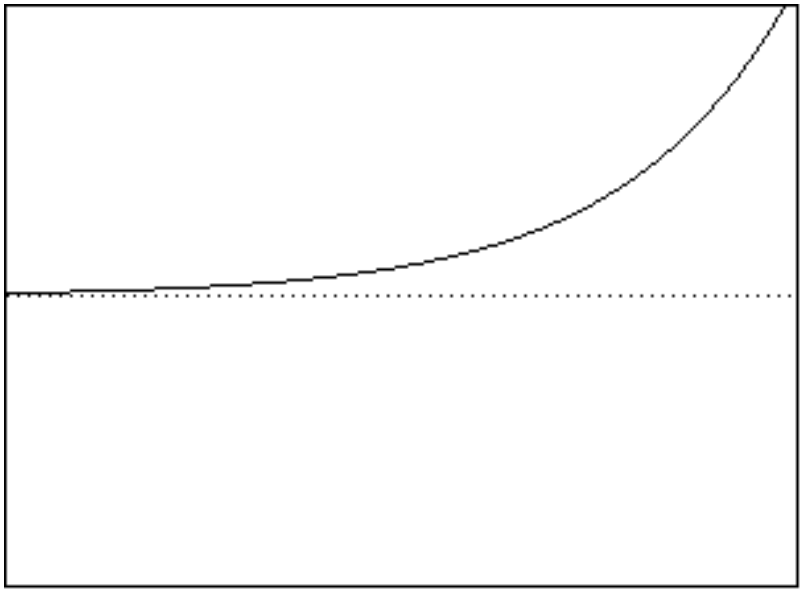


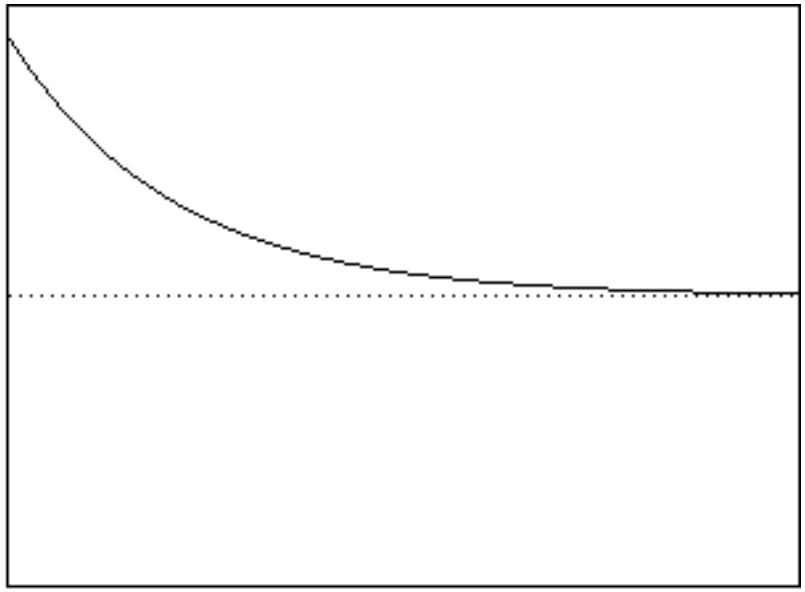
- $dR/dt = aR + bJ$
- $dJ/dt = cR + dJ$

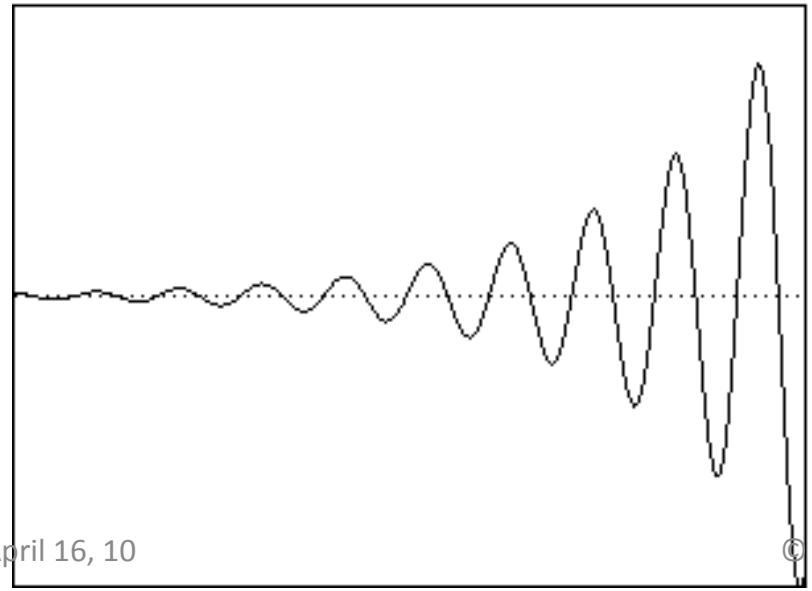


