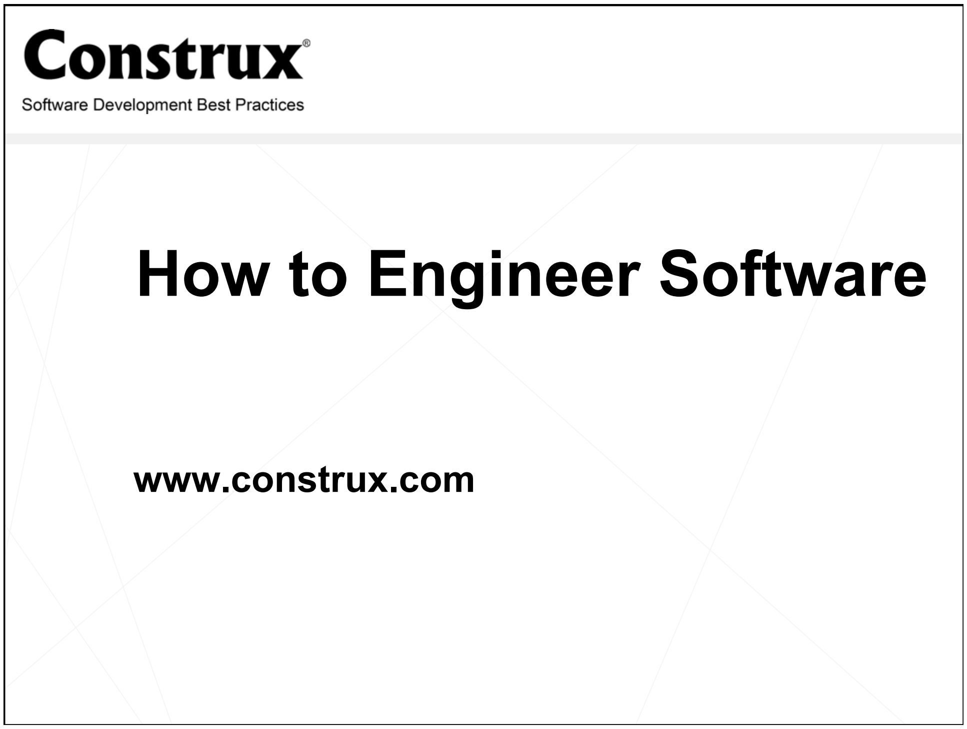
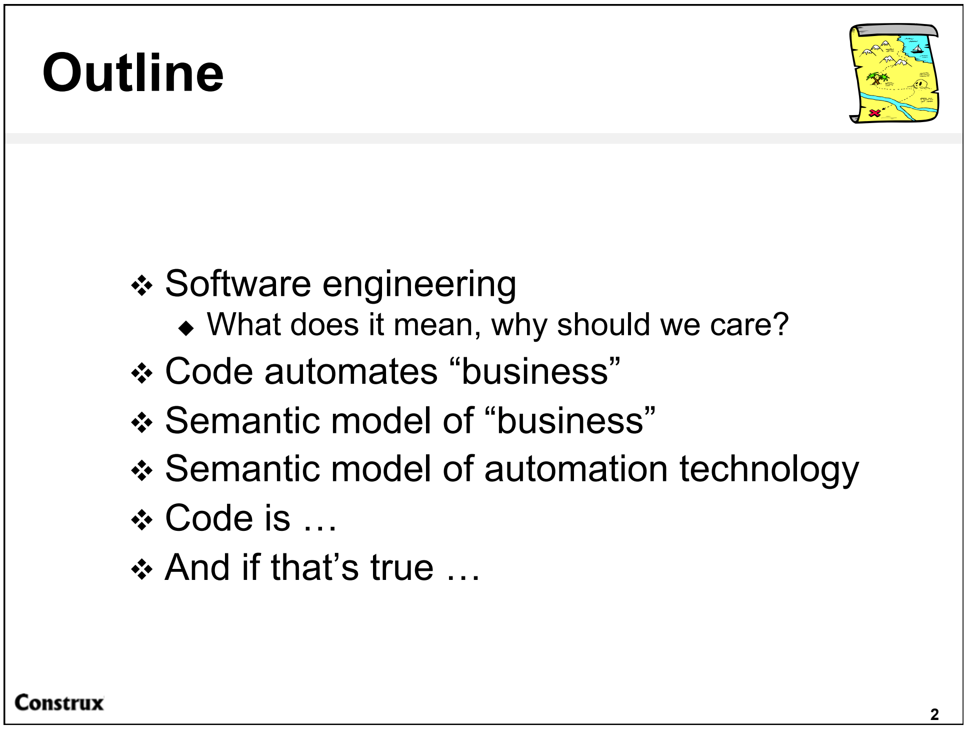


Construx®

Software Development Best Practices

How to Engineer Software

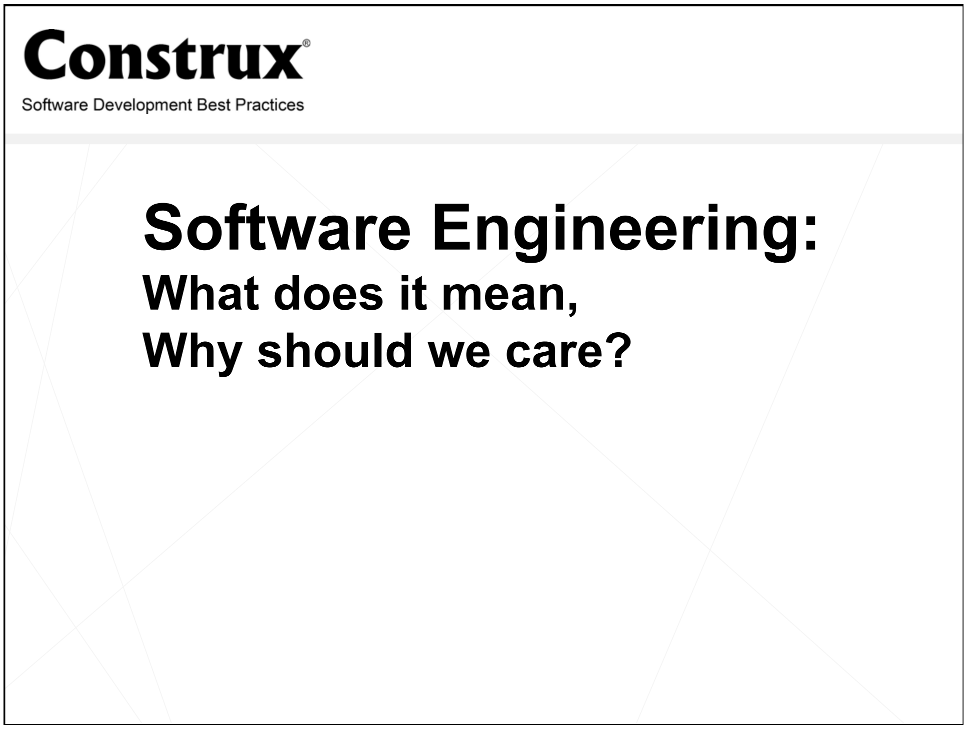
www.construx.com



Outline

- ❖ Software engineering
  - ◆ What does it mean, why should we care?
- ❖ Code automates “business”
- ❖ Semantic model of “business”
- ❖ Semantic model of automation technology
- ❖ Code is …
- ❖ And if that’s true …

Construx

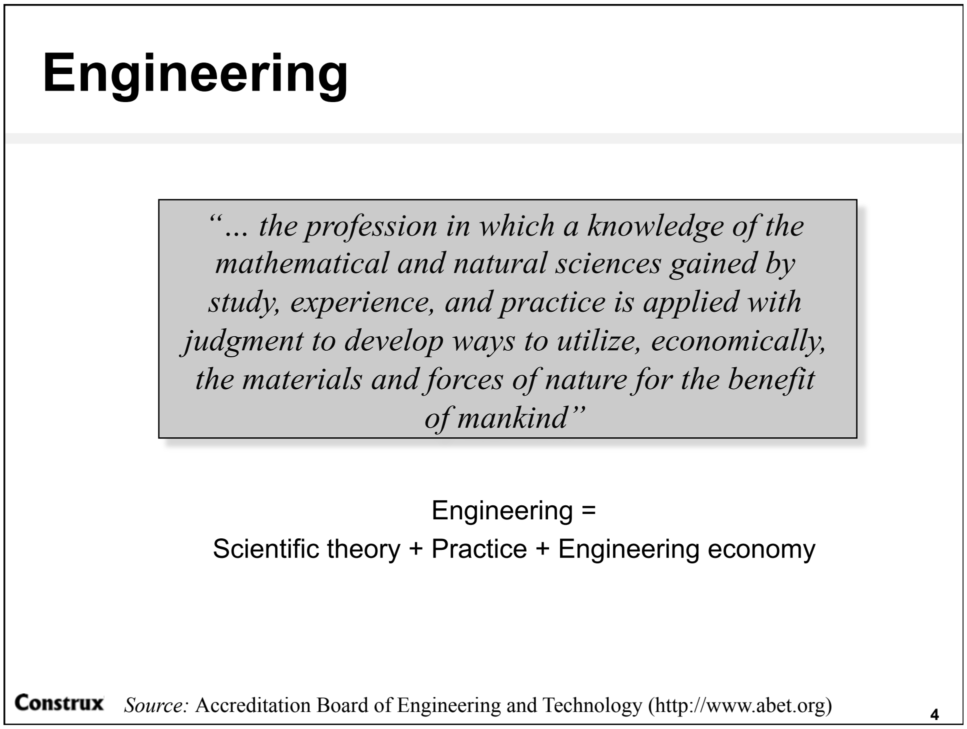


Construx®

Software Development Best Practices

Software Engineering:

What does it mean, Why should we care?



