

AGILE ARCHITECTURE

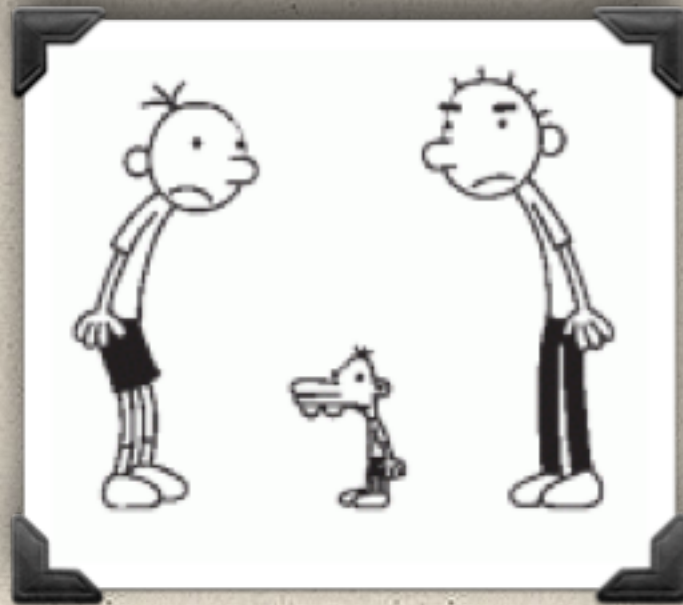
How Worse Can Be Better

Mina Boström Nakićenović

ABOUT ME



Me



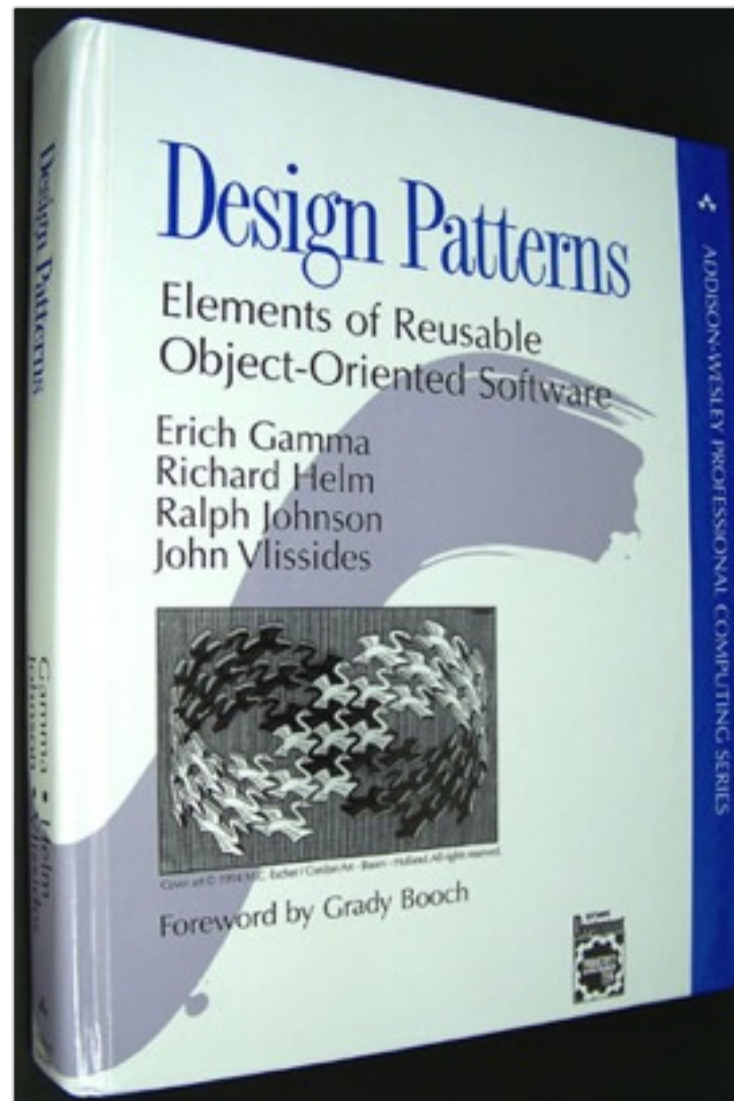
My children

Thanks to him!



My husband

REFERENCES



Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need

MY FAVORITE REFERENCE

although the MDA philosophy might be the right solution for the system?" Instead of feeling as Proust's character Swan, who said: "I've wasted the best years of my life being in love with a woman, who wasn't even my type" [20], the best years of the architect's life should not be wasted in trying to apply the full scale multi-model MDA, if it is unacceptable for the organization. Through the application of agile principles a pragmatic MDD approach could be produced and used instead. The pragmatic MDD approach relaxes the recommendations defined by the OMG's standards. In our case-study the MDA's principles were simplified by merging the PIM and PSMs to one model expressed in a custom XML dialect. In this way the MDD became less abstract, simpler and applicable in our organization. The drawback of our MDD approach was that the standard MDA tools couldn't be used and our own tools had to be developed instead. But considering the short time needed for developing our tools, this was a price worth paying. Although maintenance of our tools is required, it is a continuous process done in steps, so that the long MDA starting curve is skipped. Our approach introduced the MDD in a stepwise way, so it could fit within the short time-frame given for the project. A rough estimate of the time spent for this project is 200 developer hours spread out on a calendar period of 6 months. By using agile principles the learning curve and introduction gap of MDD methods and tools were avoided.

8. References

- [1] SunGard, www.sungard.com
- [2] TNP SDK documentation: SunGard Front Arena
- [3] Len Bass, Paul Clements, Rick Kazman: Software Architecture in Practice. Addison Wesley, 2003.
- [4] James McGovern, Scott Ambler, Michael Stevens: A practical guide to Enterprise Architecture. Prentice Hall PTR, 2003.
- [5] Jean Bezivin: Object to Model paradigm change with the OMG MDA initiative, <http://rangirola.essi.fr/cours/mda/03-bezivin.ppt>
- [6] MDA, www.omg.org/mda
- [7] AgileManifesto, www.agilemanifesto.org
- [8] Stockburger, www.psychstat.missouristate.edu/introbook/ibk04m.htm
- [9] Ron Jeffries, Ann Anderson, Chet Hendrickson: ExtremeProgramming. Addison Wesley, 2001.
- [10] GCC-XML, www.gccxml.org/HTML/index.html
- [11] Ray Carroll, Claire Fahy, Elyes Lehtihet, Sven van der Meer, Nektarios Georgalas, David Cleary: Applying the P2P paradigm to management of large-scale distributed networks using Model Driven Approach, Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP Volume, Issue , 3-7 April 2006 Page(s):1 - 14.
- [12] Anneke Kleppe, Jos Warmer: Explained: The Model Driven Architecture Promise. Addison Wesley, 2003.
- [13] R. Gabriel: Worse is Better
- [14] Stephen J. Mellor, Kendall Scott, Axel Ott: Dark Horse: MDA Distilled- Principles of Model-driven architecture. Addison Wesley, 2004.
- [15] David S. Frankel: Model Driven Architecture - Applying MDA to Enterprise Computing. Wiley, 2003.
- [16] Andrew Hunt, David Thomas: The Pragmatic programmer. Addison Wesley, 2000.
- [17] Stefan Tilkov: A critique of Fowlers MDA critique, www.innoq.com/blog/st/2004/02/a_critique_of_fowlers_mda_crit.html
- [18] Jon Kern: Pragmatic MDA - 3 keys to software development success, www.glsec.org/files/Kern_Jon_GLSEC_PragmaticMDA.ppt
- [19] Bran Selic: Reflection on 30+ years as a software developer, Software Engineering Education & Training, 18th Conference on, 18-20 April 2005, Page(s):5 - 5
- [20] Bran Selic: The Pragmatics of Model-Driven Development, IEEE Software, Vol. 20 (5) 2003.
- [21] Marcel Proust: A la recherche du temps perdu - Un amour de Swan. Paideia 2007.