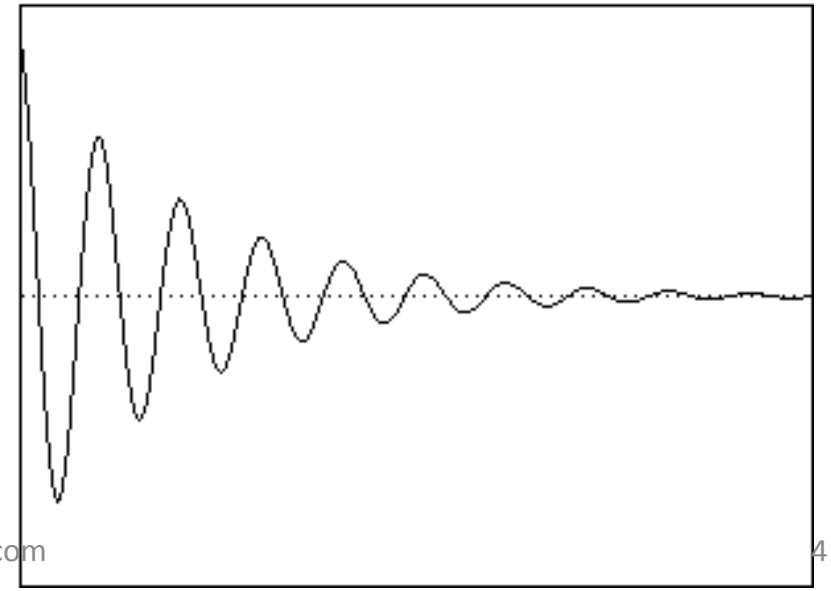
- $d^2R/dt^2 + \beta dR/dt + \omega^2 R = 0$
  - $\beta = -a - d$ (damping)
  - $\omega^2 = ad - bc$ (frequency)

