galois

High Assurance Software

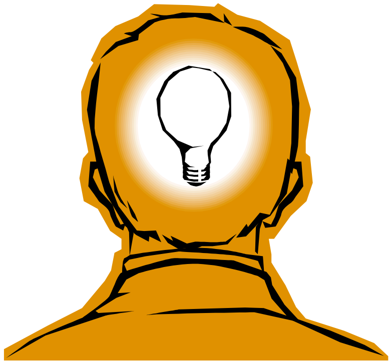
John Launchbury
CEO, Galois Inc
john@galois.com

A Personal View

- A history of Galois
- Some technology examples
- Looking forward

In the beginning…

- 1999/2000
- Service focus
- Customers
  - Government
  - Local industry



Have Functional Language, Can program

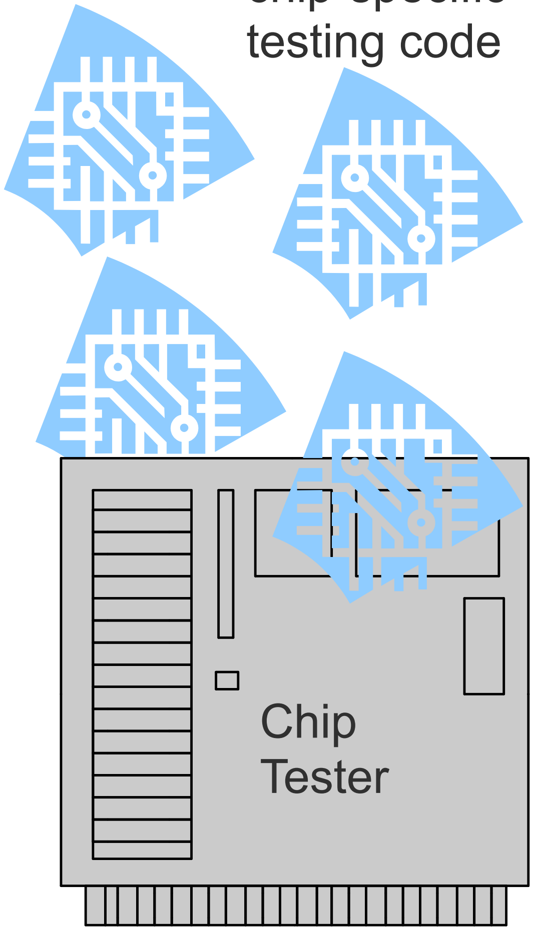
Making a Business out of FP

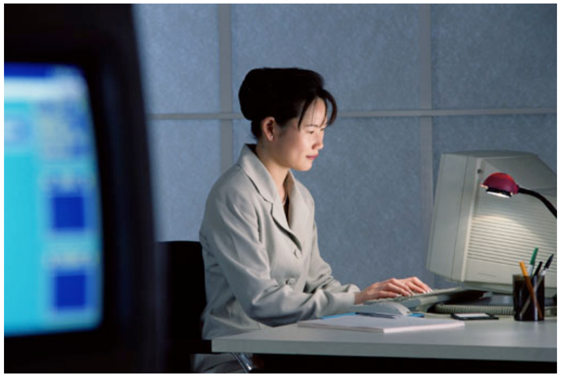
- ~~Build cool things that people should want~~
  - ~~Find sales people to sell it~~
- Sales ***is always*** the business
  - Technology is a support department for sales

Marketing identifies the right product for *Technology* to build so that *Sales* will be able to sell

Automated Test Equipment

- ATE vendor needs to provide backwards compatibility
- Translation task
  - Code cleaning to upgrade language
  - OS migration
  - API discovery & modification
- Problem: testing code contains IP
- Requirement: the code look-and-feel to remain unchanged





Partial Change List

- Insert missing headers (`#include`s)
- Change/add prototypes to match definition
- Add prototype declaration instead of implicit forward declaration
- Remove syntactic clutter
- Remove/change ill-behaved declarations (e.g., `static struct`, `static char *`)
- Make type casts explicit (i.e. `double` as case scrutinee; cast to `int`)
- Change now illegal identifier names (forced by ANSI changes)
- Change `return` statements for functions that now return `void`
- Make implicit variable declarations explicit (i.e., to `int`)
- Type changes (explicate `CONN` equivalences)
- Introduction & initialization of global and /or local variables
- Type changes/initialization of struct members
- Aggregate initialization (where array is given all its values at once; need to translate to explicit bit setting)
- Removal of redundant checks (no need to check for end of array; done inside API)
- Flag deprecated API elements
- Replacing malloc/free with API create/destroy
- API function name/type changes

API Discovery

- Old machine
  - Test programs use arrays as connection lists

```
b1 = *c;         /* set b1 to current bit */
b2 = *(c++);     /* set b2 to next bit, move focus */
*(c + 1) = b3;   /* set next bit to b3 */
```

- New machine
  - Requires use of API for building connection lists

```
b1 = conn_getbit(c, c_current);
b2 = conn_getbit(c, c_current++);
conn_setbit(c, c_current + 1, b3);
```

```
/* 1. BEFORE */

debug_printf("**** DSP error in test %s,
             occurred on bit # %d -->",
             test_name (NULL),
             (*plist & ~LASTBIT) + 1);

if ((log_ptr->vector >= f_scan_st[u])
 && (log_ptr->vector < f_scan_sp[u]))
 {
  if ((log_ptr->fail_bits[0]
       == *even_ram)
   || (log_ptr->fail_bits[1]
       == *even_ram))
  {
    ficm_write(even_ram, log_ptr->vector,
               log_ptr->vector,
               "H", UNSPECIFIED, UNSPECIFIED);
    rep_str[2*u][log_ptr->vector - f_scan_st[u]] = '1';
  }
```

```
/* 1. AFTER  */

debug_printf("**** DSP error in test %s,
             occurred on bit # %d -->",
             test_name (NULL),
             conn_getbit(plist, plist_local_counter) + 1);

if ((log_ptr->vector >= f_scan_st[u])
 && (log_ptr->vector < f_scan_sp[u]))
 {
  if ((log_ptr->fail_bits[0]
     == conn_getbit(even_ram, even_ram_global_counter))
   || (log_ptr->fail_bits[1]
     == conn_getbit(even_ram, even_ram_global_counter)))
  {
    ficm_write(even_ram, log_ptr->vector,
               log_ptr->vector,
               "H", UNSPECIFIED, UNSPECIFIED);
    rep_str[2*u][log_ptr->vector - f_scan_st[u]] = '1';
  }
```

```
/* 2. BEFORE */

for( pbl = 0; pbl < S_parConnPointer->nrbitl;
     pbl++ )
{
    close_mba_relays
          ( S_parConnPointer->bitl[pbl] );
    open_io_relays
          ( S_parConnPointer->bitl[pbl] );
    prim_wait( 3 MS );
    if ( MbaTest( S_parConnPointer->bitl[pbl],
                  SREXPD, SRESPD, DontDoMbaRly )
         == FAIL )
        goto finish;
    close_io_relays
          ( S_parConnPointer->bitl[pbl] );
    open_mba_relays
          ( S_parConnPointer->bitl[pbl] );

    if ( theSiteCount > 1 && aSiteFailed )
        update_parconn ( &S_tmpParConn, &p_sdbit );
}
```

```
/* 2. AFTER  */

for( pbl = 0; pbl < parconn_getcount(S_parConnPointer);
     pbl++ )
{
    close_mba_relays
          ( parconn_getconn(S_parConnPointer, pbl) );
    open_io_relays
          ( parconn_getconn(S_parConnPointer, pbl) );
    prim_wait( 3 MS );
    if ( MbaTest( parconn_getconn(S_parConnPointer, pbl),
                  SREXPD, SRESPD, DontDoMbaRly )
         == FAIL )
        goto finish;
    close_io_relays
          ( parconn_getconn(S_parConnPointer, pbl) );
    open_mba_relays
          ( parconn_getconn(S_parConnPointer, pbl) );

    if ( theSiteCount > 1 && aSiteFailed )
        parconn_update ( S_tmpParConn, p_sdbit );
}
```

