FUJITSU

shaping tomorrow with you

Manufacturing and Mathematics (ものづくりと数学) - Symbolic Approach -

Ryusuke Masuoka (益岡 竜介)
Fujitsu Laboratories Limited (富士通研究所)
June 20, 2012

ものづくりの過程

FUJITSU

数学に基づく設計・開発

(1) モデル化
- どのように対象のシステムをモデル化すべきか?

(2) 解析
- どのような解析をそのシステムで行うべきか?

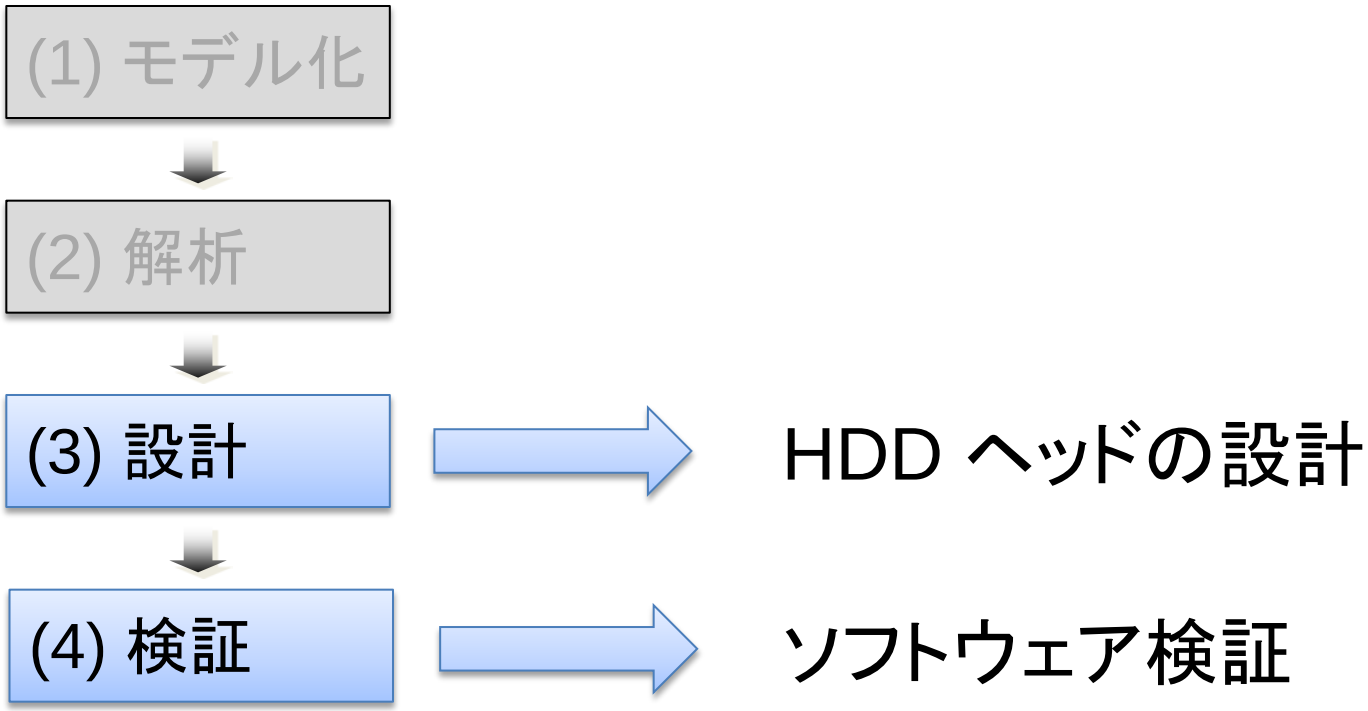
(3) 設計
- 問題をどのように定式化し、何を設計目的とするか?
- どのように設計を行う (設計問題を解く) か?

(4) 検証
- システムの設計は正しいか、不具合は?
- システムが望ましくない状態に陥らないか?

今日の話題

FUJITSU

数学に基づく設計・開発



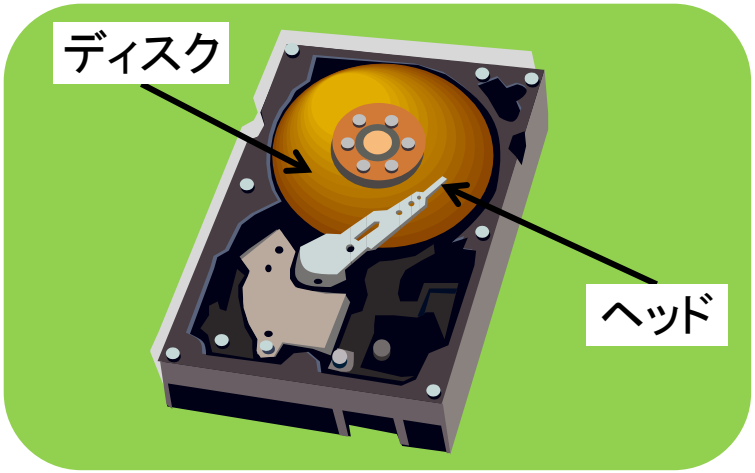
HDD: Hard Disk Drive

HDD ヘッドの設計
(HDD HEAD DESIGN)

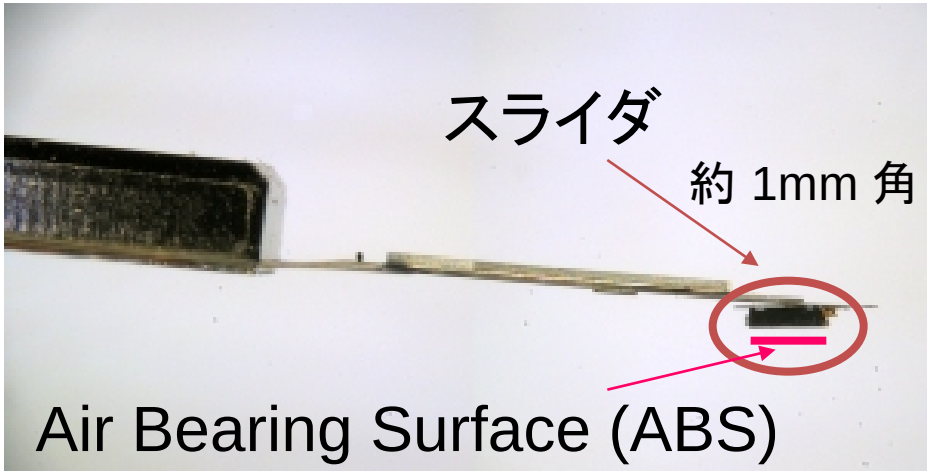
HDD 用スライダの Air Bearing Surface (ABS)

FUJITSU

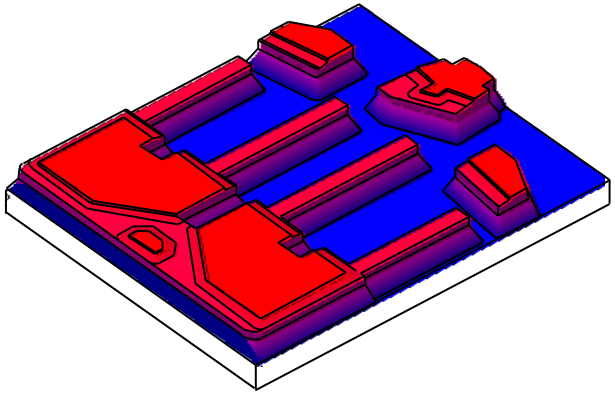
HDD



ヘッド部分の拡大図



スライダ拡大図 (上面が ABS)



スライダの ABS 形状設計

FUJITSU

- スライダの役割
  - 先端の磁気ヘッドで情報の読み取り/書き込みを行う
    - ディスクに対する相対位置が重要
    - ディスクに近いほど読み取り/書き込みエラー少ない
    - ディスクに接触するとクラッシュの原因に
- スライダとディスクの相対位置
  - 浮上量
    - スライダはディスクの回転で生じる空気の流れで 10 ～ 20 nm 程度浮上
    - 高度（気圧）などの環境変化により浮上量は変化
  - 角度
    - アームの位置により空気の流れが変わる
    - スライダに縦・横方向への回転が生じる
- ABS 形状設計
  - ABS に溝を掘ることでスライダの浮上を制御
  - 浮上量が小さく、安定した位置を保てる ABS 形状を見つける

ジャンボジェット機が地上 0.6 mm の所を飛んでいるようなもの

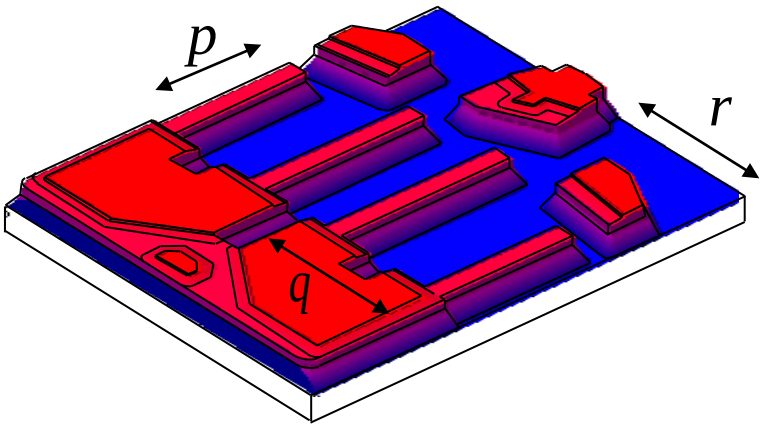
スライダをジャンボジェット機の大きさに拡大すると…

ABS 形状最適化

FUJITSU

- ABS形状 $x = (p, q, r, \ldots)$

  p, q, r, … はスライダ形状を決めるパラメタ



→ 浮上計算（レイノルズ方程式）

スライダ形状から、
フライハイト (@0m, @4000m)、ロール、ピッチ…
を計算

↓

目的関数
(コスト関数)

フライハイト (@0 m, @4000m)
ロール
ピッチ
…

- ABS 形状最適化

  目的関数 (たち) を小さくする
  ABS 形状 (パラメタ x) を求める

→ 多目的最適化

$$\begin{cases} \underset{x \in R^N}{\operatorname{argmin}} f = \{f_1(x), \ldots, f_M(x)\} \\ subject\ to \quad g(x) \leq 0 \\ \qquad\qquad h(x) = 0 \end{cases}$$