Solutions of 2-D Linear System



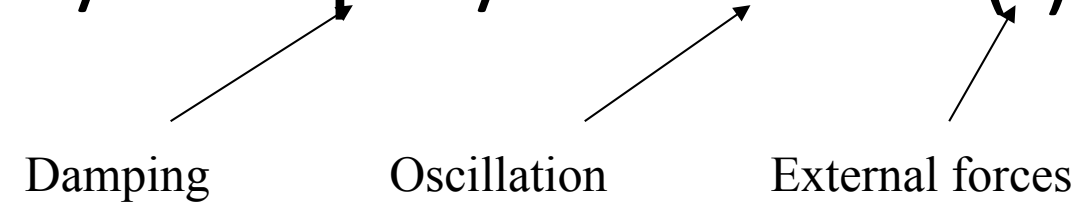






Happiness Model

- $d^2x/dt^2 + \beta dx/dt + \omega^2 x = F(t)$

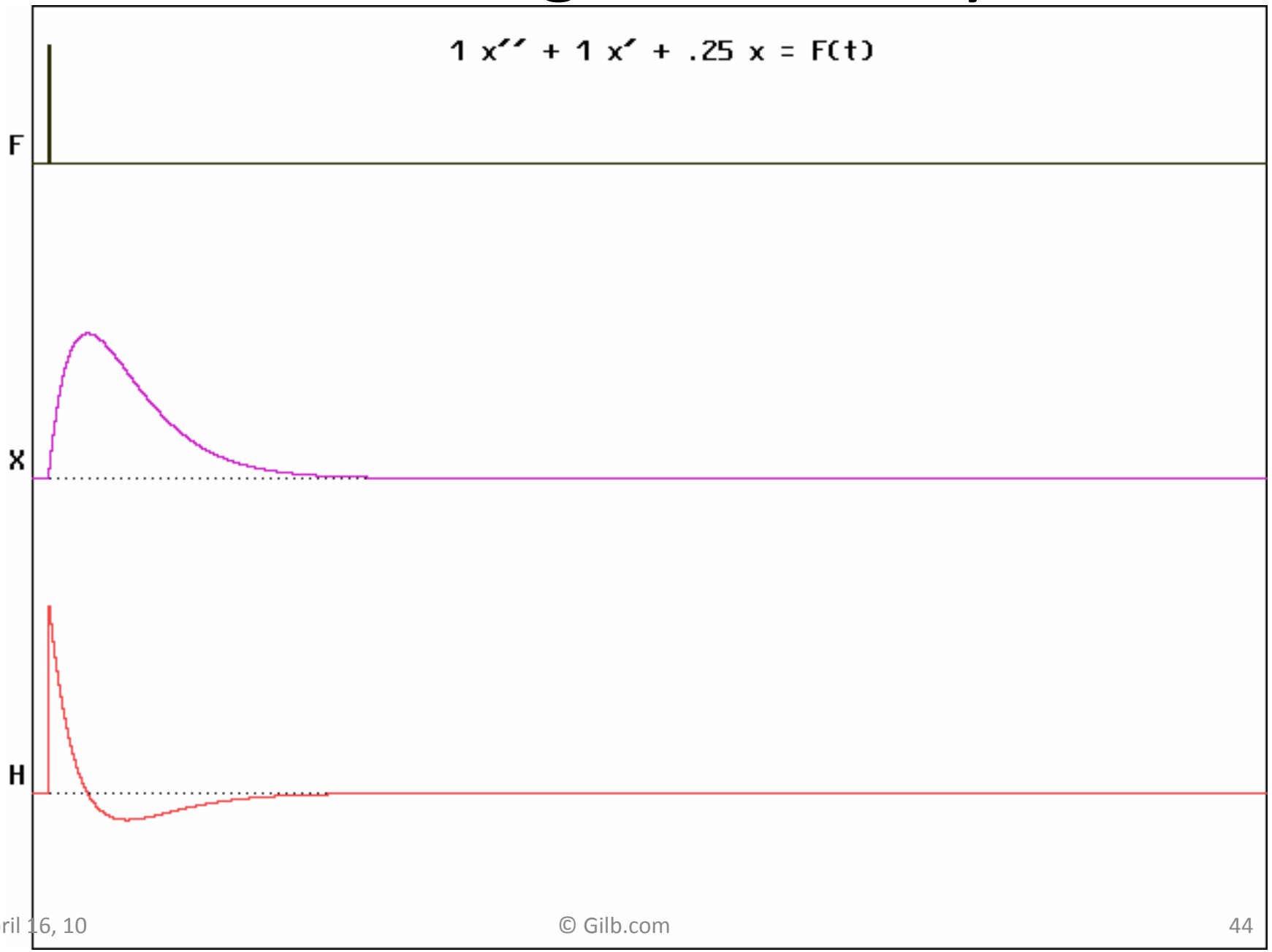


- Happiness: $H = dx/dt$
  - Habituation
  - Acclimation
  - Adaptation
- Only changes are perceived

