# Domain-Driven Design

The “What” and the “Why”

# Who am I?

- I’m Angga
- I’m a software developer

# History

- I first “met” with Domain-Driven Design in 2009, trying to implement it.
  - .. with BDD
  - .. and CQRS
  - .. and Event Sourcing
  - .. in a mobile device!
- Typical “new toys, cool!” programmer.
  - In retrospect, I was young and naïve ;).

# My aim

- DDD is help me develop better software.
- I hope it can help you, too.

# Word of warning

- There will be many domains visited during this presentation.

# Some (boring) things first

- What is “Domain”?
- What is “Model”?


“define:domain”

a specified sphere of activity or knowledge.

## What is “Domain”?

- A **specified** sphere of activity or knowledge.
- Nothing to do with “.com”, “.net”, “.org”, etc


“define:model”

the technique of representing the real world by a computer program.

## What is Model?

- the technique of **representing the real world** by a computer program.
- Nothing to do with database.
  - For now..


# Part 1: Premises

# Premises

- Computer is a very powerful tool for modeling
- Modeling is hard and takes time
- Language shapes the way we think

# Premises

- Computer is a very powerful tool for modeling
- Modeling is hard and takes time
- Language shapes the way we think

# Computer is a very powerful tool for modeling

# Painting and modeling

- Painting has the ability to model everything, from real concept to abstract concept.
- Art's scale of styles:
  - Realism on one end
  - Abstractionism on the other end
  - Expressionism in between

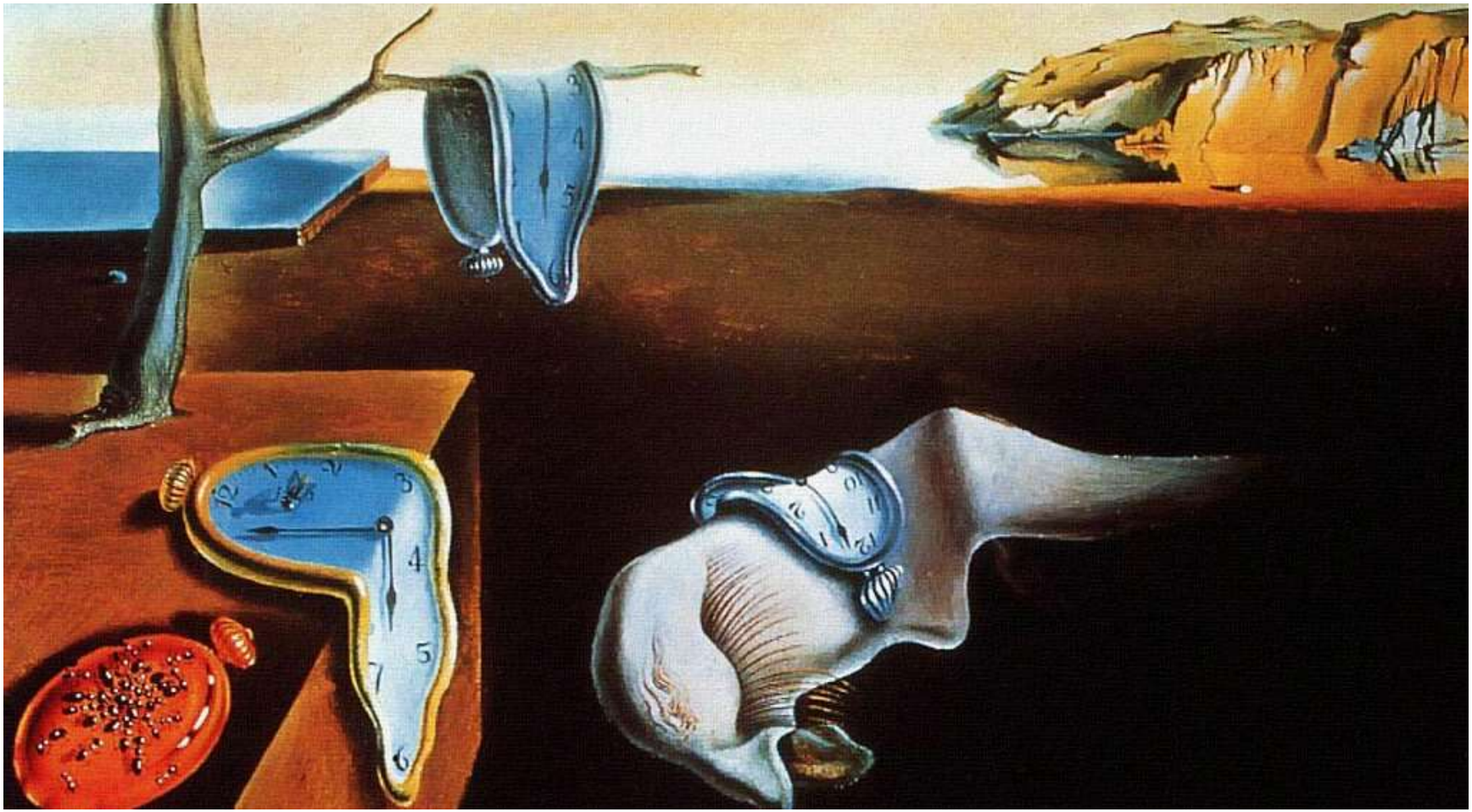
# Art’s scale of styles

# Realism

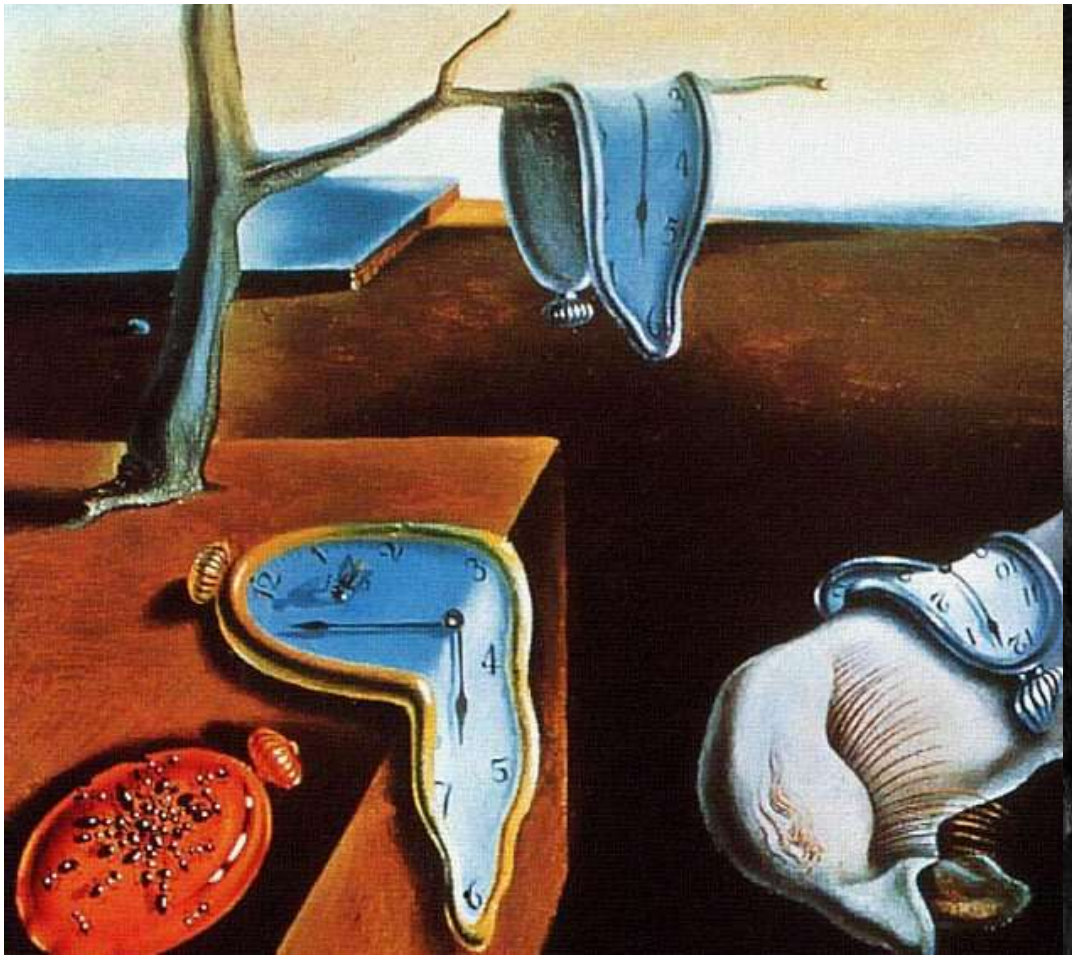
# “Girl with a Pearl Earring”

