Making it Easier:

Improving Design Productivity

MARK HOROWITZ

Lessons Learned from the 80's

Don't try to solve THE problem

- The complete problem is much too large
  - No one has the resources to do it

The problem is really many smaller problems

- Need to find natural interfaces
  - Create tools and methodologies for subproblems
- Users can then pick build "tool" sets from this work

- Clean interfaces can be
  - Between different abstraction levels
  - Along domain lines

Themes

We are interested in domain specific tools/languages

- Restricting application domain allows the tool to be more powerful
- Many of the tools are build around a domain

We are very interested in building constructors

- If systems are expensive to build, want to reuse them
  - This means we want the systems to be flexible
- Rather than building a specific instance
  - Create a tool that understands local rules
  - And can create instances from parameters

Application Frameworks

Will McGrath, Bjoern Hartmann

fabryq Statement

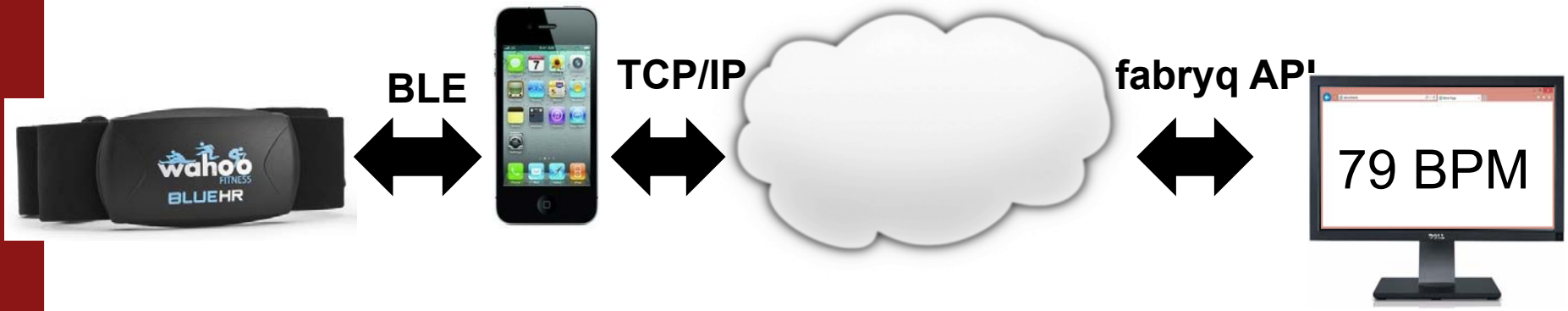
Many recent IoT applications are based on an Embedded-Gateway-Cloud (MGC) pattern

Programming complete MCG-based applications for connected devices is complicated and requires expertise in different languages on each tier (e.g. Python, Java, C)

fabryq Solution

THE FABRYQ PLATFORM LETS DEVELOPERS WRITE WEB APPS THAT CAN COMMUNICATE WITH MULTIPLE BLUETOOTH LOW ENERGY (BLE) DEVICES

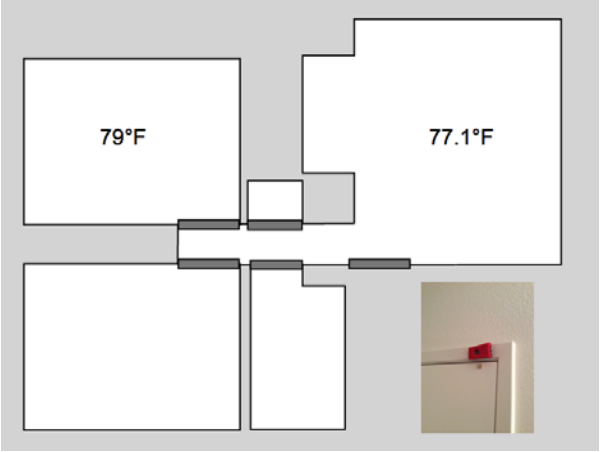
FABRYQ SUPPORTS RAPID MCG APPLICATION PROTOTYPING BY PROVIDING A REUSABLE SERVER AND MOBILE "SMART GATEWAY"



fabryq Results

APPLICATIONS DEVELOPED BY PILOT STUDY PARTICIPANTS AND THE RESEARCH TEAM DEMONSTRATED FABRYQ'S UTILITY

SUGGESTED A NEED FOR A SMOOTH WAY TO MOVE FROM PROTOTYPE TO PRODUCTION



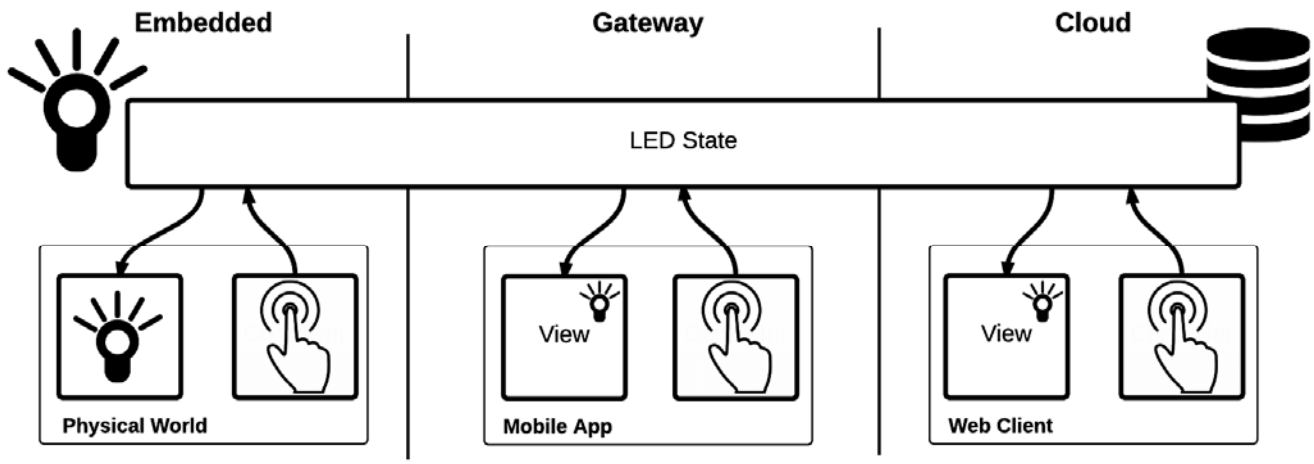
Published at EICS '15

Laurynas Riliskis, Phil Levis

Ravel

A FRAMEWORK THAT IN SINGLE LANGUAGE SUPPORTS WRITING APPLICATIONS WITH A DISTRIBUTED MODEL VIEW CONTROLLER PATTERN IN A DATA FLOW ARCHITECTURE.