Engineering

> *"… the profession in which a knowledge of the mathematical and natural sciences gained by study, experience, and practice is applied with judgment to develop ways to utilize, economically, the materials and forces of nature for the benefit of mankind"*

Engineering =
Scientific theory + Practice + Engineering economy

Construx *Source:* Accreditation Board of Engineering and Technology (http://www.abet.org)

Software Engineering

> *"… the profession in which a knowledge of the mathematical and computing sciences gained by study, experience, and practice is applied with judgment to develop ways to utilize, economically, computing systems for the benefit of mankind"*

Software engineering =
Computer science + Practice + Engineering economy

Steve Tockey

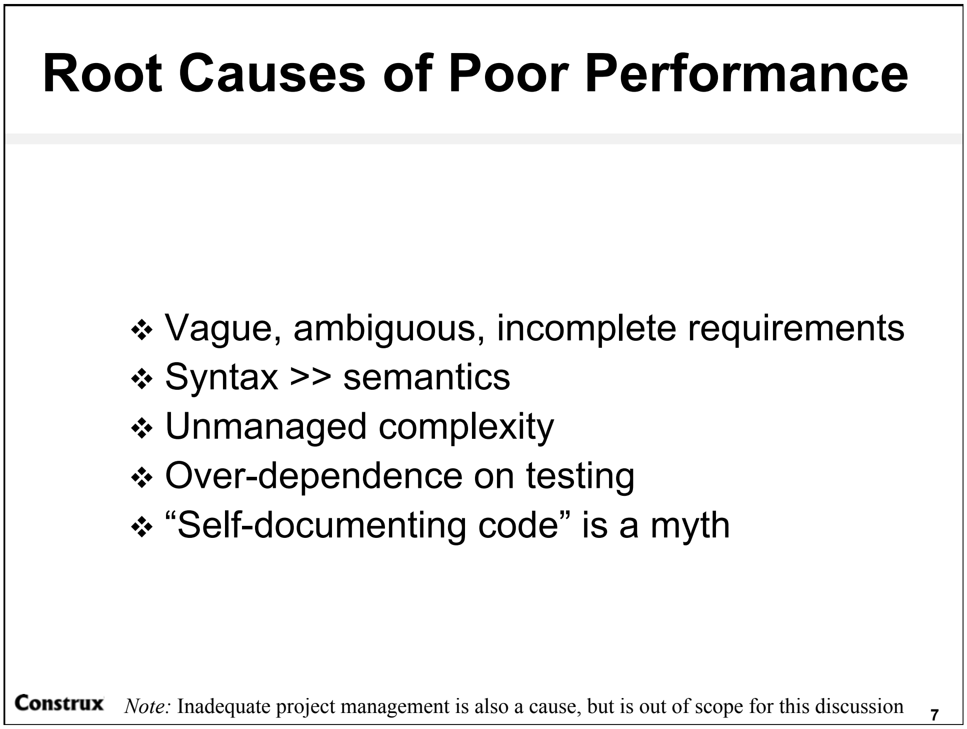
Return on Software

Source: Steve Tockey, *Return on Software*, Addison Wesley, 2005

Why Software Engineering?

- 18% of SW projects fail to deliver any value
- Of projects that deliver, average
  - 42% late
  - 35% over budget
  - 25% under scope
- Along with
  - Unhappy sponsors
  - Frustrated users
  - Team burn out

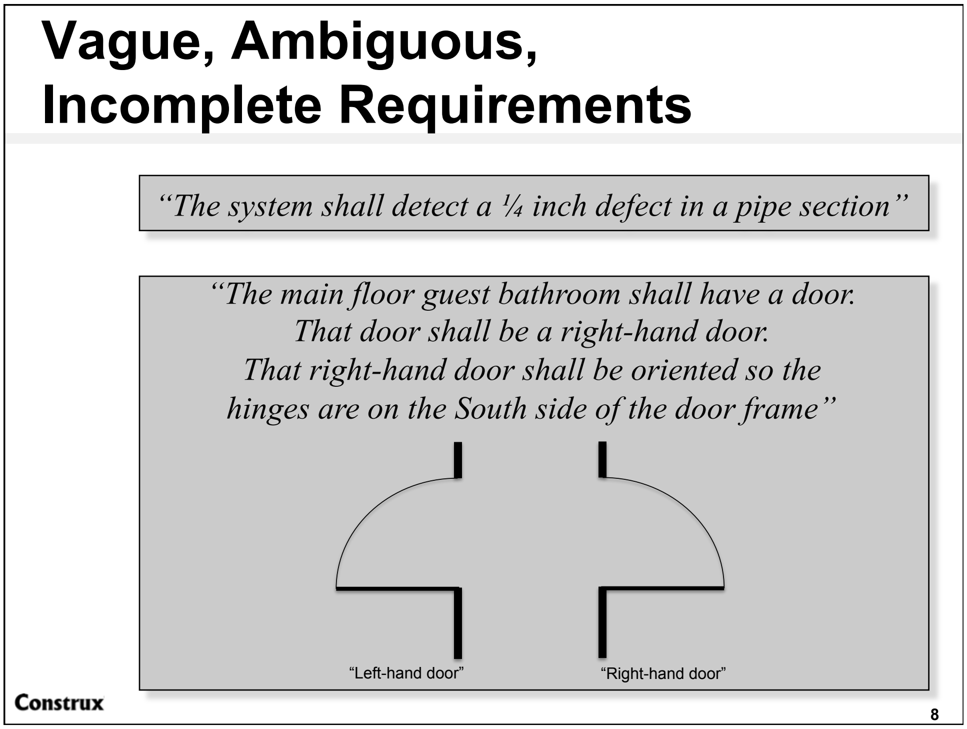
Source: Standish Group CHAOS Report 2013



Root Causes of Poor Performance

- ❖ Vague, ambiguous, incomplete requirements
- ❖ Syntax >> semantics
- ❖ Unmanaged complexity
- ❖ Over-dependence on testing
- ❖ "Self-documenting code" is a myth

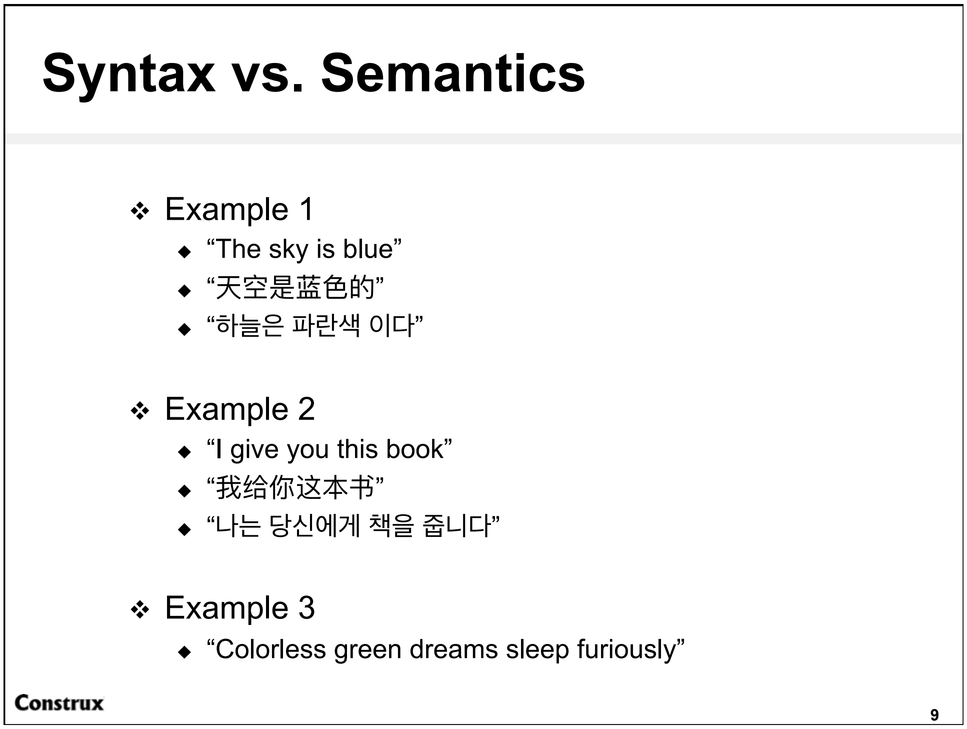
Note: Inadequate project management is also a cause, but is out of scope for this discussion



Vague, Ambiguous, Incomplete Requirements

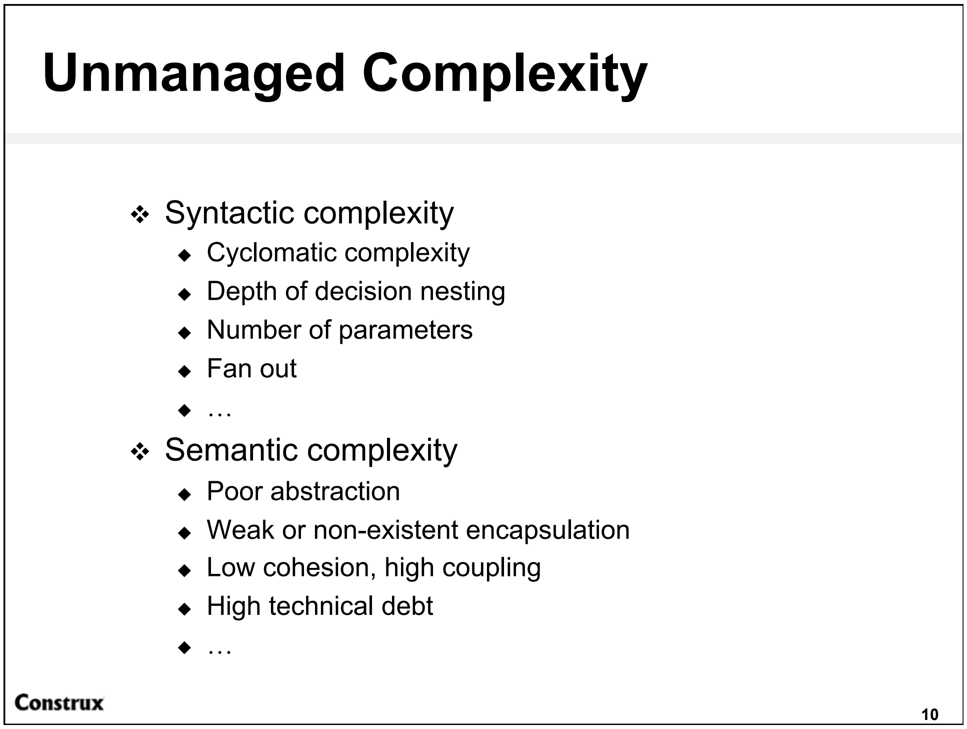
"The system shall detect a ¼ inch defect in a pipe section"

"The main floor guest bathroom shall have a door.
That door shall be a right-hand door.
That right-hand door shall be oriented so the
hinges are on the South side of the door frame"