# Abstractionism

# “The Persistence of Time”

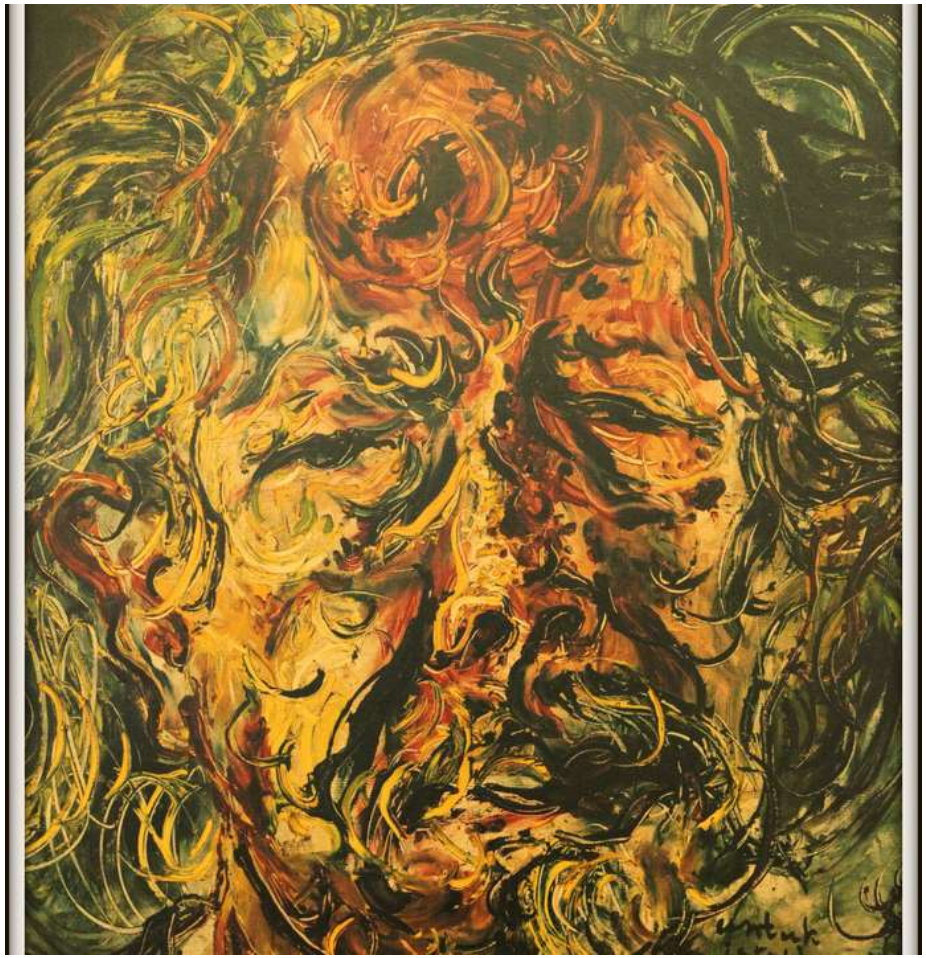


# “The Persistence of Time”



# Expressionism, somewhere in the middle

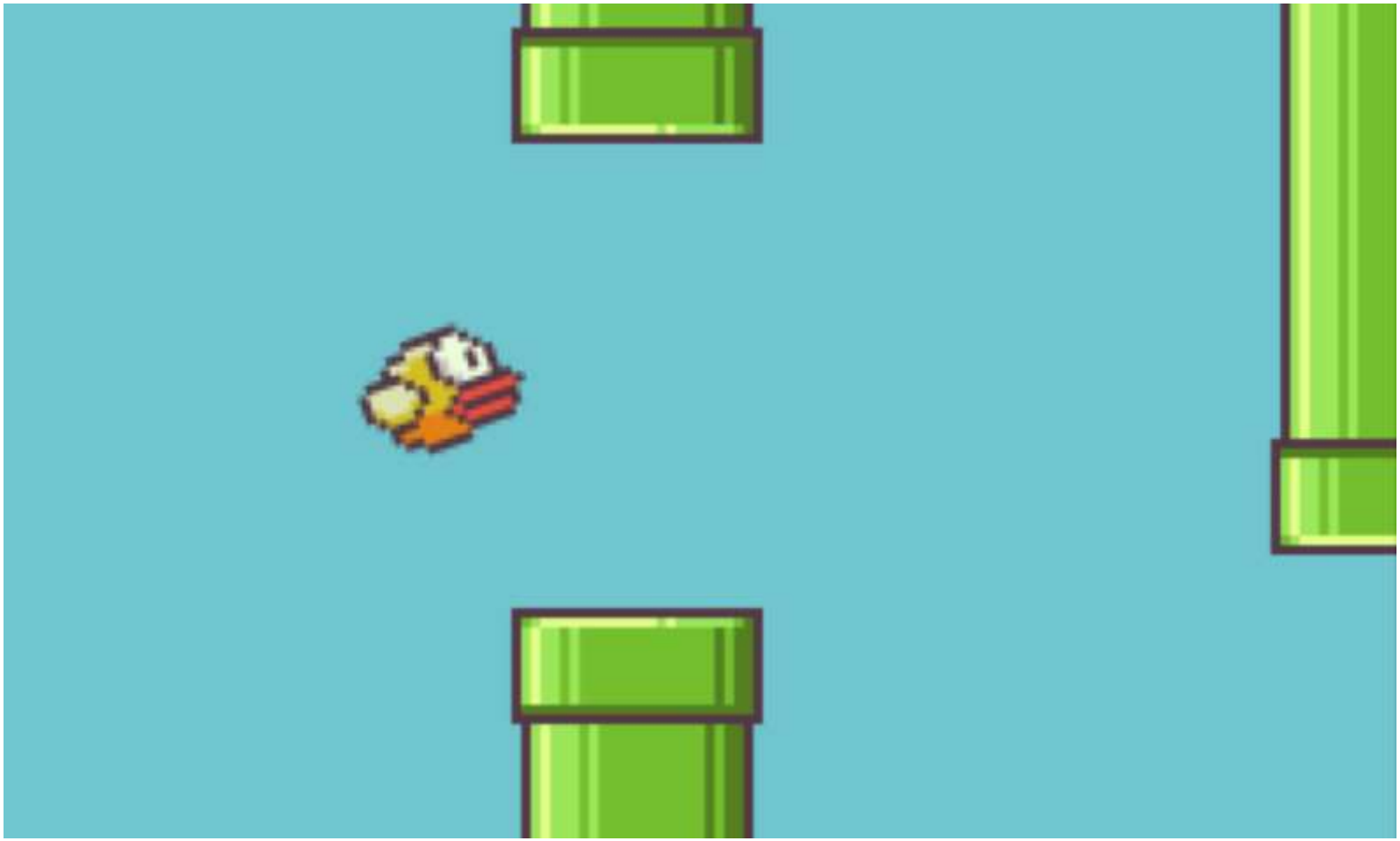
# Affandi's self-portrait



- But what does it has to do with software development?