従来手法 – 数値計算

FUJITSU

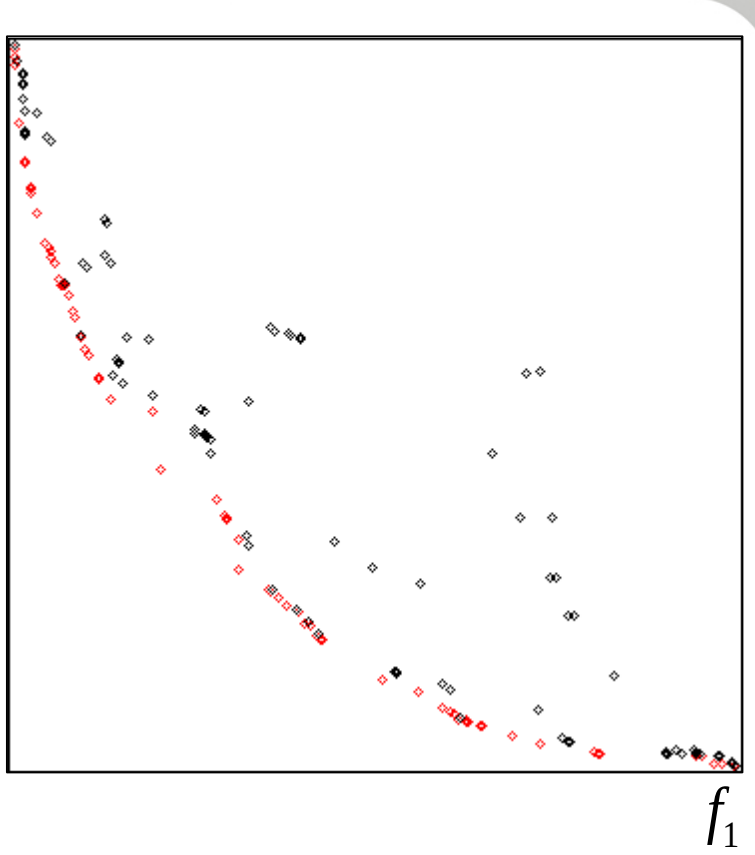
Minimize $F = (f_1(x,y), f_2(x,y))$

$$f_1(x,y) = 4x^2 + 4y^2$$
$$f_2(x,y) = (x-5)^2 + (y-5)^2$$

$$0 \geq (x-5)^2 + y^2 - 25$$
$$0 \geq -(x-8)^2 - (y+3)^2 + 7.7$$

$$0 \leq x \leq 5$$
$$0 \leq y \leq 3$$

GAで求めた解



数打ちゃあたる

GA: Genetic Algorithm

もし数式処理で解けたら

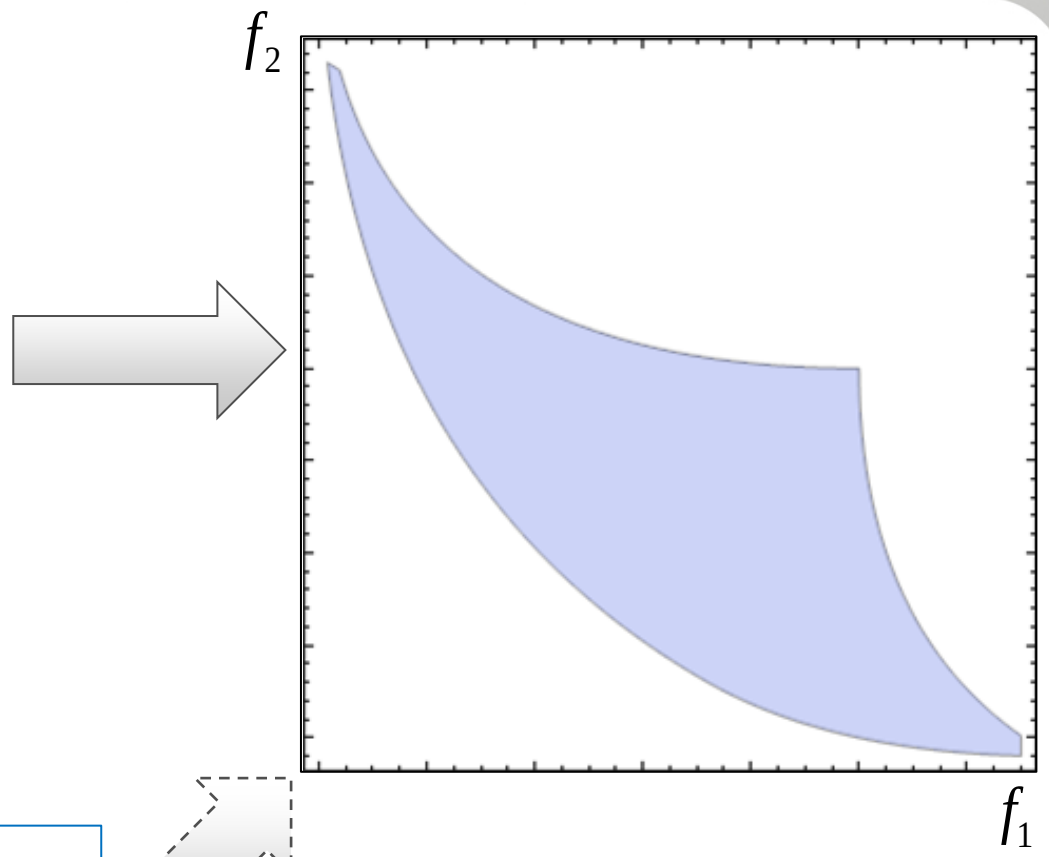
FUJITSU

Minimize $F = (f_1(x,y), f_2(x,y))$

$$f_1(x,y) = 4x^2 + 4y^2$$
$$f_2(x,y) = (x-5)^2 + (y-5)^2$$

$$0 \geq (x-5)^2 + y^2 - 25$$
$$0 \geq -(x-8)^2 - (y+3)^2 + 7.7$$

$$0 \leq x \leq 5$$
$$0 \leq y \leq 3$$



限定記号を
使った定式化

$$\exists x\, \exists y\, (4x^2 + 4y^2 - f_1 = 0 \wedge$$
$$(x-5)^2 + (y-5)^2 - f_2 = 0 \wedge$$
$$(x-5)^2 + y^2 - 25 \leq 0 \wedge$$
$$-(x-8)^2 - (y+3)^2 + 7.7 \leq 0 \wedge$$
$$x \geq 0 \wedge x \leq 5 \wedge$$
$$y \geq 0 \wedge y \leq 3\,)$$

QE 法

Yes, We Can!

限量記号消去 (Quantifier Elimination)

FUJITSU

■パラメトリックな最適化の基礎アルゴリズム

入力

限定記号がついた論理式

出力

限定記号がない等価な論理式

例

$$\forall x\ (x$$

$$b$$

$$\exists x\ (a$$

$$(a \neq 0\ \wedge\ b^2 - 4ac \geq 0)\ \vee$$
$$(a = 0\ \wedge\ b \neq 0)\ \vee$$
$$(a = 0\ \wedge\ b = 0\ \wedge\ c = 0)$$

$$\forall x \exists y\ (x^2 + xy + b > 0 \wedge$$
$$x + ay^2 + b \leq 0)$$

$$a <$$

Cylindrical Algebraic Decomposition (CAD)

FUJITSU

- QE の基礎となる代数的アルゴリズム
- G. E. Collins が導入 (1975年)
  - 入力: $F_r \subset \mathbb{Q}[x_1, \ldots, x_r]$
  - 出力: 入力の多項式が符号不変で互いに交わらない集合 (セル) への分解