[13] R. Gabriel: Worse is Better

RICHARD GABRIEL
www.dreamsongs.com

"Worse Is Better" 1990

"Worse Is Better Is Worse" 1991

"Is Worse Really Better?" 1992

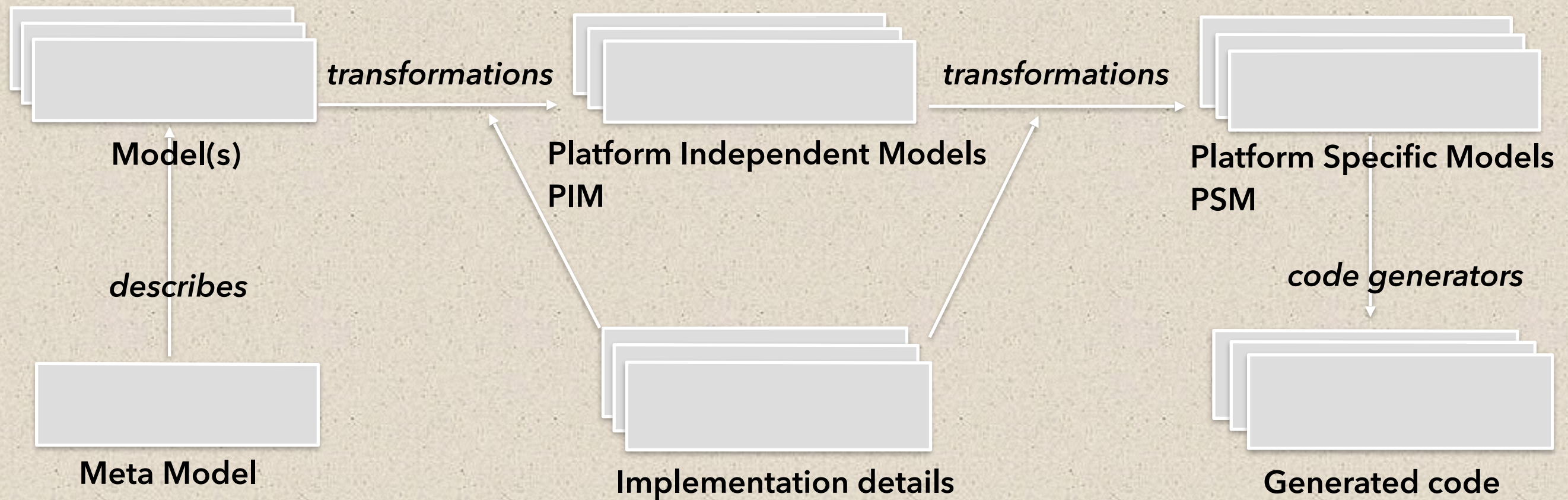
"Back to the Future: Is Worse (Still) Better?" 2000