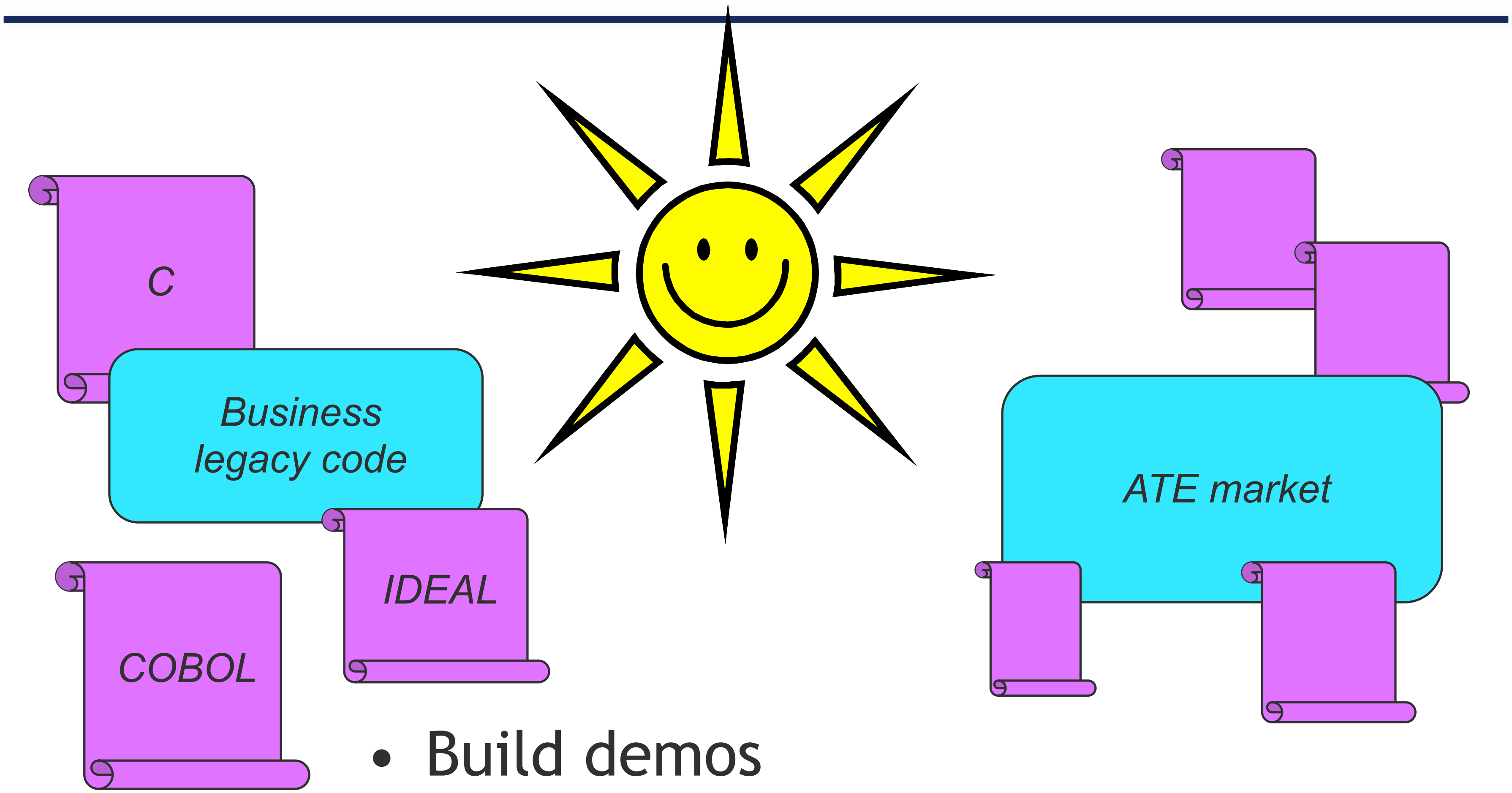
Building the translator

- C-Kit in ML
- Tight schedule, regular releases
- 6 months

Lesson 1

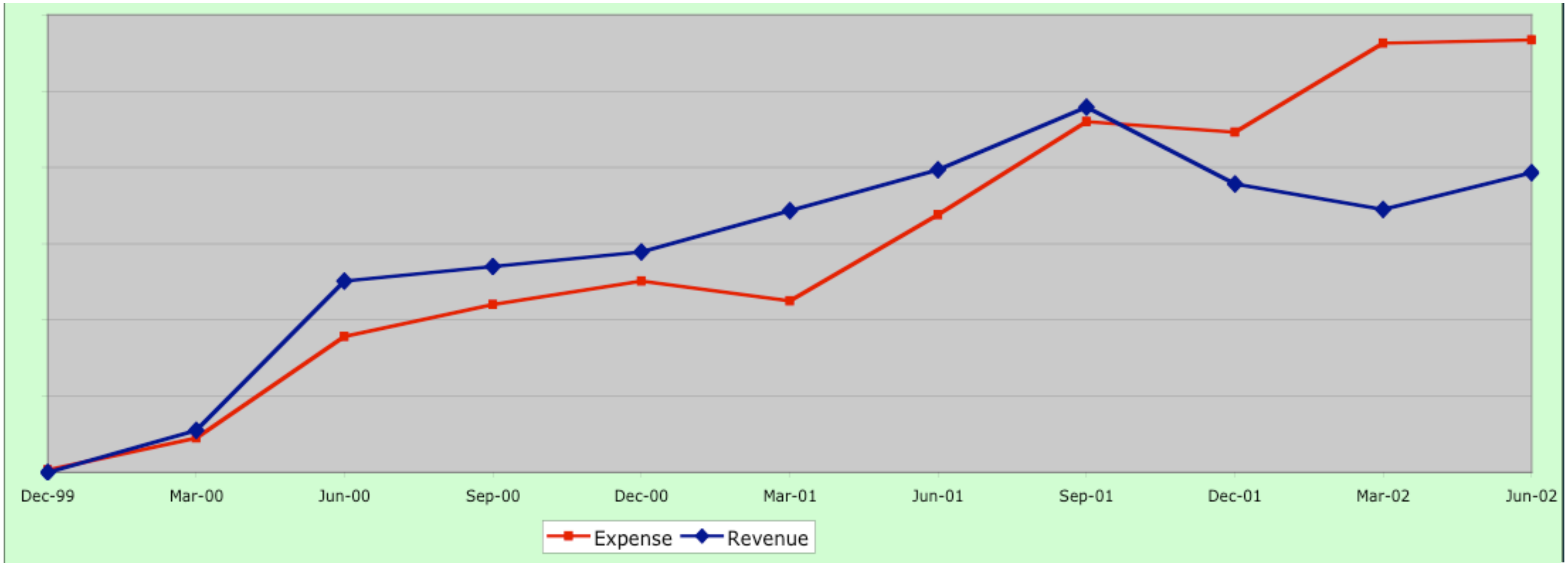
Functional programming covers over a multitude of sins

Translators!!!



- Build demos
- Visit potential customers
- Align with channel partners

Market issues



Lesson 2

Keep the blue line above the red line

Analysis

- Didn’t read the market properly
  - References
  - Budgets
- Lost focus on our core business
- Needed to re-invent Galois
  - Very challenging times

Lesson 3
It’s not about technology,
it’s about markets and relationships

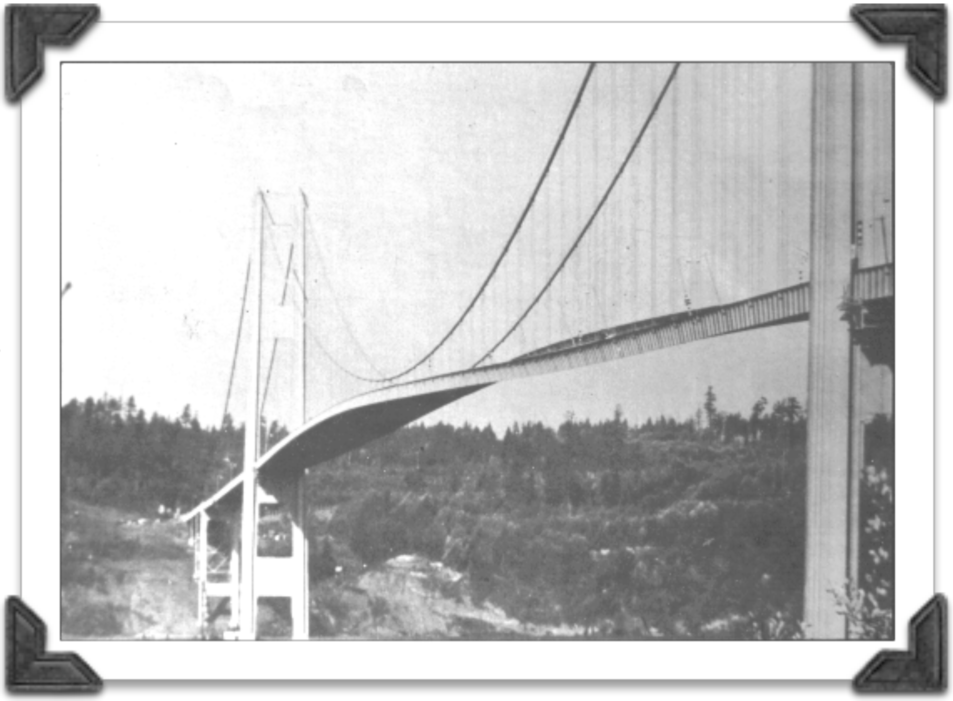
Who are we?

