sociomantic

MAY 2013

METAPROGRAMMING IN THE REAL WORLD

DON CLUGSTON

FROM RESEARCH MODE TO PRODUCTION

- My experience with Solar Photovotaics

FROM RESEARCH MODE TO PRODUCTION

- My experience with Solar Photovotaics
    - 1995 : Spacecraft and Hippies

FROM RESEARCH MODE TO PRODUCTION

- My experience with Solar Photovotaics
  - 1995 : Spacecraft and Hippies
  - 2007 : NYSE, $Billion

FROM RESEARCH MODE TO PRODUCTION

- My experience with Solar Photovotaics
  - 1995 : Spacecraft and Hippies
  - 2007 : NYSE, $Billion
  - 2011 : Commodity market

FROM RESEARCH MODE TO PRODUCTION

- My experience with Solar Photovotaics
  - 1995 : Spacecraft and Hippies
  - 2007 : NYSE, $Billion
  - 2011 : Commodity market

- Early adopters show where your guesses were wrong!

SOCIOMANTIC LABS GMBH

- Founded 2009, Berlin by 3 PhDs

SOCIOMANTIC LABS GMBH

- Founded 2009, Berlin by 3 PhDs
- Real-time bidding for online advertising

SOCIOMANTIC LABS GMBH

- Founded 2009, Berlin by 3 PhDs
- Real-time bidding for online advertising
- Technology based

SOCIOMANTIC LABS GMBH

- Founded 2009, Berlin by 3 PhDs
- Real-time bidding for online advertising
- Technology based
  - Core technology is 100% D

SOCIOMANTIC LABS GMBH

- Founded 2009, Berlin by 3 PhDs
- Real-time bidding for online advertising
- Technology based
  - Core technology is 100% D
- Expanding globally

SOCIOMANTIC LABS GMBH

- Founded 2009, Berlin by 3 PhDs
- Real-time bidding for online advertising
- Technology based
  - Core technology is 100% D
- Expanding globally
  - Serving 50+ markets on 6 continents

SOCIOMANTIC LABS GMBH

- Founded 2009, Berlin by 3 PhDs
- Real-time bidding for online advertising
- Technology based
  - Core technology is 100% D
- Expanding globally
  - Serving 50+ markets on 6 continents
  - Offices in 9 countries

SOCIOMANTIC LABS GMBH

- Founded 2009, Berlin by 3 PhDs
- Real-time bidding for online advertising
- Technology based
  - Core technology is 100% D
- Expanding globally
  - Serving 50+ markets on 6 continents
  - Offices in 9 countries
  - 100+ employees, 25+ nationalities

SOCIOMANTIC LABS GMBH

- Founded 2009, Berlin by 3 PhDs
- Real-time bidding for online advertising
- Technology based
  - Core technology is 100% D
- Expanding globally
  - Serving 50+ markets on 6 continents
  - Offices in 9 countries
  - 100+ employees, 25+ nationalities
- Profitable

SOCIOMANTIC LABS GMBH

- Founded 2009, Berlin by 3 PhDs
- Real-time bidding for online advertising
- Technology based
  - Core technology is 100% D
- Expanding globally
  - Serving 50+ markets on 6 continents
  - Offices in 9 countries
  - 100+ employees, 25+ nationalities
- Profitable
  - Growth based entirely on revenue

REAL-TIME BIDDING

- User visits web page

REAL-TIME BIDDING

- User visits web page
- While it loads, website auctions an ad space

REAL-TIME BIDDING

- User visits web page
- While it loads, website auctions an ad space
- We bid on behalf of our advertisers

REAL-TIME BIDDING

- User visits web page
- While it loads, website auctions an ad space
- We bid on behalf of our advertisers
- Highest bidder gets to show their ad in the space

REAL-TIME BIDDING

- User visits web page
- While it loads, website auctions an ad space
- We bid on behalf of our advertisers
- Highest bidder gets to show their ad in the space
- Bids must be placed within 50 milliseconds

REAL-TIME BIDDING

- User visits web page
- While it loads, website auctions an ad space
- We bid on behalf of our advertisers
- Highest bidder gets to show their ad in the space
- Bids must be placed within 50 milliseconds
  - Including internet latency