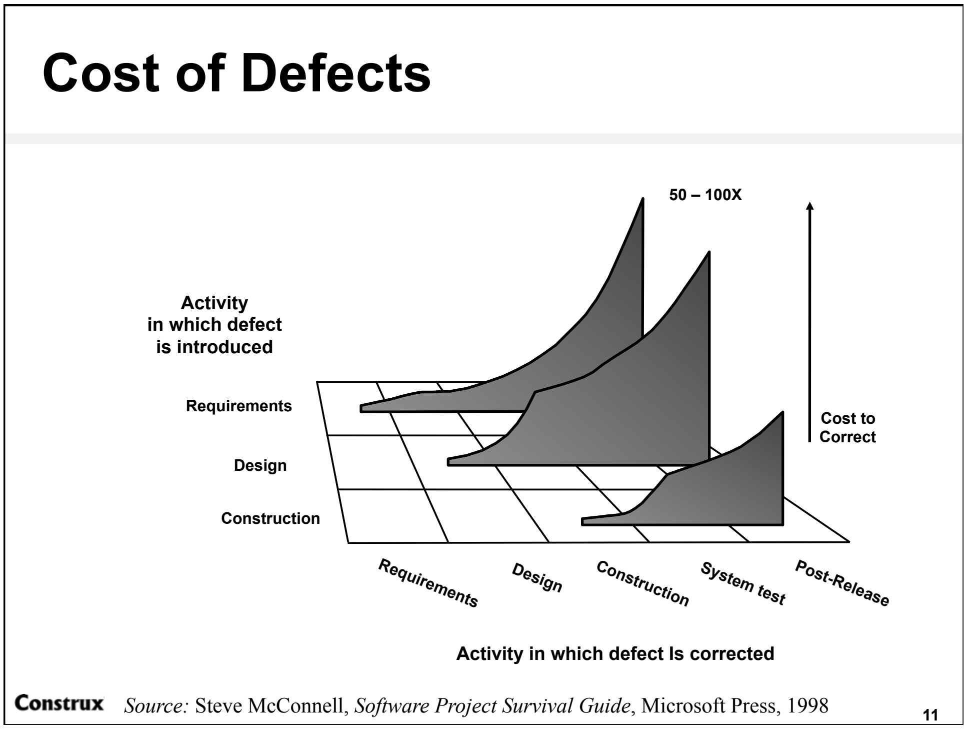
Syntax vs. Semantics

- Example 1
  - “The sky is blue”
  - “天空是蓝色的”
  - “하늘은 파란색 이다”
- Example 2
  - “I give you this book”
  - “我给你这本书”
  - “나는 당신에게 책을 줍니다”
- Example 3
  - “Colorless green dreams sleep furiously”



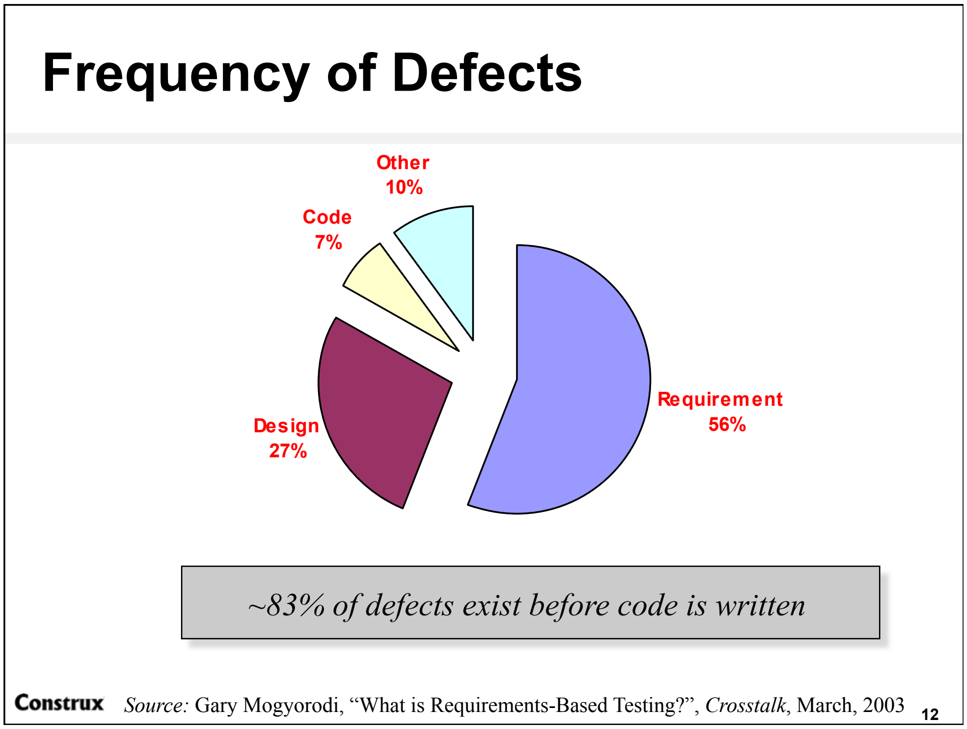
Unmanaged Complexity

- Syntactic complexity
  - Cyclomatic complexity
  - Depth of decision nesting
  - Number of parameters
  - Fan out
  - …
- Semantic complexity
  - Poor abstraction
  - Weak or non-existent encapsulation
  - Low cohesion, high coupling
  - High technical debt
  - …



Cost of Defects

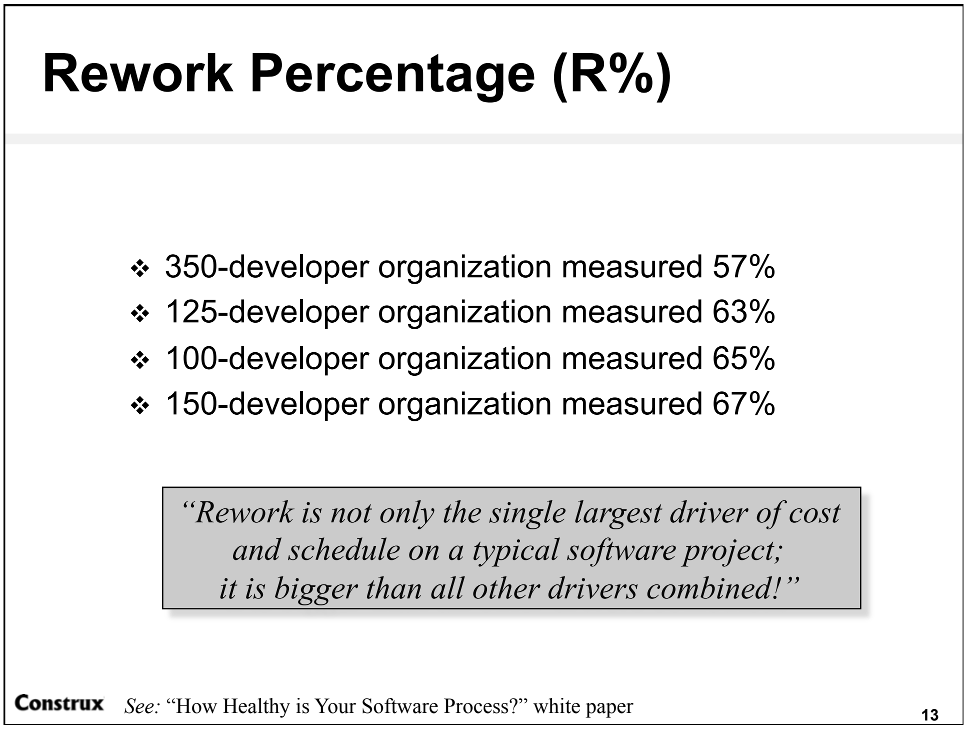
Construx *Source:* Steve McConnell, *Software Project Survival Guide*, Microsoft Press, 1998



Frequency of Defects

~83% of defects exist before code is written

Construx *Source:* Gary Mogyorodi, "What is Requirements-Based Testing?", *Crosstalk*, March, 2003

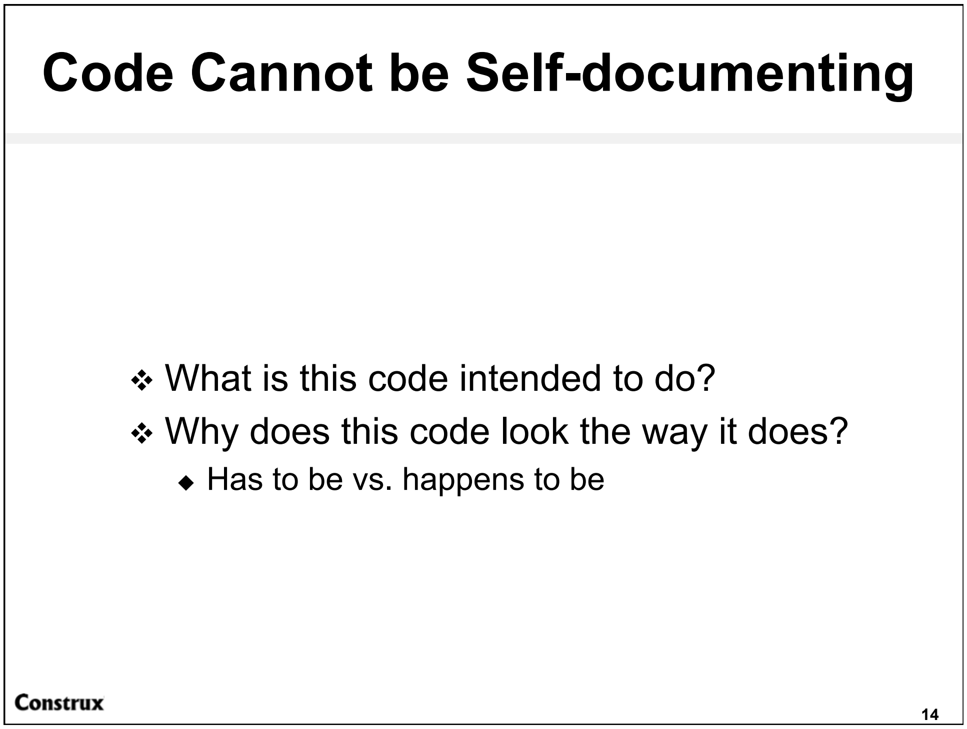


Rework Percentage (R%)

- 350-developer organization measured 57%
- 125-developer organization measured 63%
- 100-developer organization measured 65%
- 150-developer organization measured 67%