THE DEVELOPER WRITES A SINGLE MODEL WITH ONE DATA SCHEMA; RAVEL GENERATES ALL NECESSARY MECHANISMS FOR MODEL SYNCHRONIZATION ACROSS ALL TIERS OF THE IOT APPLICATION.



Leveraging Good Interfaces

Existing IoT frameworks simplifies development by providing a complete ecosystem for embedded, gateway, cloud, communications, and security. Consequently, the application and any future updates are tight up to that particular framework. Any investment in extending ecosystem and optimizing would not carry over to a different platform.

Instead of providing a runtime, Ravel translates code and uses templates to generate code for the target platforms. Such as Arduino, Intel, Nordic, Android, Amazon Cloud, Django, etc.

Switching between supported frameworks and ecosystems becomes a push of a button.

Andrew Adams, Jonathan Ragan Kelley

Halide

Language created to make writing Image/Vision Apps Easier

Image Applications

- Need to do tons of simple operations
- These operations have high locality
  - Can be done fast and efficiently

Before Halide

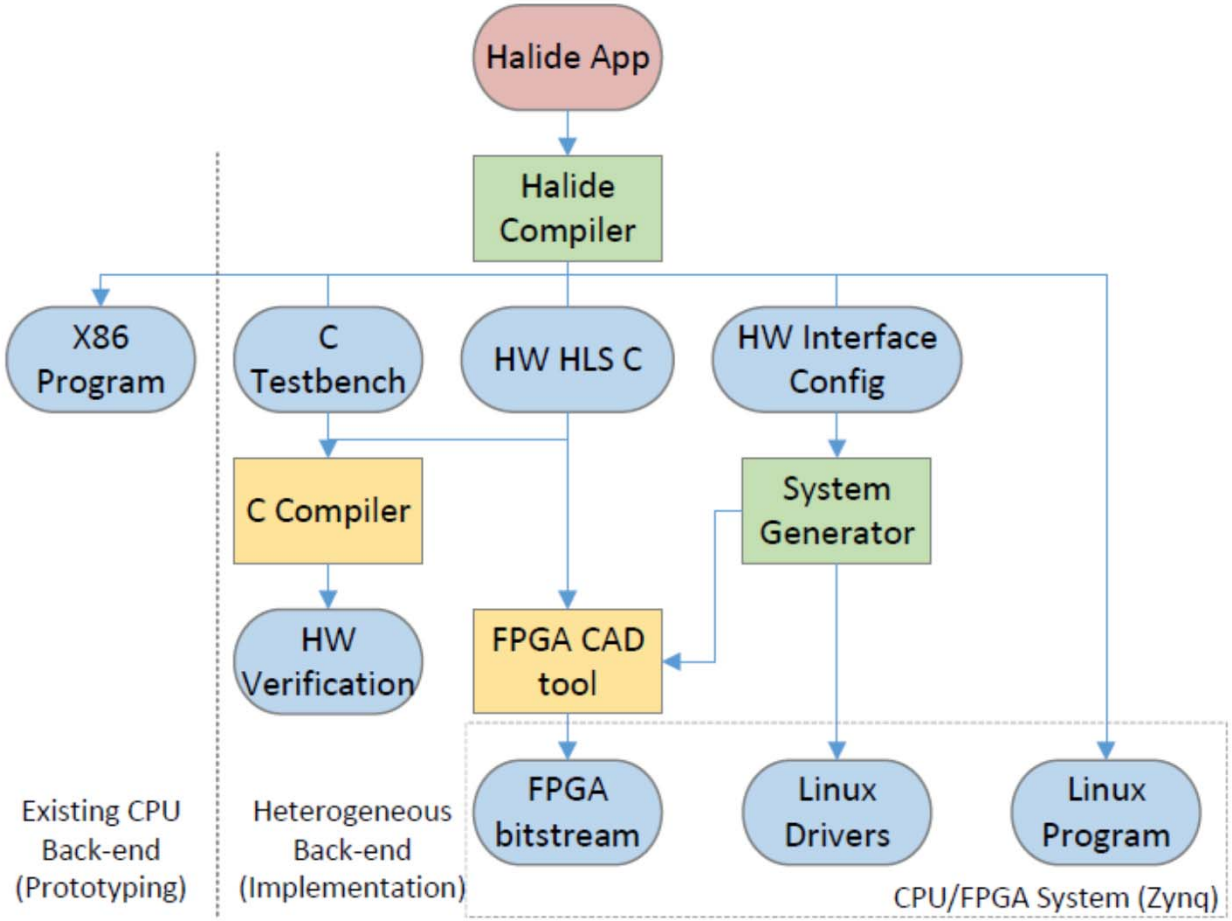
- Either wrote clean code
- Or wrote fast code that was hard to read/reuse