"Back to the Future: Worse (Still) Is Better" 2001

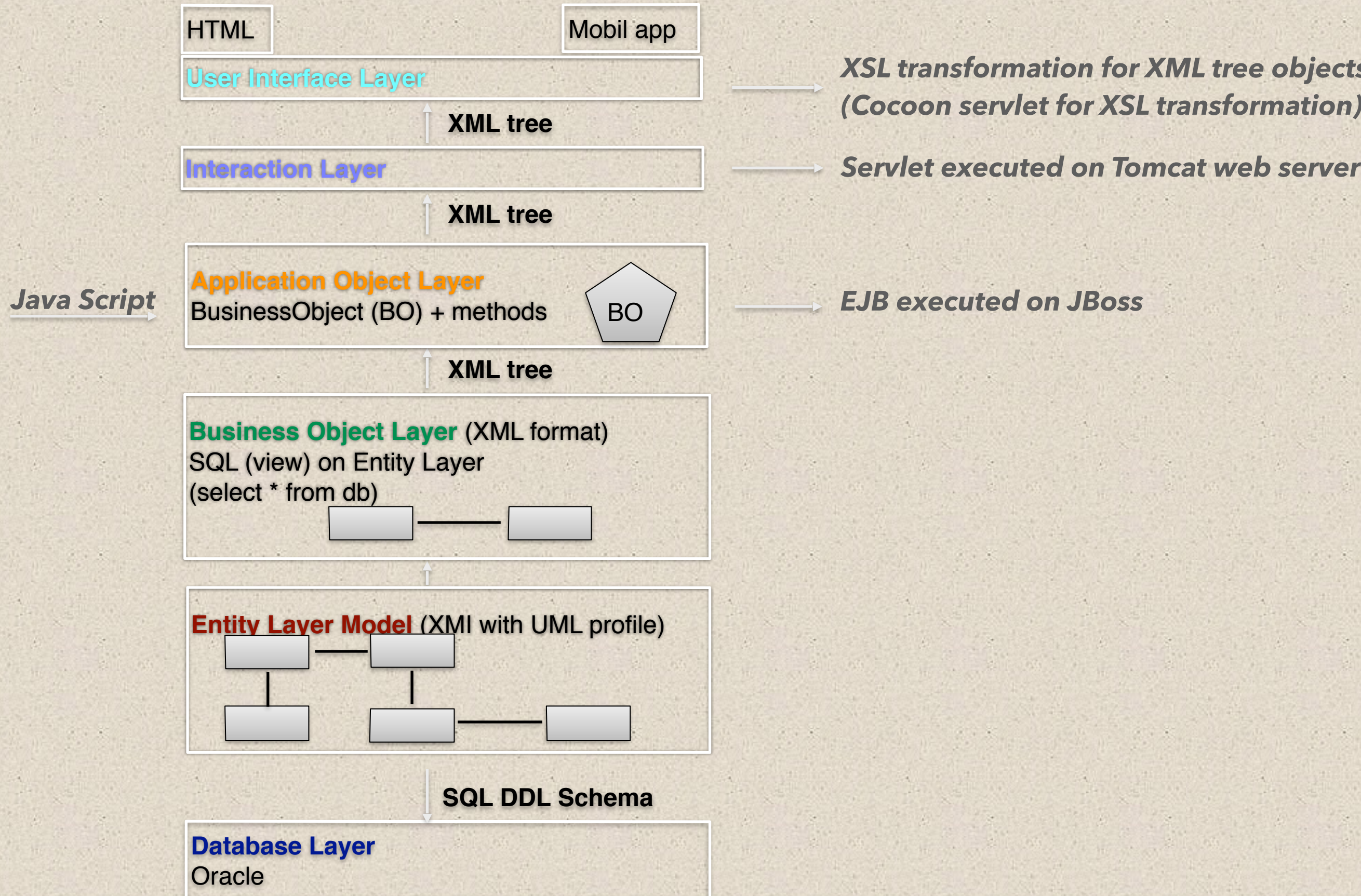
My Model-Driven Development Story

Company A (2001)

MODEL-DRIVEN DEVELOPMENT (MDD)



DATABASE DRIVEN WEB/MOBILE APPLICATION



COMPANY A - SUMMARY

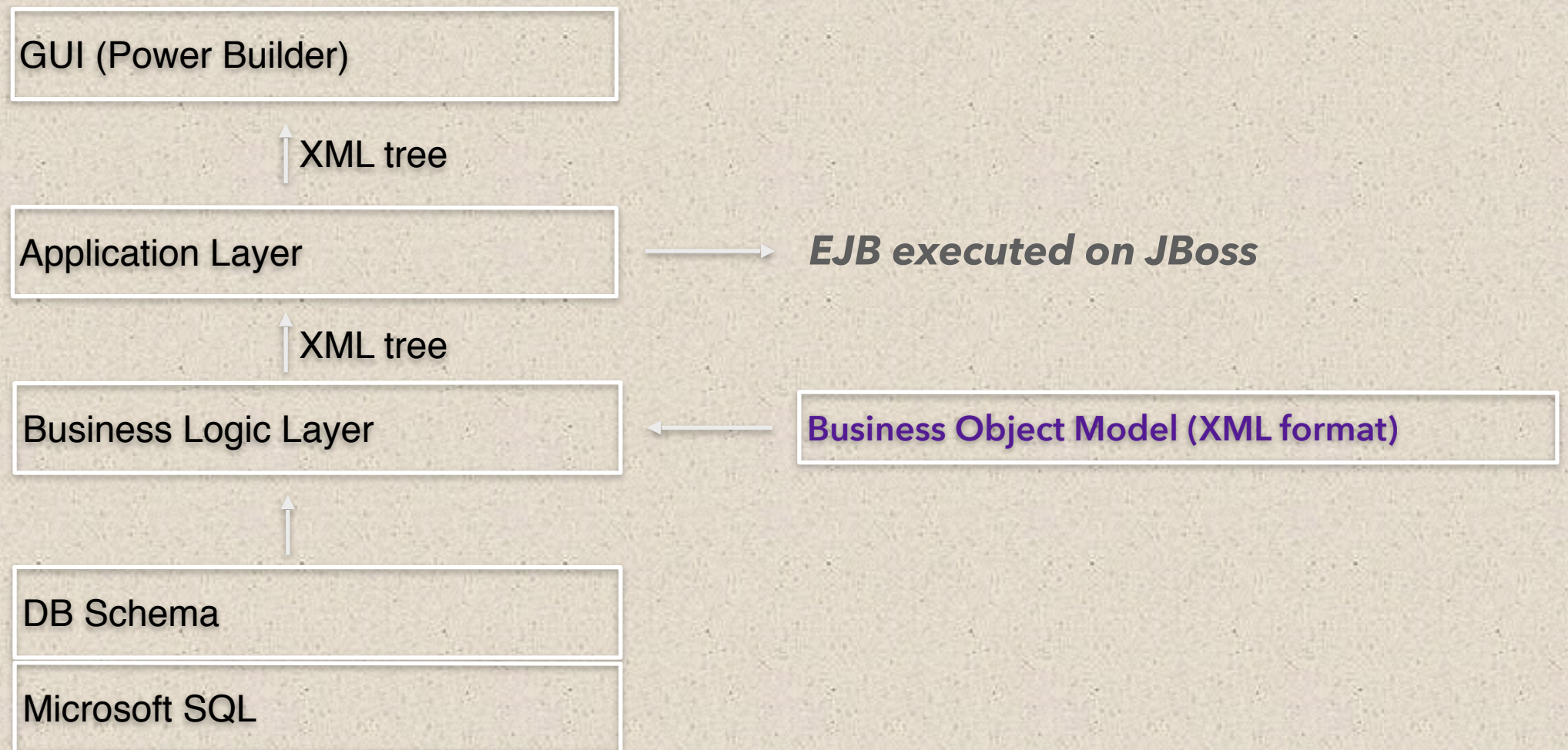
- 6 Layers Architecture => a strong Separation of Concerns (SoC)
- Model-Driven Development + SoC => "Build for change" (*Embrace change*)
- We used very sophisticated tools and very sophisticated models, so that they can reflect the realities of various deployment platform. If the architecture is flexible, it will be attractive for many customers.
- Application developers need to know just Java Script (*Simplicity*)
- Almost 100% JUnit test coverage within each layer
- Pair-programming

But

- Framework was never stable enough
- It was very complicated to implement even simple functions/queries
- Framework developers worked separated from application developers
- Only one test-field customer using our framework to run a simple application
- While we were waiting on customers, we were constantly building our framework, trying to think from the customer's point of view.

Company B (2003)

Financial Back-office System



COMPANY B - SUMMARY

- 3 Layers Architecture => a good SoC
- Model-Driven Development + SoC => "Build for change" (*Embrace change*)
- We used advanced tools and models
- Almost 100% JUnit test coverage within each layer

But

- It was very complicated to implement even simple functions/queries
- Framework was never stable enough
- Framework developers worked separated from application developers
- No customers, we were building the framework/applications 2 years

MODEL-DRIVEN DEVELOPMENT - CONCLUSION

It cannot succeed since:

- It assumes a big upfront design
- Starting curve is usually too long
It takes a long time before getting a Return Of Investment (ROI)
- It always becomes too complex and cannot be stable
- MDD idea - theoretically good, inapplicable in practice
"The difference between the theory and the practice is much greater in practice than in theory"

Model-Driven Development - Never again!!!

SUNGARD

Front Arena

SUNGARD FRONT ARENA

Sungard

- large, world-wide financial services software company
- 25 000 customers in 70 countries.