REAL-TIME BIDDING

- User visits web page
- While it loads, website auctions an ad space
- We bid on behalf of our advertisers
- Highest bidder gets to show their ad in the space
- Bids must be placed within 50 milliseconds
  - Including internet latency
- Billions of auctions per day

BIG DATA

- Must calculate how much this ad space is worth

BIG DATA

- Must calculate how much this ad space is worth
- Bid accuracy improves with more data

BIG DATA

- Must calculate how much this ad space is worth
- Bid accuracy improves with more data
    - Terabytes/day

BIG DATA

- Must calculate how much this ad space is worth
- Bid accuracy improves with more data
  - Terabytes/day
- Relational databases too slow + don´t scale

BIG DATA

- Must calculate how much this ad space is worth
- Bid accuracy improves with more data
  - Terabytes/day
- Relational databases too slow + don´t scale
- Everyone else uses an off-the shelf NoSQL product

BIG DATA

- Must calculate how much this ad space is worth
- Bid accuracy improves with more data
  - Terabytes/day
- Relational databases too slow + don´t scale
- Everyone else uses an off-the shelf NoSQL product
  - and works around the speed bottlenecks

BIG DATA

- Must calculate how much this ad space is worth
- Bid accuracy improves with more data
  - Terabytes/day
- Relational databases too slow + don´t scale
- Everyone else uses an off-the shelf NoSQL product
  - and works around the speed bottlenecks
- But we created an intrinsically fast solution, using D

BIG DATA

- Must calculate how much this ad space is worth
- Bid accuracy improves with more data
  - Terabytes/day
- Relational databases too slow + don´t scale
- Everyone else uses an off-the shelf NoSQL product
  - and works around the speed bottlenecks
- But we created an intrinsically fast solution, using D
- 50 milliseconds (minus net latency) to place a bid

BIG DATA

- Must calculate how much this ad space is worth
- Bid accuracy improves with more data
    - Terabytes/day
- Relational databases too slow + don´t scale
- Everyone else uses an off-the shelf NoSQL product
    - and works around the speed bottlenecks
- But we created an intrinsically fast solution, using D
- 50 milliseconds (minus net latency) to place a bid
    - Typical hard disk seek time is 9 ms

BIG DATA

- Must calculate how much this ad space is worth
- Bid accuracy improves with more data
  - Terabytes/day
- Relational databases too slow + don´t scale
- Everyone else uses an off-the shelf NoSQL product
  - and works around the speed bottlenecks
- But we created an intrinsically fast solution, using D
- 50 milliseconds (minus net latency) to place a bid
  - Typical hard disk seek time is 9 ms
  - For most bids we achieve <= **2** ms

OUR TECHNOLOGY STACK

- Tango-based runtime (modified), own libraries

OUR TECHNOLOGY STACK

- Tango-based runtime (modified), own libraries
    - Avoid ALL heap activity

OUR TECHNOLOGY STACK

- Tango-based runtime (modified), own libraries
  - Avoid ALL heap activity
- Fiber-based concurrency (not threads)

OUR TECHNOLOGY STACK

- Tango-based runtime (modified), own libraries
  - Avoid ALL heap activity
- Fiber-based concurrency (not threads)
- ‘Swarm’ In-memory Distributed Hash Table

OUR TECHNOLOGY STACK

- Tango-based runtime (modified), own libraries
  - Avoid ALL heap activity
- Fiber-based concurrency (not threads)
- 'Swarm' In-memory Distributed Hash Table
- Data stored in D format, no conversion

OUR TECHNOLOGY STACK

- Tango-based runtime (modified), own libraries
    - Avoid ALL heap activity
- Fiber-based concurrency (not threads)
- ‘Swarm’ In-memory Distributed Hash Table
- Data stored in D format, no conversion
- All processes stream-based and completely scalable

WHY D?

- Direct binding to C libraries

WHY D?

- Direct binding to C libraries
- Array slices

WHY D?

- Direct binding to C libraries
- Array slices
  - Avoid heap activity, but stay correct

WHY D?

- Direct binding to C libraries
- Array slices
  - Avoid heap activity, but stay correct
- Painless compile-time programming

WHY D?