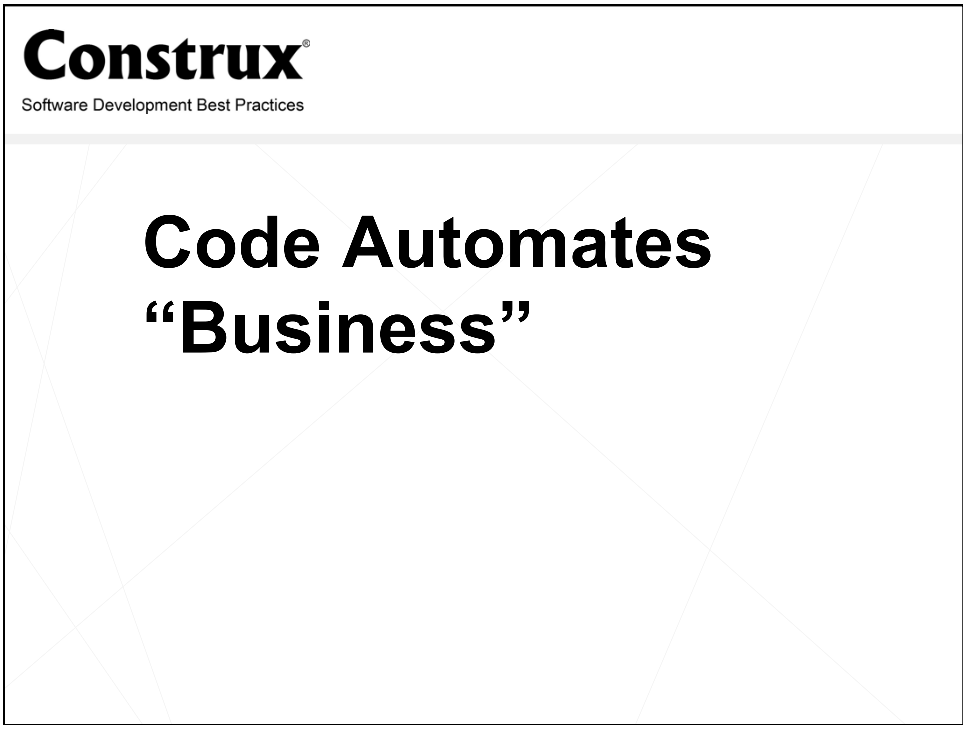
> *"Rework is not only the single largest driver of cost and schedule on a typical software project;*
> *it is bigger than all other drivers combined!"*

See: "How Healthy is Your Software Process?" white paper



Code Cannot be Self-documenting

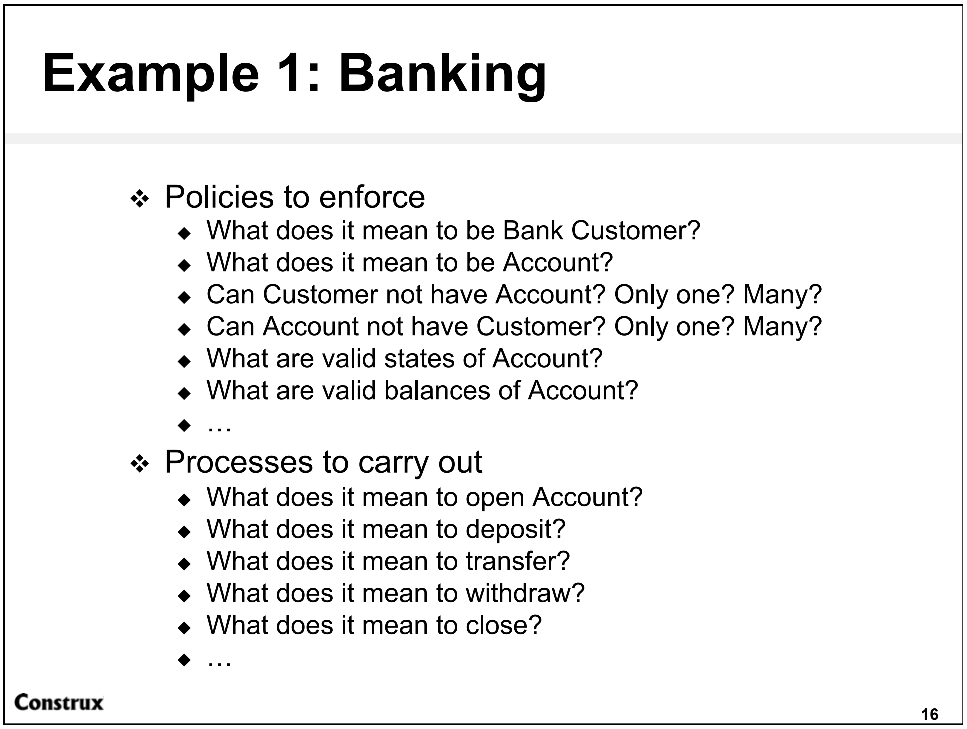
- What is this code intended to do?
- Why does this code look the way it does?
  - Has to be vs. happens to be



Construx®

Software Development Best Practices

Code Automates "Business"



Example 1: Banking

- Policies to enforce
  - What does it mean to be Bank Customer?
  - What does it mean to be Account?
  - Can Customer not have Account? Only one? Many?
  - Can Account not have Customer? Only one? Many?
  - What are valid states of Account?
  - What are valid balances of Account?
  - …
- Processes to carry out
  - What does it mean to open Account?
  - What does it mean to deposit?
  - What does it mean to transfer?
  - What does it mean to withdraw?
  - What does it mean to close?
  - …

Example 2: TCP / IP

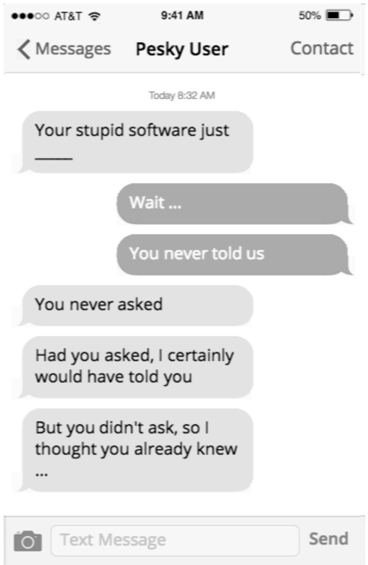
- Policies to enforce
  - What does it mean to be TCP Port?
  - What does it mean to be TCP Connection?
  - Can Port not have Connection? Only one? Many?
  - Can Connection not have Port? Only one? Many?
  - What are valid states of TCP Connection?
  - What are valid IP Addresses for IP Datagram?
  - …
- Processes to carry out
  - What does it mean to Ack Segment?
  - What does it mean to Window probe?
  - What does it mean to fragment IP Datagram?
  - What does it mean to reassemble IP Datagram?
  - What does it mean when Time to live = 0?
  - …

Success Depends on …

> *For software developers to be successful automating someone's business, those developers need to understand that business at least as well as—if not better than—the business experts understand it**

**To the extent that business is being automated*

Dreaded SMS Syndrome



Your stupid software just _____

Wait ...

You never told us

You never asked

Had you asked, I certainly would have told you

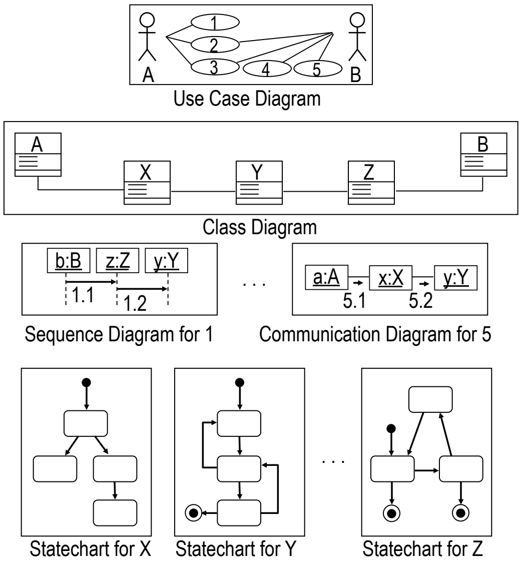
But you didn't ask, so I thought you already knew ...

Construx®

Software Development Best Practices

Semantic Model of “Business”

Semantic Model of “Business”