Halide's Big Idea

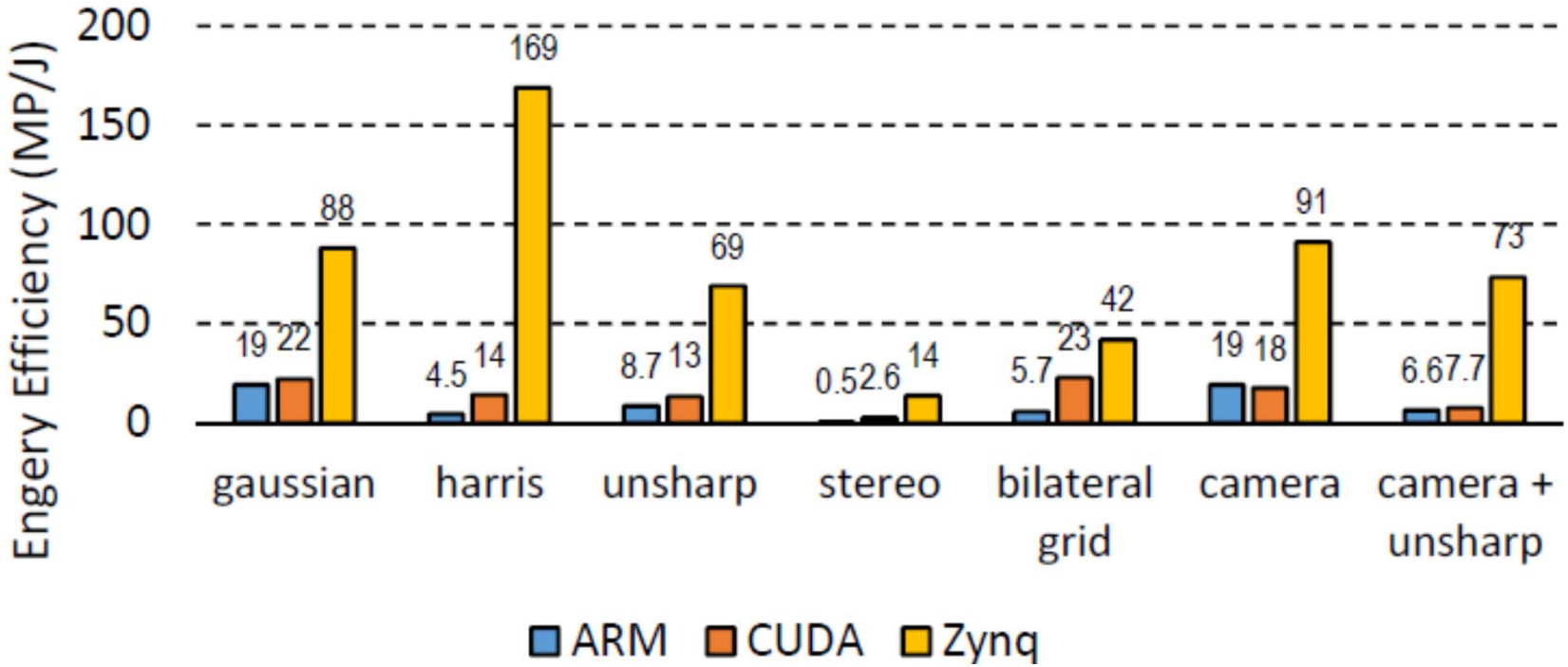
- Separate schedule of code from function of code

Jing Pu, Mark Horowitz

Halide to Zynq FGPA



Energy Results

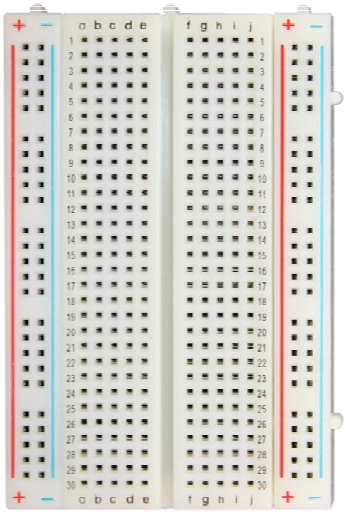


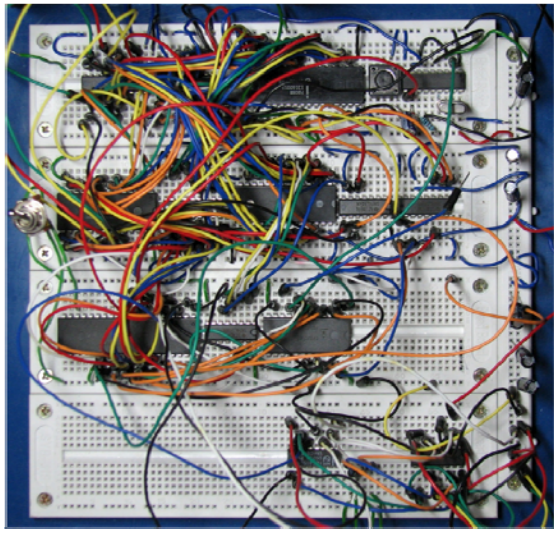
Hardware Platforms

Daniel Drew, Bjoern Hartmann

Toastboard Statement

WHILE SOLDERLESS BREADBOARDS ALLOW FOR QUICK CIRCUIT PROTOTYPING, NOVICE USERS CAN MAKE ERRORS THAT ARE DIFFICULT AND TIME-CONSUMING TO DIAGNOSE