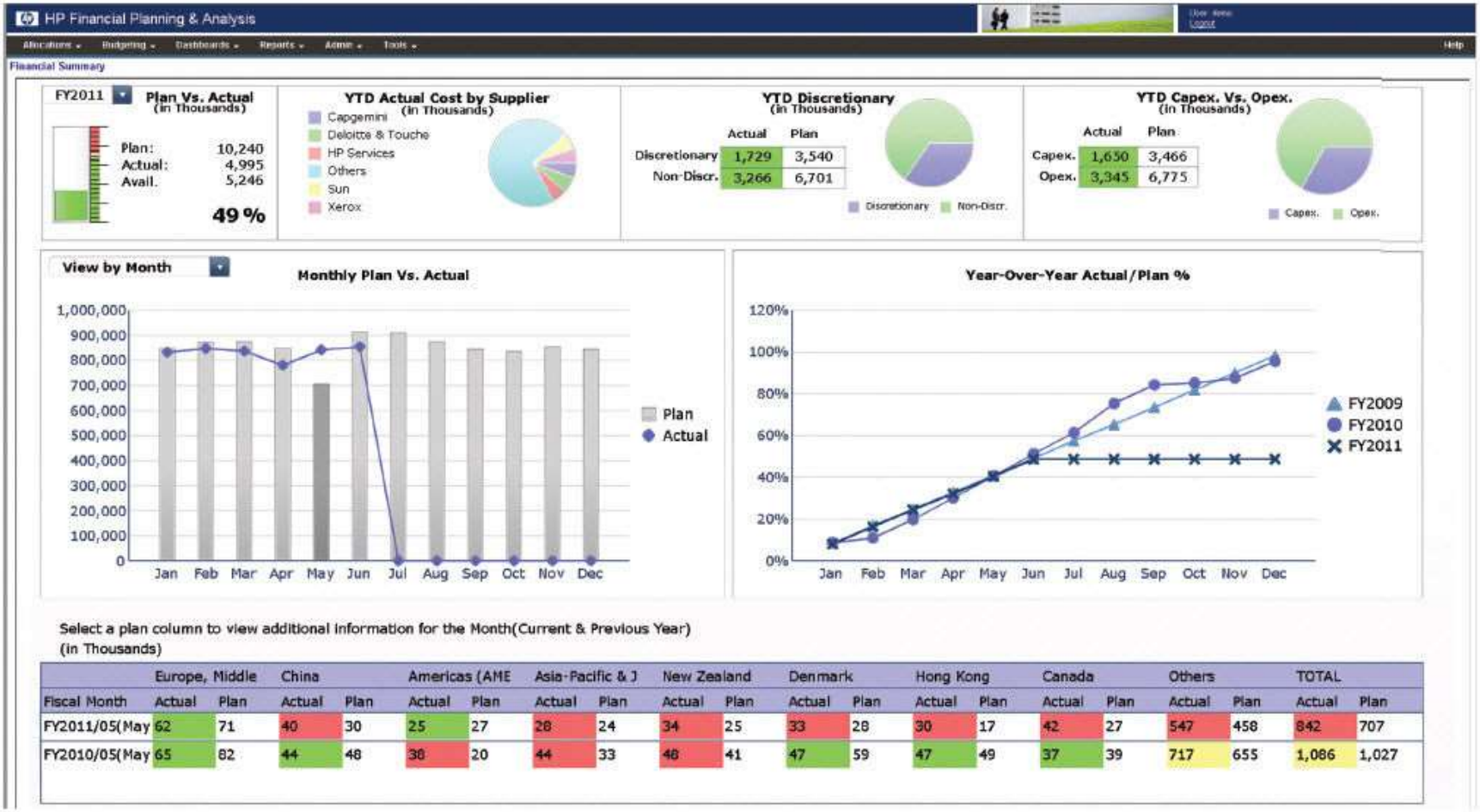
- Software also has “scale of styles”.