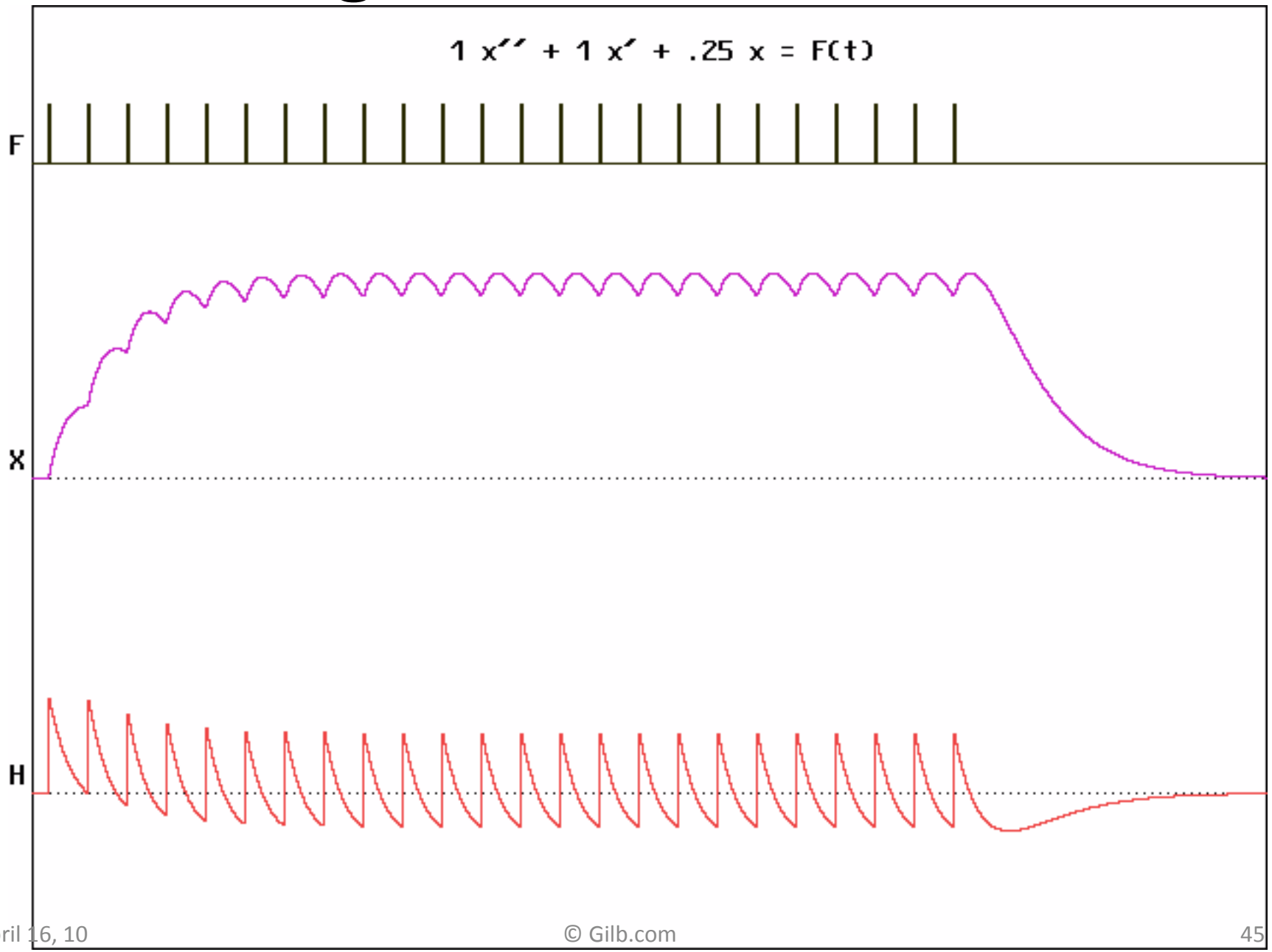
What is *x*?

- *x* = integral of *H*
- *x* is what others perceive
- *H* (your happiness) must average to zero (with positive damping)
- *x* does not average to zero