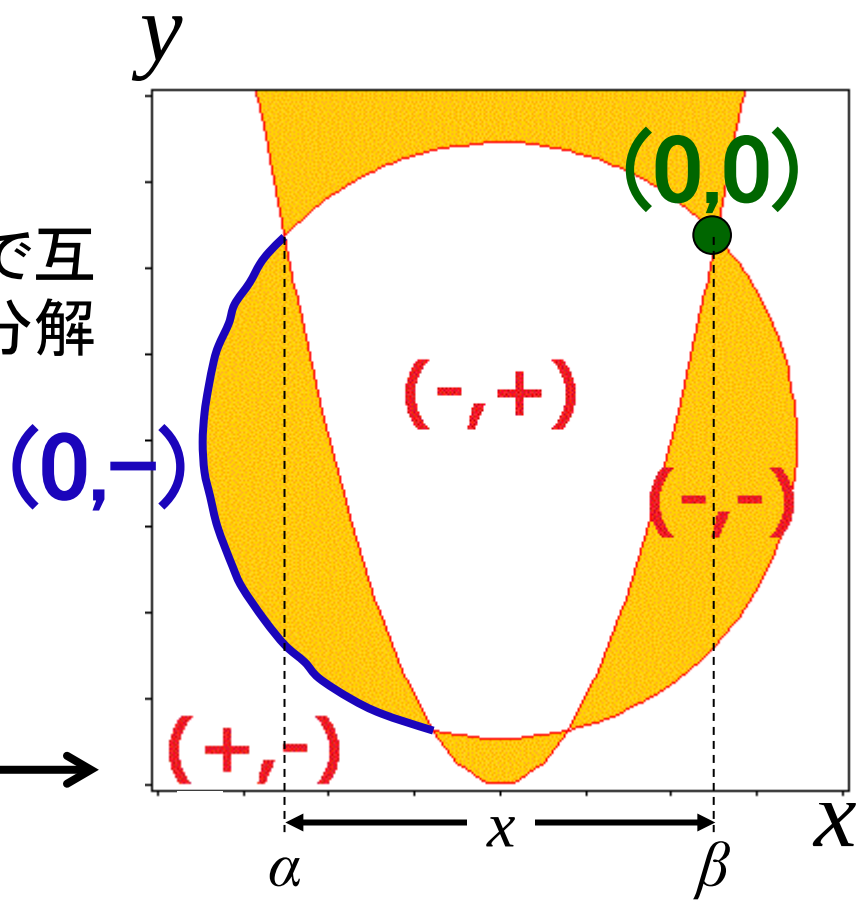
例

$F_2 = |$

$|$

⇒

$\mathbb{R}^2$ $C_i^{(2)}$ ⟶

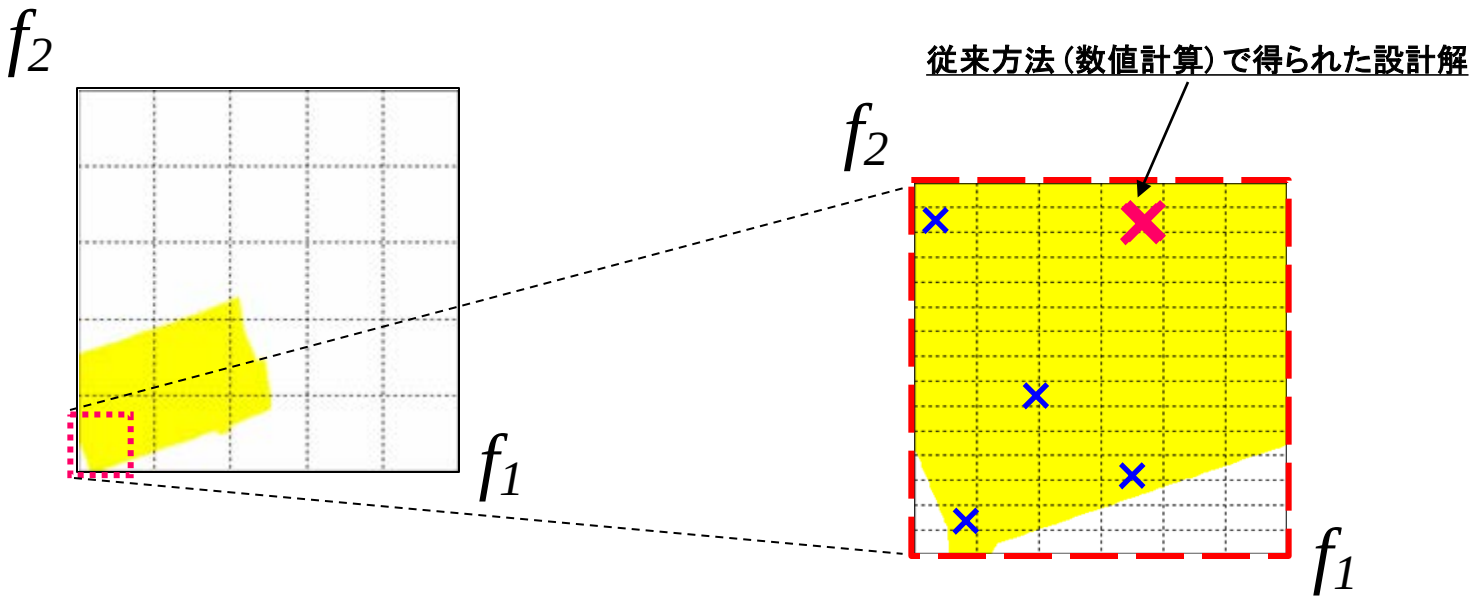


問 以下と同値な式をもとめよ

$\exists y\,(x^2 + y^2 - 3 < 0 \wedge y - 2x^2 + 2 > 0)$

答 $\alpha < x < \beta$

ABS 形状 (パラメタ) 決定



$\{f_1, f_2\}$ の可能領域の正確な可視化により、
多目的最適化を見通しよく実現

QE tools

FUJITSU

SyNRAC

Maple 15
The Essential Tool for Mathematics and Modeling

CAD, VS, SDC
Fujitsu Laboratories Ltd.
(H. Yanami, H. Iwane, H. Anai)

Wolfram Mathematica 8

CAD, VS
Wolfram Research, Inc.
(A. Strzebonski)

QEPCAD

CAD
RISC-Linz + etc.
(G. Collins, H. Hong, C. Brown)

REDLOG

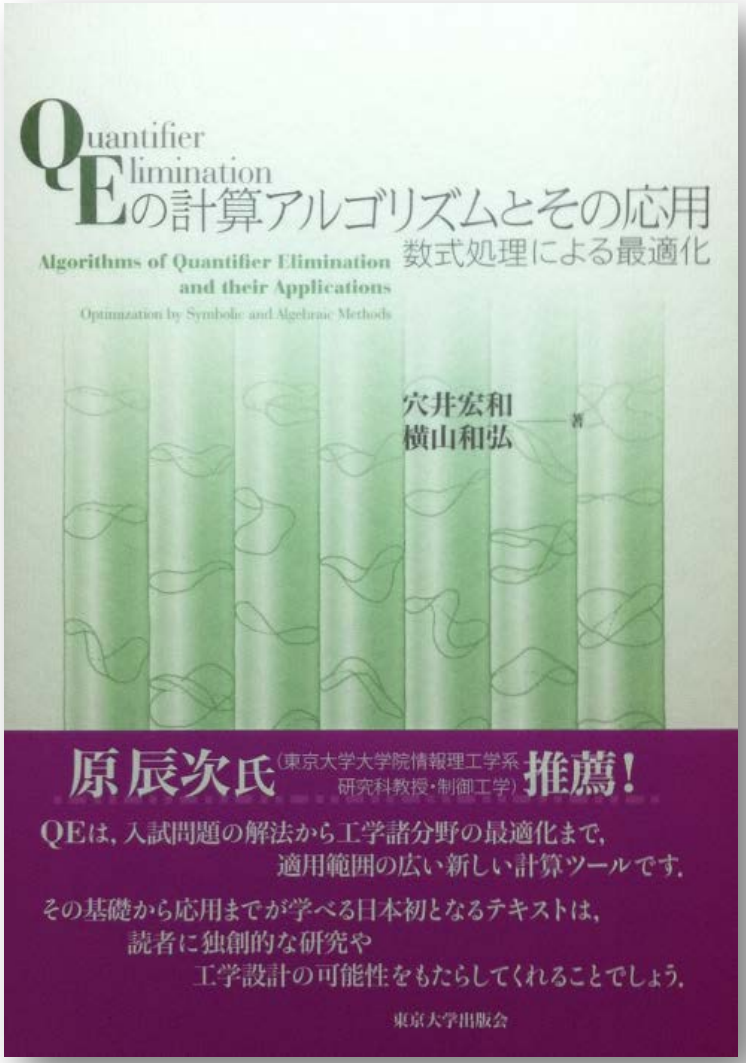
REDUCE

CAD, VS
Univ. Passau
(T. Sturm, A. Dolzmann, A. Seidl)

CAD: Cylindrical Algebraic Decomposition, VS: Virtual Substitution, SDC: Sign Definite Condition

References and a Seminar

- 穴井 宏和他, “数式処理を用いた設計技術,” 雑誌 Fujitsu, Vol.60, No.5, pp. 514 – 521,Sep. 2009
- 穴井 宏和, 横山 和弘, “QEの計算アルゴリズムとその応用 ～数式処理による最適化,” 東京大学出版会, Aug. 2011
- QE 関係 九州大学 集中講座
  - 講演者：穴井 宏和
  - 日時: 6 月 25 日 (月) – 6 月 29 日 (金) 初回は 13 時 00 分から。後の日程は初回に決定。
  - 授業科目：計算数理学Ⅱ (院)
  - 場所: 中セミナー室 6 (6 月 26 日 (火) のみ 中セミナー室 4)

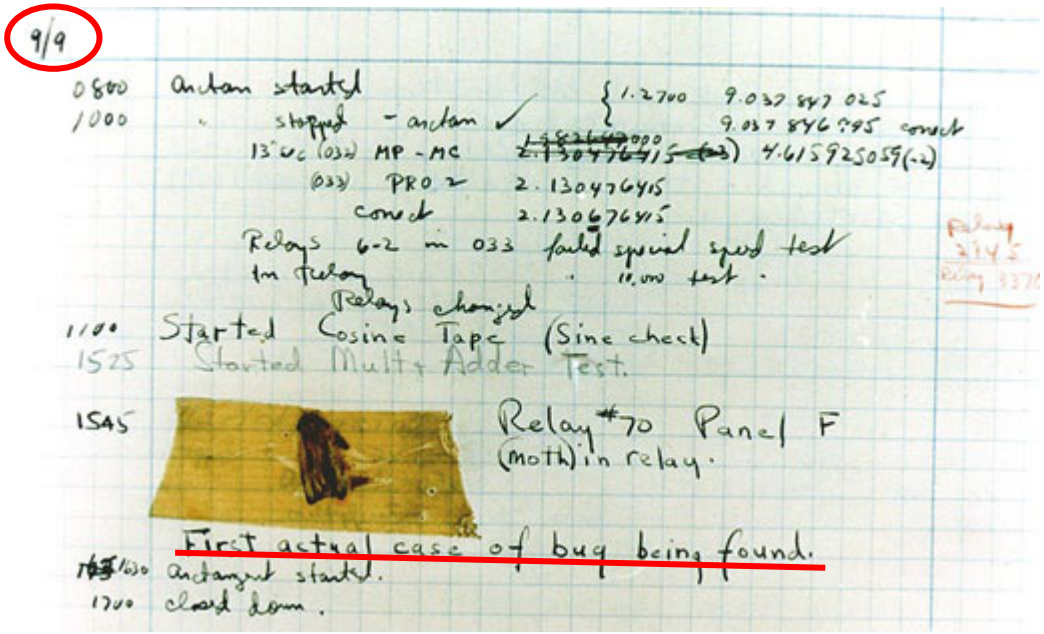


ソフトウェア検証
(SOFTWARE VALIDATION)

September 9th

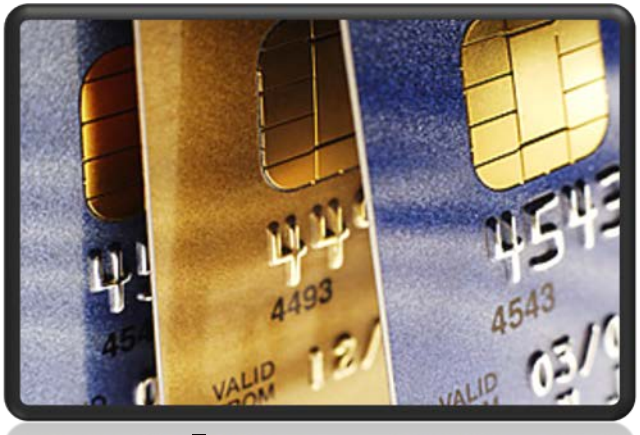
What day is September 9th?