- Examination
  - Look at what we’ve been successful at
  - Look at our skill sets
  - Ask our clients
- Synthesize
- Define the brand

Lesson 4
If you don’t know who you are,
then neither does anyone else

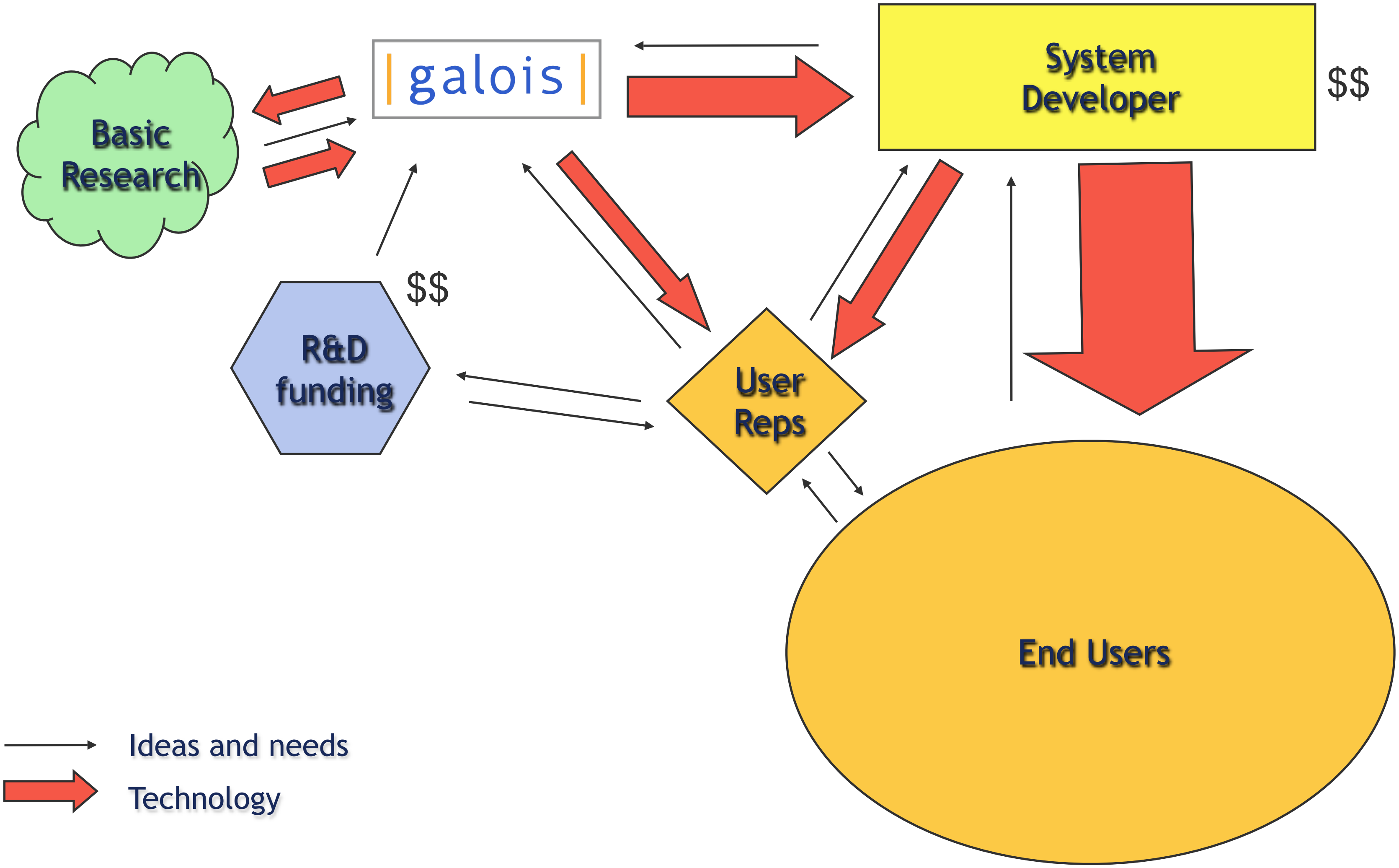
High Assurance Software

- Let the software itself be trustworthy
  - Software artifacts to speak for themselves
  - ... rather than hoping to rely on the process that created them
- Use mathematical models to enable tractable analysis
  - Executable models and formal methods
  - A model is an abstraction that allows thought at a higher level
- Follow open standards
  - Build components with high internal integrity
  - Maximize interoperability

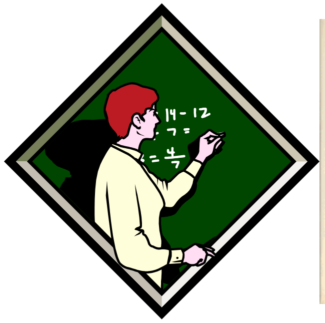


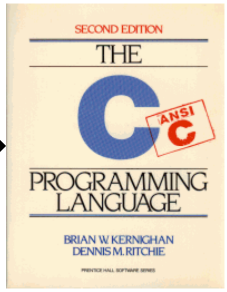
We want to see software built with the same diligence and analysis as other engineers build bridges

Galois Business Model



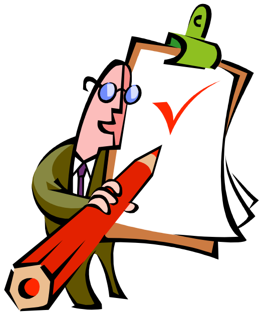
Challenge: Correctness of Crypto





Requires skills in math AND programming

Variety of target architectures



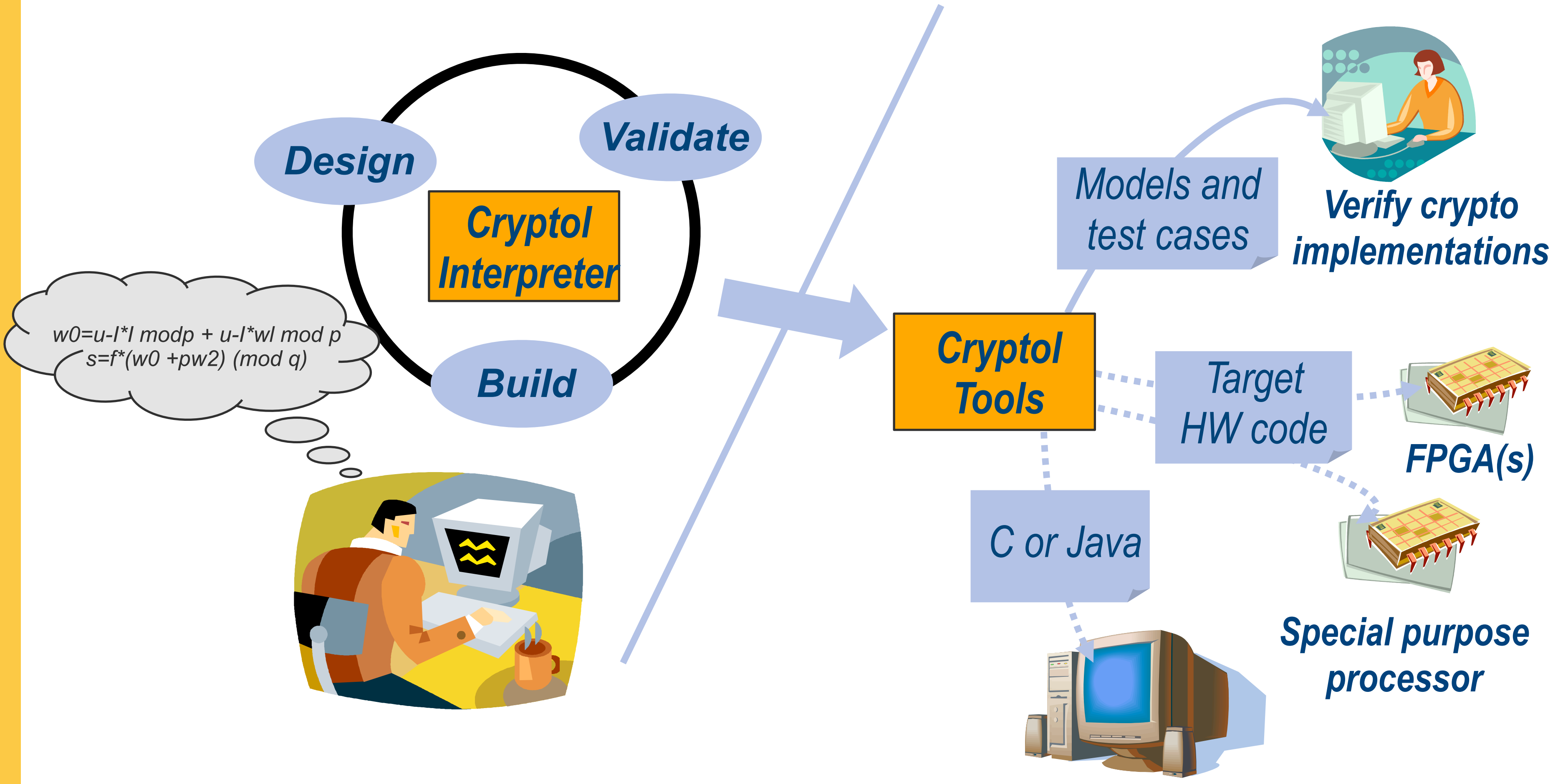
Validation is complex and tedious

- Validation and verification of crypto implementations is critical to crypto-modernization programs
- Not just the DoD
  - "25% of algorithms submitted for FIPS validation had security flaws"
    Director NIST CMVP, March 26, 2002