- Direct binding to C libraries
- Array slices
  - Avoid heap activity, but stay correct
- Painless compile-time programming
  - eg, for serialization

D METAPROGRAMMING IN 2001

- Features to drop from C++
    - C source code compatibility
    - Link compatibility with C++
    - Multiple inheritance
    - Preprocessor
    - Templates

-- Walter Bright, "D Spec Draft 1", (Aug 2001)

D METAPROGRAMMING IN 2005

- Templates!

D METAPROGRAMMING IN 2005

- Templates!
- static if, static assert

D METAPROGRAMMING IN 2005

- Templates!
- static if, static assert
- Some reflection -- is() expressions

D METAPROGRAMMING IN 2005

- Templates!
- static if, static assert
- Some reflection -- is() expressions
- Still defensive w.r.t C++

D METAPROGRAMMING IN 2005

- Templates!
- static if, static assert
- Some reflection -- is() expressions
- Still defensive w.r.t C++
    - "If a language can capture 90% of the power of C++ with 10% of its complexity, I argue that is a worthwhile tradeoff." - DMD FAQ

D METAPROGRAMMING IN 2007

- Improved constant folding

D METAPROGRAMMING IN 2007

- Improved constant folding
- Compile Time Function Execution (CTFE)

D METAPROGRAMMING IN 2007

- Improved constant folding
- Compile Time Function Execution (CTFE)
- string mixins

D METAPROGRAMMING IN 2007

- Improved constant folding
- Compile Time Function Execution (CTFE)
- string mixins
- stringof

D METAPROGRAMMING IN 2013

- Template constraints

D METAPROGRAMMING IN 2013

- Template constraints
- __traits (just as ugly as is() expressions!)

D METAPROGRAMMING IN 2013

- Template constraints
- __traits (just as ugly as is() expressions!)
- alias this

D METAPROGRAMMING IN 2013

- Template constraints
- __traits (just as ugly as is() expressions!)
- alias this
- opDispatch

D METAPROGRAMMING IN 2013

- Template constraints
- __traits (just as ugly as is() expressions!)
- alias this
- opDispatch
- Dramatic implementation improvements

WHY THE HISTORY MATTERS

- We got here by incremental improvements

WHY THE HISTORY MATTERS

- We got here by incremental improvements
- Programmers follow the same learning curve

WHY THE HISTORY MATTERS

- We got here by incremental improvements
- Programmers follow the same learning curve
- Metaprogramming is an unexpected strength of D

WHY THE HISTORY MATTERS

- We got here by incremental improvements
- Programmers follow the same learning curve
- Metaprogramming is an unexpected strength of D
- We still have some detritus

RETURN ON INVESTMENT (ROI)

- (Benefit – Cost) / Cost

RETURN ON INVESTMENT (ROI)

- (Benefit - Cost) / Cost
- At what time does this become positive?

RETURN ON INVESTMENT (ROI)

- (Benefit - Cost) / Cost
- At what time does this become positive?
- The time until you obtain benefit can be as important as the cost!

RETURN ON INVESTMENT (ROI)

- (Benefit - Cost) / Cost
- At what time does this become positive?
- The time until you obtain benefit can be as important as the cost!
- Benefit > Cost at t = infinity is not enough!

RETURN ON INVESTMENT (ROI)

- (Benefit - Cost) / Cost
- At what time does this become positive?
- The time until you obtain benefit can be as important as the cost!
- Benefit > Cost at t = infinity is not enough!
- Who gets the benefit?

BACKWARDS COMPATIBILITY - EXPECTATION

- Language changes must NEVER break code

BACKWARDS COMPATIBILITY – EXPECTATION

- Language changes must NEVER break code
- Except in extreme cases

BACKWARDS COMPATIBILITY – EXPERIENCE

- Breaking code is an up-front cost

BACKWARDS COMPATIBILITY – EXPERIENCE

- Breaking code is an up-front cost
- But keeping mis-features is worse!