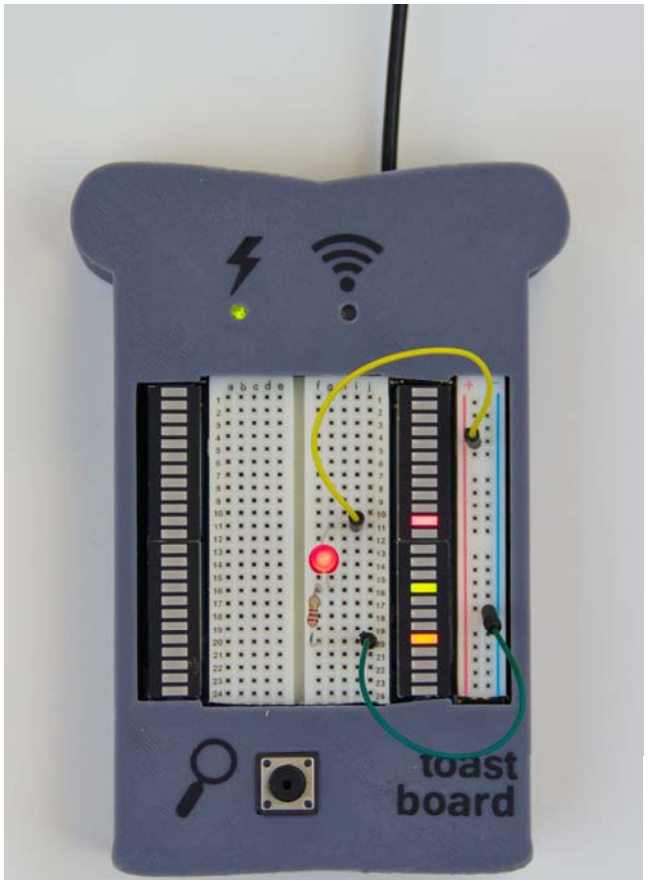




Toastboard Solution

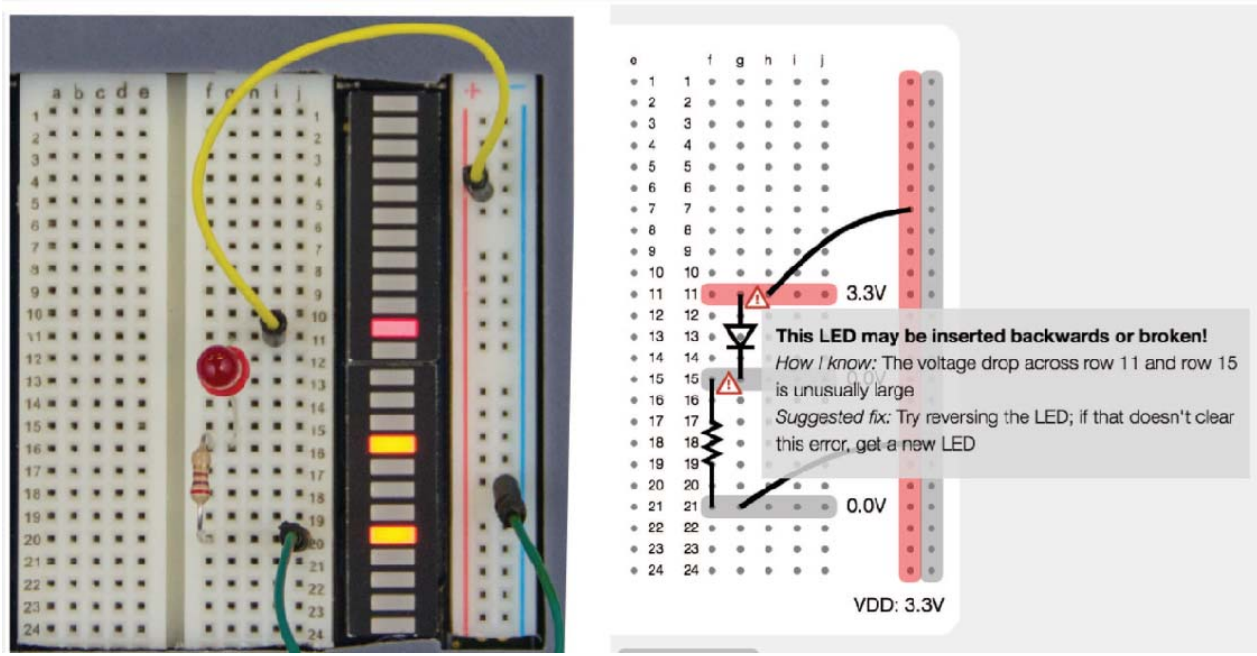
THE TOASTBOARD IS AN INSTRUMENTED BREADBOARD THAT CAN AUTOMATICALLY SENSE THE VOLTAGE ON ALL OF ITS ROWS

IT INCLUDES AN EXTENSIBLE SET OF AUTOMATED CHECKER FUNCTIONS THAT CAN DETECT AND EXPLAIN CIRCUIT ERRORS IN A USER-FRIENDLY WAY



Toastboard Results

A PRELIMINARY EVALUATION DEMONSTRATED THAT THE TOASTBOARD CAN HELP NOVICE USERS CORRECT COMMON CIRCUIT ISSUES WITH CHECKER FUNCTIONS.



Accepted to UIST

Shane Leonard, Phil Levis

Imix:
Enabling Next-Generation Research for Secure IoT

Standalone IoT system

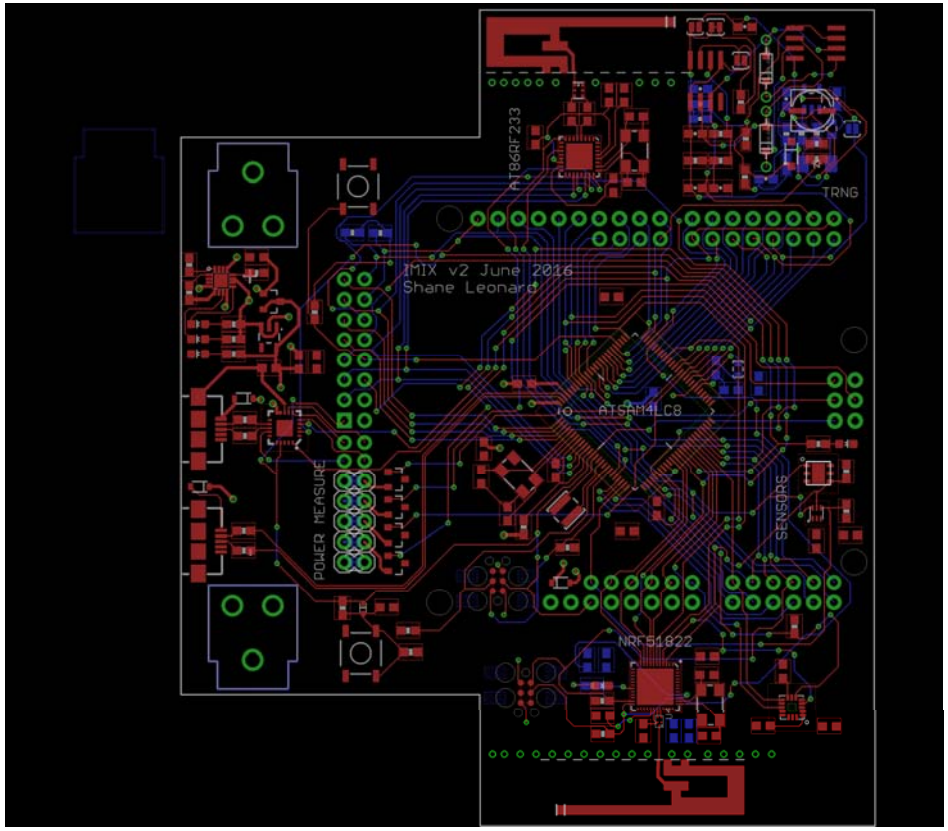
- Crypto, wireless, power, sensors, software

Maximize system visibility

- All internal signals exposed for debugging
- Power measurement header

Usability and control

- Fully open source
- Software backbone: Tock



Imix: Features List

Open source

Two wireless protocols (802.15.4 and BLE)