Cryptol: One Specification — Many Uses

Domain-Specific Design Capture

Assured Implementation



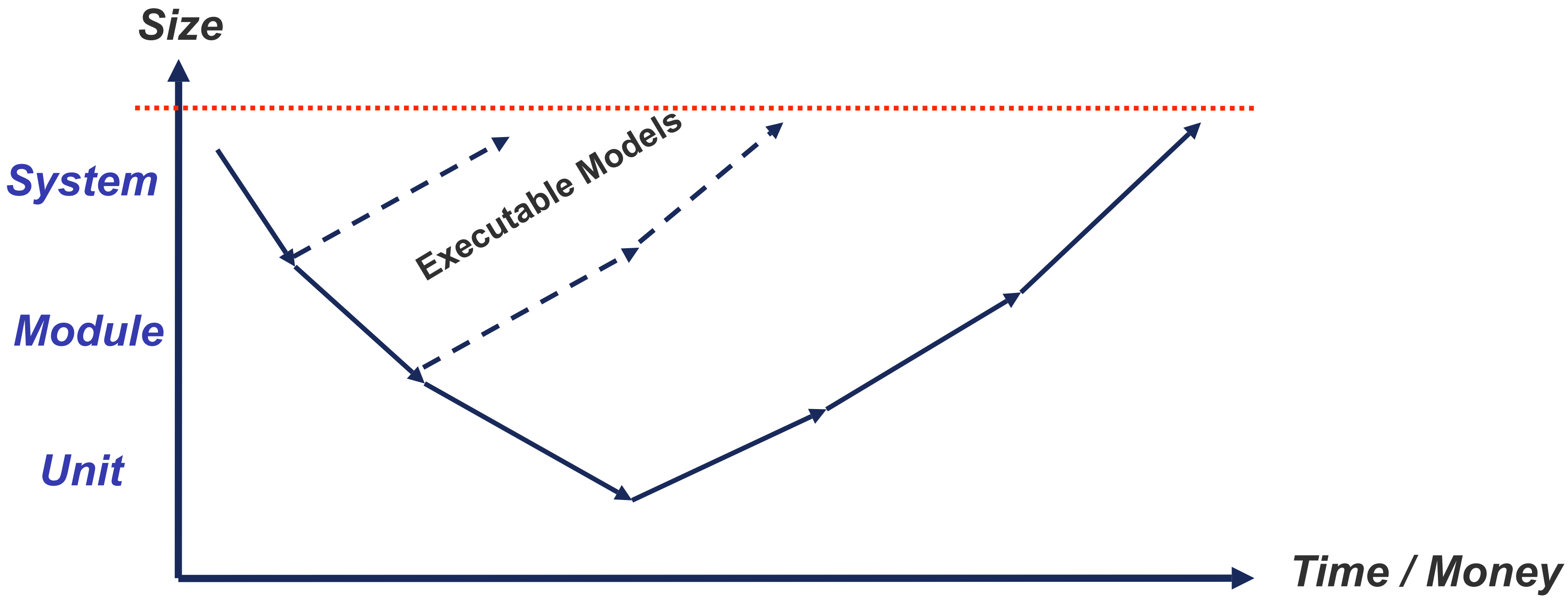
Cryptol Toolset

- Cryptol™ Interpreter
  - Enables algorithm exploration and debugging
- Cryptol™ FPGA Compiler
  - Automates algorithm-to-hardware
  - Throughputs up to 50Gbps+
  - Algorithm- and device-agile
- Cryptol™ Verifier
  - Performs equivalence checking between algorithm representations
  - Works at any level from Cryptol source to netlist
  - Extremely fast

Basic Cryptol tools currently used by GDIT in AIM crypto-development
Reported 25% reduction in development times

Early Use of Models

- Early testing/analysis has a profound cost benefit



Engineering in Haskell

Modeling

- Mathematical foundation
  - Allows for mathematical guarantees of behavior
  - High assurance
- Very powerful abstraction
  - Say what needs to be said, nothing more
  - Easier to build smarter software
- Executable models
  - Automatic memory access and protection
  - Non-deterministic timings
  - Flexible and powerful

Production

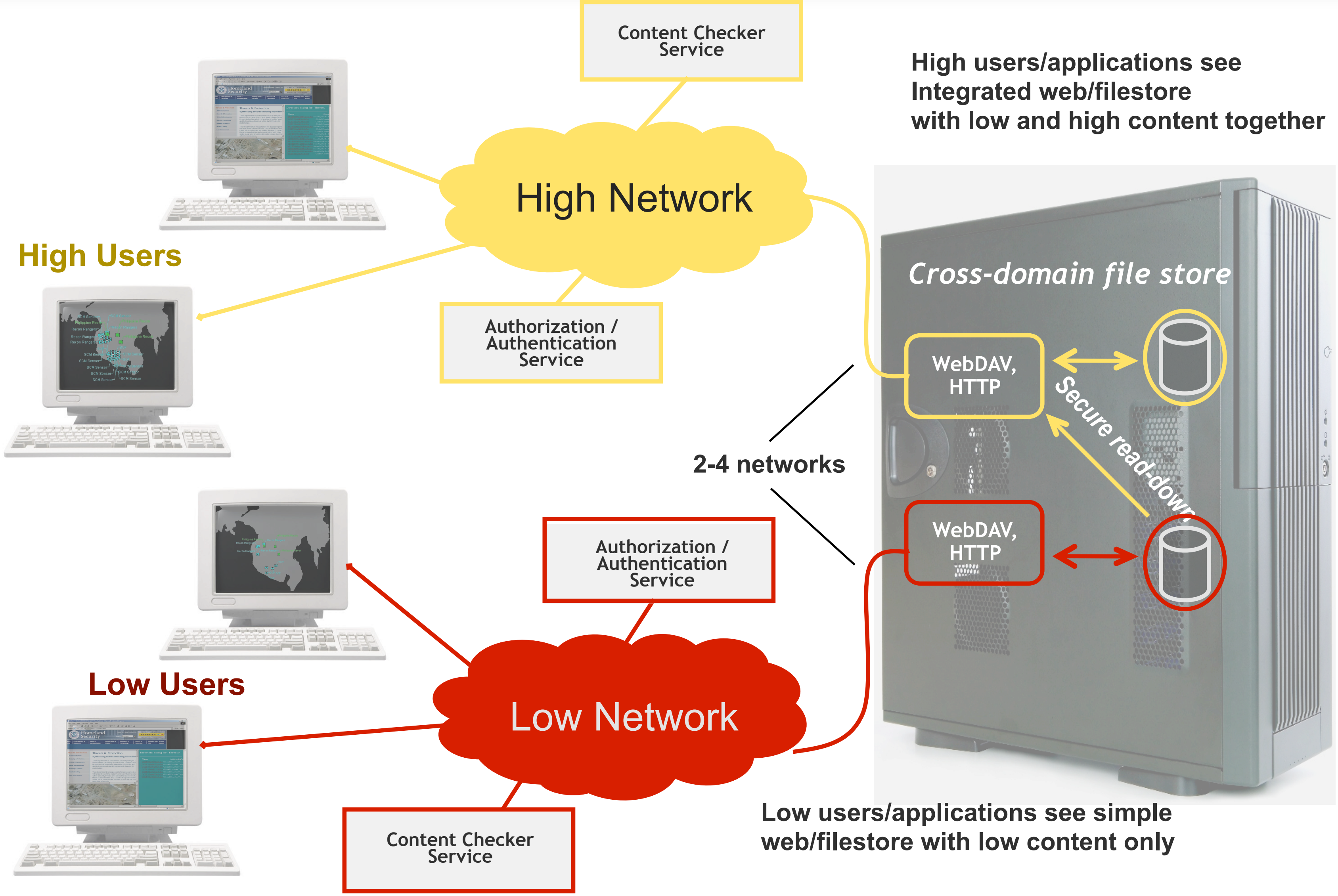
- Smooth path from model to product
  - The executable model is the first prototype
  - Incremental refinements from problem focus to solution focus
- Huge productivity benefits
  - Shorter (2-10x), clearer, and more maintainable code
  - Reducing time-to-deployment
- Scalable to complex systems
  - Concise expression
  - Overcomes limitations of earlier formal and semi-formal methods
  - Multi-core ready !!