BACKWARDS COMPATIBILITY - EXPERIENCE

- Breaking code is an up-front cost
- But keeping mis-features is worse!
  - an on-going cost

BACKWARDS COMPATIBILITY – EXPERIENCE

- Breaking code is an up-front cost
- But keeping mis-features is worse!
  - an on-going cost
- Gratuitous name changes have very poor ROI

BACKWARDS COMPATIBILITY – EXPERIENCE

- Breaking code is an up-front cost
- But keeping mis-features is worse!
  - an on-going cost
- Gratuitous name changes have very poor ROI
- If the benefit is instant, any cost is OK

BACKWARDS COMPATIBILITY – EXPERIENCE

- Breaking code is an up-front cost
- But keeping mis-features is worse!
  - an on-going cost
- Gratuitous name changes have very poor ROI
- If the benefit is instant, any cost is OK
  - eg if it catches a bug

BACKWARDS COMPATIBILITY – EXPERIENCE

- Breaking code is an up-front cost
- But keeping mis-features is worse!
  - – an on-going cost
- Gratuitous name changes have very poor ROI
- If the benefit is instant, any cost is OK
  - – eg if it catches a bug
- Breaking changes can be met with enthusiasm!

METAPROGRAMMING

- Expectation

METAPROGRAMMING

- Expectation
  - Easier than in C++

METAPROGRAMMING

- Expectation
  - Easier than in C++
  - But still only used by wizards, in libraries

METAPROGRAMMING

- Expectation
  - Easier than in C++
  - But still only used by wizards, in libraries
- Experience

METAPROGRAMMING

- Expectation
  - Easier than in C++
  - But still only used by wizards, in libraries
- Experience
  - Used even in in application code!

METAPROGRAMMING

- Expectation
  - Easier than in C++
  - But still only used by wizards, in libraries
- Experience
  - Used even in in application code!
  - Used even by new D programmers!

METAPROGRAMMING

- Expectation
  - Easier than in C++
  - But still only used by wizards, in libraries
- Experience
  - Used even in in application code!
  - Used even by new D programmers!
  - Entry level is very low

METAPROGRAMMING

- Expectation
  - Easier than in C++
  - But still only used by wizards, in libraries

- Experience
  - Used even in in application code!
  - Used even by new D programmers!
  - Entry level is very low
    - ´static if´ is instantly understood

METAPROGRAMMING

- Expectation
  - Easier than in C++
  - But still only used by wizards, in libraries
- Experience
  - Used even in in application code!
  - Used even by new D programmers!
  - Entry level is very low
    - 'static if' is instantly understood
    - ROI is excellent

METAPROGRAMMING

- Expectation
  - Easier than in C++
  - But still only used by wizards, in libraries
- Experience
  - Used even in in application code!
  - Used even by new D programmers!
  - Entry level is very low
    - 'static if' is instantly understood
    - ROI is excellent
- Improves programmer morale

ERROR MESSAGES

- Expectation

ERROR MESSAGES

- Expectation
  - Lowest importance of any type of compiler bug

ERROR MESSAGES

- Expectation
  - Lowest importance of any type of compiler bug
- Experience

ERROR MESSAGES

- Expectation
  - Lowest importance of any type of compiler bug
- Experience
  - Make advanced features seem simpler

ERROR MESSAGES

- Expectation
  - Lowest importance of any type of compiler bug
- Experience
  - Make advanced features seem simpler
  - Have a pedagogic role

ERROR MESSAGES

- Expectation
  - Lowest importance of any type of compiler bug
- Experience
  - Make advanced features seem simpler
  - Have a pedagogic role
  - Good error messages save time.. and time is money

ERROR MESSAGES

- Expectation
  - Lowest importance of any type of compiler bug
- Experience
  - Make advanced features seem simpler
  - Have a pedagogic role
  - Good error messages save time.. and time is money
  - Error messages are the reason we use statically-typed languages!

COMPILE TIME FUNCTION EXECUTION

- Expectation

COMPILE TIME FUNCTION EXECUTION

- Expectation
- Huge win! Used everywhere

COMPILE TIME FUNCTION EXECUTION

- Expectation
- Huge win! Used everywhere
- Subliminal metaprogramming!

COMPILE TIME FUNCTION EXECUTION

- Expectation
- Huge win! Used everywhere
- Subliminal metaprogramming!
- Increased power will increase adoption

COMPILE TIME FUNCTION EXECUTION

- Expectation
- Huge win! Used everywhere
- Subliminal metaprogramming!
- Increased power will increase adoption
  - Pointers, throw exceptions, …

COMPILE TIME FUNCTION EXECUTION

- Expectation
- Huge win! Used everywhere
- Subliminal metaprogramming!
- Increased power will increase adoption
  - Pointers, throw exceptions, …
- Experience

COMPILE TIME FUNCTION EXECUTION

- Expectation
- Huge win! Used everywhere
- Subliminal metaprogramming!
- Increased power will increase adoption
  - Pointers, throw exceptions, …
- Experience
- CTFE hardly gets used, because it´s too slow

COMPILE TIME FUNCTION EXECUTION

- Expectation
- Huge win! Used everywhere
- Subliminal metaprogramming!
- Increased power will increase adoption
  - Pointers, throw exceptions, ...
- Experience
- CTFE hardly gets used, because it´s too slow
  - Fast compilation is addictive!