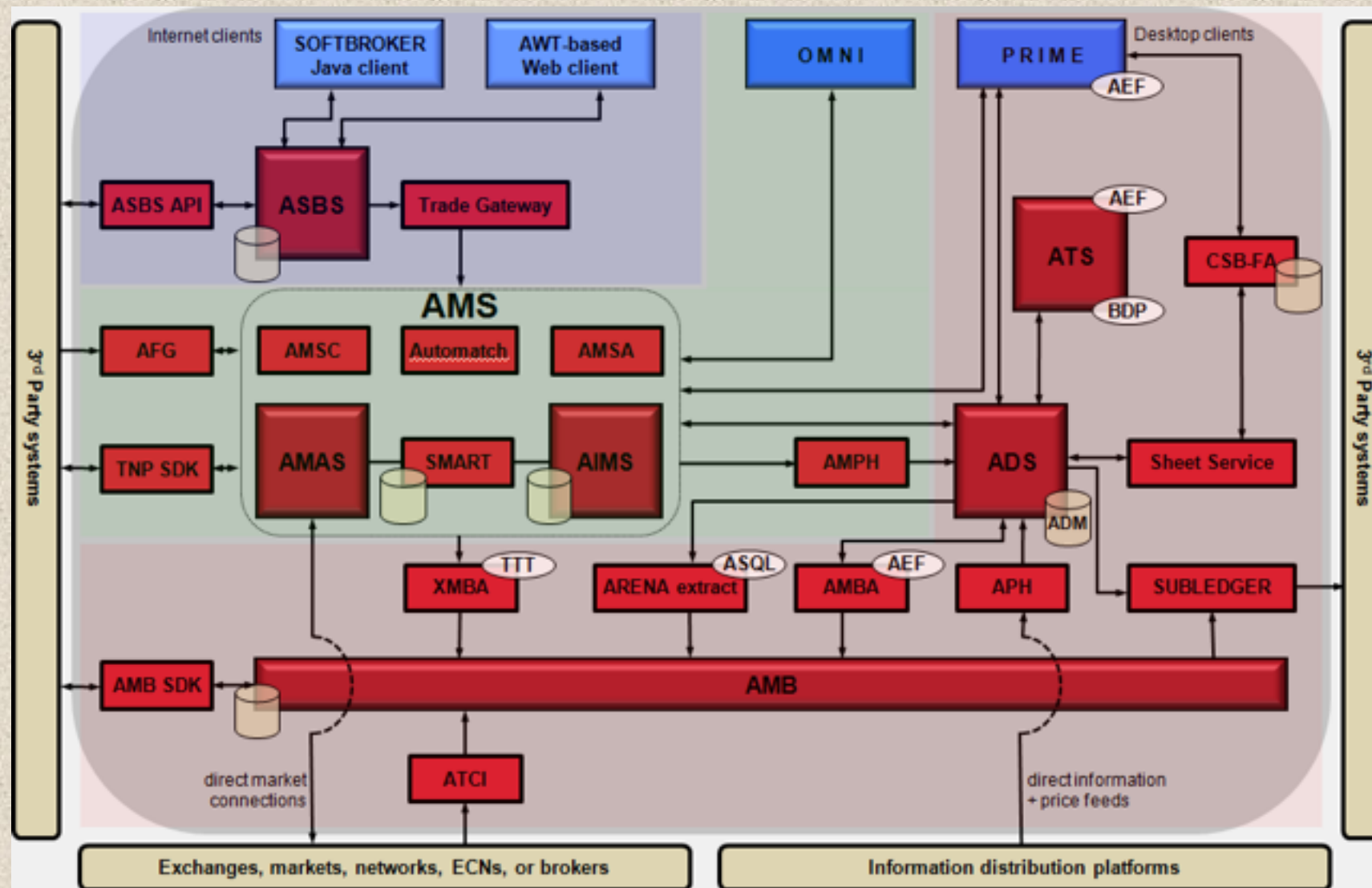
Front Arena (FA) system

- order management and deal capture on electronic exchanges

Market access

- based on a client/server architecture

SYSTEM ARCHITECTURE



Transaction Network Protocol - TNP

- An internal financial message protocol for transaction handling
It is built on top of TCP/IP
- Used for the clients/servers components communication

TNP PROTOCOL MESSAGE DEFINITIONS

TNP message definitions have only C++ format

Components that use the TNP protocol:

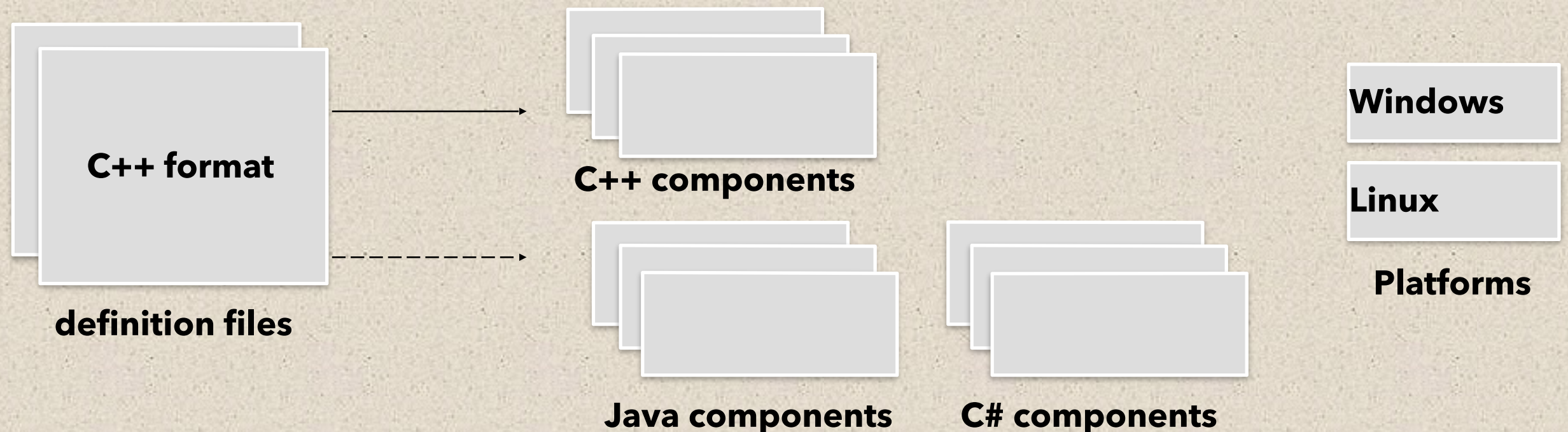
- implemented in different programming languages (C++, Java, C#)
- must use the same TNP message definition.

The same definition is defined in different C++ files:

- none contained the complete definition
- some information was duplicated.