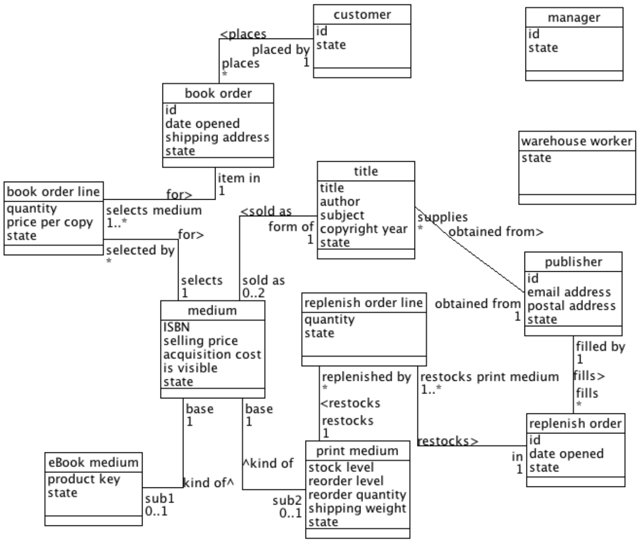
Process:
high level

Policy

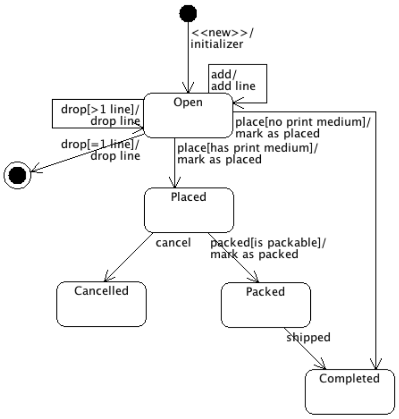
Process:
intermediate level

Process:
detailed level

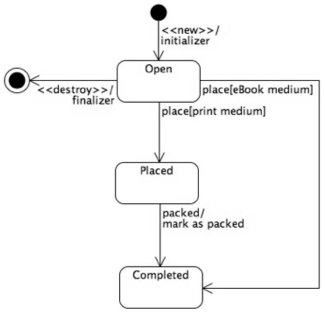
JAL Model Editor: Policy



JAL Model Editor: Detailed Process



book order

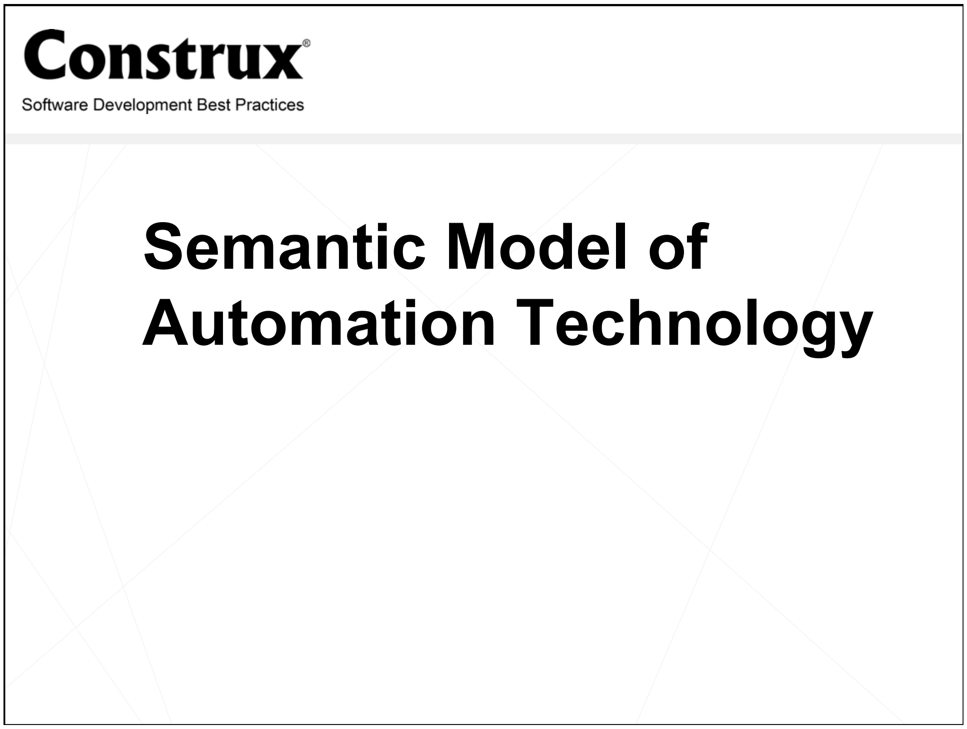


book order line

Avoid Requirements Defects

- Unambiguous
  - Single interpretation derived from computer science, discrete math
- Precise
  - Association multiplicities
  - Attribute ranges
  - Action preconditions, postconditions
  - Generalization completeness
- Concise

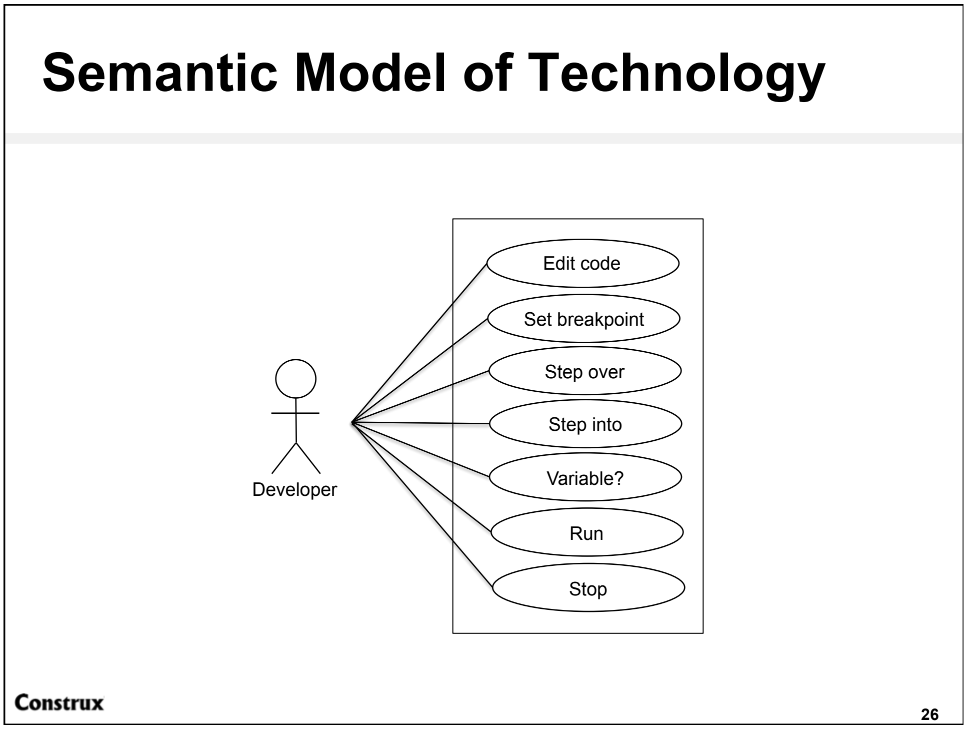
- Completeness guidelines
  - Categories of use cases
  - All events in all states
- Checklists
- Simulation



Construx®

Software Development Best Practices

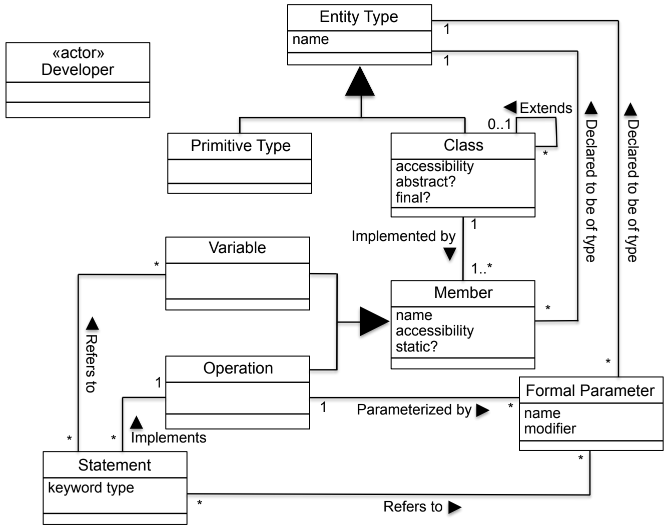
Semantic Model of Automation Technology



Semantic Model of Technology

Construx

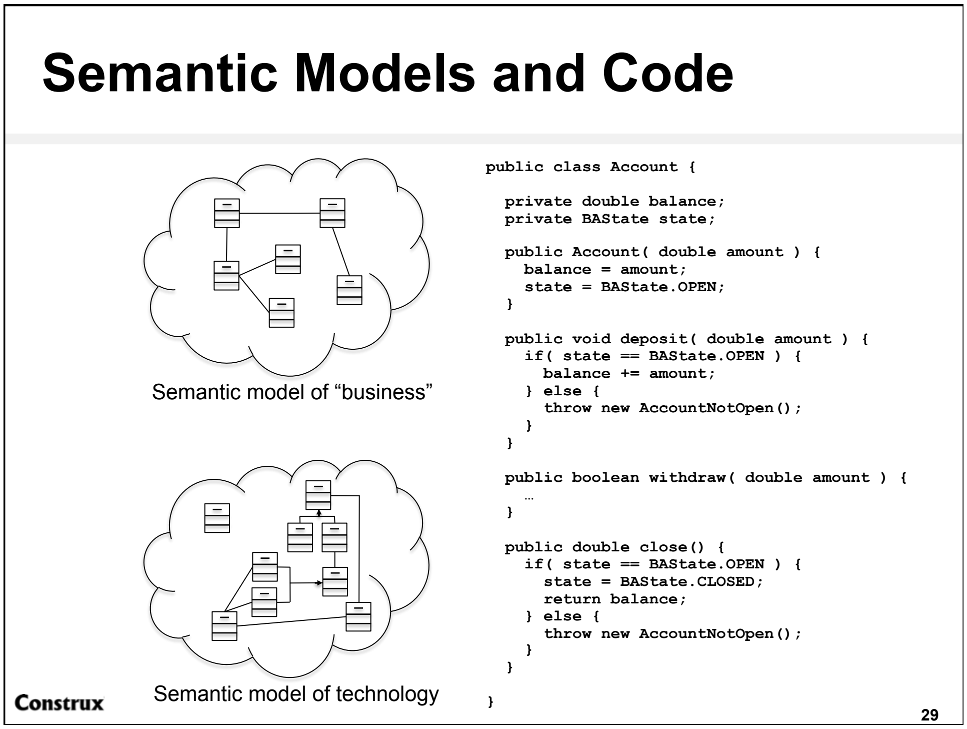
Semantic Model of Technology (cont)



Construx®

Software Development Best Practices

Code is …



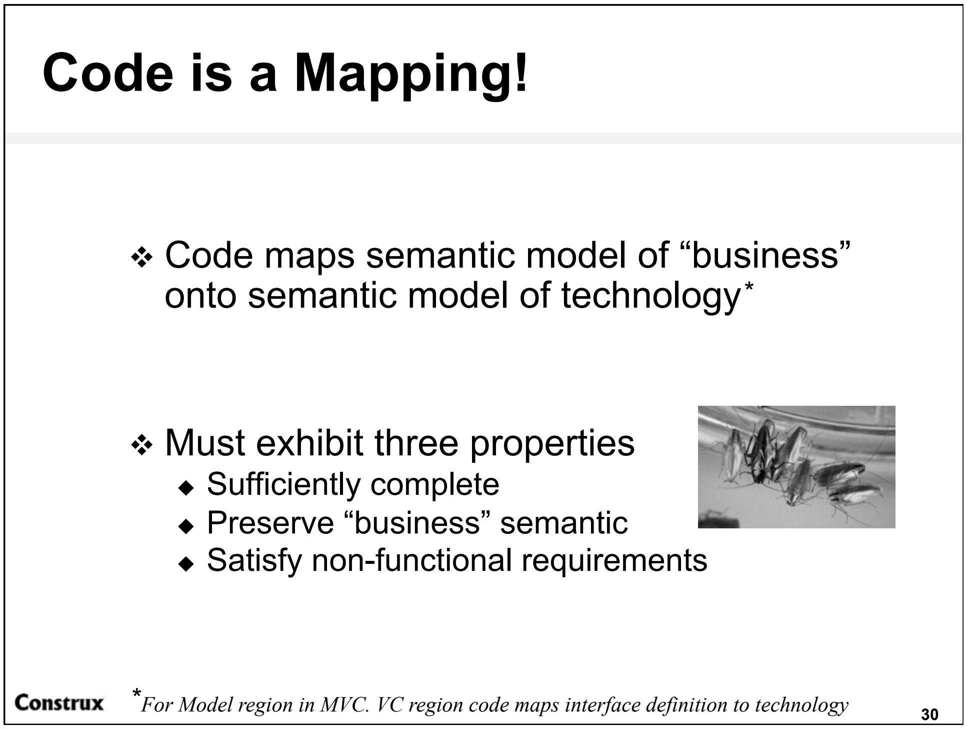
Semantic Models and Code

Semantic model of “business”

