Better Stories, Better Languages

What Would Alyssa P. Hacker Do?

François-René Rideau, *TUNES Project*

Bay Area Lispers, 2009-12-29

http://fare.tunes.org/computing/bal2009.pdf

The Take Home Points

Software tools imply a story

New stories can help invent new tools

Explicit stories are a great meta-tool...

This Talk

Take a sad so-o-ong, and make it be-e-etter

Let’s start with easy ones you already know...

How to fund software?

Issue:	Software costs money to produce
Story:	Software is owned and sold Vendors & customers
Solution:	Proprietary Software Closed binaries

Sad...

How to fund programming?

Issue: Starved programmers don't code

Story: Labor is owned and sold
Contributors & users

Solution: Free Software
Open Source

Better!

Decompose programs?

Issue: Software doesn't fit in one brainful

Story: Hierarchical design into components
by fully informed designer

Solution: Top-down management
UML diagrams

Sad...

Decompose programming?

Issue: Many programmers must cooperate

Story: Partially informed programmers
Growing a network of programs

Solution: Software distributions
Distributed version control

Better!

Achieve great software?

Issue: Software is hard to design

Story: Disseminate expert information
Restrict each component to best experts

Solution: Standards
Segregation by expertise

Sad...

Foster better programming?

Issue:	Do our best, compete with others
Story:	Experience through experiments Cultivate good incentives
Solution:	Select from abundant market Learn in communities

Better!

Choose features in a PL?

Issue: Access all features of the machine

Story: PLs are for machines

Solution: Match machine features
Turing tar pit

Sad...

Express programming ideas?

Issue: Convey all meanings of the humans

Story: PLs are for humans

Solution: Match human cognition
Simpler programming languages

Better!

Handle repetitive programs?

Issue: Lots of repetition in programs

Story: Programmer as grunt worker
Language as a given

Solution: Theorize repetitions as Design Patterns
More programmers, more drudge

Sad...

Remove programming drudge?

Issue: Drudge in programming

Story: Programmer as abstract thinker
Language as a platform

Solution: Metaprograms
PL extensibility (macros…)

Better!

Have an extensible syntax?

Issue: Hooks into existing syntax

Story: Side-effect the One True Syntax

Solution: Bind symbols to macros
Dynamic readtable

Sad...

Explore useful syntaxes?

Issue: Best express each program fragment

Story: Exploration of many syntaxes

Solution: Scoped syntax specification
Racket languages, OMeta

Better!

Users ≠ Programmers

Issue: Users and Programmers differ

Story: Dumbed down UI for Users
All-Power for Devs (in VM)

Solution: Unrelated UI and PL
Segregation

Sad...

Using = Programming

Issue:	One language, spoken or written
Story:	Computer interaction is programming Continuum of proficiency in users
Solution:	Integrated interactive interface Language levels and dialects

Better!

Programmer ≠ PL Implementer

Issue: Writing a compiler is hard

Story: Specialists write compiler
Mere programmers use

Solution: Closed implementation for each PL

Sad...

Programming = PL Implementing

Issue: Incremental DSL development

Story: Special case of Using = Programming
P implements PL spoken by users

Solution: Support for DSL
First-class implementations

Better!

PL Definer ≠ PL Implementer

Issue: Designing a language is hard

Story: Specialists define big language
Others implement

Solution: Standard for language
Decades-old design

Sad...

PL Defining = PL Implementing

Issue: Specifying Semantics = Implementing

Story: Declarative specification
Orthogonal implementation strategies

Solution: Grammatical mixins
Monadic lifting

Better!

Get a specialized language?

Issue: SW involves heterogeneous activities

Story: Each domain its experts
Segregation of experts

Solution: External DSLs
Scripting languages

Sad...

Specialize conversation?

Issue: Express domain expertise

Story: One brain, many topics
Adapt PL to domain

Solution: Internal DSLs
Universal PL, many contexts

Better!

Get Programs Debugged?