Powered by USB, Li-ion battery, or external voltage source

Integrated battery charger

Power measurement header for each subsystem

6-axis accelerometer/magnetometer, temperature sensor, light sensor

Ultra low power ARM Cortex-M4 processor with MPU

FTDI USB to Serial link

Native USB

Arduino-compatible

Compatible with Firestorm bootloader

True random number generator based on Lampert circuit

Runs Tock, a secure Rust OS for IoT

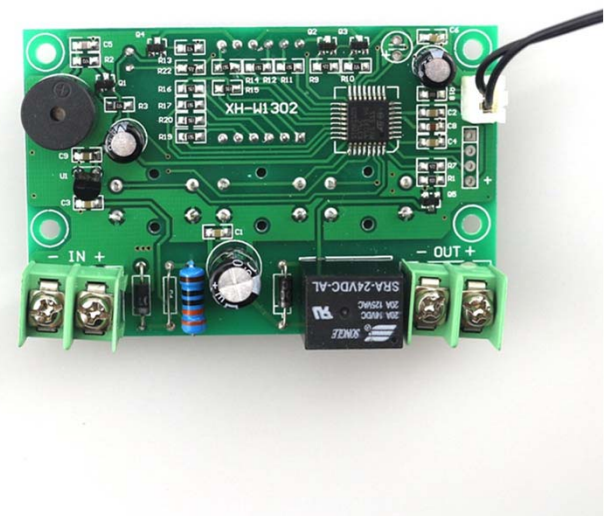
Rohit Ramesh, Prabal Dutta

Going Meta: Embedded Device Generation

- Compile high-level code into embedded hardware.
- Expand access to hardware development.
- Automate the development process for hobbyists
- Improve development tools for professionals.

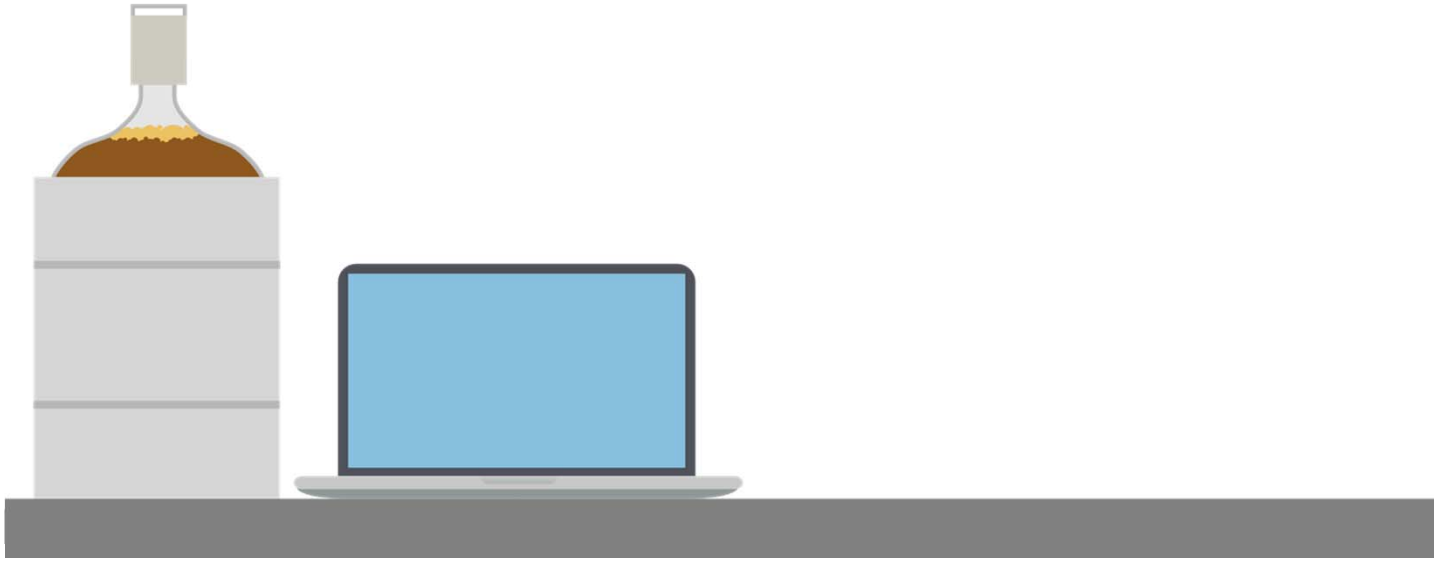
```
// Component Declarations
component thermometer = new Thermometer(immersion,
  min-temp <= 0c, max-temp >= 100c);
component heater      = new Heater(immersion, power > 10w);
component cooler      = new Cooler(immersion);

// Control Logic
fn main(){
  while(true){
    if(thermometer.temp() < 20c){
      // Water temp too low
      cooler.off();
      heater.on();
    } else if(thermometer.temp() > 20c){
      // Water temp too high
      heater.off();
      cooler.on();
    } else {
      // Water temp just right
      heater.off();
      cooler.off();
    }
  }
}
```