Semantic model of technology

```
public class Account {

  private double balance;
  private BAState state;

  public Account( double amount ) {
    balance = amount;
    state = BAState.OPEN;
  }

  public void deposit( double amount ) {
    if( state == BAState.OPEN ) {
      balance += amount;
    } else {
      throw new AccountNotOpen();
    }
  }

  public boolean withdraw( double amount ) {
    …
  }

  public double close() {
    if( state == BAState.OPEN ) {
      state = BAState.CLOSED;
      return balance;
    } else {
      throw new AccountNotOpen();
    }
  }

}
```

Construx

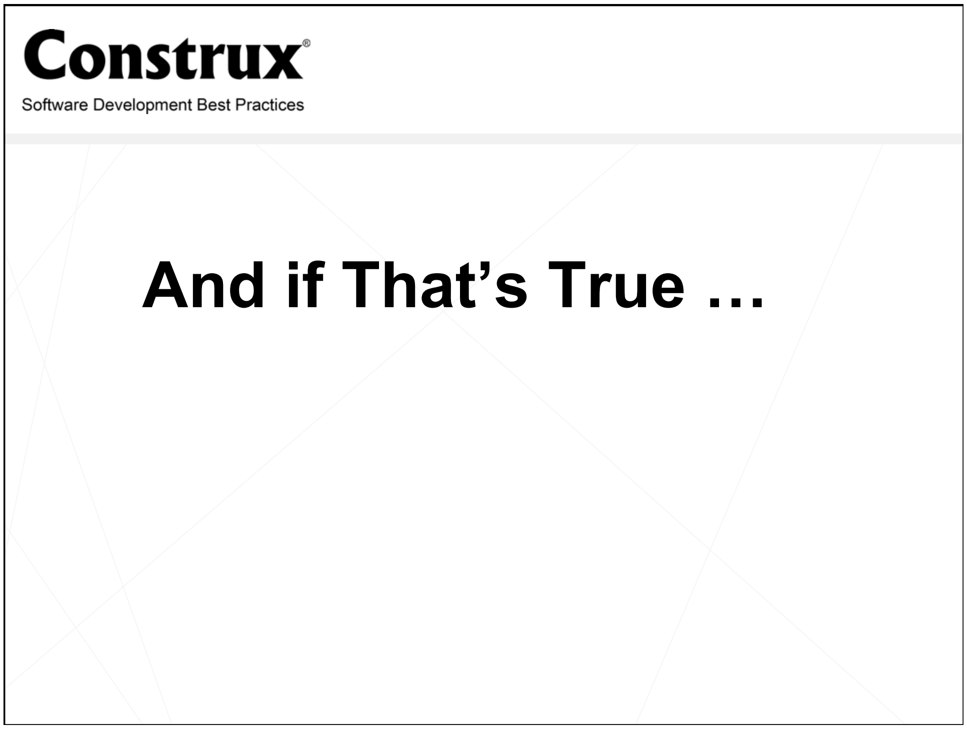


Code is a Mapping!

- Code maps semantic model of “business” onto semantic model of technology*
- Must exhibit three properties
  - Sufficiently complete
  - Preserve “business” semantic
  - Satisfy non-functional requirements

Construx

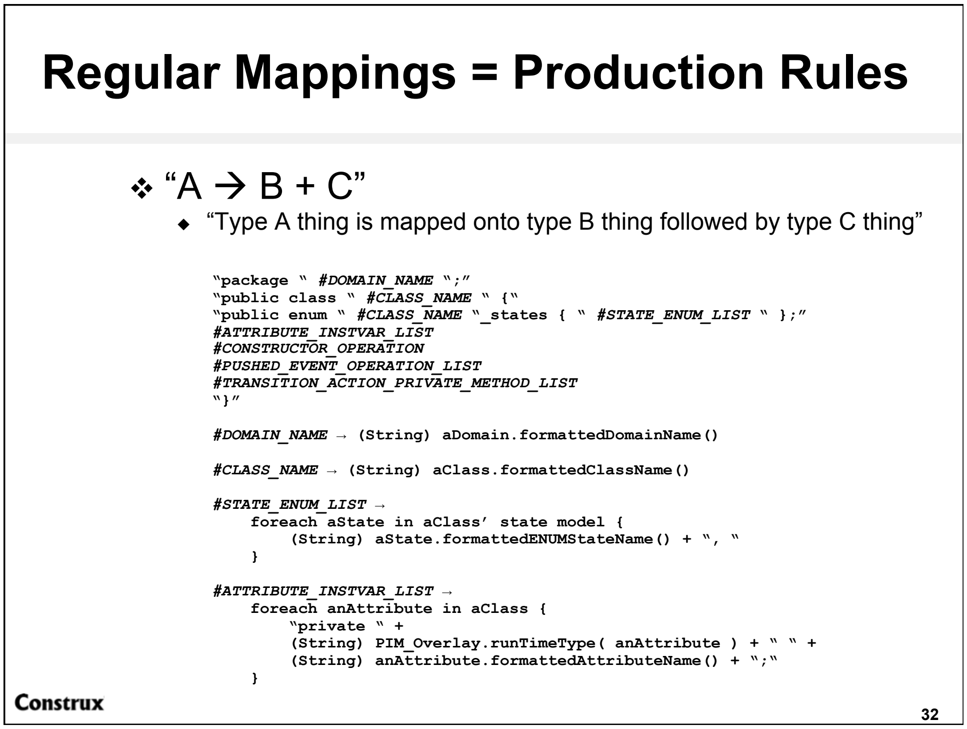
*For Model region in MVC. VC region code maps interface definition to technology



Construx®

Software Development Best Practices

And if That’s True …



Regular Mappings = Production Rules

- “A → B + C”
  - “Type A thing is mapped onto type B thing followed by type C thing”

```
“package “ #DOMAIN_NAME “;”
“public class “ #CLASS_NAME “ {“
“public enum “ #CLASS_NAME “_states { “ #STATE_ENUM_LIST “ };”
#ATTRIBUTE_INSTVAR_LIST
#CONSTRUCTOR_OPERATION
#PUSHED_EVENT_OPERATION_LIST
#TRANSITION_ACTION_PRIVATE_METHOD_LIST
“}”

#DOMAIN_NAME → (String) aDomain.formattedDomainName()

#CLASS_NAME → (String) aClass.formattedClassName()

#STATE_ENUM_LIST →
    foreach aState in aClass’ state model {
        (String) aState.formattedENUMStateName() + “, “
    }

#ATTRIBUTE_INSTVAR_LIST →
    foreach anAttribute in aClass {
        “private “ +
        (String) PIM_Overlay.runTimeType( anAttribute ) + “ “ +
        (String) anAttribute.formattedAttributeName() + “;“
    }
```

Construx

More Production Rules

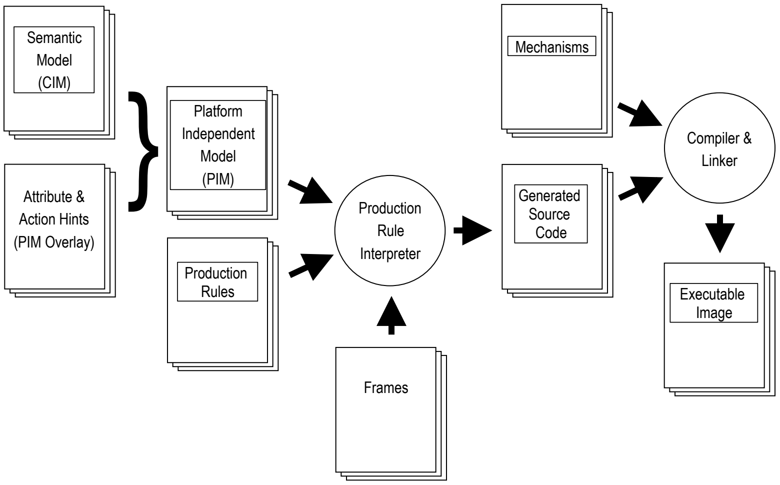
```
#PUSHED_EVENTS_OPERATION_LIST →
    foreach anEvent in aClass' state model {
        "public void " +
        (String) anEvent.formattedEventName() +
        "(" + #OPERATION_FORMAL_PARAMETERS + ") {" +
        #EVENT_METHOD_BODY +
        "}"
    }

#EVENT_METHOD_BODY →
    foreach aTransition triggered by anEvent {
        "if( state == " +
        (String) aClass.formattedClassName() +
        "_states." +
        (String) aTransition.formattedStartState() +
        #OPTIONAL_GUARD + " ) {"
        #TRANSITION_ACTIONS_LIST +
        if( aTransition.startState() != aTransition.endState() ) {
            "state = " +
            (String) aClass.formattedClassName() +
            "_states." +
            (String) aTransition.formattedEndState() +
        }
        "}"
    }

#OPTIONAL_GUARD →
    if( aTransition.hasGuard() ) {
        " && " +
        (String) PIM_Overlay.guardCondition( aTransition.guard() )
    }
```

CIMs, PIMs, PSMs

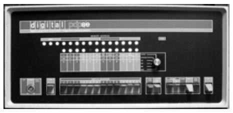
- CIM
  - *Computation Independent Model*
  - Purely “business” semantics, no automation technology
    - Not translate-able to fully executable code
- PIM
  - *Platform Independent Model*
  - Sufficient guidance to produce executable code, but generic enough to be translated into different computing platforms
    - Range → run time type, action contract → algorithm, …
- PSM
  - *Platform Specific Model*
  - Targets one technology environment, e.g., Java on single-user desktop, distributed C#, C++ on mobile device, Ruby on Rails, Python for cloud, …

Source: Object Management Group, “Model Driven Architecture”