“Debugging Day!”

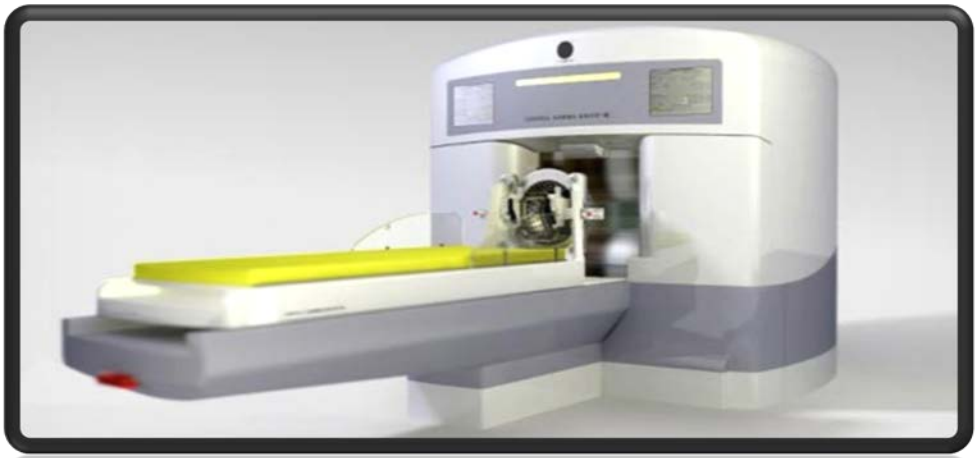


Is a Bug an Inconvenience?

Calls dropped
- 1990 AT&T 4ESS long-distance switches



Money lost
- 2010 German Bank Card Year 2010 Bug



Lives lost
- 1985 Therac-25 Medical Accelerator

… Or Human Extinction

FUJITSU

False Alarms

- 1980 NORAD
- 1983 Soviet Union

Testing, Of Course, But … It Is Expensive

FUJITSU

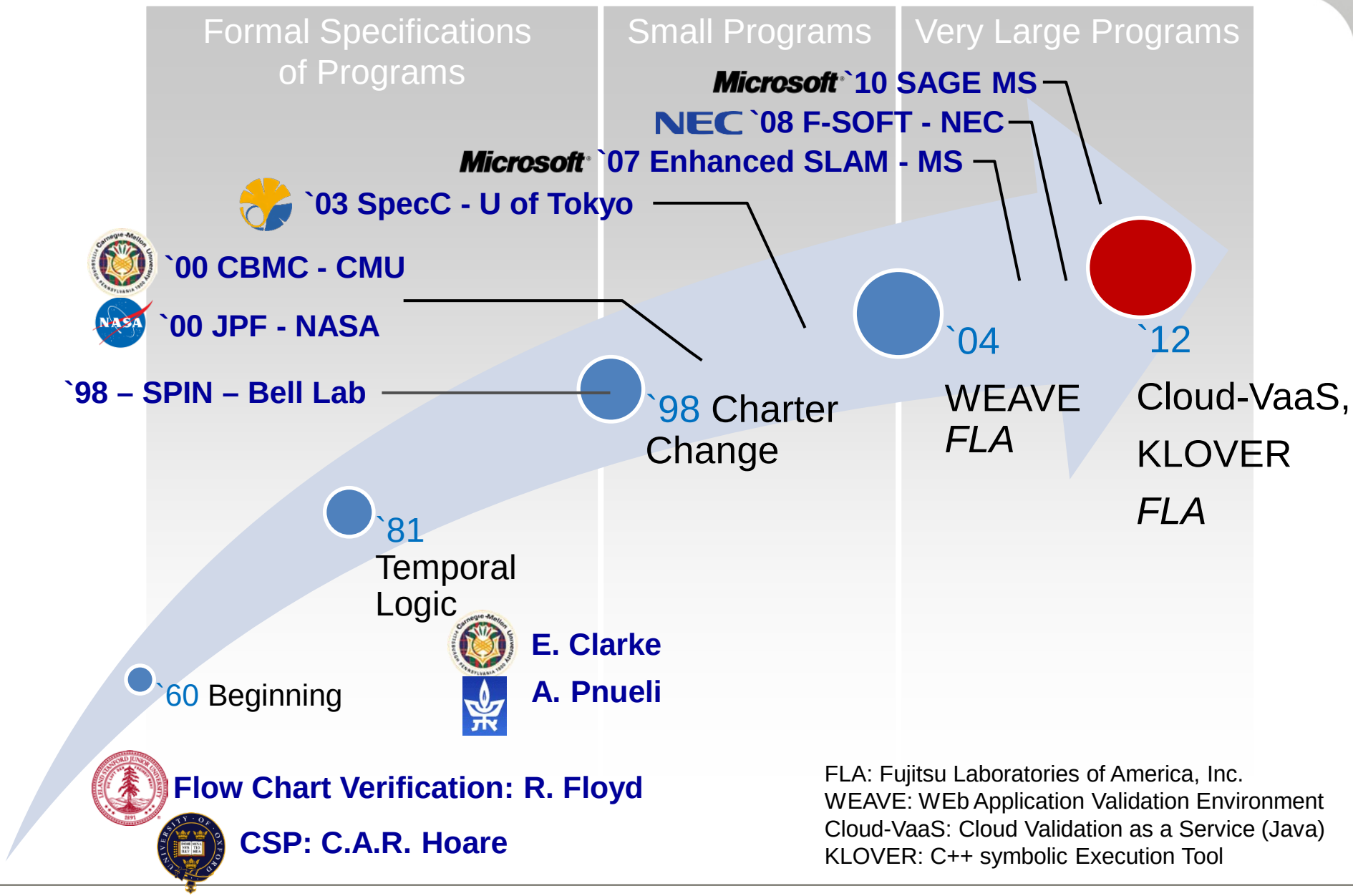
- Largely manual labor-intensive process
- Software getting larger and more complex every day
  - Beyond human capacity to grasp!
  - More expensive

What is Software Validation?

FUJITSU

- Calculate whether a system satisfies a certain *behavioral property*:
  - Whenever a packet is sent will it eventually be received?
  - Is the system deadlock free?
- So it is like testing? No, there is a major difference:
  - Try exploring *all* possible behaviors of a system in *one* run
  - Two different techniques:
    - Model checking (suitable for hardware, finite state)
    - Symbolic execution (suitable for software, infinite state)
- Automation
  - Push-button technology needs deep algorithmic research
  - Most important for commercial acceptance

History of Software Validation

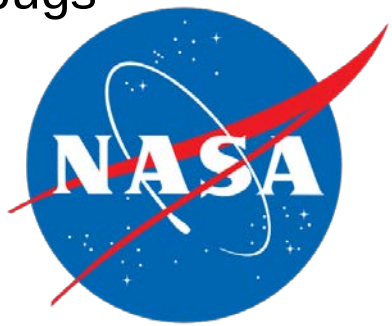


FLA: Fujitsu Laboratories of America, Inc.
WEAVE: WEb Application Validation Environment
Cloud-VaaS: Cloud Validation as a Service (Java)
KLOVER: C++ symbolic Execution Tool

Major Players

FUJITSU

- NASA
  - Why NASA? - History of blowing up rockets to software bugs
  - Checking Toyota's brake software
  - FLA closely collaborating – JPF, String
- Fujitsu - FLA
  - As a provider of ICT for critical social infrastructure
- IBM, NEC, HP, Intel, Siemens, MS
- Startups
  - Coverity, RealIntent
- Universities
  - UC Berkeley, Oxford Univ., Stanford, CMU, NYU, UT Austin, Kansas U
  - University of Tokyo, JAIST



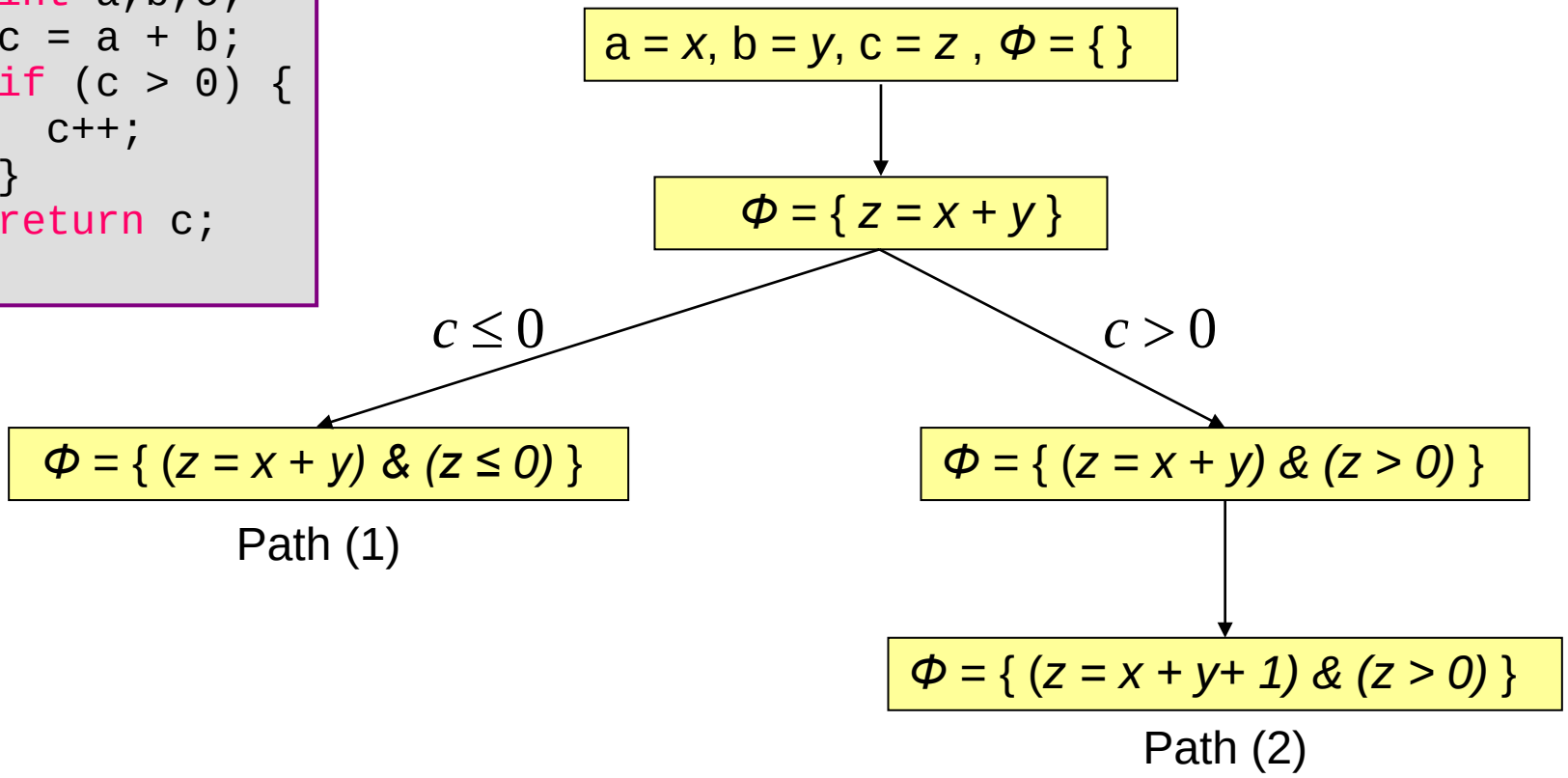
FLA: Fujitsu Laboratories of America, Inc.
JPF: Java Path Finder