Issue: Programs have bugs, need be fixed

Story: Bug is an exceptional situation
Ad-hoc tools retrofitted

Solution: Low-level debugger
Ad-hoc debugging information

Sad...

Exploring Program Semantics?

Issue: P'ing isn't obvious, needs exploring

Story: Imperfect is the default situation
Environment for exploration

Solution: Compiler as reversible lens
Experiment in Virtual World

Better!

Secure existing software?

Issue: Security is a hard, requires specialists

Story: Security as afterthought
Security by independent experts

Solution: Forever patch leaks
Low-level protection

Sad...

Build software securely?

Issue: Security is intrinsic to SW design

Story: Design with security from start
Educate programmers

Solution: Whole-system protocol design
Capabilities

Better!

Dealing with catastrophes?

Issue: Bad manipulations cause data loss

Story: Error is exceptional, catastrophic

Solution: Confirm menus, remove bin
Undo last (few)

Sad...

Eliminating catastrophes?

Issue: Make bad manipulations unexpressible

Story: Error is normal, casual

Solution: monotonic storage never loses data
Infinite undo

Better!

Document software interfaces?

Issue: PL can't formalize SW intention

Story: Document what can't be expressed
PL as a given

Solution: Informal contracts
between heterogenous teams

Sad...

Agree on Responsibilities?

Issue: Formalism has costs and benefits

Story: Formalize contracts
PL extended as needed

Solution: Better contracts and types
Cost benefit analysis

Better!

Arbitrate Resource?

Issue: Need invariants on shared resources

Story: Central dictator needed
Schedule resource possession

Solution: (OS or App) Kernel
Static set of resources

Sad...

Resolve Conflicts?

Issue: Identify owner of resource bundles

Story: Self-enforcing contracts
Linear logic expresses ownership

Solution: Invariant-enforcing linker
Dynamic resource bundles

Better!

Connect Computers?

Issue: Single machine can't do it all

Story: Putting together many systems

Solution: Remote method invocation
Shipping information around

Sad...

Distribute Computation?

Issue: System is made of many machines

Story: One system, many agents

Solution: Declarative deployment
Content-based addressing

Better!

Handle mistrust?

Issue: Need protection barriers

Story: Kernel-managed domains
Expensive rigid model

Solution: processes, containers, virtual PCs
Expensive and inexpressive

Sad...

Express limited trust?

Issue: Dynamic bundles of capabilities

Story: Everyone "root" in own VW
Recursively so, by default

Solution: PL support for VW
Cheap to create sub-user

Better!

Persist important data?

Issue: Important data must persist

Story: Programmer must manually persist data
Transient by default

Solution: Ad-hoc filesystems, databases
Explicit I/O

Sad...

Write persistent software?

Issue: All data is important

Story: Why else program about it?
Persistence by default

Solution: Orthogonal persistent
Implicit support in PL

Better!

Model a changing world?

Issue: Mutations happen — Object-Oriented

Story: Data is always mutable... by others!
Can't trust anything or anyone

Solution: Imperative programming
Locks: expensive temporary protection

Sad...

Model changes to the world?

Issue:	Transforms compose — Value-Oriented
Story:	Immutable values, at least by default Can always reason about programs
Solution:	Functional PL (OCaml, Haskell) Effects or Monads

Better!

Discuss relevant change?

Issue: First-class change — Change-Oriented

Story: State is meta-level modularity
Mutable vs immutable views on code

Solution: Differentiation and Integration
Switch view to/from Monadic style

Even better!

The Grand Challenge

The Grand Challenge

None of these Stories is revolutionary

Each has been foretold in past systems

The Grand Challenge

None of these Stories is revolutionary

Each has been foretold in past systems

But no *system* embodies them all at once

Missing: *vision*, not technical ability

The Meta-Story

The Meta-Story

Sad Stories: about Things Created

Better Stories: about People Creating

The Meta-Story

Sad Stories: about Things Created

Better Stories: about People Creating

Sad Stories: bind good early

Better Stories: ban bad early