# Abstract: Flappy bird



# Abstract: Super Mario Bros.

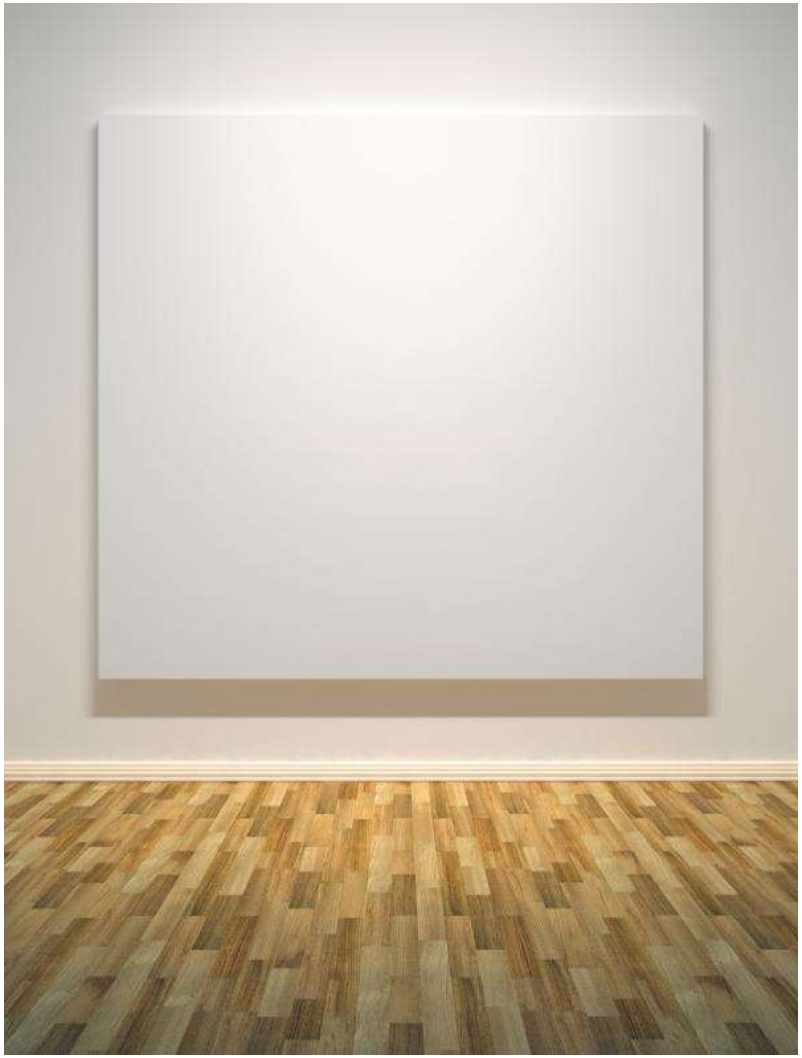
# Realism: Financial Software

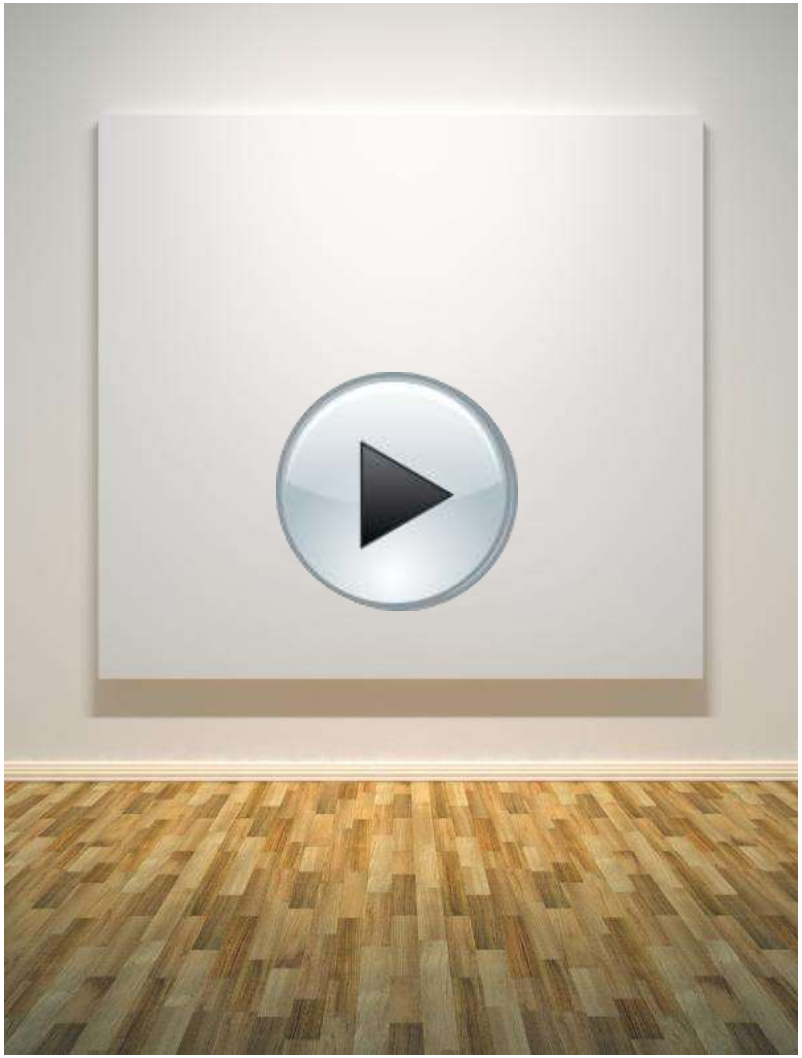



| | Europe, Middle | | China | | Americas (AME | | Asia-Pacific & J | | New Zealand | | Denmark | | Hong Kong | | Canada | | Others | | TOTAL | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Fiscal Month | Actual | Plan | Actual | Plan | Actual | Plan | Actual | Plan | Actual | Plan | Actual | Plan | Actual | Plan | Actual | Plan | Actual | Plan | Actual | Plan |
| FY2011/05(May | 62 | 71 | 40 | 30 | 25 | 27 | 28 | 24 | 34 | 25 | 33 | 28 | 30 | 17 | 42 | 27 | 547 | 458 | 842 | 707 |
| FY2010/05(May | 65 | 82 | 44 | 48 | 38 | 20 | 44 | 33 | 48 | 41 | 47 | 59 | 47 | 49 | 37 | 39 | 717 | 655 | 1,086 | 1,027 |

# So, what's the different?

- Computer is (arguably) more powerful!





# “Why Software is Eating the World” (2011)

- Software companies dominance over hardware companies
  - A software company bought hardware company (Google bought Motorola)
  - Virtual stores crumples “hardware” stores (Amazon, Netflix)
- Even “physical” company uses software for their unfair advantage
  - FedEx is best thought of as a software network that happens to have trucks, planes and distribution hubs attached.

- Computer is a very powerful modeling tool.

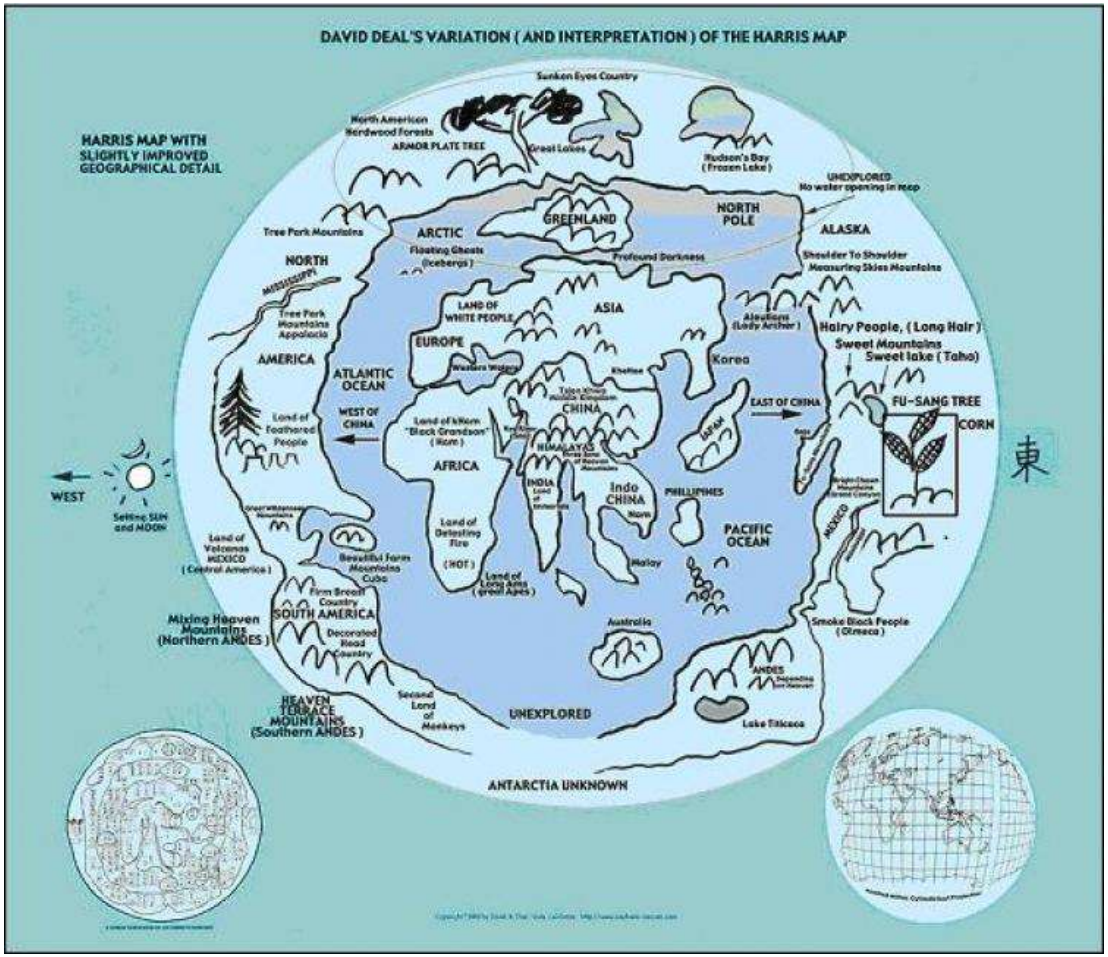
# Premises

- Computer is a very powerful tool for modeling
- Modeling is hard and takes time
- Language shapes the way we think

# Modeling is hard and takes time

# Ancient Chinese map

# Ancient Chinese map



# Old world map

# Modern “map”

- Again, what does it have to do with the software development?

# “There are known knowns” problem

- The knowing phase.
  - The known knowns.
  - The known unknowns.
  - The unknown unknowns.
- Ancient Chinese explorers doesn’t know about Americas continents.
- European explorers doesn’t know about Australia continent.

# Example: my first effort to learn web development (~2004)

- The known knowns
  - I know basic stuff how to use a computer.
- The known unknowns
  - I know I have to learn HTML+CSS.
  - I know I have to learn PHP.
- The unknown unknowns
  - I need to install Apache.
  - I need to pick a PHP version.
  - I need to learn MySQL.
  - I need to install PHP extension.
  - ..