Symbolic Execution (SE)

- Each path in the tree represents a (possibly infinite) set of execution paths

Φ is symbolic expression
x, y, z are symbolic integers

```
foo(a, b, c) {
  int a,b,c;
  c = a + b;
  if (c > 0) {
    c++;
  }
  return c;
}
```



Finding a Bug

- **Property to check : if ((a > 1) & (b > 0)) → (c > 4)**
- **Negate property:**
  - **if ((a > 1) & (b > 0)) → (c <= 4)**
- **Check at the end of each path of the symbolic execution**

Equations at the end of path (1)

x > 1
y > 0 } Preconditions
z = x + y
z <= 0
z <= 4 } Post condition

Solve using ILP
- No solutions
- Property holds

Equations at the end of path (2)

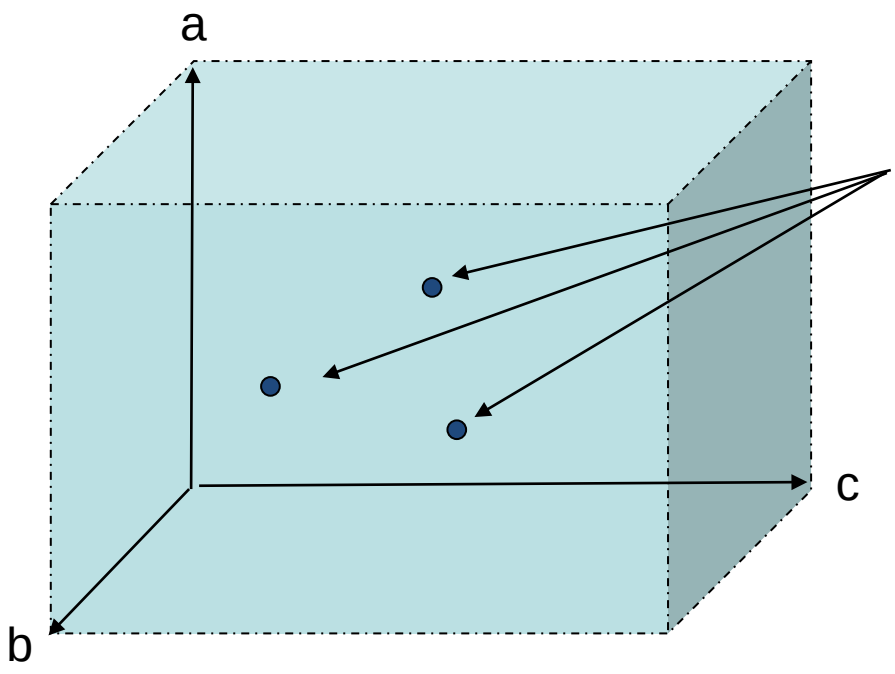
x > 1
y > 0 } Preconditions
z = x + y + 1
z > 0
z <= 4 } Post condition

Solve using ILP
- **SOLUTION FOUND !!**
- Counter example: x = 2, y = 1, z = 4

- Bug Uncovered: if a = 2 and b = 1 then c > 4 does not hold in this path

Symbolic Execution: Advantages

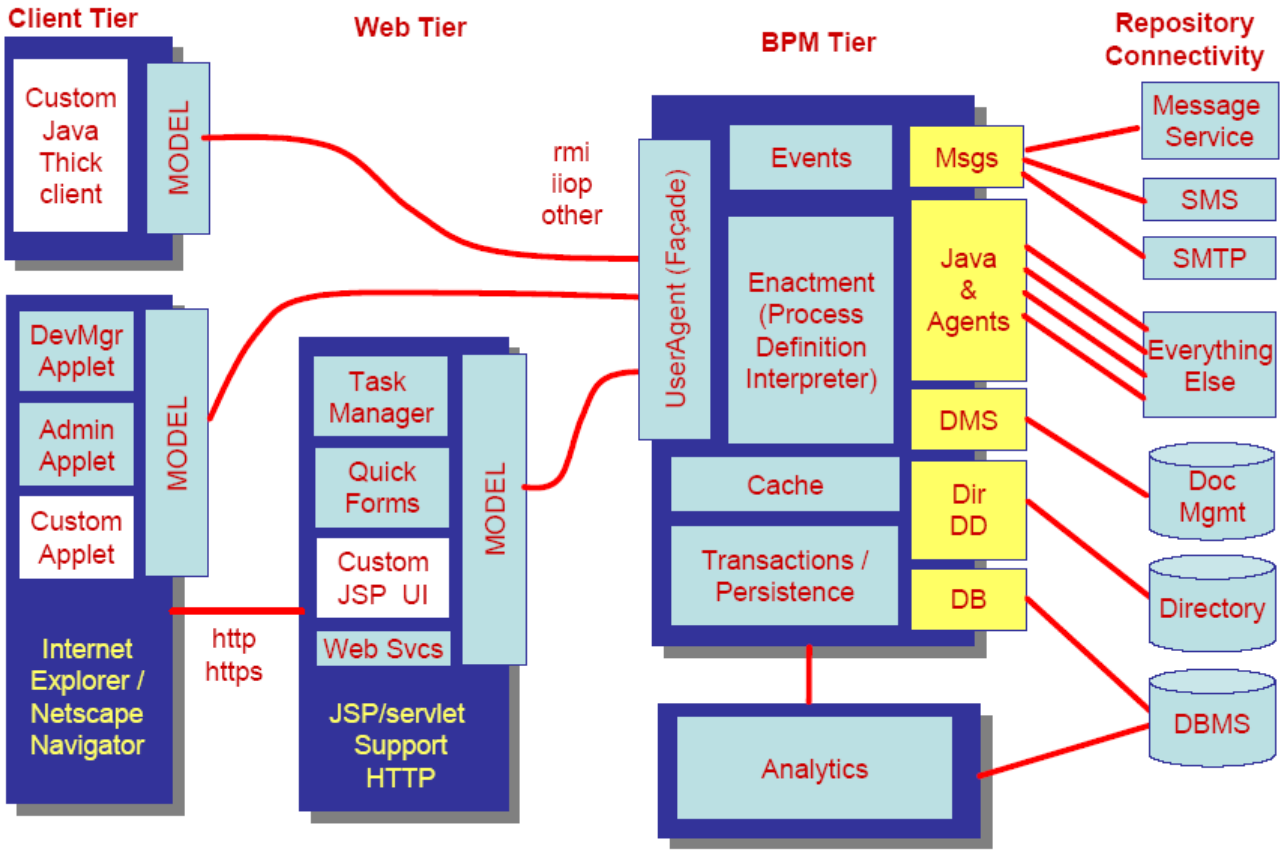
- Solution of these symbolic expressions can give concrete traces representing all distinct paths in the program tree
  - Complete input space examined in one shot



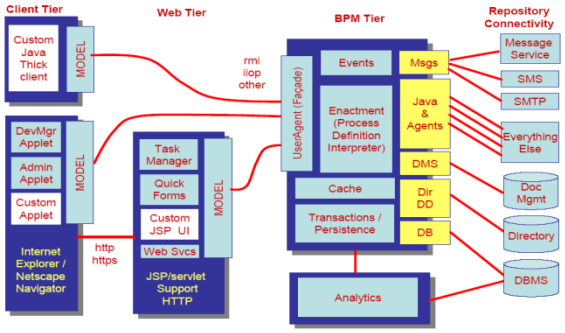
But, Industry Software too Large to Process

FUJITSU

Scenario: Given a large software code base, consisting of interacting heterogeneous components such as client, server, and database. Its server "locks up" some times and clients receive incorrect data