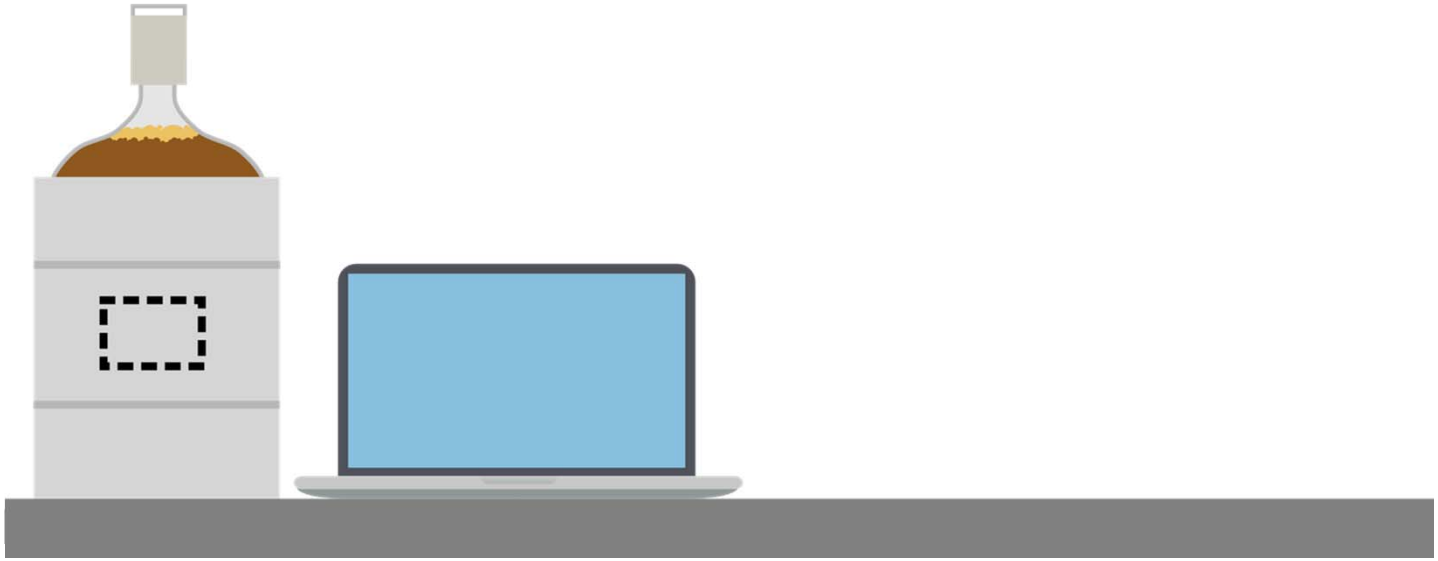


19

Brewing Beer at Home



Hobbyist wants a bespoke temperature controller.



21

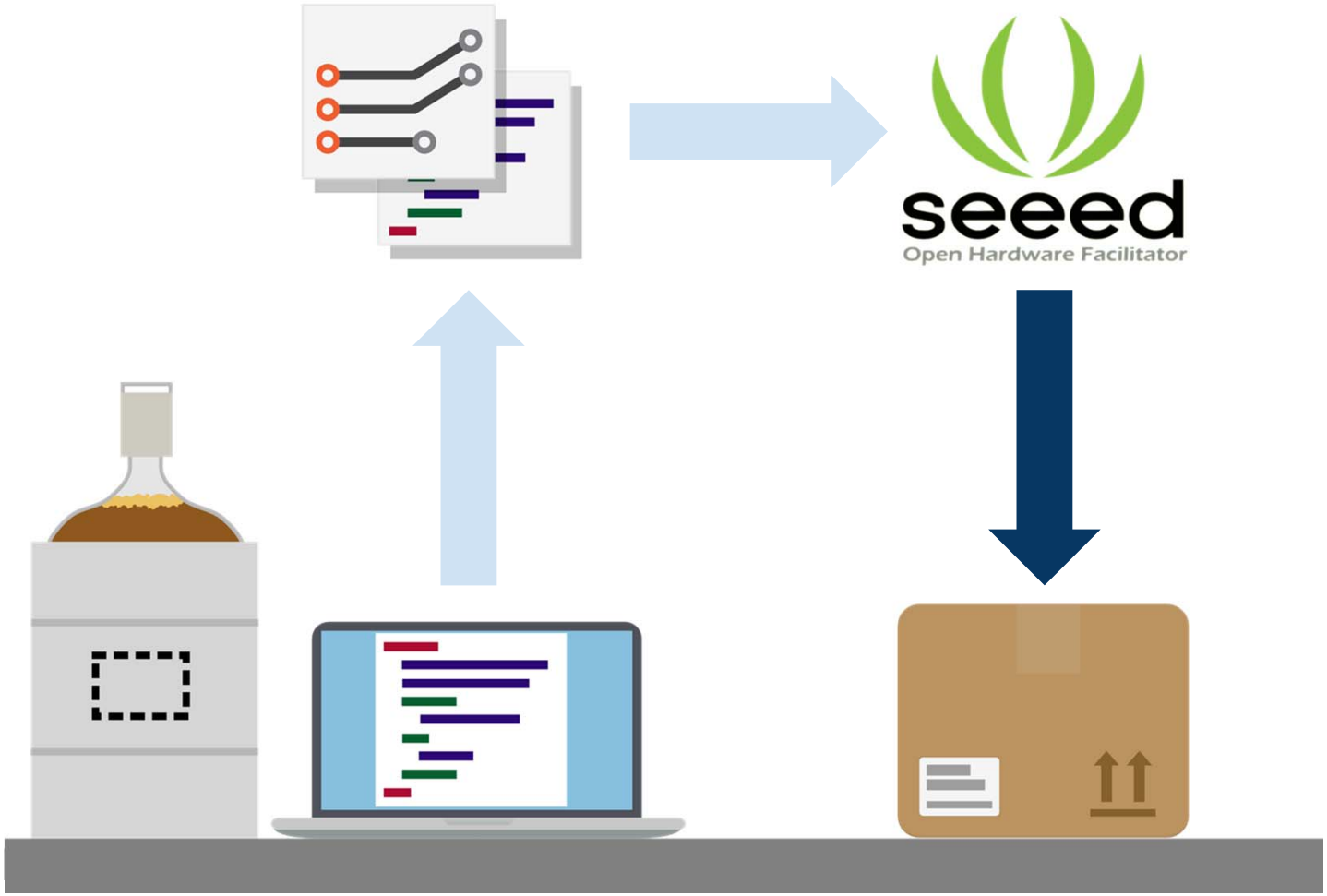
They write some Application Logic.