# The danger of unknown unknowns

- Examples (totally made up):
  - “You can’t book a cleaning service in holiday”
    - “Unless it’s public leave holiday”
      - “Unless it’s after new year’s eve holiday”

- Modeling is hard and takes time

# Premises

- Computer is a very powerful tool for modeling
- Modeling is hard and takes time
- Language shapes the way we think

# Language shapes the way we think

# Sapir-Whorf Hypothesis

- “[..] the idea that differences in the way languages encode cultural and cognitive categories affect the way people think, so that speakers of different languages will tend to think and behave differently depending on the language they use.”

# Examples

- English: He ate the melon.
- Indonesia: Dia memakan melon.
- At least, two facts are lost:
  - The gender
  - The time

- Study have shown that children who speaks languages with gender bias know their gender faster than the children who speaks language without one (Guiora, 1983)
- What this means: Children in England knows their gender before children in Indonesia.

# Javanese expressiveness

- Jatuh ke belakang: NGGEBLAK
- Jatuh dari atas: CEBLOK
- Jatuh ke depan: NYUNGSEP
- Jatuh terlempar: NJUNGKEL
- Jatuh tersandung: NJLUNGUP
- Jatuh meluncur: NDLOSOR
- Nyaris jatuh: MINGKLIK-MINGKLIK

- Language shapes the way we think!

# Premises

- Computer is a very powerful tool for modeling
- Modeling is hard and takes time
- Language shapes the way we think

# Domain-Driven Design

# What DDD looks like?

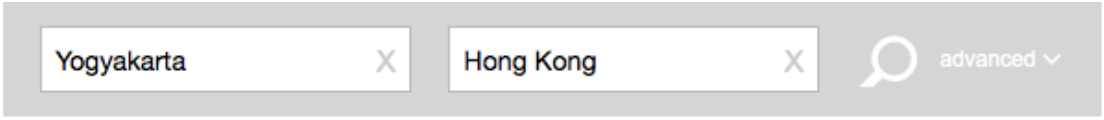
- In the communication between team-members.
- In the code.

# What DDD looks like?

- In the communication between team-members.
- In the code.

# Comparison

- Two dialogs between the user (domain expert) and the developer.
- Domain: Cargo Shipping
- Problem: How to efficiently generate Itinerary?




**Combined freight rate index: 2 689, transit time estimate: 11.85 days, CO2 emission index: 1 369**

- Yogyakarta (origin) - Semarang, Java (Tanjung Emas) port
- Semarang, Java (Tanjung Emas) port - Hong Kong port
- Hong Kong port - Hong Kong (destination)

**Combined freight rate index: 3 536, transit time estimate: 7.65 days, CO2 emission index: 1 437**

- Yogyakarta (origin) - Jakarta, Java (Tanjung Priok) port
- Jakarta, Java (Tanjung Priok) port - Tanjung Pelepas, Johor port
- Tanjung Pelepas, Johor port - Hong Kong port
- Hong Kong port - Hong Kong (destination)

# Core concepts

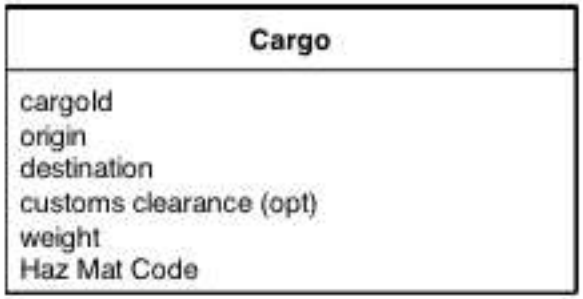
- Cargo
- Custom Clearance Point
- Route
- Route Specification
- Itinerary

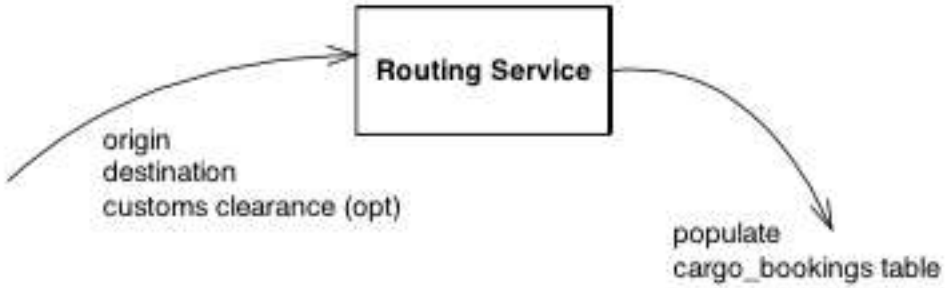
# In the Communication (A)

- USER: So when we change the **customs clearance point**, we need to redo the whole routing plan.

- DEVELOPER: Right. We'll delete all the rows in the shipment table with that **cargo id**, then we'll pass the origin, destination and the new **customs clearance point** into the **Routing Service** and it will repopulate the table. We'll have to have a Boolean in the **Cargo** so we'll know there is data in the shipment table.

## Scenario 1: Minimal abstraction of the domain




Database table: cargo_bookings

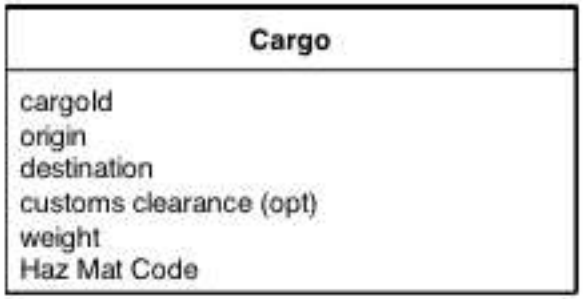
| Cargo_ID | Transport | Load | Unload |
|---|---|---|---|
| | | | |

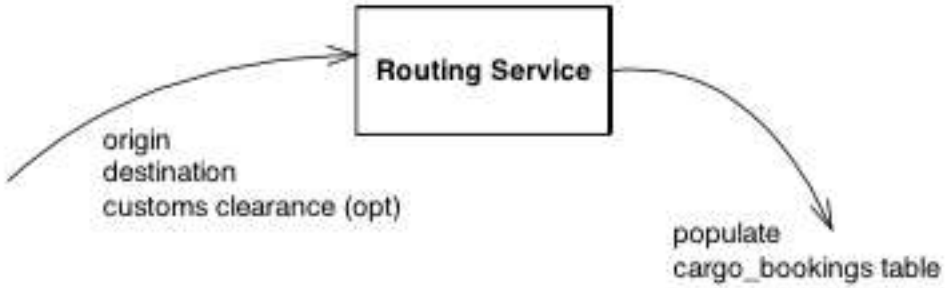
# In the Communication (continued) (A)

- USER: Delete the rows? Ok, whatever. Anyway, if we didn't have a **customs clearance point** at all before, we'll have to do the same thing.

- DEVELOPER: Sure, any time you change the origin, destination or **customs clearance point** (or enter one for the first time) we'll check to see if we have shipment data and then we'll delete it and then let the **Routing Service** regenerate it.

## Scenario 1: Minimal abstraction of the domain




Database table: cargo_bookings

| Cargo_ID | Transport | Load | Unload |
|---|---|---|---|
| | | | |

# In the Communication (continued) (A)

- USER: Of course, if the old **customs clearance** just happened to be the right one, we wouldn't want to do that.

- DEVELOPER: Oh, no problem. It is easier to just make the **Routing Service** redo the loads and unloads every time.

- USER: Yes, but it is extra work for us to make all the supporting plans for a new **itinerary**, so we don't want to reroute unless the change necessitates it.