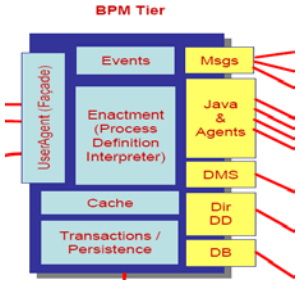
Three-Pronged Approach for Reduction

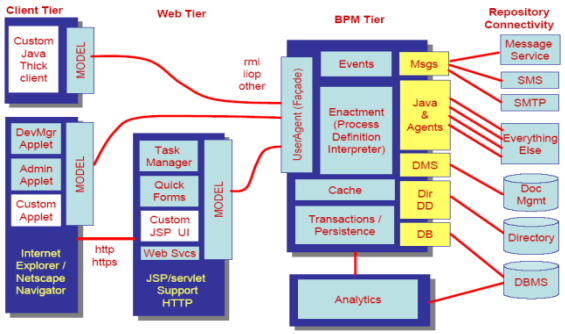


Environment Generation

Reduce scenarios of the system behavior

Env. Gen

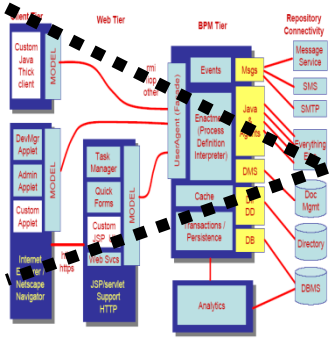


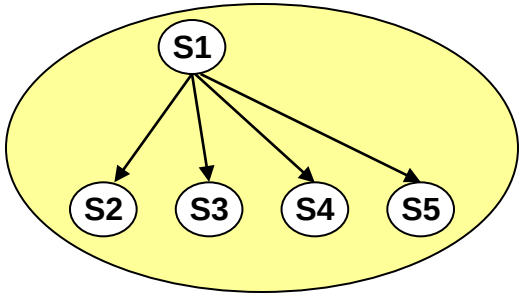


Slicing

Reduce irrelevant statements through static analysis

Slicing

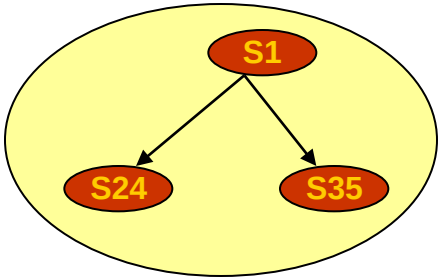




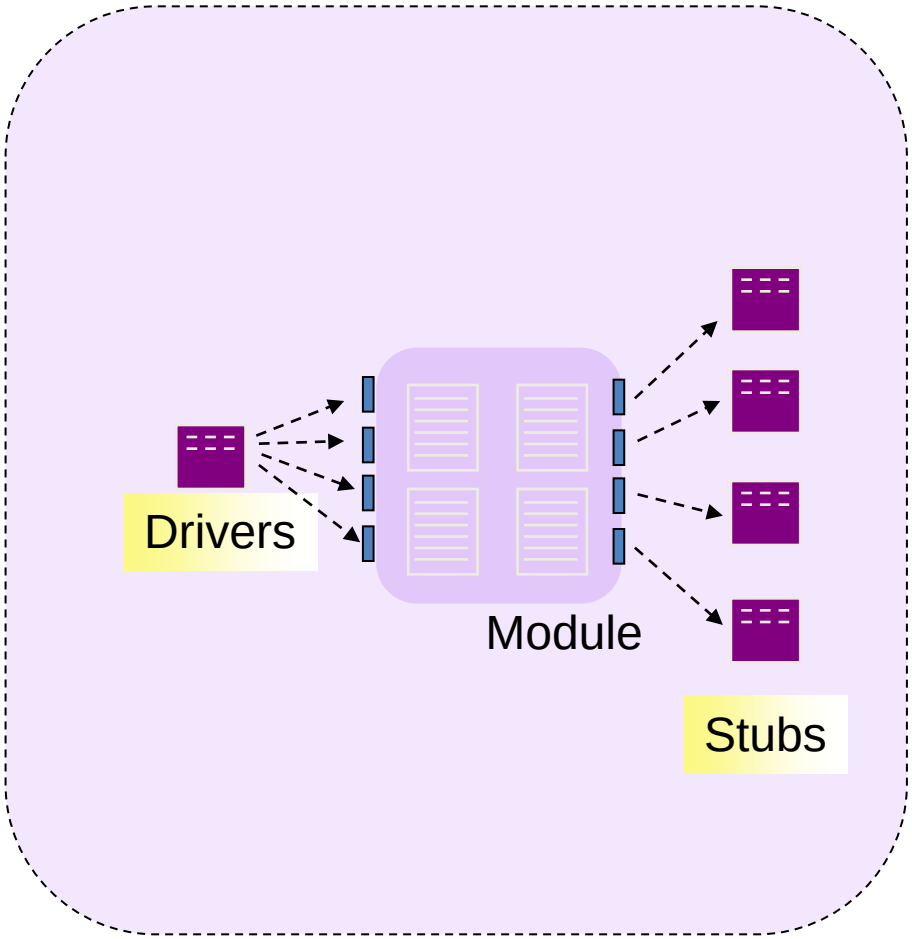
Abstraction

Reduce the state space by consolidating states

Abstraction



Environment Generation: Reduces Scenarios



Code Base

- Drivers (Test harness, test drivers)
  - Call the module
  - Java classes/threads that instantiate classes inside the module and perform sequences of method calls to the module
- Stubs
  - Are called by the module
  - Java classes/methods called by classes inside the module, e.g., library methods

Slicing: Reduces Code Size to Process

FUJITSU

- Chop irrelevant program statements through static analysis

Example below taken from slide 4 of
"The Bandera Model Reduction Tools"
[Linked from "slicing in Bandera" in
http://bandera.projects.cis.ksu.edu/talks.shtml]

Raise m to the power of n:

```
init: m      := 5;              [init.1]
      n      := 2;              [init.2]
      result := 1;              [init.3]
      goto test;

test: if (n<1)                  [test.1]
      then end
      else loop;

loop: result := result*m;       [loop.1]
      n := n-1;                 [loop.2]
      goto test;                [loop.3]

end:  return;                   [end.1]
```

Data dependence

Control dependence

Original Program

```
init:
      n      := 2;              [init.2]

goto test;

test: if (n<1)                  [test.1]
      then end
      else loop;

loop:
      n := n-1;                 [loop.2]
      goto test;                [loop.3]

end:  return;                   [end.1]
```

Sliced Program

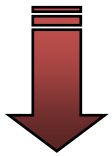
Slicing Criterion: C = { [loop.2] }

Abstraction: Example

FUJITSU

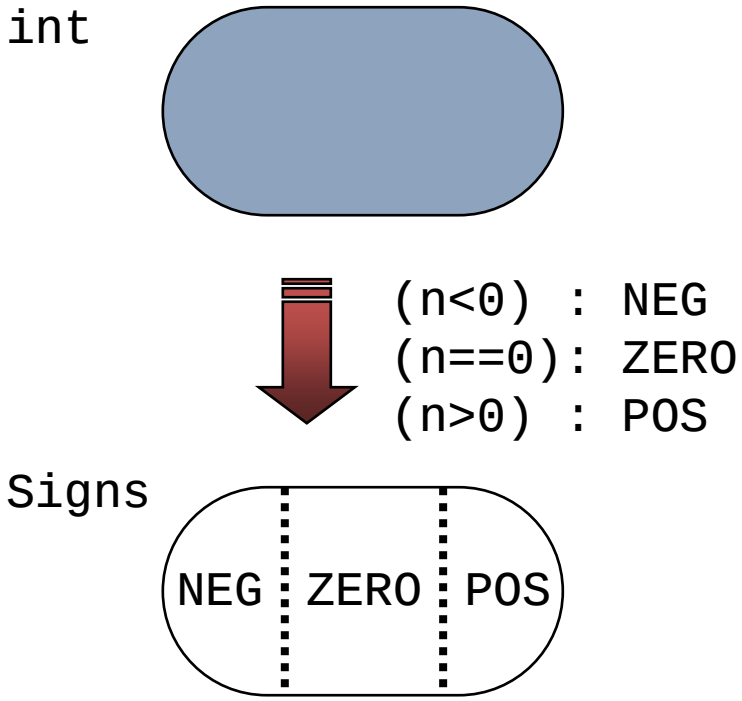
Example Code

```
int x = 0;
if (x == 0)
  x = x + 1;
```



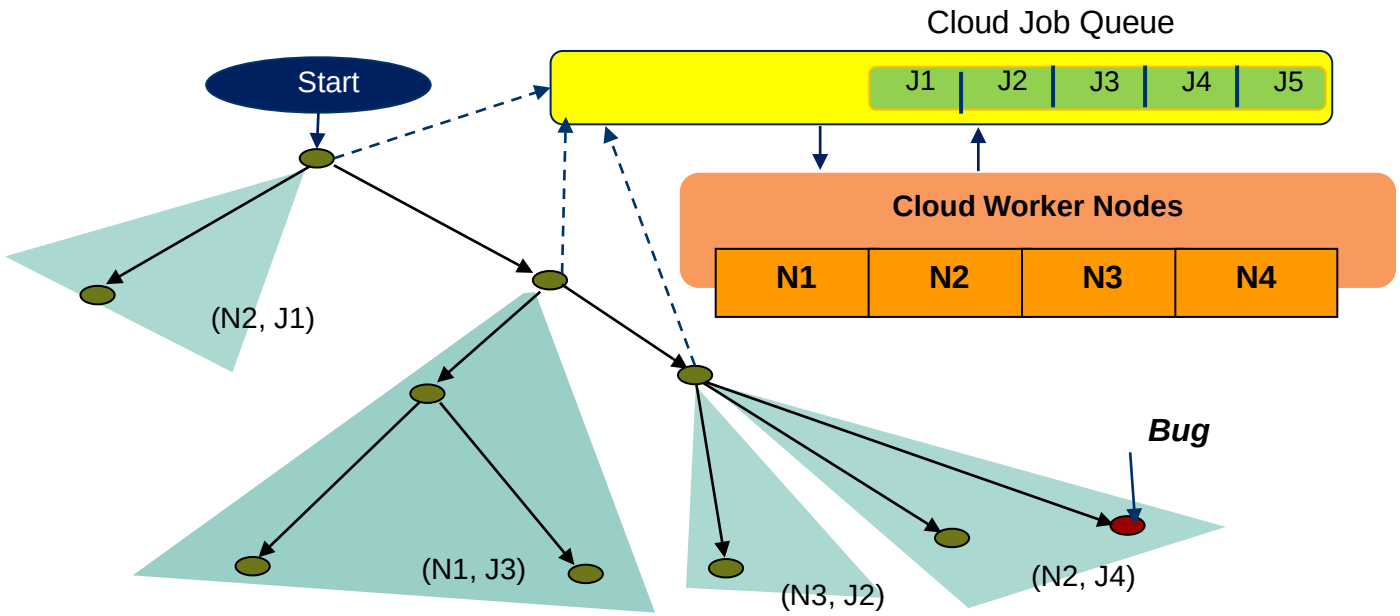
```
Signs x = ZERO;
if (Signs.eq(x,ZERO))
  x = Signs.add(x,POS);
```