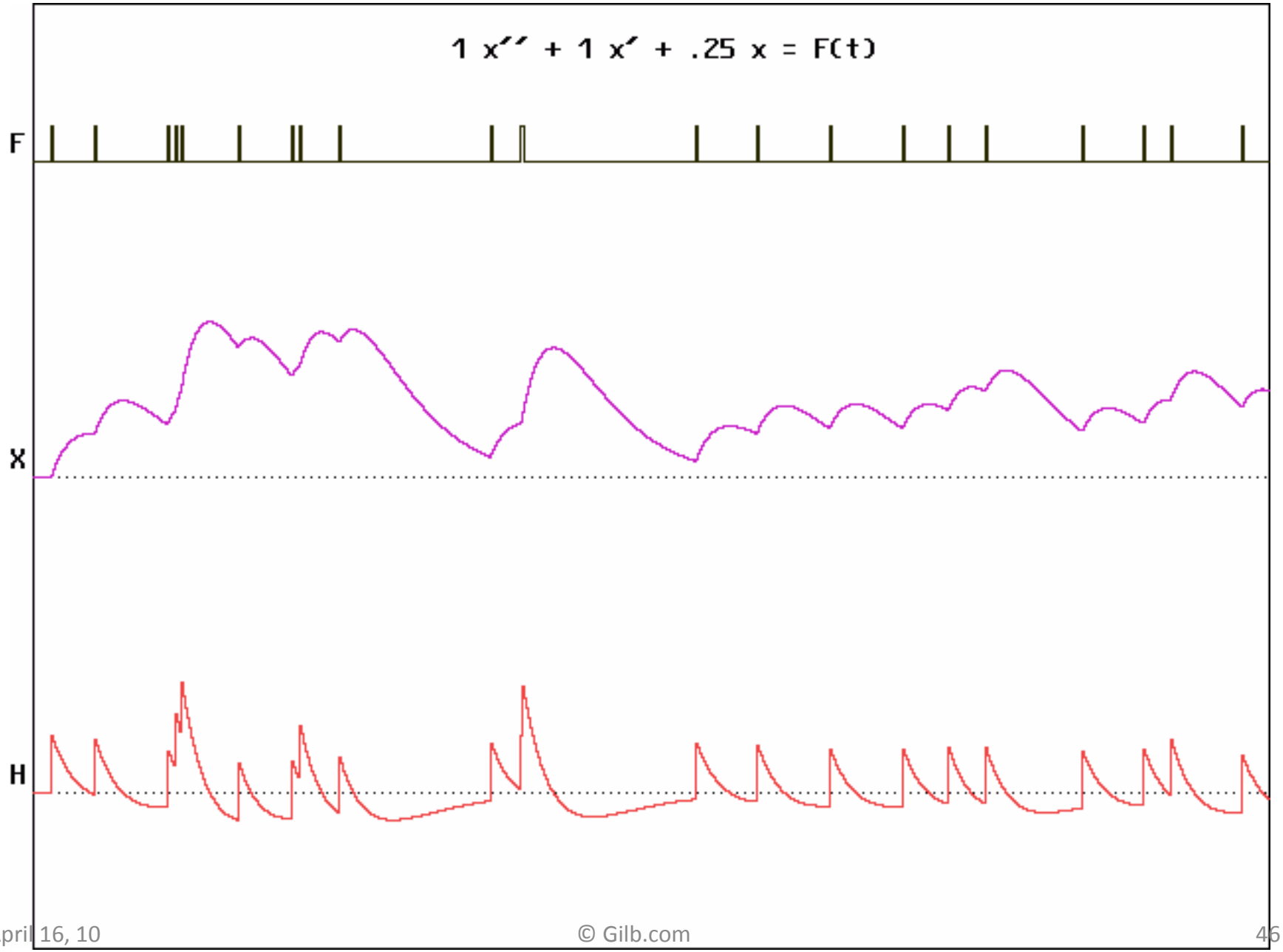
Winning the Lottery



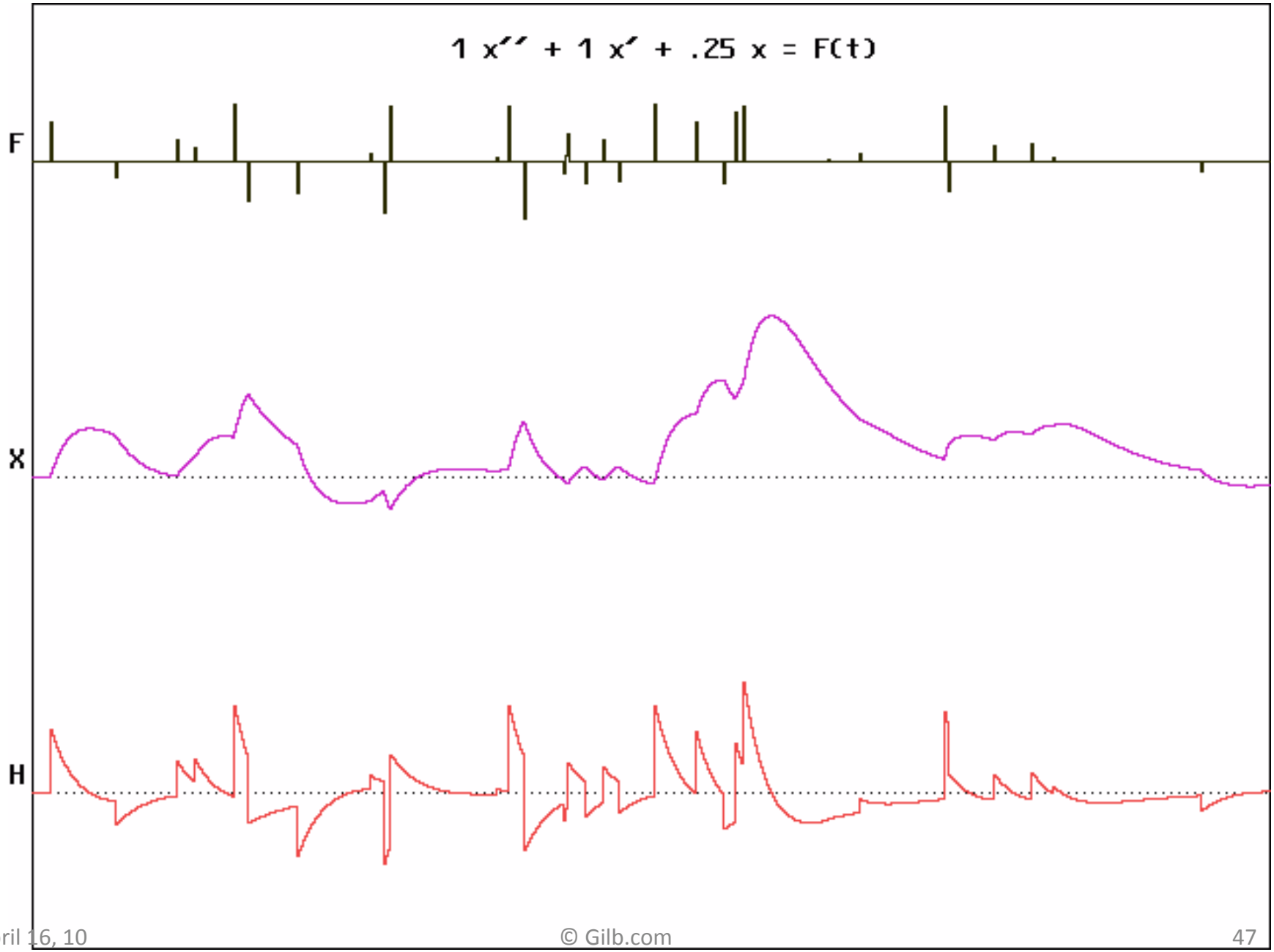
Drug or Other Addiction



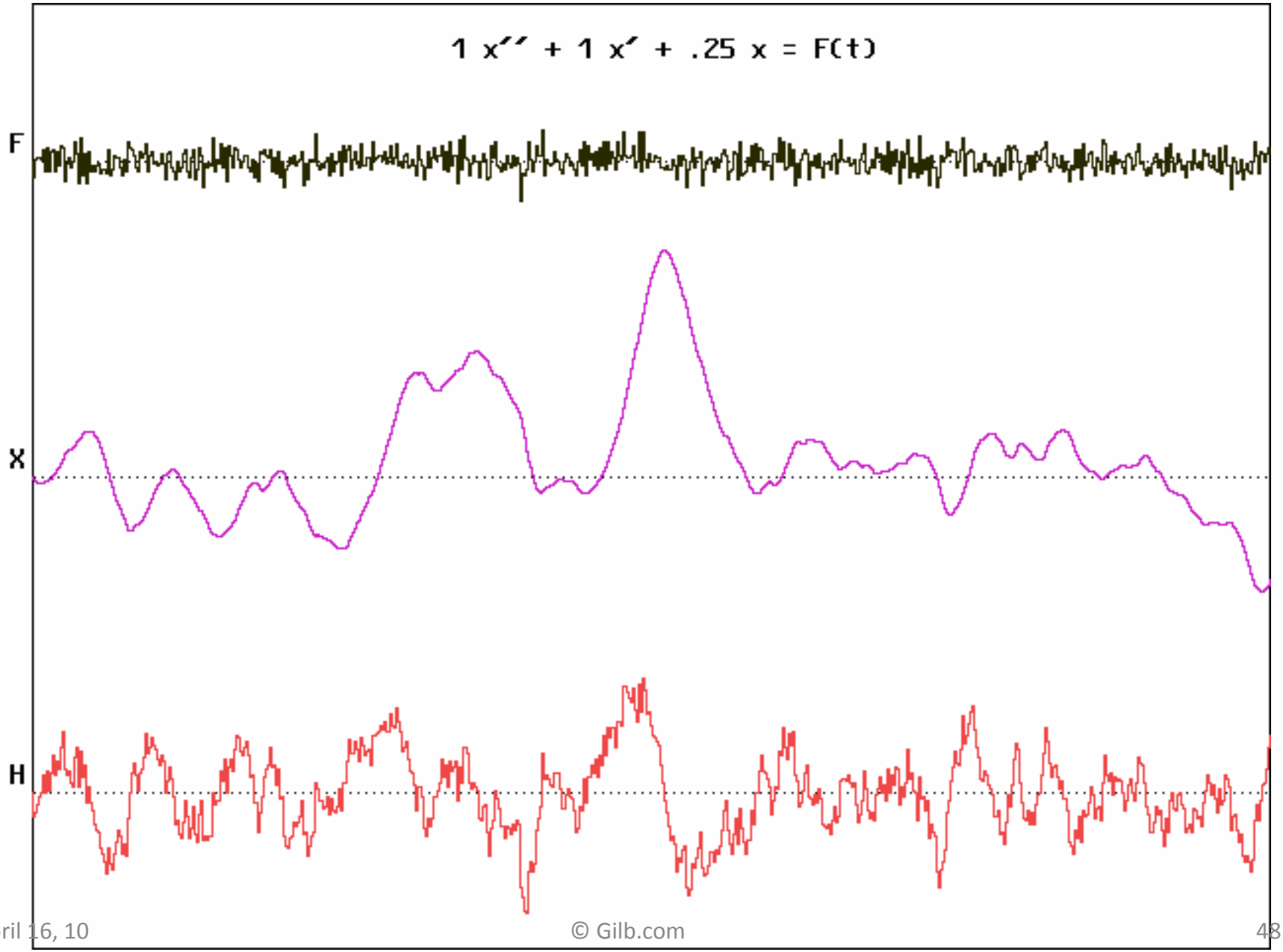
Intermittent Reinforcement



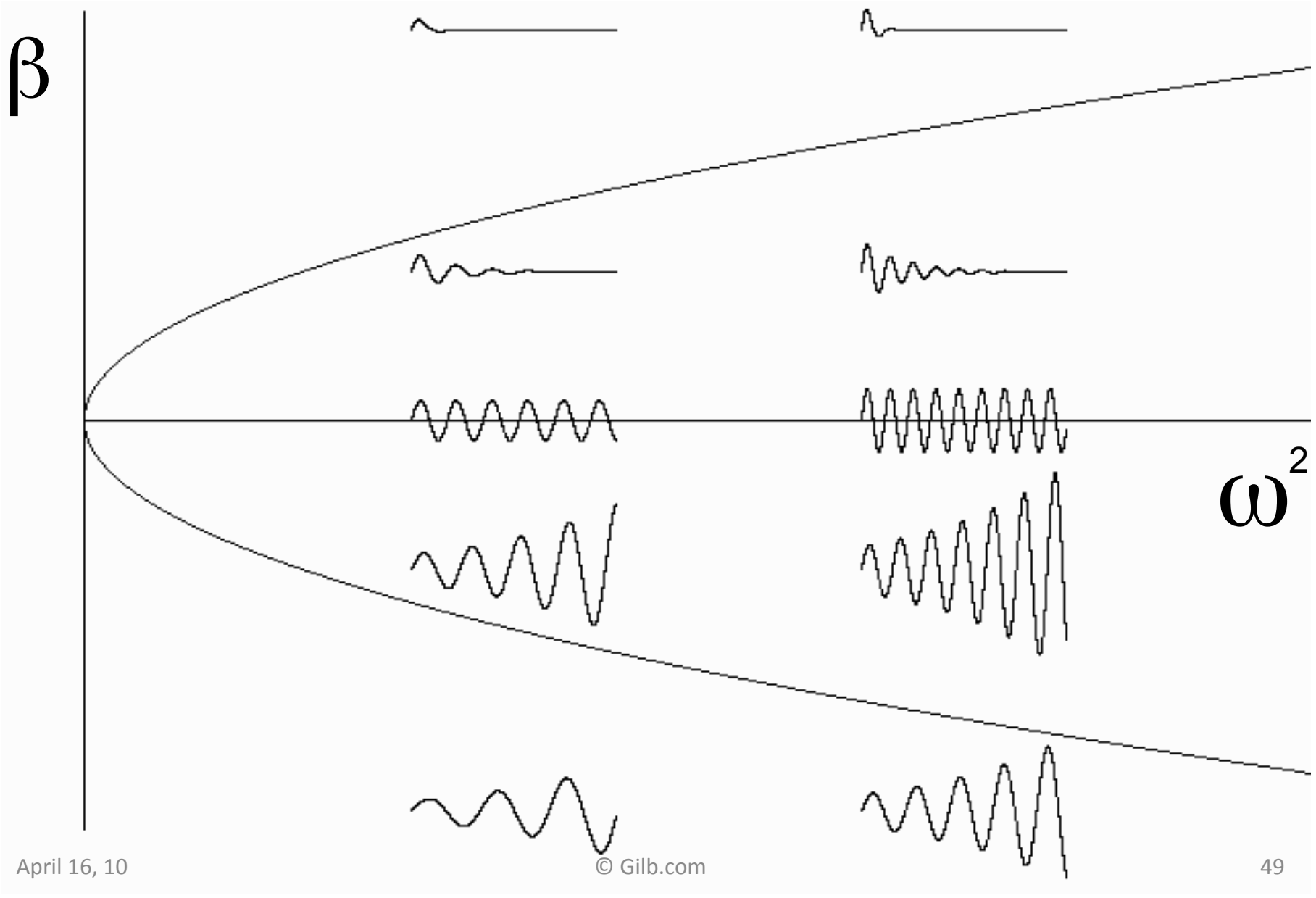
Random Events



Real Life



Parameter Space



Some Implications

- Constant happiness is an unrealistic goal.
- Others see less volatility in you and often wrongly conclude how you feel.
- Individuals can be categorized by their values of β and ω.
- Manic depression may correspond to $\beta \sim 0$.
- Long prison terms may be ineffective.