Architecture had several critical problems:

- An unnecessary duplication of the data definition (risk for data inconsistency)
- A programming language dependency (one of the definition files was a C++ file)



MANAGEMENT LIMITATIONS

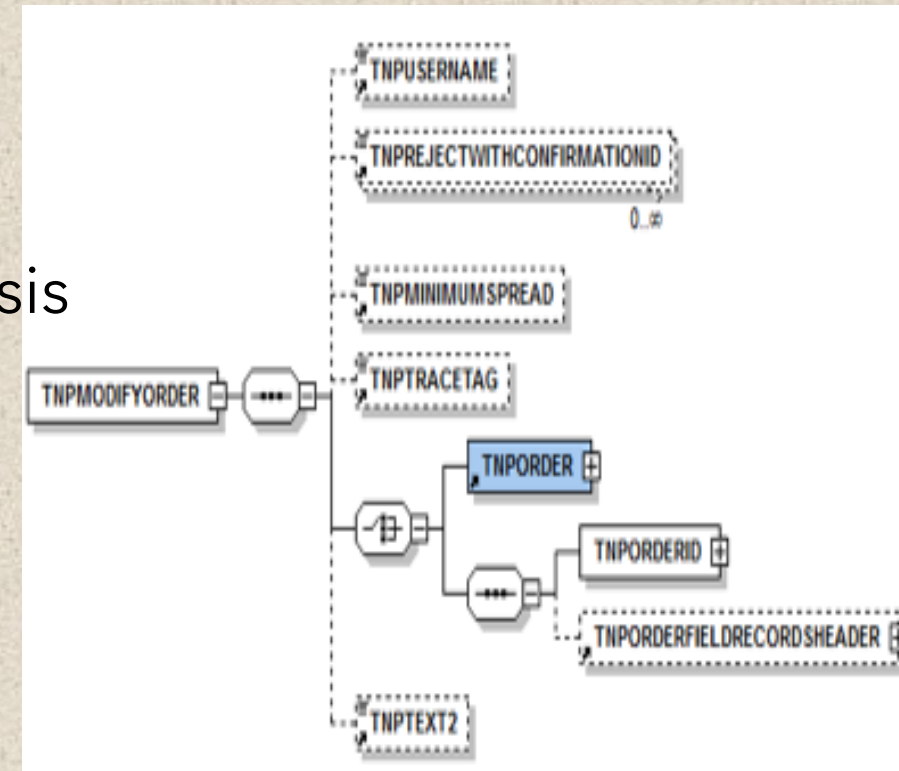
- Short time-frame
- No investment in change management
- Only already used or open-source tools



NEW ARCHITECTURE - ATTRIBUTE DRIVEN DESIGN

TNP Message definitions

- There are more than 1000 TNP messages defined
- Message definitions updates frequently, on the daily basis
- All developing teams are updating it



Quality Attributes

- Modifiability
- Usability
- Interoperability
- Reusability
- Efficiency

Architectural Drivers

- Centralization
- Data consistency
- Programming language independency
- Presentation

ARCHITECTURAL DECISION

MDD is the most suitable solution for our problem

But haven't I said: "MDD never again"?!?!



Edvard Munch "Scream"

ANALYZING FAILED MDD ARCHITECTURES

- **Starting curve was too long**

We have never achieved to reach production and get a ROI

- **We had a lot of waste**

We built complex architectures, although most parts of them were never used.

- **We built for future**

We were focused on creating perfect technical solutions, believing that they can "embrace change" i.e. that it can provide an architecture which is easy to change.

SOFTWARE ARCHITECTURE DEFINITION

1. The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

Philippe Kruchten, Grady Booch

2. Software architecture is the art and science of designing and delivering valuable technology strategy.

International Association of Software Architects (IASA)

3. Software architecture encompasses the set of significant decisions about the organization of a software system including the selection of the structural elements and their interfaces by which the system is composed; behavior as specified in collaboration among those elements; composition of these structural and behavioral elements into larger subsystems; and an architectural style that guides this organization.

Len Bass, Paul Clements, Rick Kazman

CAN MDD BE AGILE AND LEAN ?



"MDD is a such big design upfront and how can it be agile?!"

What is a heart of the MDD?

- Accelerated development, achieved by the centralized architecture and automatic generations.
- Separations of concerns both on technical and business aspects, making the system architecture flexible for the changes.

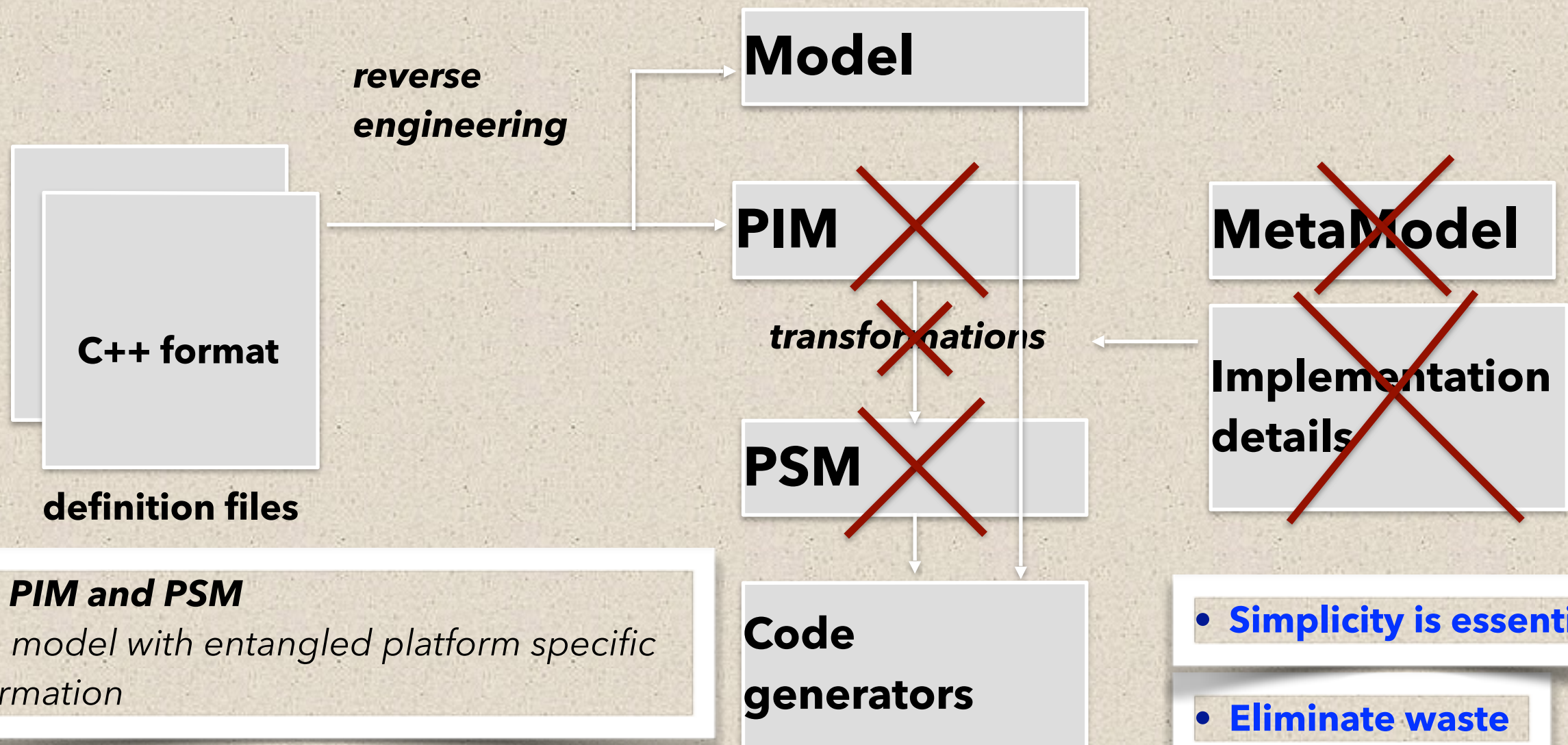
What are the main goals of the Agile and Lean principles?