COMPILE TIME FUNCTION EXECUTION

- Expectation
- Huge win! Used everywhere
- Subliminal metaprogramming!
- Increased power will increase adoption
  - Pointers, throw exceptions, …
- Experience
- CTFE hardly gets used, because it´s too slow
  - Fast compilation is addictive!
- Why isn´t it fast yet?

COMPILE TIME FUNCTION EXECUTION

- Expectation
- Huge win! Used everywhere
- Subliminal metaprogramming!
- Increased power will increase adoption
  - Pointers, throw exceptions, ...
- Experience
- CTFE hardly gets used, because it´s too slow
  - Fast compilation is addictive!
- Why isn´t it fast yet?
  - Because of the history

COMPILE TIME FUNCTION EXECUTION

- Expectation
- Huge win! Used everywhere
- Subliminal metaprogramming!
- Increased power will increase adoption
  - Pointers, throw exceptions, ...
- Experience
- CTFE hardly gets used, because it´s too slow
  - Fast compilation is addictive!
- Why isn´t it fast yet?
  - Because of the history
  - Many unintended dependencies

COMPILE TIME FUNCTION EXECUTION

- Expectation
- Huge win! Used everywhere
- Subliminal metaprogramming!
- Increased power will increase adoption
  - Pointers, throw exceptions, ...
- Experience
- CTFE hardly gets used, because it´s too slow
  - Fast compilation is addictive!
- Why isn´t it fast yet?
  - Because of the history
  - Many unintended dependencies
  - Front-end must be in a valid state!

Expectation

TUTORIALS

- Expectation
  - Tutorials are almost irrelevant

TUTORIALS

- Expectation
  - Tutorials are almost irrelevant
- Experience

TUTORIALS

- Expectation
  - Tutorials are almost irrelevant
- Experience
  - Absence of tutorials is an embarassment

COMPILER BUGS

- Much smaller problem than expected

COMPILER BUGS

- Much smaller problem than expected
- Template bugs rarely encountered in D1

COMPILER BUGS

- Much smaller problem than expected
- Template bugs rarely encountered in D1
- 64 bit code generation a nightmare

COMPILER BUGS

- Much smaller problem than expected
- Template bugs rarely encountered in D1
- 64 bit code generation a nightmare
  - But mostly a one-off cost borne by us

COMPILER BUGS

- Much smaller problem than expected
- Template bugs rarely encountered in D1
- 64 bit code generation a nightmare
  - But mostly a one-off cost borne by us
- Otherwise, IDE bugs much worse

SUMMARY

- D is moving out of research mode

SUMMARY

- D is moving out of research mode
  - We can no longer ignore implementation issues

SUMMARY

- D is moving out of research mode
  - We can no longer ignore implementation issues
- A Return-On-Investment model is useful

SUMMARY

- D is moving out of research mode
  - We can no longer ignore implementation issues
- A Return-On-Investment model is useful
  - D must deliver value in the near-term

SUMMARY

- D is moving out of research mode
  - We can no longer ignore implementation issues
- A Return-On-Investment model is useful
  - D must deliver value in the near-term
- Metaprogramming is a strength of D in the real world

SUMMARY

- D is moving out of research mode
  - We can no longer ignore implementation issues
- A Return-On-Investment model is useful
  - D must deliver value in the near-term
- Metaprogramming is a strength of D in the real world
  - D does deliver ROI for Sociomantic Labs

SUMMARY

- D is moving out of research mode
  - We can no longer ignore implementation issues
- A Return-On-Investment model is useful
  - D must deliver value in the near-term
- Metaprogramming is a strength of D in the real world
  - D does deliver ROI for Sociomantic Labs
  - But not yet in all areas

WE'RE HIRING!

www.sociomantic.com