Cross Domain Collaboration

- Problem: information sharing across security domains (e.g. unclass -> TS)
  - Ensuring information flows from low to high
  - High assurance (SABI, TSABI environments)
  - Compatible with existing and future networks

- Shared network file server, with read down
  - Designed certifiable architecture up front
  - Leverage assurance efforts elsewhere (MILS, trusted separation kernel)
  - Runs on standard COTS (Intel) hardware

Cross-domain webDAV server



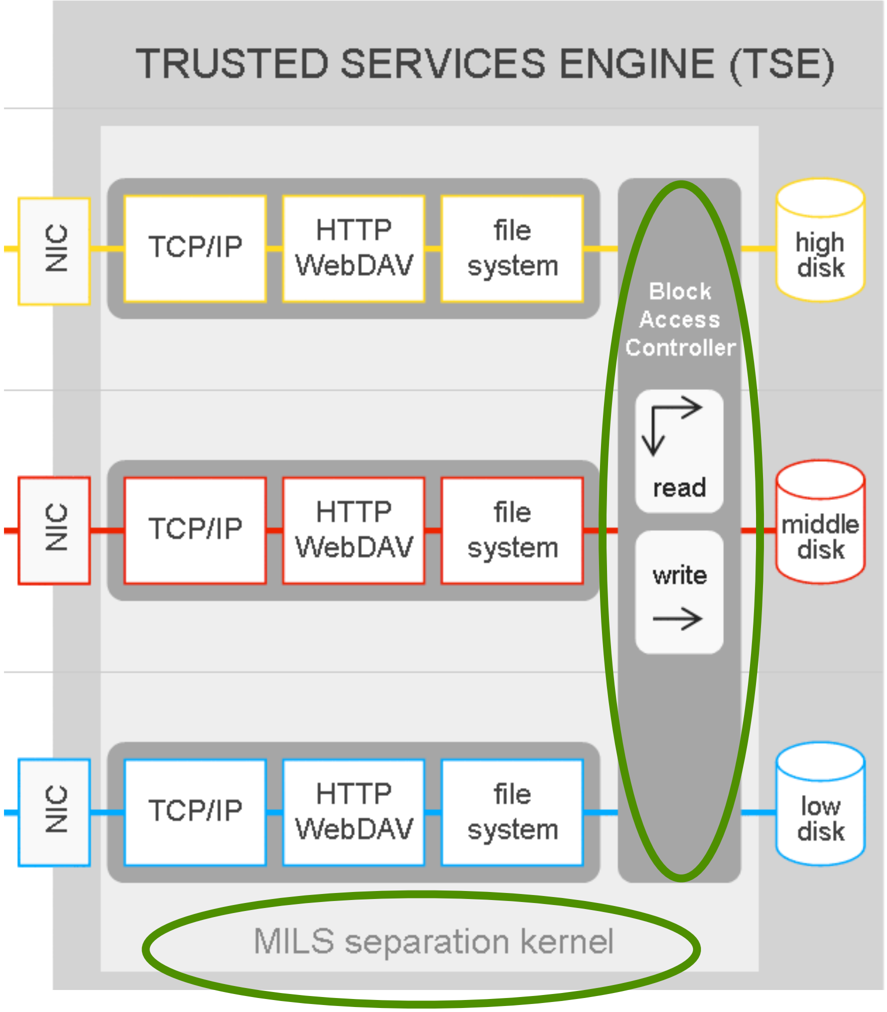
High users/applications see Integrated web/filestore with low and high content together

Low users/applications see simple web/filestore with low content only

Internal Architecture

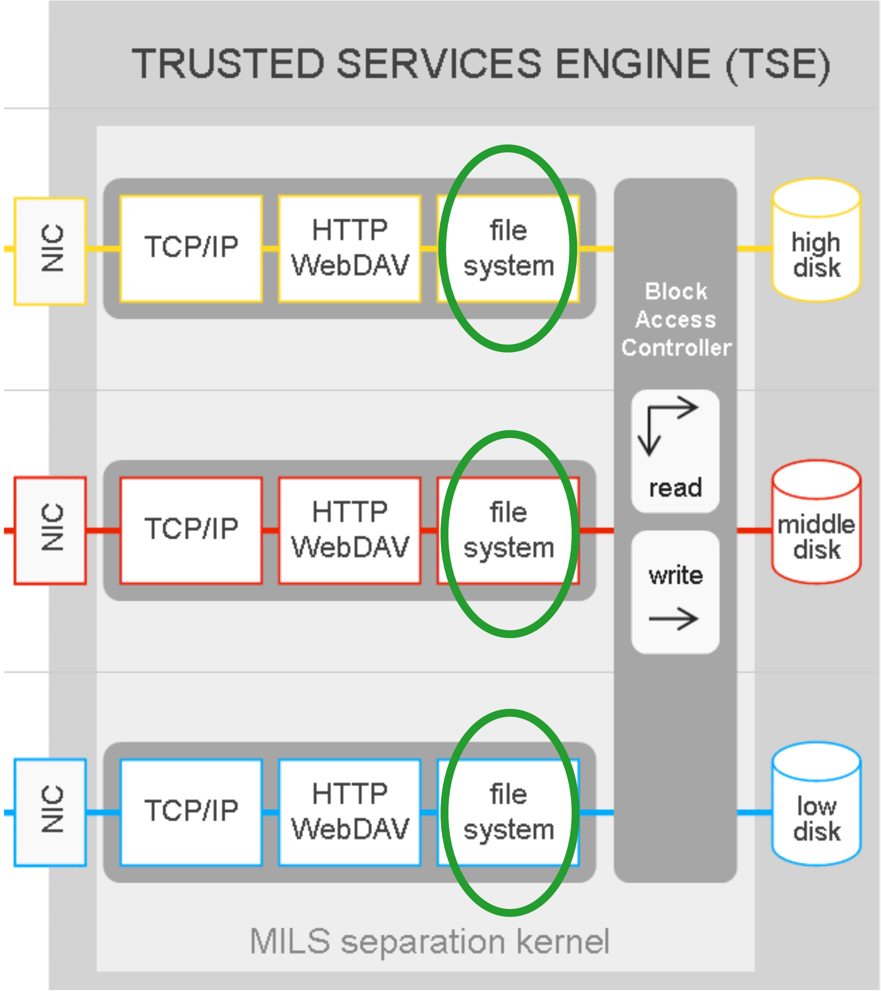
MILS = Multiple Independent Levels of Security

1. Factor the security architecture
2. Minimize the number of components requiring high assurance
3. Keep each as simple as possible
4. Use formal methods in critical places