- To develop qualitative and no cost-effective solutions and deliver them quickly
- Welcome changing requirements even late in the project
("decide in the last responsible moment")
- Deliver business value
- Eliminate waste
- See the whole

Both address the same goals:

- Making systems less sensitive to frequent changes
- Accelerated development
- Eliminating wastes

APPLYING AGILE AND LEAN PRINCIPLES



- **No PIM and PSM**

One model with entangled platform specific information

- **XML format is good enough**

A tradeoff between XML simplicity and UML's abstraction benefits

- **Code generators - XSL transformations**

Common standard for C++, Java, .NET developers

- **Simplicity is essential**

- **Eliminate waste**

- **Don't build for tomorrow**

- **Organizational maturity**

- **Collective code ownership**

WHAT DID WE GET?

"Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away."



Antoine de Saint-Exupery

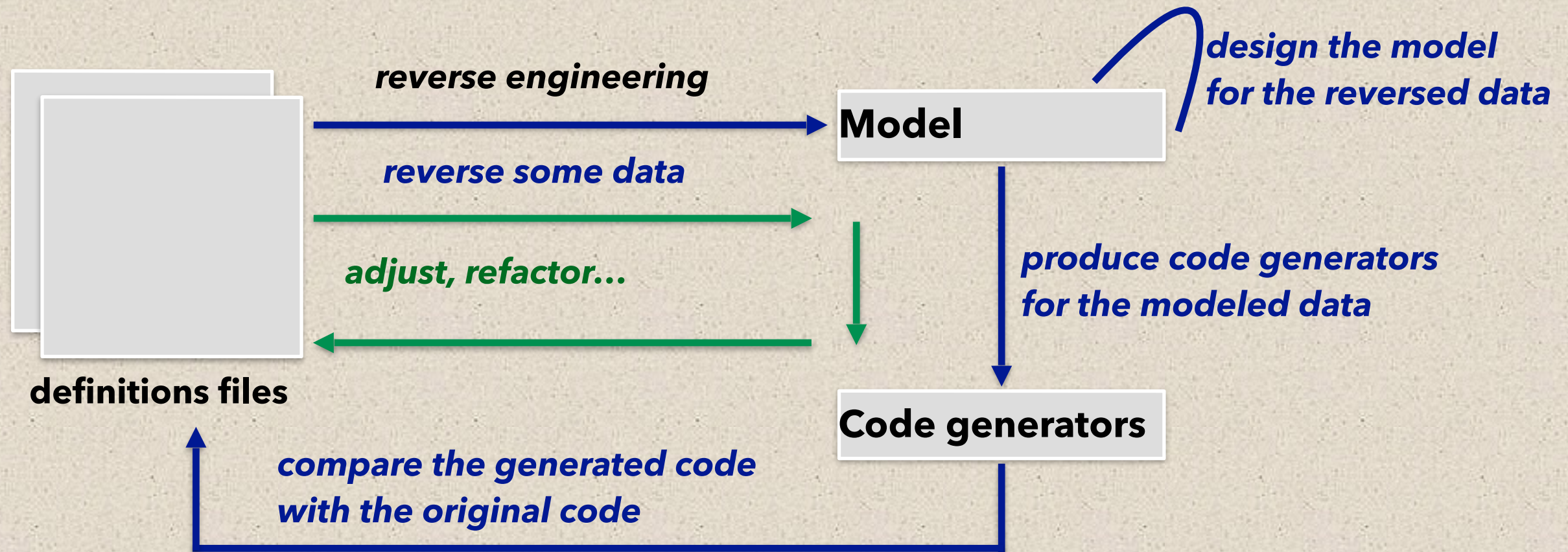
"Simple does not mean simplistic." James Coplien

"Good developers add values and remove code", Thomas Riha

HOW TO PRODUCE THE ARCHITECTURE?

- To build the right product
- To build the product in a right way

HOW TO PRODUCE THE ARCHITECTURE?



- Model first? There is a risk that we specify things which will never be used - waste!
- Reverse the existing data first? But we cannot proceed without the model design.
- Forward engineering (code generation) first? But we cannot proceed without the model design.

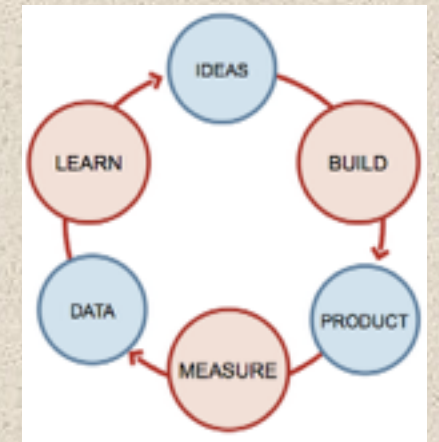
"Deliver working software frequently"

- Do all three activities within the same sprint! Set a spike for each round-trip scenario.

LEAN STARTUP METHODOLOGY (ERIC RIES)

Build-Measure-Learn Loop

Build a feature, measure whether it delivers, learn from the results



Minimum Viable Product (MVP)

Version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort.

MVP is a bare-bones product that includes just enough features to allow useful feedback from early adopters.