Data Type Abstraction



One Last Trick: Distribution on Cloud

FUJITSU



- Distributed it over compute nodes in cloud to enhance scalability

  Performance improvement: 15x speed-up with 25 nodes

Conclusion (Software Validation)

FUJITSU

To check suspicious modules

To reduce computational loads

Symbolic Execution

+

On Cloud

Environment Generation
Slicing
Abstraction

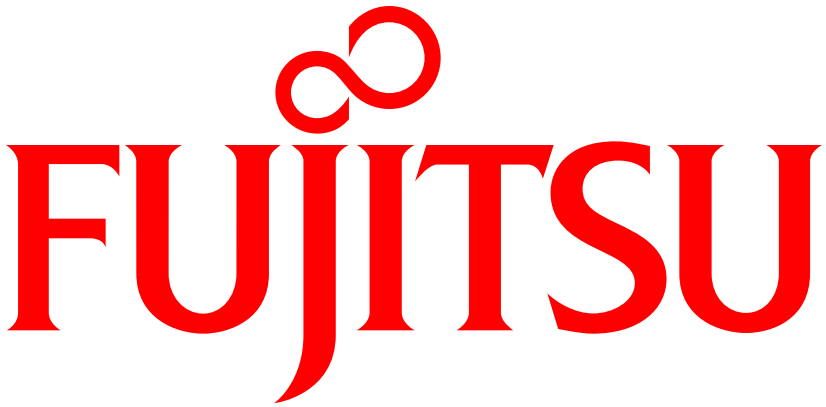
To enhance scalability

Software validation for large software

References

FUJITSU

- News Articles
  - “Epic failures: 11 infamous software bugs,” *ComputerWorld,* Sep. 9, 2010, http://www.computerworld.com/s/article/9183580/
  - “The Year 2010 Bug Strikes German Bank Cards,” *TIME,* Jan. 7, 2010, http://www.time.com/time/business/article/0,8599,1952305,00.html
  - “10 historical software bugs with extreme consequences,” *Royal Pingdom,* Mar. 9, 2009, http://royal.pingdom.com/2009/03/19/10-historical-software-bugs-with-extreme-consequences/
- Surveys
  - Cadar, C., Godefroid, P., Khurshid, S., Pasareanu, C., Sen, K., Tillmann, N., “Symbolic Execution for Software Testing in Practice : Preliminary Assessment,” *ICSE'11*
  - D'Silva, V., Kroening, D. & Weissenbacher, G., "A Survey of Automated Techniques for Formal Software Verification,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD),* July 2008.
- Software Validation Research at FLA
  - Li, G., Ghosh, I., Rajan, S.P., “KLOVER: A Symbolic Execution and Automatic Test Generation Tool for C++ Programs,” *CAV 2011*:
  - Rajan, S. P., Tkachuk, O., Prasad, M. R., Ghosh, I., Goel, N., & Uehara T., "WEAVE: WEb Applications Validation Environment,” *ICSE 2009*



shaping tomorrow with you

WORKING IN US

Personal History

FUJITSU

FLA, Sunnyvale, CA

FLA, College Park, MD

FLL, Momochi, Fukuoka

Visiting Scientist,
CMU, Pittsburgh, PA

2012

Back to FLL,
Kawasaki, Kanagawa

2009

2001

FLL, Mukuhari, Chiba

2000

Got a Ph.D.
(Mathematical Sciences)

1995

1993

Back to FLL,
Kawasaki, Kanagawa

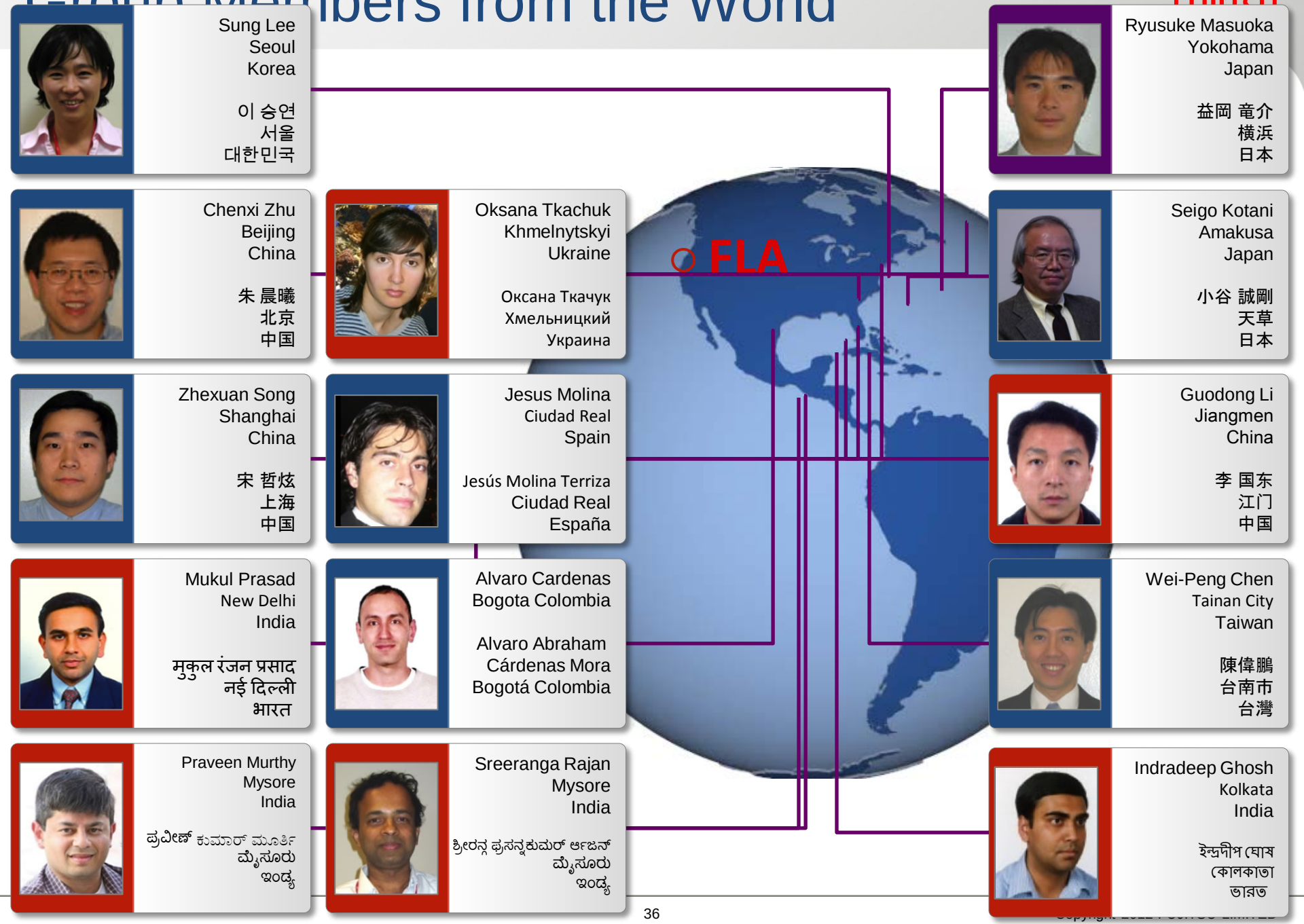
1991

1988

Joined FLL,
Kawasaki, Kanagawa

FLL: Fujitsu Laboratories Limited
FLA: Fujitsu Laboratories of America, Inc.

Group Members from the World



Sung Lee
Seoul
Korea

이 승연
서울
대한민국

Chenxi Zhu
Beijing
China

朱 晨曦
北京
中国

Zhexuan Song
Shanghai
China

宋 哲炫
上海
中国

Mukul Prasad
New Delhi
India

मुकुल रंजन प्रसाद
नई दिल्ली
भारत

Praveen Murthy
Mysore
India

ಪ್ರವೀಣ್ ಕುಮಾರ್ ಮೂರ್ತಿ
ಮೈಸೂರು
ಇಂಡ್ಯ

Oksana Tkachuk
Khmelnytskyi
Ukraine

Оксана Ткачук
Хмельницкий
Украина

Jesus Molina
Ciudad Real
Spain

Jesús Molina Terriza
Ciudad Real
España

Alvaro Cardenas
Bogota Colombia

Alvaro Abraham
Cárdenas Mora
Bogotá Colombia

Sreeranga Rajan
Mysore
India

ಶ್ರೀರಂಗ ಪ್ರಸನ್ನಕುಮಾರ್ ರಾಜನ್
ಮೈಸೂರು
ಇಂಡ್ಯ

Ryusuke Masuoka
Yokohama
Japan

益岡 竜介
横浜
日本

Seigo Kotani
Amakusa
Japan

小谷 誠剛
天草
日本

Guodong Li
Jiangmen
China

李 国东
江门
中国

Wei-Peng Chen
Tainan City
Taiwan

陳偉鵬
台南市
台灣

Indradeep Ghosh
Kolkata
India

ইন্দ্রদীপ ঘোষ
কোলকাতা
ভারত

Diversity

FUJITSU

- India, China, Korea, Taiwan, Columbia, Greece, Germany, France, Spain, Russia, Saudi Arabia, Nigeria, Vietnam, Iran, … Of course, a few from Japan and US
- Status もばらばら
  - Visa Status
    - 米国市民 - 前の代から/自分の代から/途中から
    - グリーンカード (米国永住権)
    - 学生ビザ、就労ビザ
    - 一時訪問者
  - FLA での Status
    - 研究者 (正社員)、テンプ、インターン (学生)、コンサルタント
  - 米国の特に研究開発現場では当たり前
    - 国防関係以外
- メンバーの歳を知らず
  - 能力・成果のみで給与・昇進を判断

人の動かし方

FUJITSU

- 米国でも以外にコネは重要
  - ただしそれから後は、自分の能力しだい
- High turnover
  - 出来る人ほどすぐいなくなる
  - 最先端なので、バックグラウンドのある人などいない
  - いかに満足度を高めるか
  - インターン: 新しい技術の習得、レジメ
- 使えるものは何でも使う
- 自主独立
  - 明確な Job Description
  - 彼らの多くも異国で戦っている
- 柔らかく、しかししっかりと、煙たがれない程度に頻繁に、手を変え、品を変え、押す