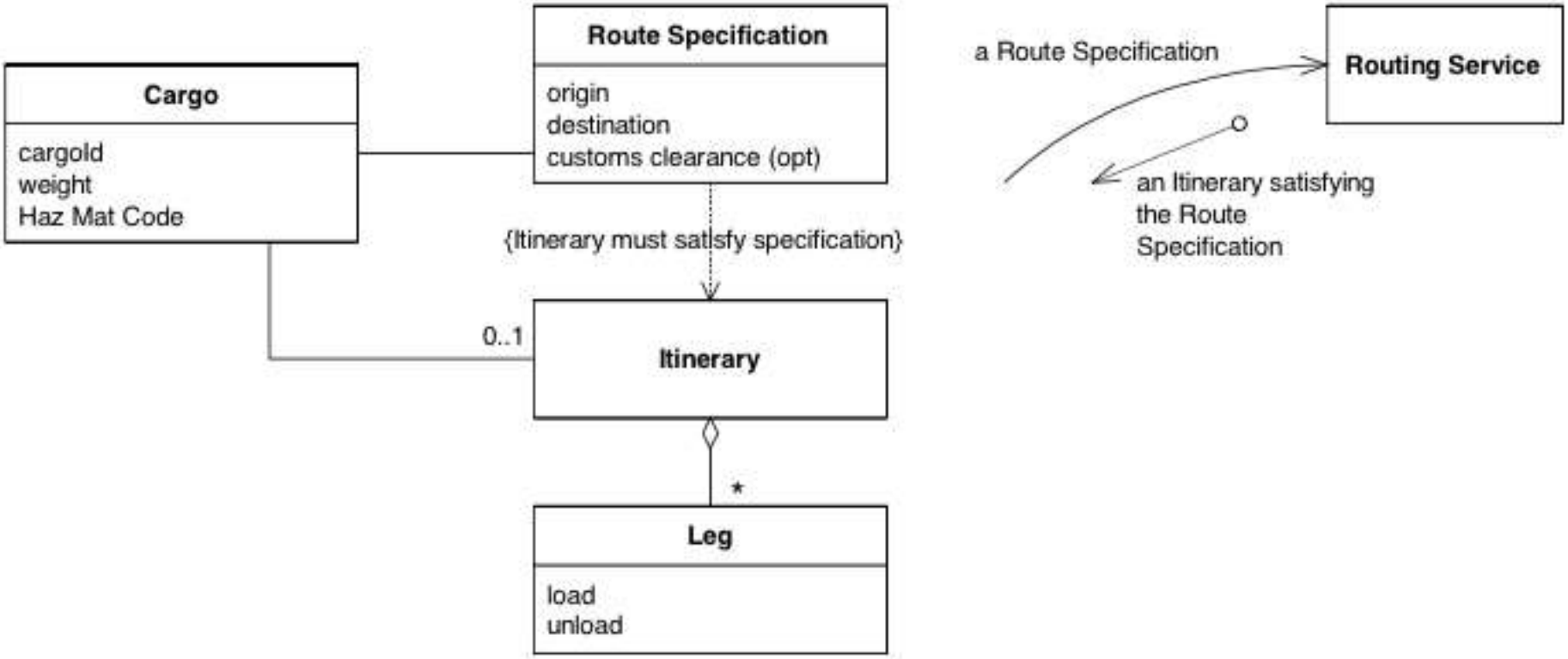
- DEVELOPER: Ugh. Well, then, if you are entering a **customs clearance point** for the first time, we'll have to query the table to find the old derived **customs clearance point**, and then compare it to the new one. Then we'll know if we need to redo it.

# In the Communication (B)

- USER: So when we change the **customs clearance point**, we need to redo the whole routing plan.

- DEVELOPER: Right. When you change any of the attributes in the **Route Specification**, we'll delete the old Itinerary and ask the **Routing Service** to generate a new one based on the new **Route Specification**.

## Scenario 2: Domain model enriched to support discussion

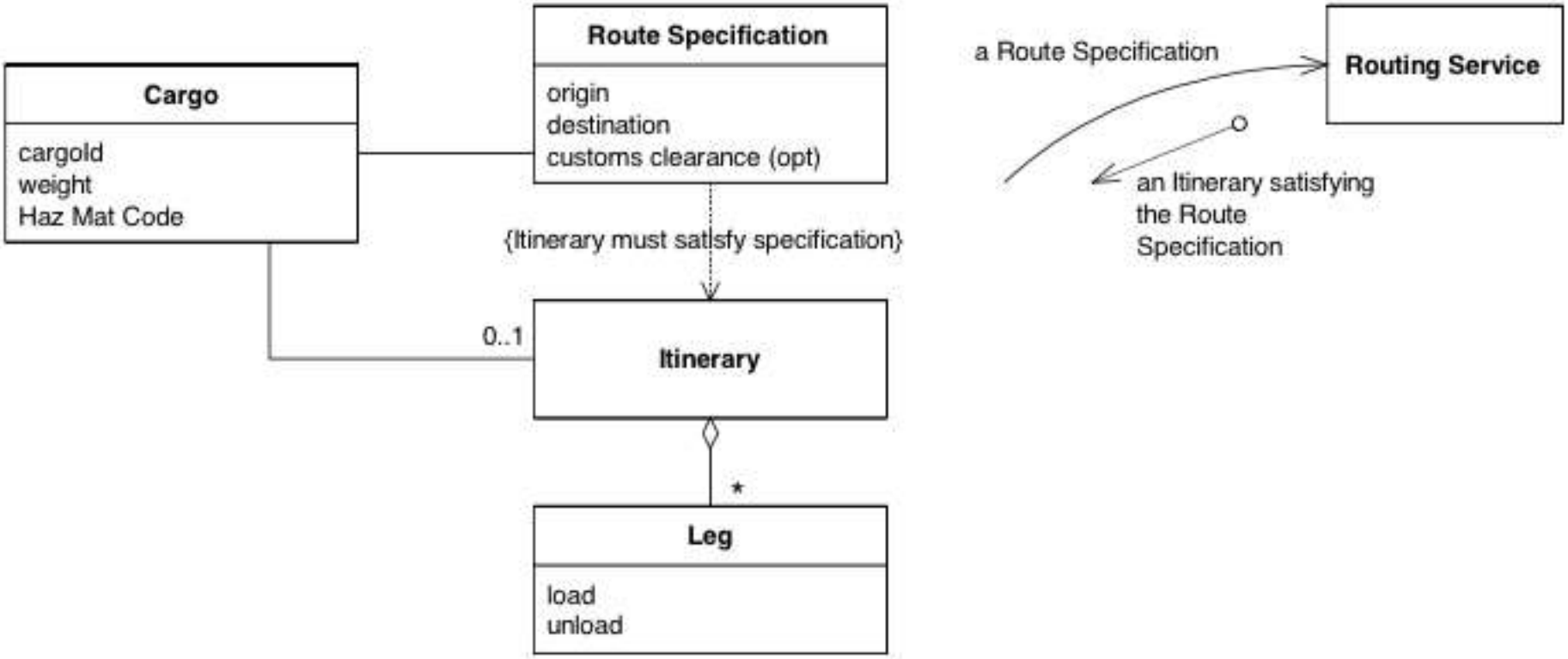


# In the Communication (continued) (B)

- USER: If we hadn't specified a **customs clearance point** at all before, we'll have to do the same thing.

- DEVELOPER: Sure, any time you change anything in the **Route Specification**, we'll regenerate the **Itinerary**. That includes entering something for the first time.