6

"The concept known as "worse is better" holds that in software making it is better to start with a minimal creation and grow it as needed.", R. Gabriel (1989)

AGILE AND LEAN MODEL-DRIVEN DEVELOPMENT

- We modeled just what we needed, according to the input from the definition files. Hence the model was **"pulled by the request"** and a **"potential waste was not introduced"**.
"All businesses have costs. Waste is optional", Stephen Parry
- We generalized model only for the repeating artifacts. We did not put any effort to model things that appeared just once. We **"did not build for tomorrow"**.
- After one spike/round tripping we learnt how to improve things for the next cycle. **"Learn first"**
- We could see, early in the project, which parts of the MDD had to be improved. **"Build quality in", "Improve constantly"**
- Consequently, MDD's long starting curve was shortened. We produced a piece of working software in each first sprint. **"Working software is a primary measure of progress"**

OUR AGILE OR "GOOD-ENOUGH" MODEL

```
RTY name="RTY_MARKETSERVERATTRIBUTES" value="0179" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="MarketServerAttributes"/>
RTY name="RTY_SETTLEMENTPRICE" value="017A" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="SettlementPrice"/>
RTY name="RTY_DEALID" value="017D" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="DealId"/>
RTY name="RTY_LISTID" value="017E" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="ListId"/>
RTY name="RTY_CHARGECURVENAME" value="017F" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="ChargeCurveName"/>
RTY name="RTY_ORDERID" value="0180" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="OrderId"/>
RTY name="RTY_ACCOUNTID" value="0196" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="AccountId"/>
RTY name="RTY_REFERENCE" value="0197" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="Reference"/>
RTY name="RTY_IMORDERID" value="01AB" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="IMOrderId"/>
RTY name="RTY_IMINFOHEADER" value="01AF" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="IMInfoHeader"/>
RTY name="RTY_EMINFOHEADER" value="01B0" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="EMInfoHeader"/>
RTY name="RTY_GETORDERATTRIBUTES" value="01B1" type="Child record" sdk="yes" section="Record Types (RTYs)" nameCS="GetOrderAttributes"/>
RTY name="RTY_MEMBER" value="01B4" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="Member"/>
RTY name="RTY_LISTNAME" value="01B5" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="ListName"/>
RTY name="RTY_SEQNORECOVERY" value="01B6" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="SeqNoRecovery"/>
RTY name="RTY_ORDERBOOKID" value="01B7" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="OrderBookId"/>
RTY name="RTY_TICKINTERVAL" value="01B8" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="TickInterval"/>
RTY name="RTY_FREETEXTLISTITEM" value="01B9" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="FreeTextListItem"/>
RTY name="RTY_FREETEXTFORMAT" value="01BA" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="FreeTextFormat"/>
RTY name="RTY_DELETEORDERID" value="01BB" type="Child record" sdk="yes" section="Record Types (RTYs)" nameCS="DeleteOrderId"/>
RTY name="RTY_HINTINFO" value="01BC" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="HintInfo"/>
RTY name="RTY_ORDERHINT" value="01BD" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="OrderHint"/>
RTY name="RTY_INSTRUMENTTYPE" value="01BE" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="InstrumentType"/>
RTY name="RTY_DEAL" value="01BF" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="Deal"/>
RTY name="RTY_ALTERNATIVELISTNAME" value="01C3" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="AlternativeListName"/>
RTY name="RTY_ORDERBOOKSTATUS" value="01C4" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="OrderBookStatus"/>
RTY name="RTY_PRICEDETAIL" value="01C5" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="PriceDetail"/>
RTY name="RTY_NEWDEAL" value="01C6" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="NewDeal"/>
RTY name="RTY_PARTY" value="01C7" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="Party"/>
RTY name="RTY_MARKETPRICE" value="01C9" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="MarketPrice"/>
RTY name="RTY_DISCLAIMERTEXT" value="01CA" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="DisclaimerText"/>
RTY name="RTY_ORDER" value="01CB" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="Order"/>
RTY name="RTY_ORDERINFO" value="01CC" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="OrderInfo"/>
RTY name="RTY_BROKERID" value="01CD" sdk="yes" section="Record Types (RTYs)" type="Child record" nameCS="BrokerId"/>
RTY name="RTY_MOVEORDERID" value="01CF" type="Child record" sdk="yes" section="Record Types (RTYs)" nameCS="MoveOrderId"/>
```

Merged PIM and PSM?
"Is it an architectural error?!"

C++ memory location?!

MVP CONCEPT

*"Bugs were all over the place, extremely ugly looking, and only the most rudimentary features. While I used to enjoy showing my family what I'd built, I remember thinking: **I never want my family to know I've released this pitiful of a product.**"*

*But it allowed us to test the market and get **feedback**, which allowed us to drop certain features that didn't resonate with our audience, and focus on others that our customers really liked."*



Eric Ries

AFTER SEVERAL BUILD-MEASURE-LOOP CYCLES

- When we finished with the reverse engineering we could release the new architecture and deploy to production.
- As soon as the developers stopped wasting time on the manual updates in the old architecture and started using the new architecture, we started receiving a ROI.
- We got a working architecture which creates a business value.
- We got an ugly but "good-enough" model, which creates a value. We are constantly improving it, with the concurrent receipt of ROI.

WHY DID WE SUCCED THE THIRD TIME?

- By accident?

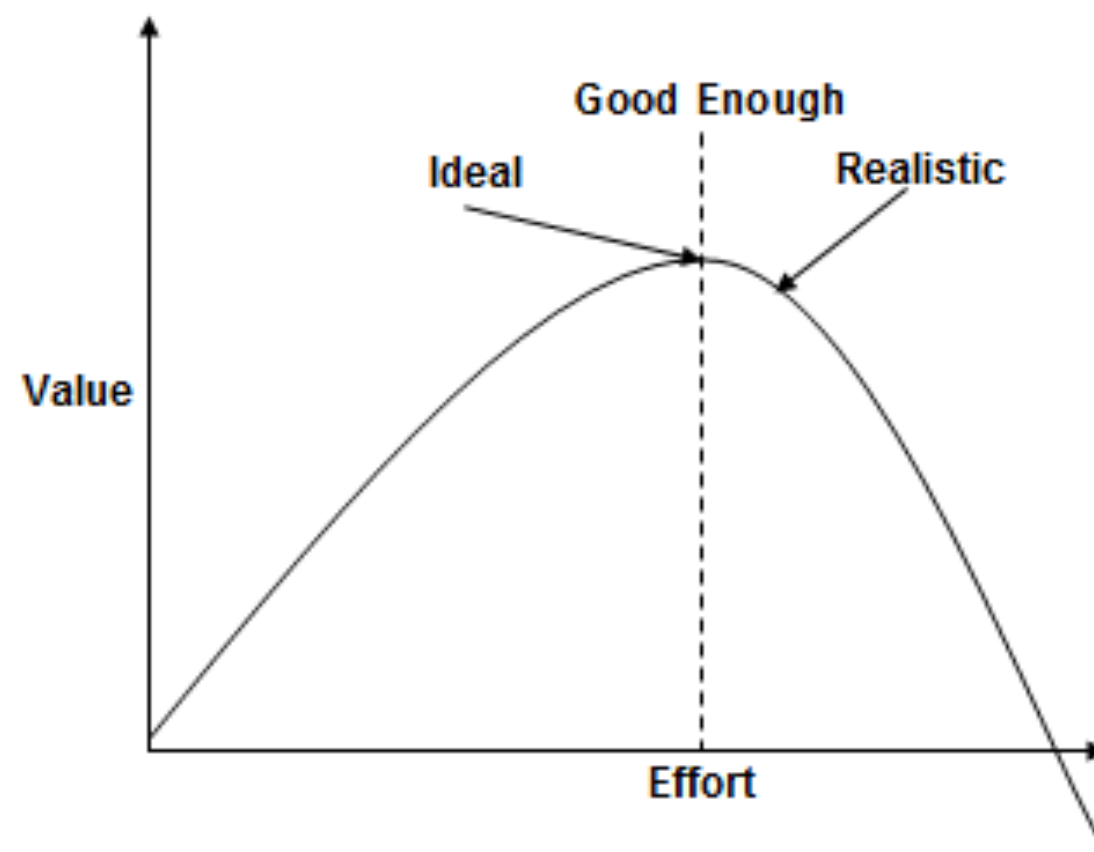
- Third time lucky?