“Open” Model Compiler

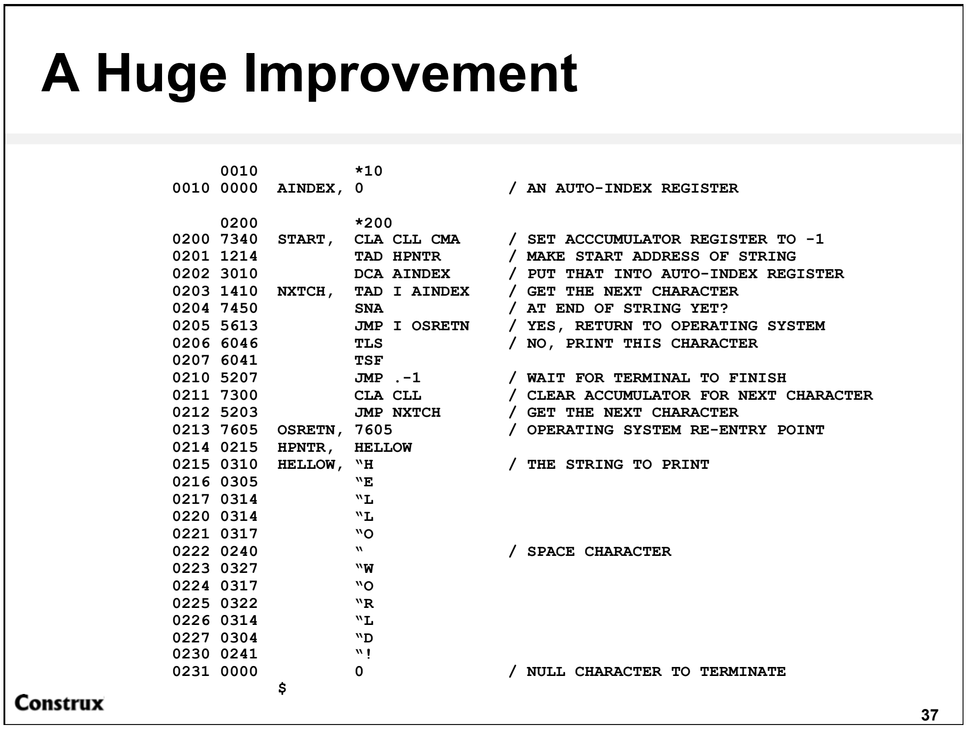


To the Computer …



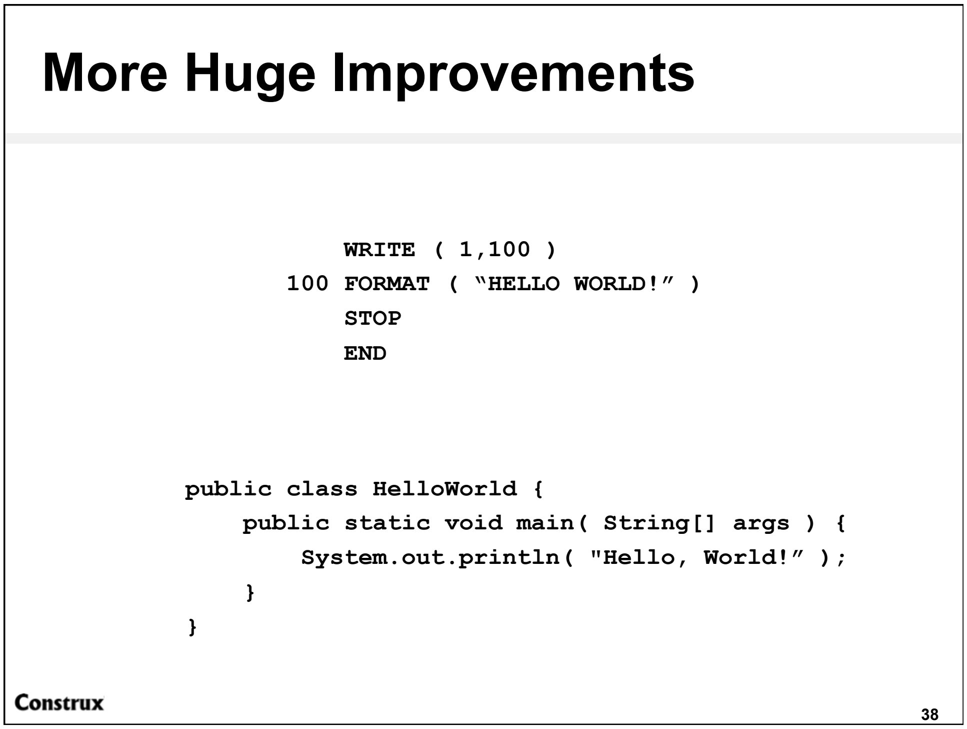
```
   Memory Address       Memory Content
   000 000 001 000      000 000 000 000
→  000 010 000 000      111 011 100 000
   000 010 000 001      001 010 001 100
   000 010 000 010      011 000 010 000
   000 010 000 011      001 100 001 000
   000 010 000 100      111 100 101 000
   000 010 000 101      101 110 001 011
   000 010 000 110      110 000 100 110
   000 010 000 111      110 000 100 001
   000 010 001 000      101 010 000 111
   000 010 001 001      111 011 000 000
   000 010 001 010      101 010 000 011
   000 010 001 011      111 110 000 101
   000 010 001 100      000 010 001 101
   000 010 001 101      000 011 001 000
   000 010 001 110      000 011 000 101
   000 010 001 111      000 011 001 100
   000 010 010 000      000 011 001 100
   000 010 010 001      000 011 001 111
   000 010 010 010      000 010 100 000
   000 010 010 011      000 011 010 111
   000 010 010 100      000 011 001 111
   000 010 010 101      000 011 010 010
   000 010 010 110      000 011 001 110
   000 010 010 111      000 011 000 100
   000 010 011 000      000 010 100 001
   000 010 011 001      000 000 000 000
```

→ *Starting memory address*



A Huge Improvement

```
     0010          *10
0010 0000  AINDEX, 0               / AN AUTO-INDEX REGISTER

     0200          *200
0200 7340  START,  CLA CLL CMA     / SET ACCCUMULATOR REGISTER TO -1
0201 1214          TAD HPNTR       / MAKE START ADDRESS OF STRING
0202 3010          DCA AINDEX      / PUT THAT INTO AUTO-INDEX REGISTER
0203 1410  NXTCH,  TAD I AINDEX    / GET THE NEXT CHARACTER
0204 7450          SNA             / AT END OF STRING YET?
0205 5613          JMP I OSRETN    / YES, RETURN TO OPERATING SYSTEM
0206 6046          TLS             / NO, PRINT THIS CHARACTER
0207 6041          TSF
0210 5207          JMP .-1         / WAIT FOR TERMINAL TO FINISH
0211 7300          CLA CLL         / CLEAR ACCUMULATOR FOR NEXT CHARACTER
0212 5203          JMP NXTCH       / GET THE NEXT CHARACTER
0213 7605  OSRETN, 7605            / OPERATING SYSTEM RE-ENTRY POINT
0214 0215  HPNTR,  HELLOW
0215 0310  HELLOW, "H              / THE STRING TO PRINT
0216 0305          "E
0217 0314          "L
0220 0314          "L
0221 0317          "O
0222 0240          "               / SPACE CHARACTER
0223 0327          "W
0224 0317          "O
0225 0322          "R
0226 0314          "L
0227 0304          "D
0230 0241          "!
0231 0000          0               / NULL CHARACTER TO TERMINATE
           $
```

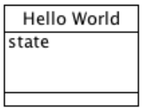


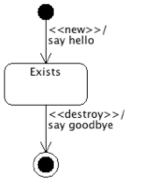
More Huge Improvements

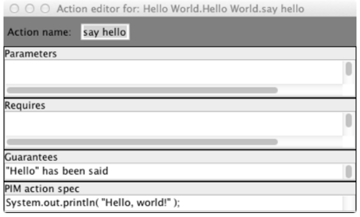
```
      WRITE ( 1,100 )
  100 FORMAT ( "HELLO WORLD!" )
      STOP
      END
```

```
public class HelloWorld {
    public static void main( String[] args ) {
        System.out.println( "Hello, World!" );
    }
}
```