Logistics

FUJITSU

- 管理職はサービス業
  - 担当者が効率よく (気持ちよく) 仕事が出来るように、段取りよく準備しておく
  - 判断などをすばやく、はっきりと下す
    - 日本とのプロジェクトではここが難しい
- 簡単に Okay というが …
  - 分担がはっきりしている (明確な job description) …
    - どこかで間に落ちて止まる
  - 何度もチェックを入れる
- 日本とのプロジェクト
  - うまく回せば 24 時間動かし続けられる
  - 逆に何日も無駄に過ぎることも

コミュニケーション

FUJITSU

- 英会話だけで意図を伝えるのは難しい
  - 特に Ontology (存在論) とか Semantics (意味論) といった抽象的なもの
  - 作戦
    - Visualization
    - モックアップ
- Email
  - 何でも文書にして、関係する範囲に流しておく
    - 特に打ち合わせはまとめを Email で送る
    - 後で言った、言わないがなくなる
    - 自分の記録にもなる
  - 最初のパラグラフに用件を遠慮せずはっきりと書く
    - 背景や詳細は別に後ろに書く
  - 何種類もの依頼と感謝の言い方を持っておく

技術関連

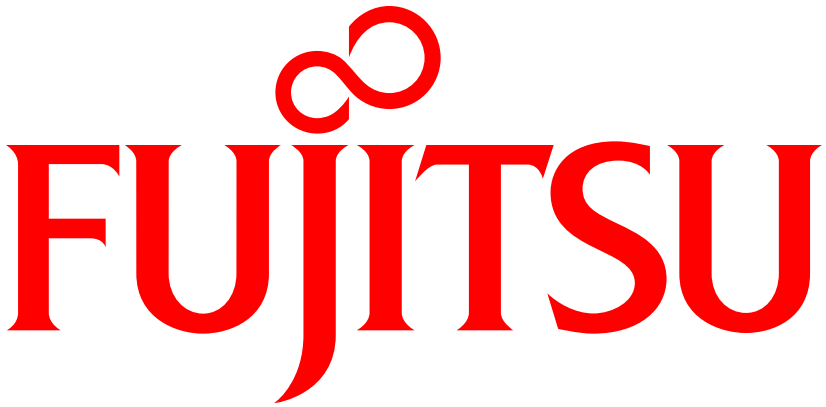
FUJITSU

- 知的所有権 (IPR)
  - 最近、大学も IPR に厳しくなってきた
  - Open Source のソフト
- 標準化
  - 同じ船作戦

でも、最後は…

FUJITSU

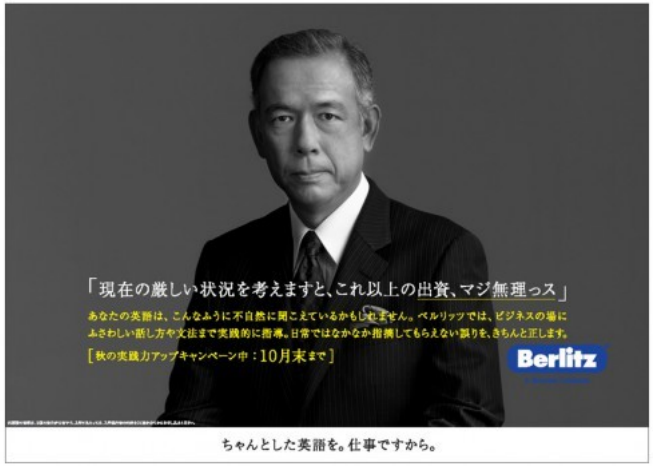
- 自分が何をやりたいかの明確かつ強い意志を持ち、
  - しかし頑迷にならず、大きな目的のためには柔軟に対処する
- 一度ではあきらめないで、
  - 手を変え、品を変え、試してみる
- 誠意を尽くし、がんばる
- 対立しているように見えても、考えればいくらでも両者のメリットになりうる形はありうる
  - 共同研究やインターン



shaping tomorrow with you

IMPROVING YOUR ENGLISH

These Are Okay, but …



Never!

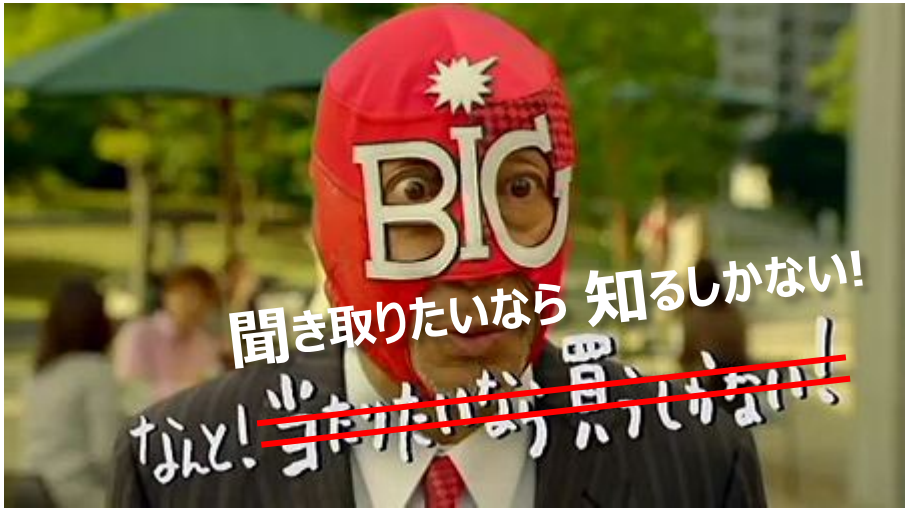
FUJITSU

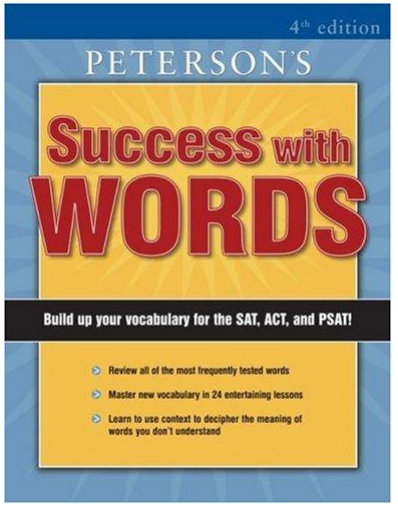
伝える気のない奴は、
英語で仕事をするな!

違いは?

「ナサケの女 ～国税局査察官～」
（テレビ朝日系列）風に

Vocabulary





- 小学生の発表のよう…にならないため
- 知的な会話ができるようになるため

なんと! 英語で言えないことはない

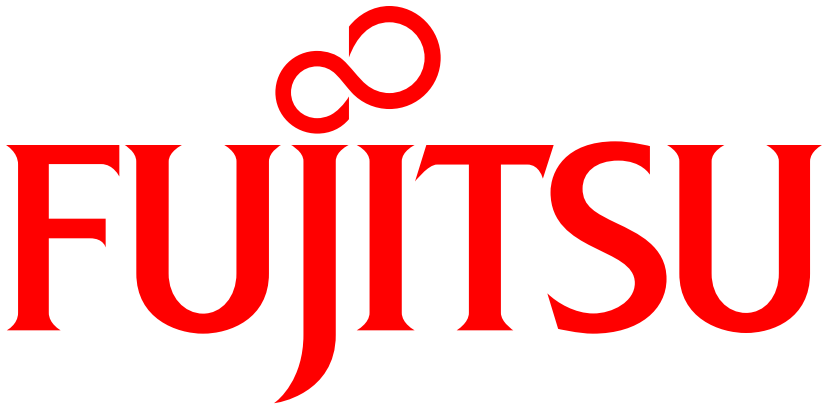
FUJITSU

- … もし英語だけで考えれば
  - 本当にそう。… 表現できないことは思いつきもしない。
    - 英語圏の子供はそう
  - 日本語でも全く同じ – 考える言語の重要性
    - 日本語で考えたことを英語にしようとするとすごく大変、スムースな会話は失われる
  - だけどそれは必ずしも rich な会話を保証しない
    - You know …, you know …, … like …, … like …etc.
    - 限定された語彙の範囲
  - 日本語もしゃべれるあなたは英語しかしゃべれない人たちに比べてより豊かな思考世界を持ちうる可能性がある!
- だからこそ、語彙を増やすことが重要
  - 語彙を増やすことは英語 (欧米) 圏で蓄積されてきたいろんな考えを学ぶこと
  - 余力があれば、rich な表現も

よりよく伝えるために

FUJITSU

- Distraction を少なくする
  - Stress や Articles を正確に使う
  - 間違えるとそれらが stand out してしまい、文のコンテントから聴衆の気が逸れてしまう
- Stress
  - Stress に情報を多く乗せている。ここを違えると通じない
  - 文でも、単語でも、ちゃんと stress を乗せていく
  - 恥ずかしいくらい stress を強調してちょうどいいくらい
  - Stress が違うとその言葉が stands out する
- Articles (A, An, The) and Single/Plural Forms
  - 冠詞が違ったり、単数のところを複数にすると、その単語が stand out する

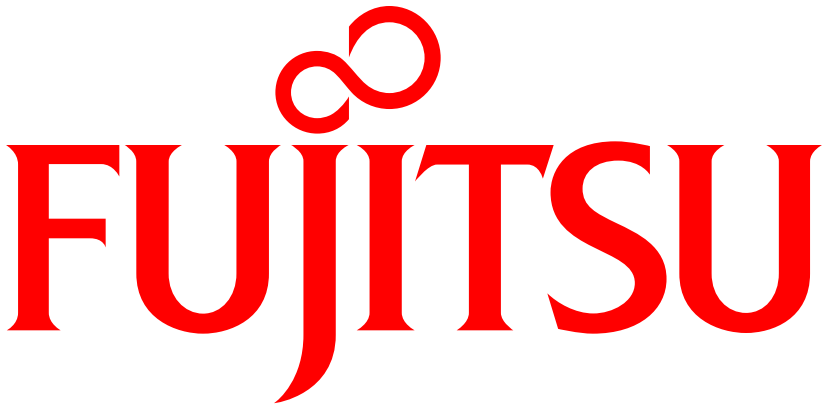


shaping tomorrow with you

YOUR GREATEST ASSET

FUJITSU

訳のわからないものを面白がれる能力



shaping tomorrow with you