- Because WORSE IS BETTER?

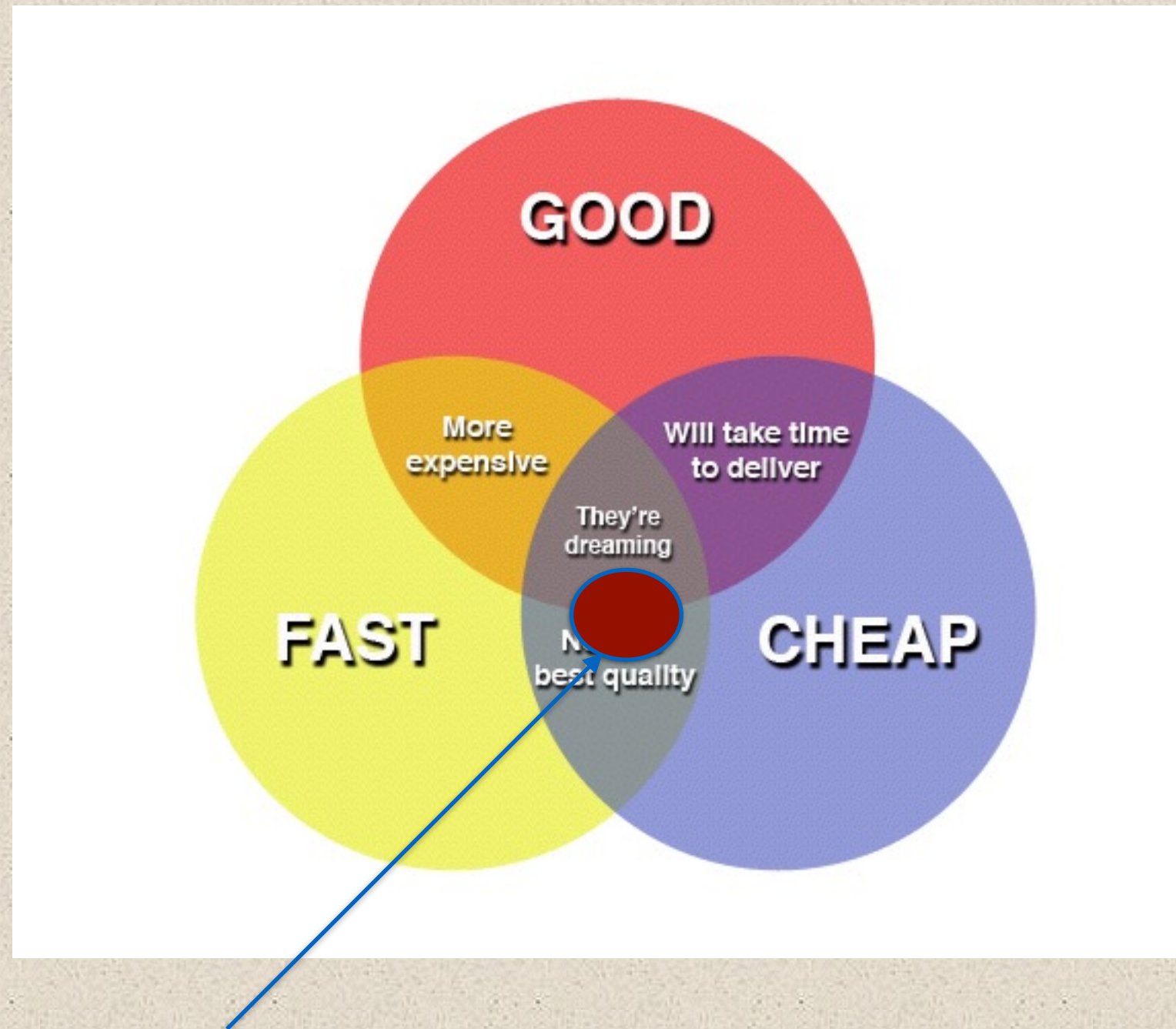
"GOOD-ENOUGH" SOLUTIONS (Scott Ambler)

"Good enough" solution is a practices-based.

"Just barely good-enough" solution, where the fundamental challenge with "just barely good-enough" is situational and therefore the most efficient. Hence such solutions usually create a **biggest business value**.



"GOOD-ENOUGH" SOLUTIONS



GOOD ENOUGH

A "HELLO WORLD" PROGRAM

```
10 PRINT "HELLO WORLD"  
20 END
```

```
program Hello(input, output)  
begin  
    writeln('Hello World')  
end.
```

```
#include <stdio.h>  
void main(void)  
{  
    char *message[] = {"Hello ", "World"};  
    int i;  
  
    for(i = 0; i < 2; ++i)  
        printf("%s", message[i]);  
    printf("\n");  
}
```

```
#include <iostream.h>  
#include <string.h>  
  
class string  
{  
private:  
    int size;  
    char *ptr;  
  
string() : size(0), ptr(new char[1]) { ptr[0] = 0; }  
  
    string(const string &s) : size(s.size)  
    {  
        ptr = new char[size + 1];  
        strcpy(ptr, s.ptr);  
    }  
  
    ~string()  
    {  
        delete [] ptr;  
    }  
  
    friend ostream &operator <<(ostream &, const string &);  
    string &operator=(const char *);  
};  
  
ostream &operator<<(ostream &stream, const string &s)  
{  
    return(stream << s.ptr);  
}  
  
string &string::operator=(const char *chrs)  
{  
    if (this != &chrs)  
    {  
        delete [] ptr;  
        size = strlen(chrs);  
        ptr = new char[size + 1];  
        strcpy(ptr, chrs);  
    }  
    return(*this);  
}  
  
int main()  
{  
    string str;  
  
    str = "Hello World";  
    cout << str << endl;  
  
    return(0);  
}
```

"IS WORSE BETTER?"

"I still can't decide."



R. Gabriel

It is (often)!



Mina

What do you think?