## Scenario 2: Domain model enriched to support discussion



# In the Communication (continued) (B)

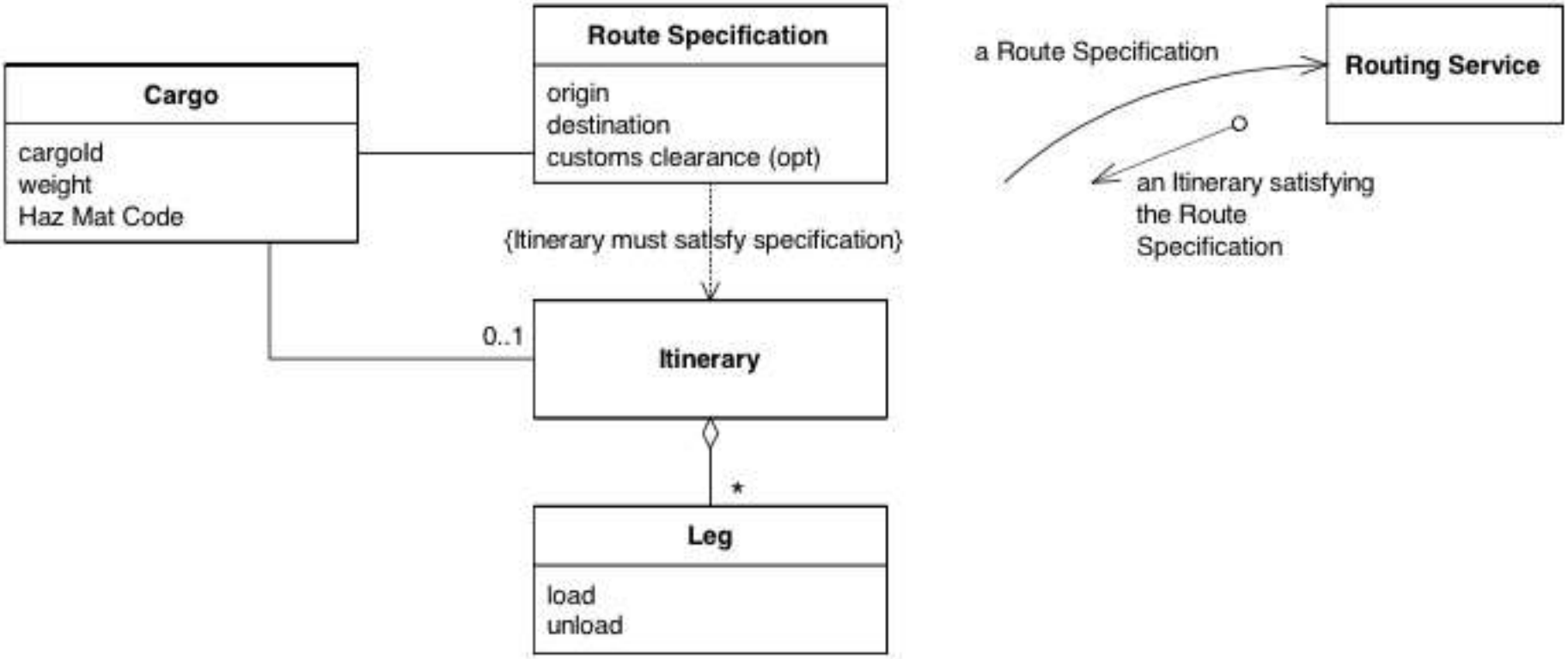
- USER: Of course, if the old **customs clearance** just happened to be the right one, we wouldn't want to do that.

- DEVELOPER: Oh, no problem. It is easier to just make the **Routing Service** redo the **Itinerary** every time.

- USER: Yes, but it is extra work for us to make all the supporting plans for a new Itinerary, so we don't want to reroute unless the change necessitates it.

- DEVELOPER: Oh. Then we'll have to add some functionality to the **Route Specification**. Then, whenever you change anything in the **Specification**, we'll see if the Itinerary still satisfies the **Specification**. If it doesn't, we'll have the **Routing Service** regenerate the **Itinerary**.

## Scenario 2: Domain model enriched to support discussion



# In the Code

- Domain: Time and Date
- Problem: How to generates due date based on the work date.
  - Rule: payment is due 30 days after the end of the month in which the work is done.

## November

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|
| | | | | | | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| 30 | | | | | | |

## December

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 | | | |

# In the Code
# (A)

```
1  Date workDate = //...;
2  Calendar calendar = Calendar.getInstance();
3  calendar.set(workDate.getYear(), workDate.getMonth() - 1, 1); // Month starts from 0.
4  calendar.set(Calendar.DATE, calendar.getActualMaximum(Calendar.DATE));
5  calendar.add(Calendar.DATE, 30);
6
7  Date dueDate = calendar.getTime();
8
```

# In the Code
# (B)

```
1   CalendarDate workDate = CalendarDate.of(/* ... */);
2   Duration thirtyDays = Duration.days(30);
3   CalendarDate dueDate = workDate.month().end().plus(thirtyDays);
4
```

# What DDD emphasizes on?

- The binding of Model
- The Iterative improvement of the Model
- The expressiveness of the Model

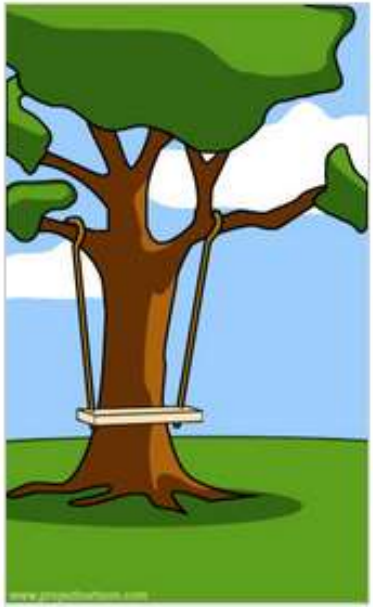
# What DDD emphasizes on?

- The binding of Model
- The Iterative improvement of the Model
- The expressiveness of the Model

# The binding of Model

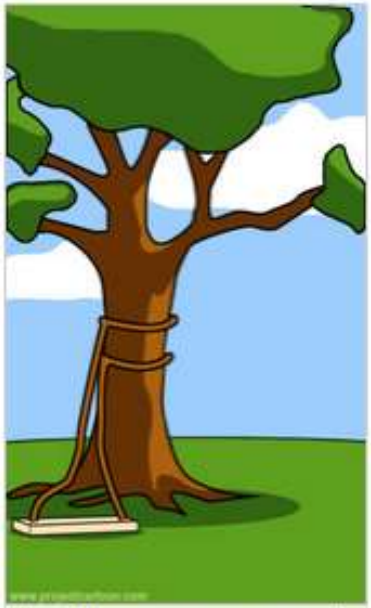
- Remember Sapir-Whorf Hypothesis?

How the customer explained it



How the project leader understood it

How the analyst designed it



How the programmer wrote it

What the beta testers received

What the customer really needed

# What DDD emphasizes on?

- The binding of Model
- The Iterative improvement of the Model
- The expressiveness of the Model

# The iterative improvement of the Model

- Remember the unknown-unknown zone?
  - The only way to know what's in there is by continuous learning.
  - And then applied it to the Model.

# What DDD emphasizes on?

- The binding of Model
- The Iterative improvement of the Model
- The expressiveness of the Model

# The expressiveness of the Model

- It’s like the Time and Code example.
- Readable is good, but expressive is better.