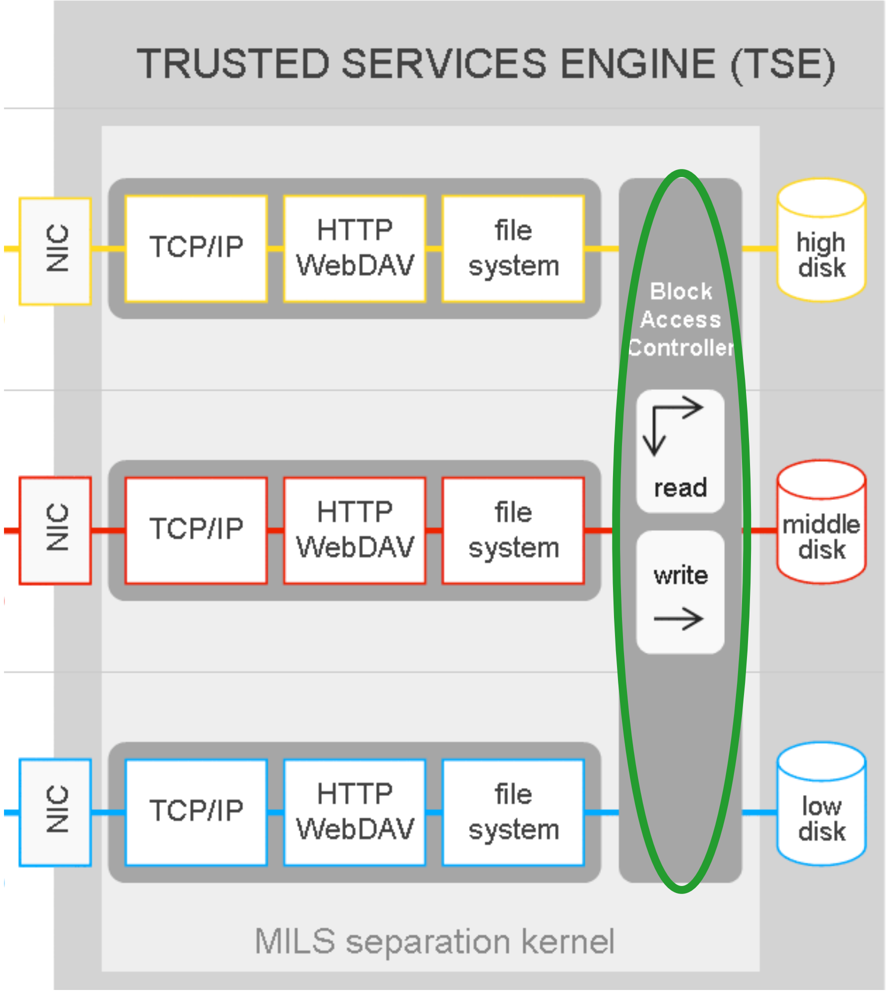
File System Characteristics

- Single-level file systems access multiple disk drives
  - Drives and their firmware are outside the TCB
- Read access from high must be invisible to low
  - No locking (hi-lo channel)
  - No abort/retry (lo-hi denial of service)
- No existing file system would work
  - Traditional cache coherency is infeasible across security boundaries
  - Designed WFFS (wait-free file system)



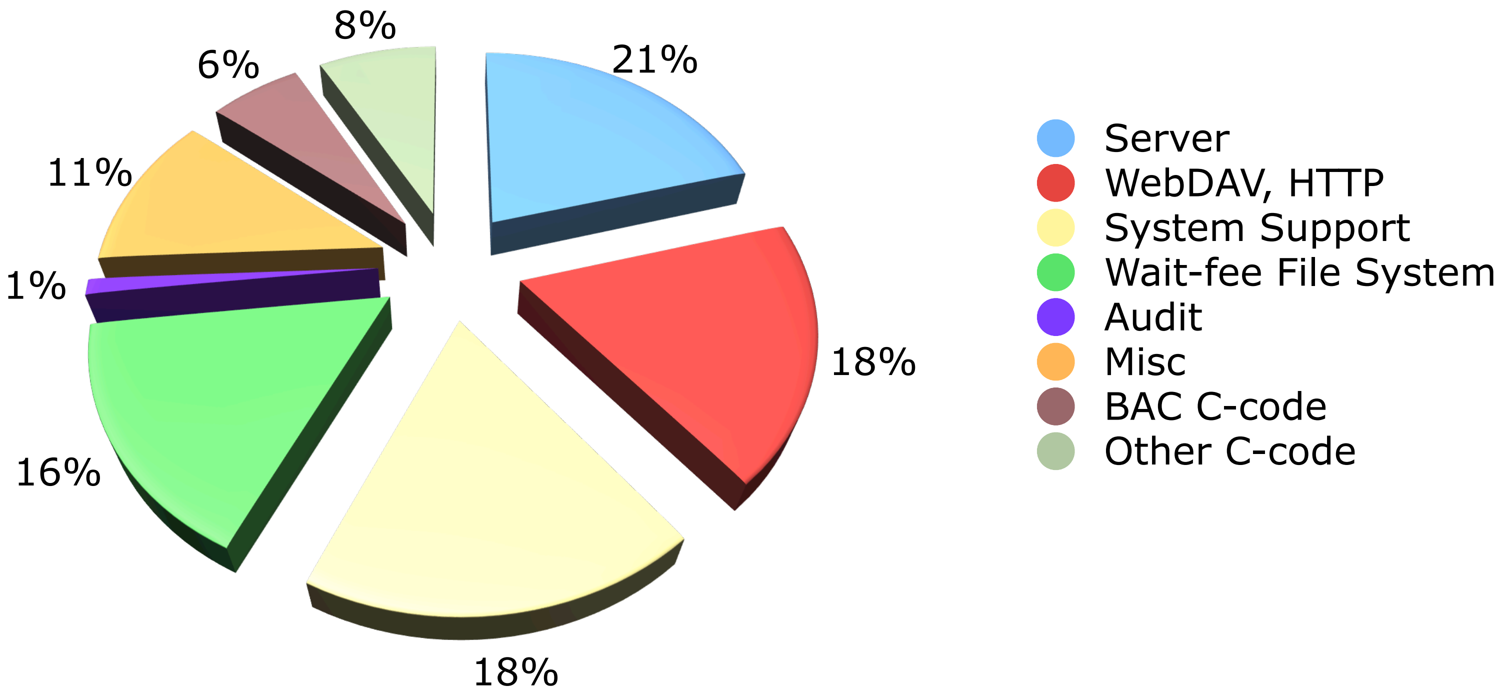
Block Access Controller

- BAC directs accesses across multiple disk drives
  - Block-level requests
  - Request queuing
- BAC spans partitions at multiple levels
  - High assurance component
  - Eliminate data channels between levels
  - Control timing channels between levels
- Modeled in Haskell
  - Proven in Isabelle
  - Implemented in simple C !



galois

Line Count Breakdown of Web-Server



Total: 54KLOC

Haskell

- Flexibility of Haskell enable substantial changes
  - Authentication and a plugin architecture added after the system was essentially finished as web server
  - Multi-disk buffer block cache added to our file system after the file system was complete
- FFI worked well and reliably
  - The low-level access to the BAC
  - The interface to the SSL library written in C
- Modules system mostly worked well
  - Would have liked a way to manage imports more flexibly
- Heap and RTS
  - Prohibitively complex for creating secure components

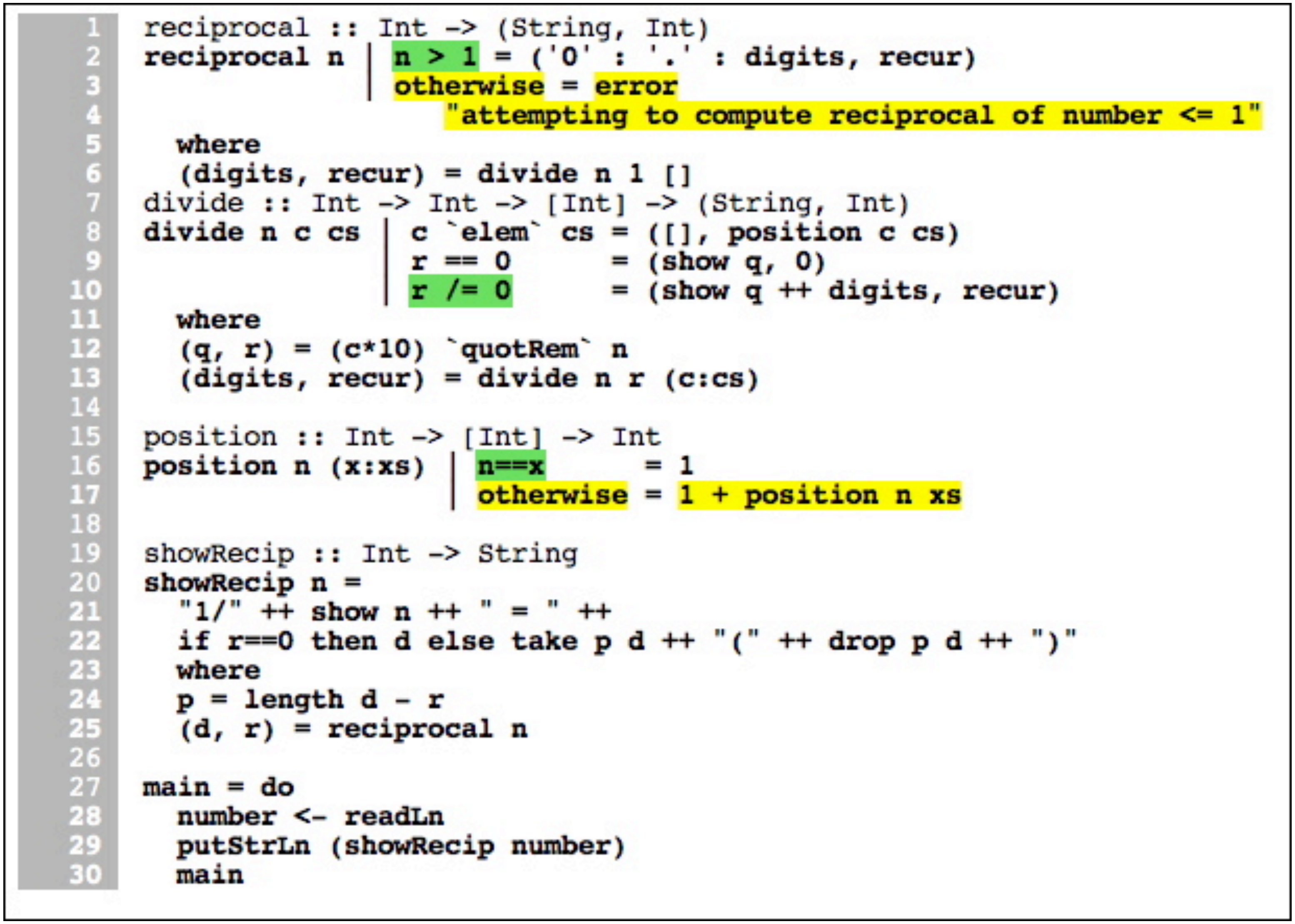
Feedback on Haskell

- “My favorite pro: ease of maintenance! Change the data type and let the compiler walk you through the entire code base pointing to every single place you need to worry about.”
- “GHC got quite slow at one point. We had generated programs whose datatypes has ~200 constructors, with about the same number of type parameters. The took about 30 minutes to compile.”
- “We had 30+ methods to access the model. Newtype deriving was great for generating all these components automatically.”