A few other happiness studies

- Brickman, Coates & Janoff-Bulman (1978) report only small differences in life satisfaction between paraplegics, control subjects, and lottery winners.
- Lykken (1981) reports that religious people are not noticeably happier than freethinkers.
- Diener & Diener (1996) review studies indicating that all American socioeconomic groups score above neutral in life satisfaction, as do people with severe disabilities.

What disabilities, you ask?

- Hellmich (1995) reports that 84% of individuals with extreme quadriplegia say that their life is average or above average.
- Delespaul & DeVries (1987) report that people with chronic mental problems claim positive well-being.

As for the dynamics

- Silver (1982) reports that individuals with spinal cord injuries are very unhappy immediately following their injury, but that 58% state that happiness is their strongest emotion by the third week after their injuries.
- Suh, Diener, & Fujita (1996) report that good and bad events have almost no effect on happiness after 6 months.

In Summary ... (Lykken 1999)

- There seem to be no permanent ups and downs; natural selection has made us this way, because, by accommodating to both adversity and to good fortune in this fashion, we remain more productive, more adaptable to changing circumstances, and more likely to have viable offspring.

Other Similar Qualities

- Sense of wealth
- Health
- Intelligence
- Skills
- Senses
  - hot/cold
  - smell
  - vision
  - hearing ...

Summary

- Love and happiness are wonderful
- So is mathematics

References

- http://sprott.physics.wisc.edu/ lectures/ love&hap/ (This talk)

- Steven H. Strogatz, *Nonlinear Dynamics and Chaos* (Addison-Wesley, 1994)

- sprott@juno.physics.wisc.edu