Another Huge Improvement







Action name: say hello

Parameters

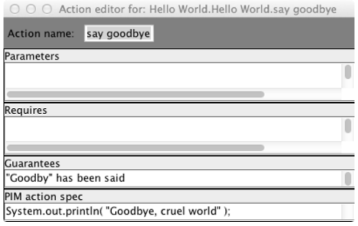
Requires

Guarantees

"Hello" has been said

PIM action spec

```
System.out.println( "Hello, world!" );
```



Action name: say goodbye

Parameters

Requires

Guarantees

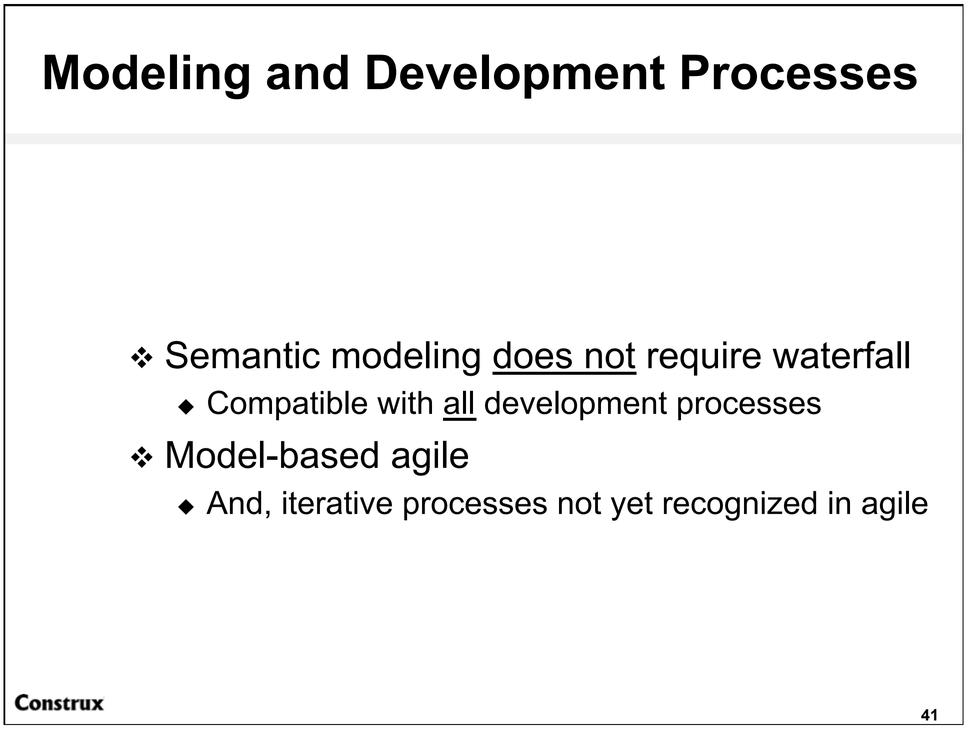
"Goodby" has been said

PIM action spec

```
System.out.println( "Goodbye, cruel world" );
```

Open Model Compiler: Other Uses

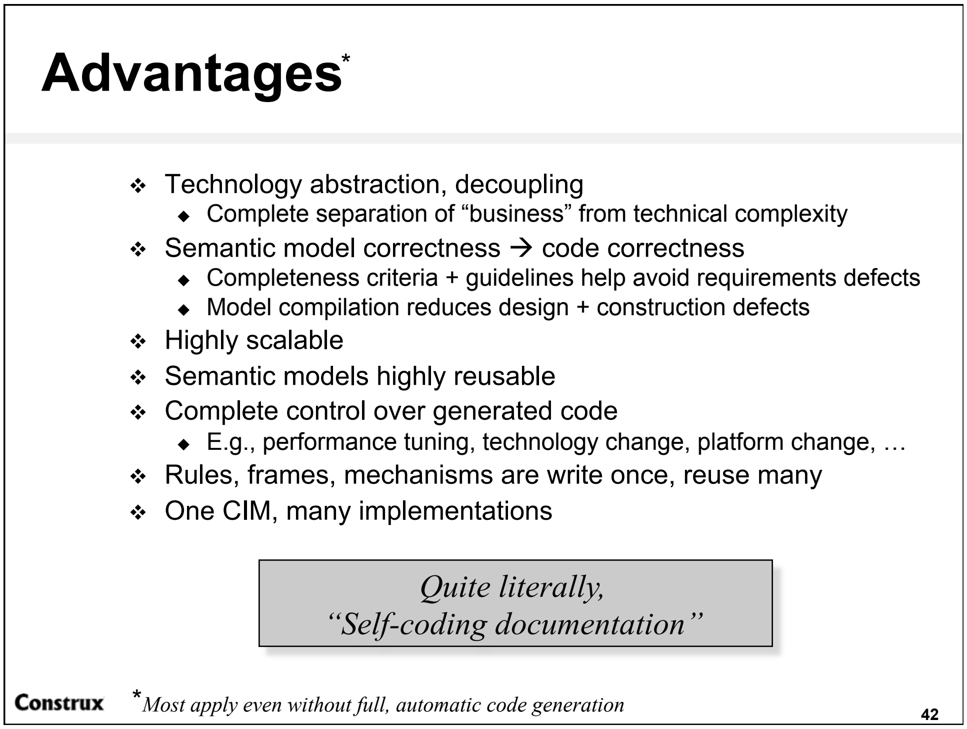
- Derive verification test cases
- Generate formal documentation
  - Including “The system shall …”
- Compute semantic model complexity metrics
- …



Modeling and Development Processes

- Semantic modeling does not require waterfall
  - Compatible with all development processes
- Model-based agile
  - And, iterative processes not yet recognized in agile

Construx



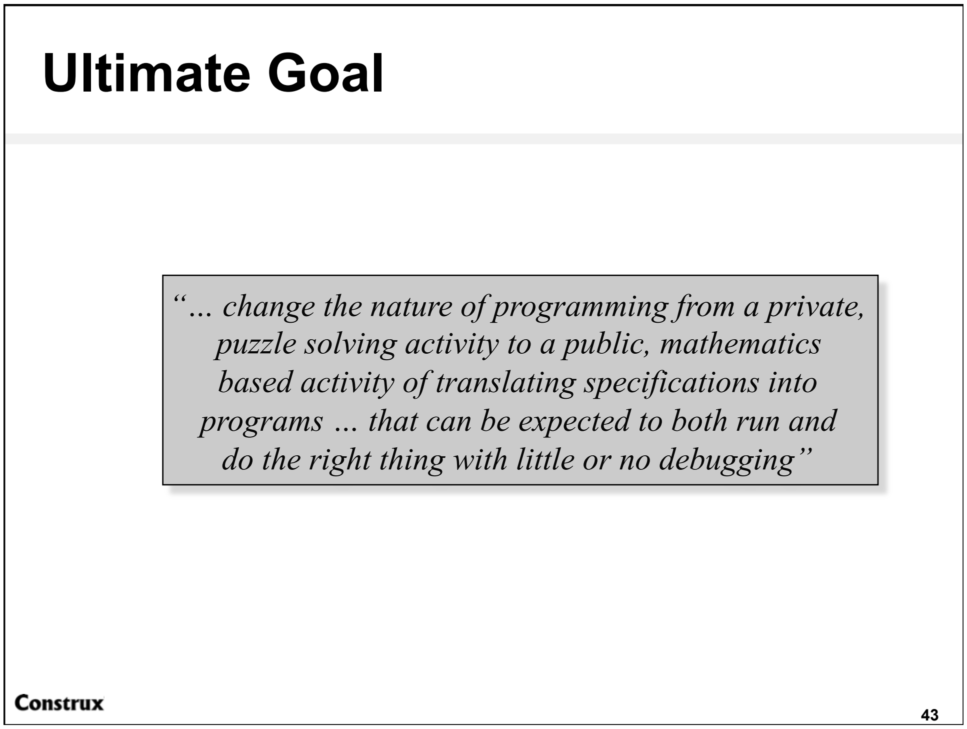
Advantages*

- Technology abstraction, decoupling
  - Complete separation of “business” from technical complexity
- Semantic model correctness → code correctness
  - Completeness criteria + guidelines help avoid requirements defects
  - Model compilation reduces design + construction defects
- Highly scalable
- Semantic models highly reusable
- Complete control over generated code
  - E.g., performance tuning, technology change, platform change, …
- Rules, frames, mechanisms are write once, reuse many
- One CIM, many implementations

Quite literally,
“Self-coding documentation”

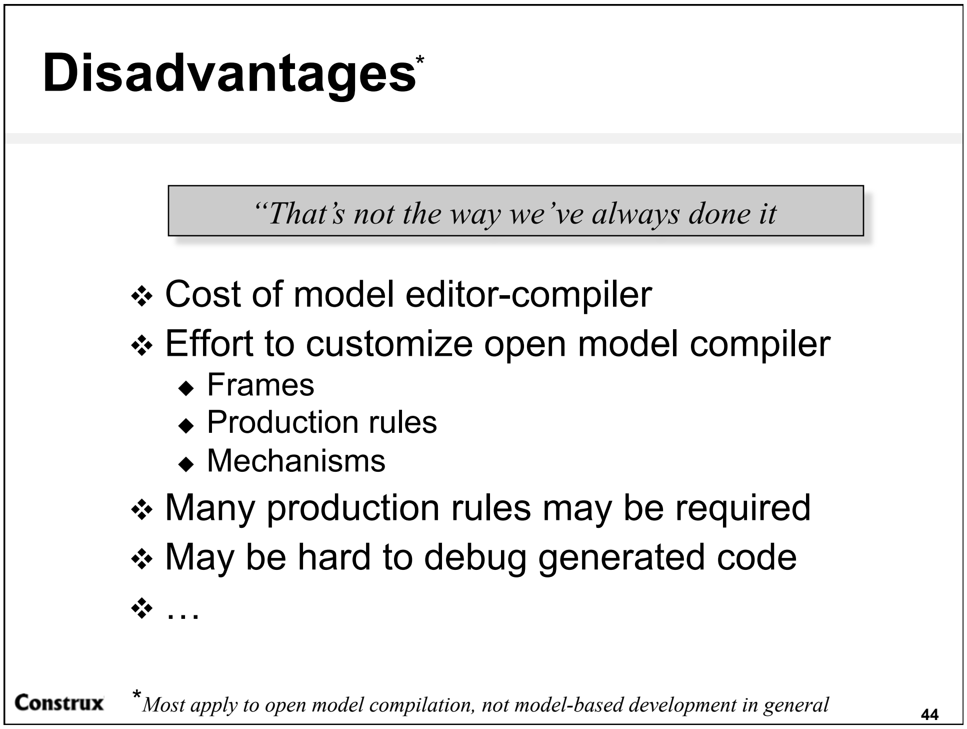
Construx

**Most apply even without full, automatic code generation*



Ultimate Goal

> *"… change the nature of programming from a private, puzzle solving activity to a public, mathematics based activity of translating specifications into programs … that can be expected to both run and do the right thing with little or no debugging"*



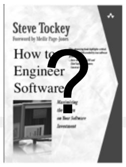
Disadvantages*

> *"That's not the way we've always done it*

- ❖ Cost of model editor-compiler
- ❖ Effort to customize open model compiler
  - ◆ Frames
  - ◆ Production rules
  - ◆ Mechanisms
- ❖ Many production rules may be required
- ❖ May be hard to debug generated code
- ❖ …

**Most apply to open model compilation, not model-based development in general*

Book Outline



- Part I: Intro and Foundations
  - Introduction
  - Nature of code
  - Fundamental principles
  - Functional and non-functional requirements
  - UML overview
  - Partitioning into domains
- Part II: Semantic modeling
  - Use case diagrams
  - Class models
  - Interaction diagrams
  - State models
  - Partitioning into subdomains
  - Wrapping up semantic modeling
- Part III: Design and code
  - Introduction to design and code
  - Designing interfaces
  - HLD: Classes and operations
  - HLD: Contracts and signatures
  - Detailed design and code
- Part III: Design and code (cont)
  - Formal disciplines
  - Optimization
  - Model compilation
  - Advanced open model compilation
  - Wrapping up design and code
- Part IV: Related topics
  - Estimation
  - Development processes
  - Economics of error handling
  - Arguments against MBSE
- Part V: Summary
  - Closing remarks
- References
- Part VI: Appendices
  - Documentation principles
  - WebBooks 2.0 case study
  - Semantics of semantic modeling
  - Sample production rules
  - Structural complexity metrics

Summary

- Software projects perform poorly
  - Poor requirements, syntax >> semantics, unmanaged complexity, over dependence on test, code not self-documenting
- Semantics >> syntax
  - Bug == defect == semantic inconsistency
- Code automates “business”
- Can precisely, concisely specify business semantic
- Can precisely, concisely specify automation technology semantic
- Code maps business semantic onto automation technology semantic
  - Source of most defects!
- Mapping can be expressed as production rules
  - Open model compiler interprets rules
  - Different rules generate different application source code
    - Executable code for different platforms
    - Executable code with different performance characteristics
    - Verification test cases
    - Formal documentation
    - Semantic model complexity metrics
    - …

Contact Information

Construx®

Software Development Best Practices

- Seminars
- Consulting
- Resources

stevet@construx.com

www.construx.com

+1(425) 636-0100