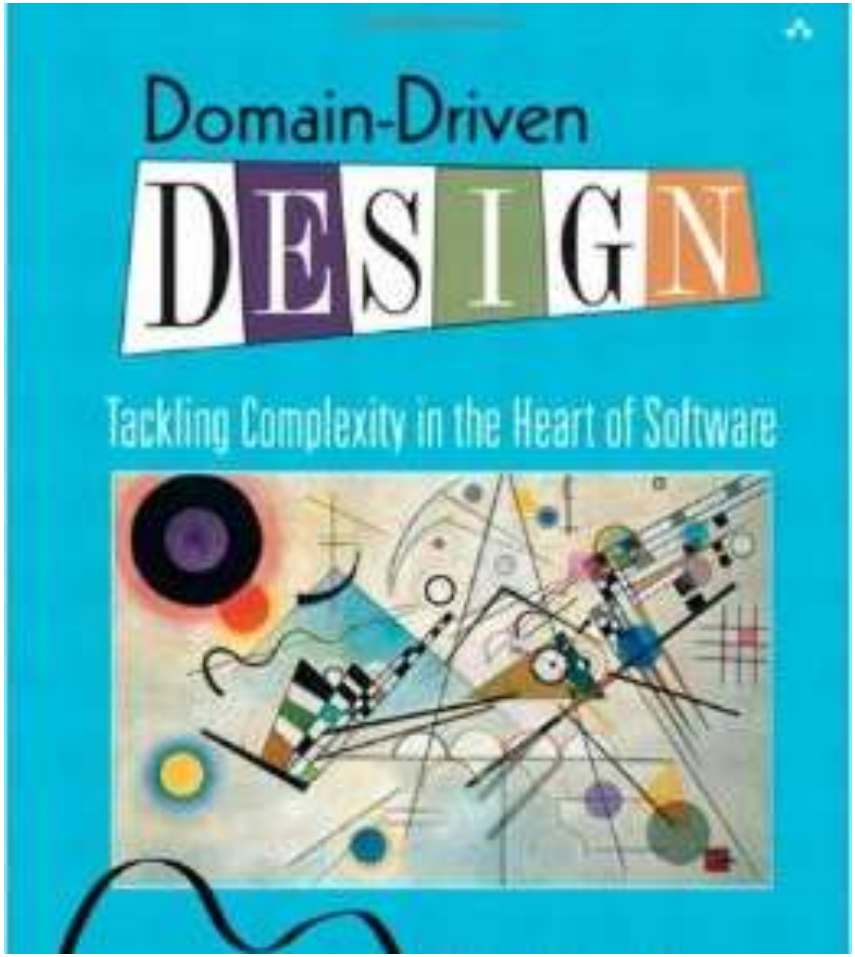
# The expressiveness of the Model

- But there are better examples
  - You already use them!
- The “holy grail” of Domain-Driven Design is Domain Specific Language
  - HTML + CSS (Domain: document representation on the web)
  - SQL (Domain: RDMS manipulation)
  - Regex (Domain: String processing)
  - etc

# When to use DDD

- When the domain is non-trivial (or even complex).
  - To-Do List doesn’t count ;)
- When the developer has access to the Domain Expert.

# So, what is DDD and why?




- Domain-Driven Design: **Tackling Complexity in the Heart of Software**.

# So, what is DDD and why?

- The heart of software is its ability to solve domain-related problem for the user. (Evans, 2003)

- Two types of complexity:
  - Essential complexity
  - Accidental complexity

# Accidental Complexity

- Twitter

# Essential Complexity

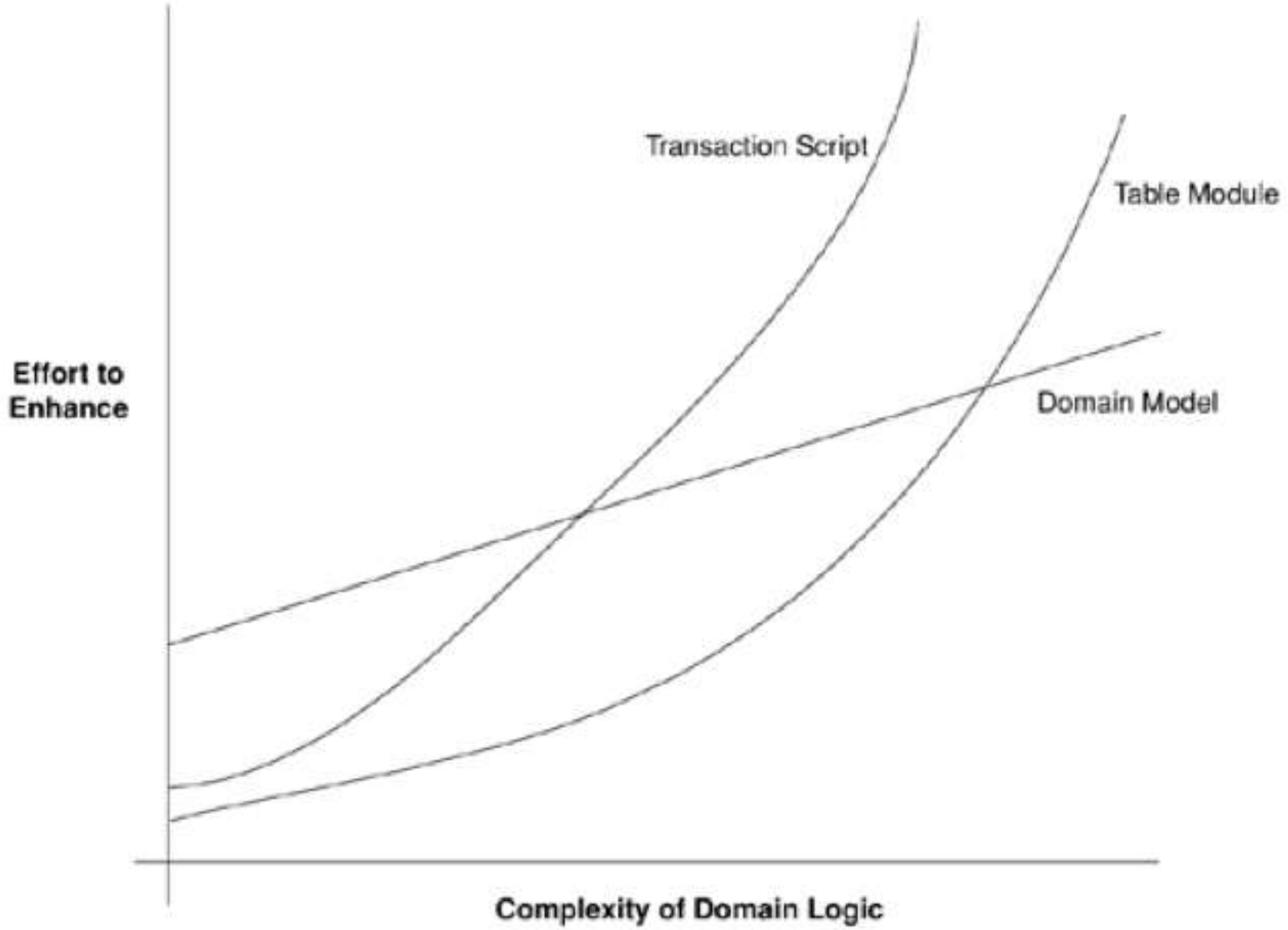
- Financial Software
- Your software? ;)

# So, what is DDD and why?

- Why DDD?
  - Because we need to tackle complexity on the domain.
- What is DDD?
  - A set of thinking tools to tackle complexity on the domain.

# The Bad News

- DDD takes a long time
  - Remember the map? It takes hundreds of years.
  - Many many times of refinements.
    - Think of constantly refactoring.


Transaction Script
Table Module
Domain Model
Effort to Enhance
Complexity of Domain Logic

- “Your problem, of course, is to figure out where on that x axis your application lies. The good news is that I can say that you should use a Domain Model whenever the complexity of your domain logic is greater than 7.42. The bad news is that nobody knows how to measure the complexity of domain logic.” (Fowler, 2003)

- “People are overly interested about some of the DDD patterns like Entities, Value objects and Events, Domain Events, these things have a certain manifestation in the object-oriented world, but the core of Domain-Driven Design is the Ubiquitous Languages.” (Evans, 2003)