22

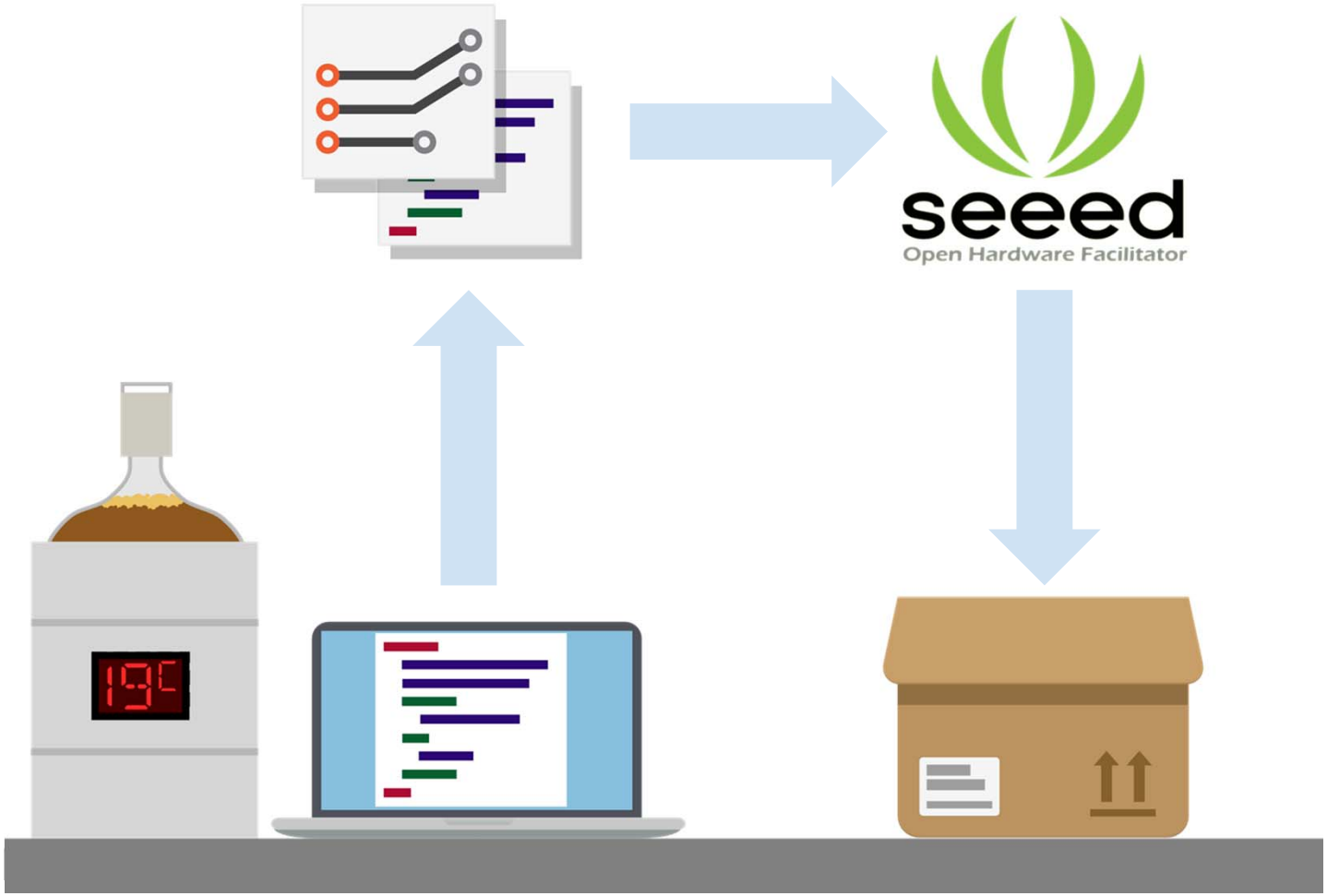
Device Generation turns code into design files.

Hobbyist sends the design to a fabricator.

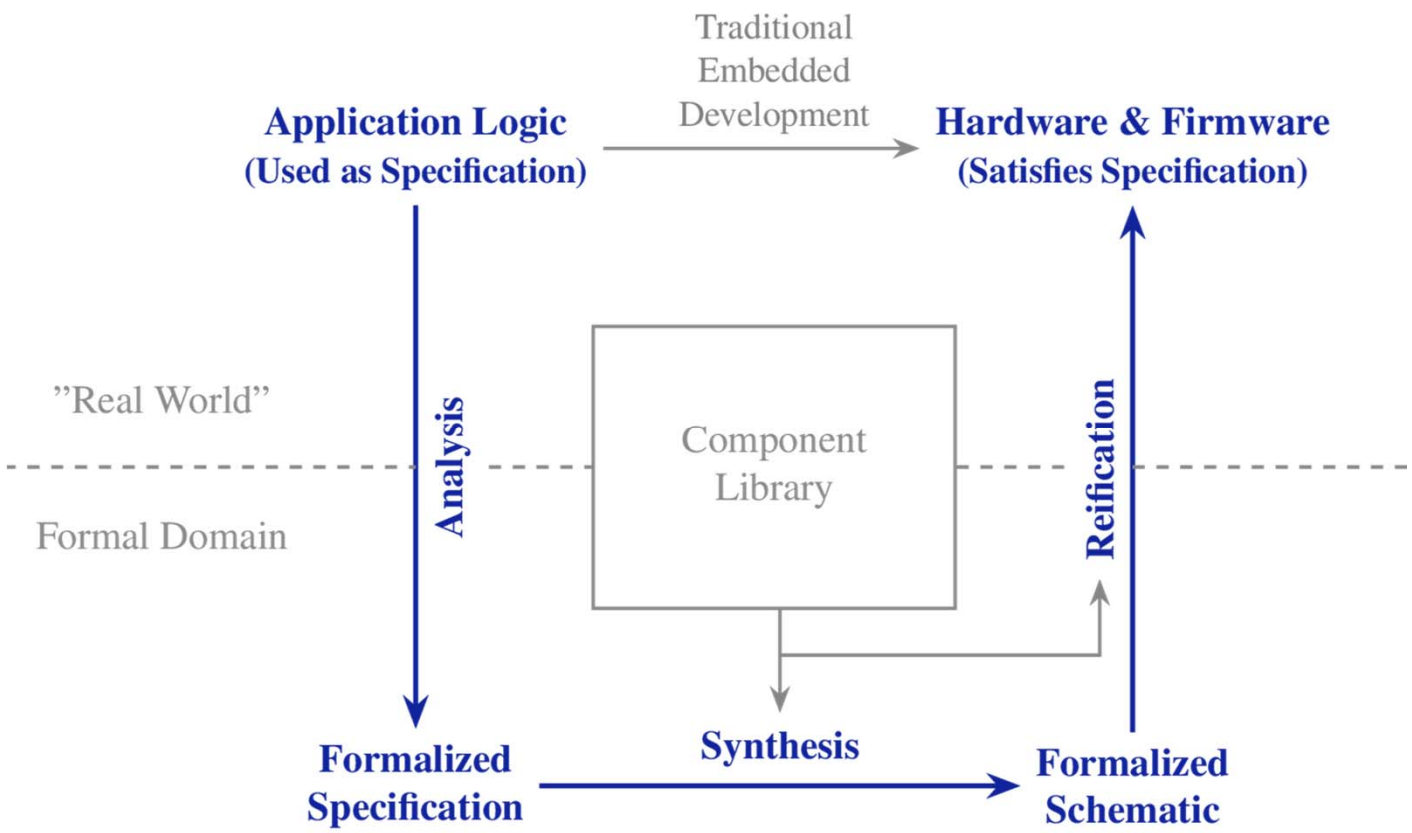
A box arrives in the mail ...