Haskell Tools

- Profiler guided all the performance improvements
- Testing by both QuickCheck and HUnit
  - Many tests were IO based, used lots of HUnit tests
  - HPC developed to address coverage needs; not yet used in detail
- Tried to use Cabal to put multiple packages into a system hierarchy
  - Had to make a complex build system that had to figure out dependencies
  - Also had to customize Cabal also to allow build-local packages
- Haddock
  - Haddock lets you add comments to types, generating HTML docs
  - Worked beautifully!

Example Coverage Markup



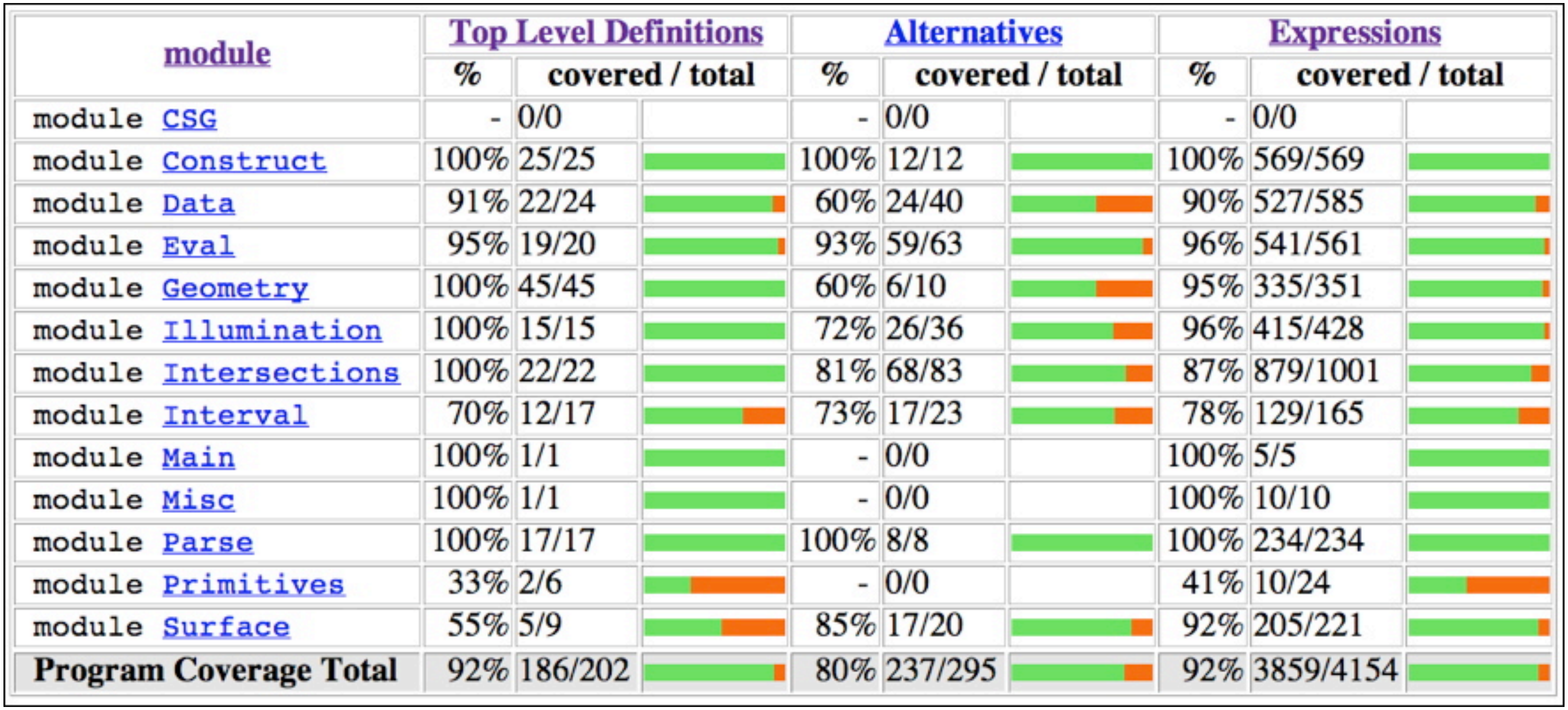
```
reciprocal :: Int -> (String, Int)
reciprocal n | n > 1 = ('0' : '.' : digits, recur)
             | otherwise = error
                  "attempting to compute reciprocal of number <= 1"
  where
  (digits, recur) = divide n 1 []
divide :: Int -> Int -> [Int] -> (String, Int)
divide n c cs | c `elem` cs = ([], position c cs)
              | r == 0      = (show q, 0)
              | r /= 0      = (show q ++ digits, recur)
  where
  (q, r) = (c*10) `quotRem` n
  (digits, recur) = divide n r (c:cs)

position :: Int -> [Int] -> Int
position n (x:xs) | n==x      = 1
                  | otherwise = 1 + position n xs

showRecip :: Int -> String
showRecip n =
  "1/" ++ show n ++ " = " ++
  if r==0 then d else take p d ++ "(" ++ drop p d ++ ")"
  where
  p = length d - r
  (d, r) = reciprocal n

main = do
  number <- readLn
  putStrLn (showRecip number)
  main
```

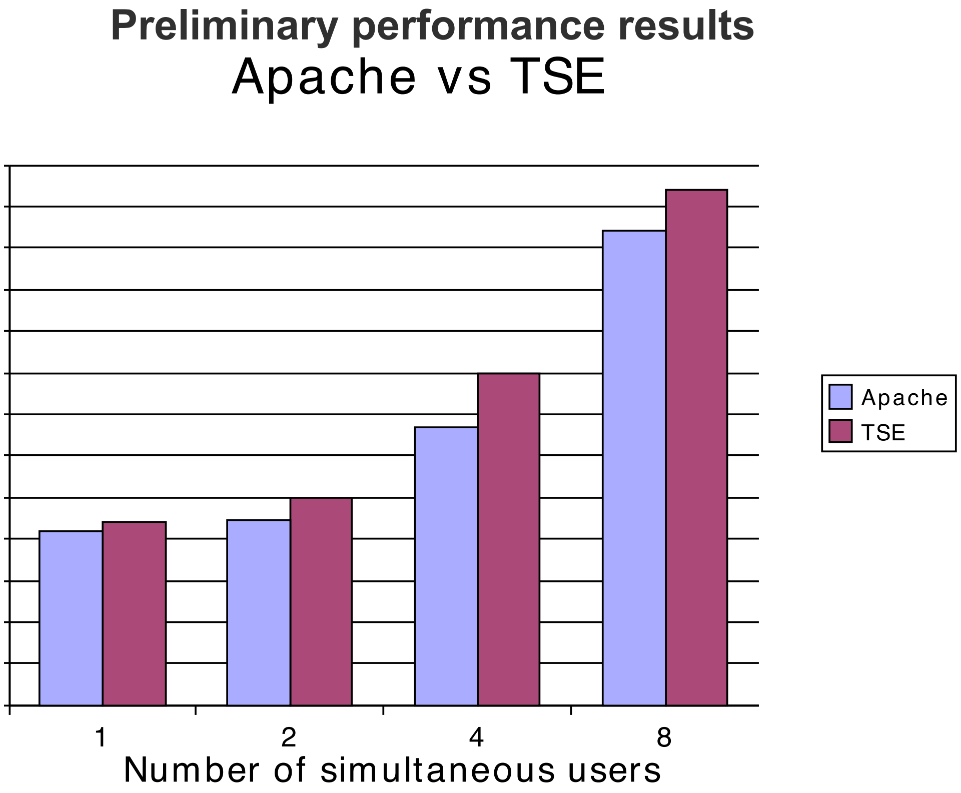
Haskell Program Coverage Dashboard



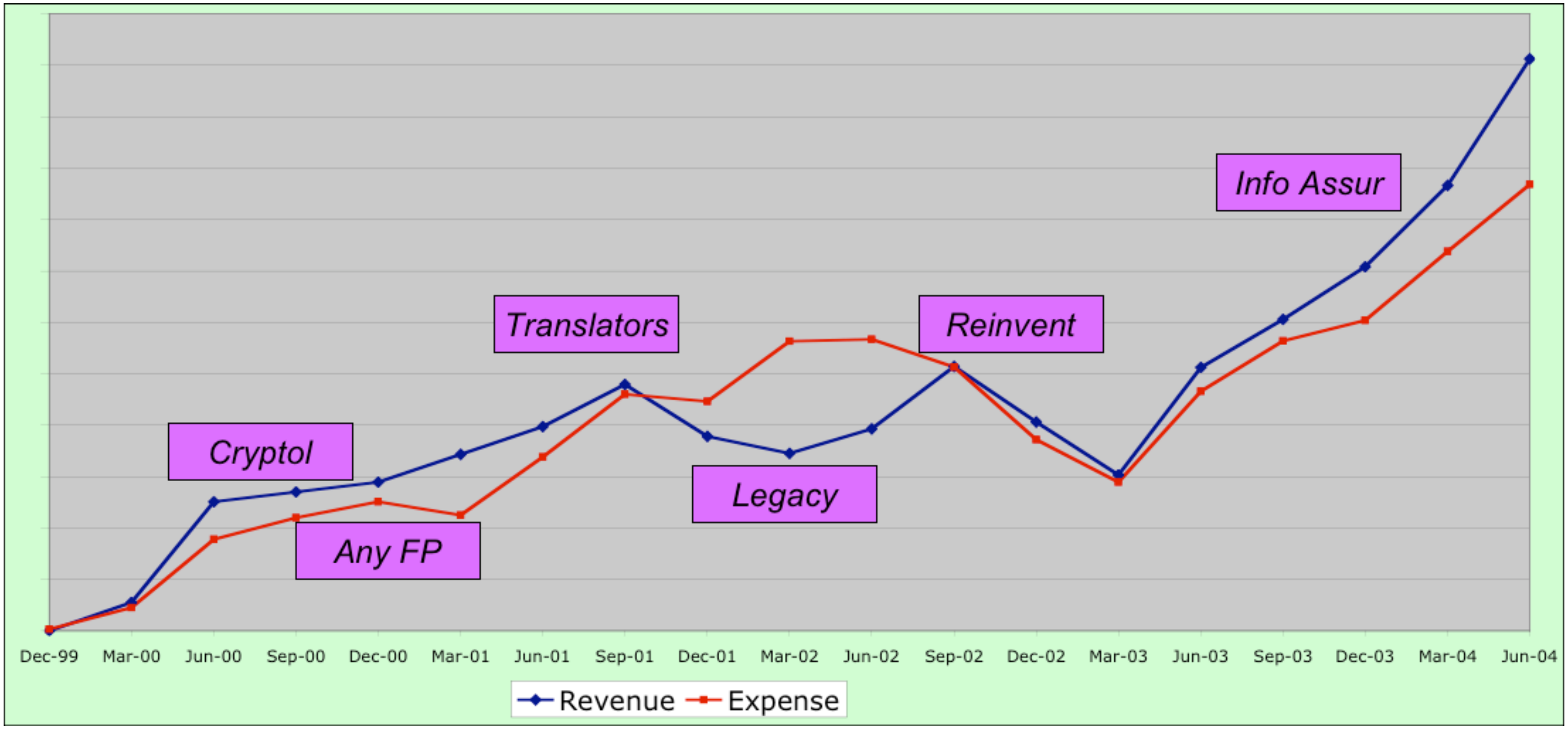
module	Top Level Definitions			Alternatives			Expressions		
	%	covered / total		%	covered / total		%	covered / total	
module CSG	-	0/0		-	0/0		-	0/0	
module Construct	100%	25/25	🟩	100%	12/12	🟩	100%	569/569	🟩
module Data	91%	22/24	🟩🟥	60%	24/40	🟩🟥	90%	527/585	🟩🟥
module Eval	95%	19/20	🟩🟥	93%	59/63	🟩🟥	96%	541/561	🟩🟥
module Geometry	100%	45/45	🟩	60%	6/10	🟩🟥	95%	335/351	🟩🟥
module Illumination	100%	15/15	🟩	72%	26/36	🟩🟥	96%	415/428	🟩🟥
module Intersections	100%	22/22	🟩	81%	68/83	🟩🟥	87%	879/1001	🟩🟥
module Interval	70%	12/17	🟩🟥	73%	17/23	🟩🟥	78%	129/165	🟩🟥
module Main	100%	1/1	🟩	-	0/0		100%	5/5	🟩
module Misc	100%	1/1	🟩	-	0/0		100%	10/10	🟩
module Parse	100%	17/17	🟩	100%	8/8	🟩	100%	234/234	🟩
module Primitives	33%	2/6	🟩🟥	-	0/0		41%	10/24	🟩🟥
module Surface	55%	5/9	🟩🟥	85%	17/20	🟩🟥	92%	205/221	🟩🟥
Program Coverage Total	92%	186/202	🟩🟥	80%	237/295	🟩🟥	92%	3859/4154	🟩🟥

Performance

- Laziness
  - “To a first approximation, strictness versus laziness didn’t matter squat.” Andy Gill, key developer
- Most early performance problems were with manipulation of binaries
  - Original bhPut copied Binary objects byte-by-byte
  - Modified version used clib’s memmove



Focus, focus, focus



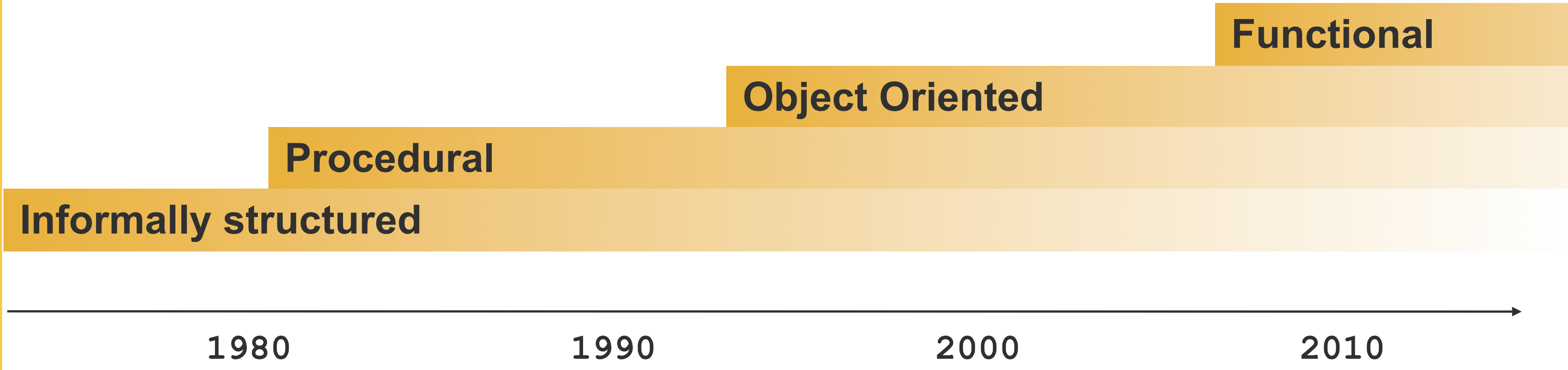
Galois Today

- 40 employees, primarily technical/engineering
  - Flat organization: a “collaborative web”
  - Stable and profitable, growing diversification of clients
- Products
  - Translator for chip-test programs
  - AIM development environment
  - Cryptol system
  - Flexible syntax front-end
- In beta
  - Cross-domain network filestore
  - Tearline federation of media-wikis
  - Crypto IP cores on FPGAs
  - OS and virtualization models

Technology conclusion

- Haskell gave us the engineering freedom to build systems right
  - Implement big new capabilities like the file-system
  - Performance is great
  - Concurrency was really easy to use, multi-core for free
  - Straightforward to enable interaction with non-Haskell parts
  - Types are wonderful
- Big systems could benefit from better Haskell infrastructure
  - Flexible module import
  - Compilation manager
- Security
  - Cannot build high-security components directly in Haskell because of the runtime system and heap, but still good for modeling
  - Really could do with a better handle on space usage

Era of Functional Languages?



Functional

Object Oriented

Procedural

Informally structured

1980

1990

2000

2010