… with the finished device.



Overall Framework



Luke Hsiao, Phil, Mark, Chris

One Problem: How to Read Data Sheets

1. There are a lot of components
2. Components are complicated
3. Existing search tools and online databases only store limited information

Embedded system designers need to choose between spending lots of time curating components and the optimality of their choices.

Component Family	Number of Entries on Parts.io
Connectors	35,835,399
Power Circuits	4,308,382
Diodes	2,200,889
Sensors/Transducers	1,429,819
Memories	1,319,651
Microcontrollers	818,329
Transistors	659,628
Drivers/Interfaces	121,955
Amplifiers	98,506
Transformers	86,021

Solution

Create a framework to automatically build component databases from their datasheets. This component library will enable a wave of new applications that have access to detailed, accurate component information.

- Component search
- Datasheet error checking
- Hardware optimization
- Embedded device generation

Results

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Collector - Emitter Voltage BC546 BC547 BC548	V_{CEO}	 65 45 30	Vdc
Collector - Base Voltage BC546 BC547 BC548	V_{CBO}	 80 50 30	Vdc
Emitter - Base Voltage	V_{EBO}	6.0	Vdc
Collector Current – Continuous	I_C	100	mAdc
Total Device Dissipation @ T_A = 25°C Derate above 25°C	P_D	625 5.0	mW mW/°C
Total Device Dissipation @ T_C = 25°C Derate above 25°C	P_D	1.5 12	W mW/°C
Operating and Storage Junction Temperature Range	T_J, T_{stg}	–55 to +150	°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Max	Unit
Thermal Resistance, Junction–to–Ambient	$R_{\theta JA}$	200	°C/W
Thermal Resistance, Junction–to–Case	$R_{\theta JC}$	83.3	°C/W

Stresses exceeding Maximum Ratings may damage the device. Maximum Ratings are stress ratings only. Functional operation above the Recommended Operating Conditions is not implied. Extended exposure to stresses above the Recommended Operating Conditions may affect device reliability.

Table Extractor

Process Cell Features

```
 doc   | part_num | storage_temp_min
------+----------+----------------
X.pdf |  BC546   | -55
X.pdf |  BC547   | -55
X.pdf |  BC548   | -55
```

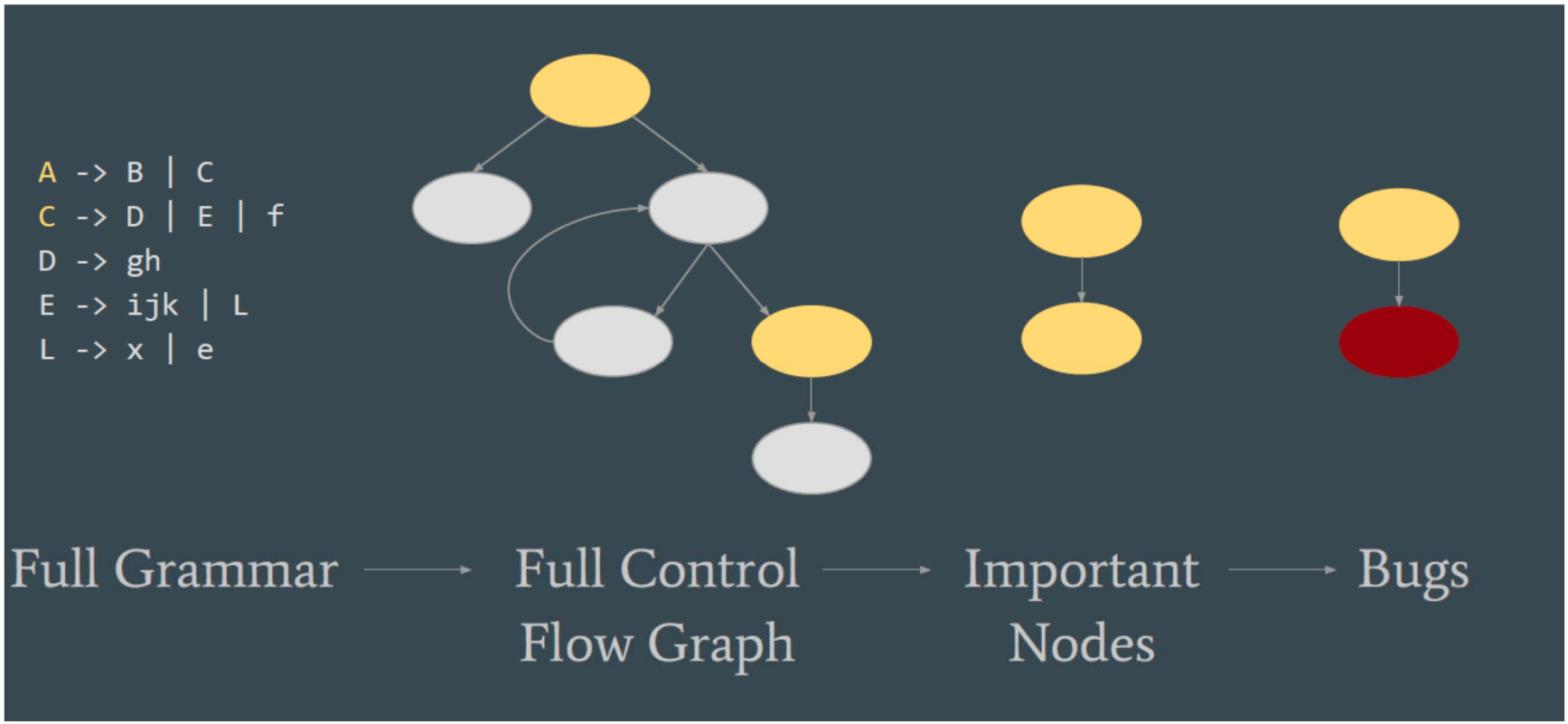
Validation

Fraser Brown, Dawson Engler

Code Checkers

TRADITION SOFTWARE CHECKERS ARE COMPLEX SOFTWARE

- Hard to write and add rules



Big Idea

IF YOU ARE GOING TO THROW STUFF AWAY, WHY PARSE IT IN THE FIRST PLACE