
Please stand by

Automating System Assembly of Aerospace Systems

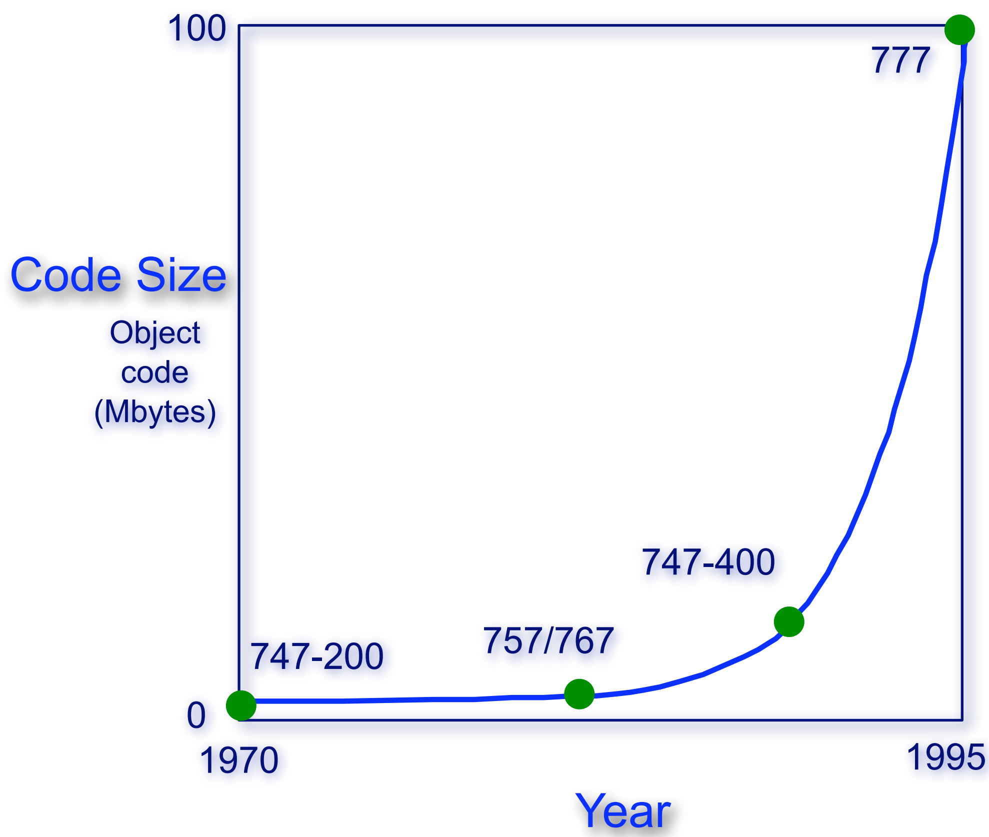
Pete Manolios
Northeastern University
Joint work with
Gayatri Subramanian and Daron Vroon

LANGLEY FORMAL METHODS

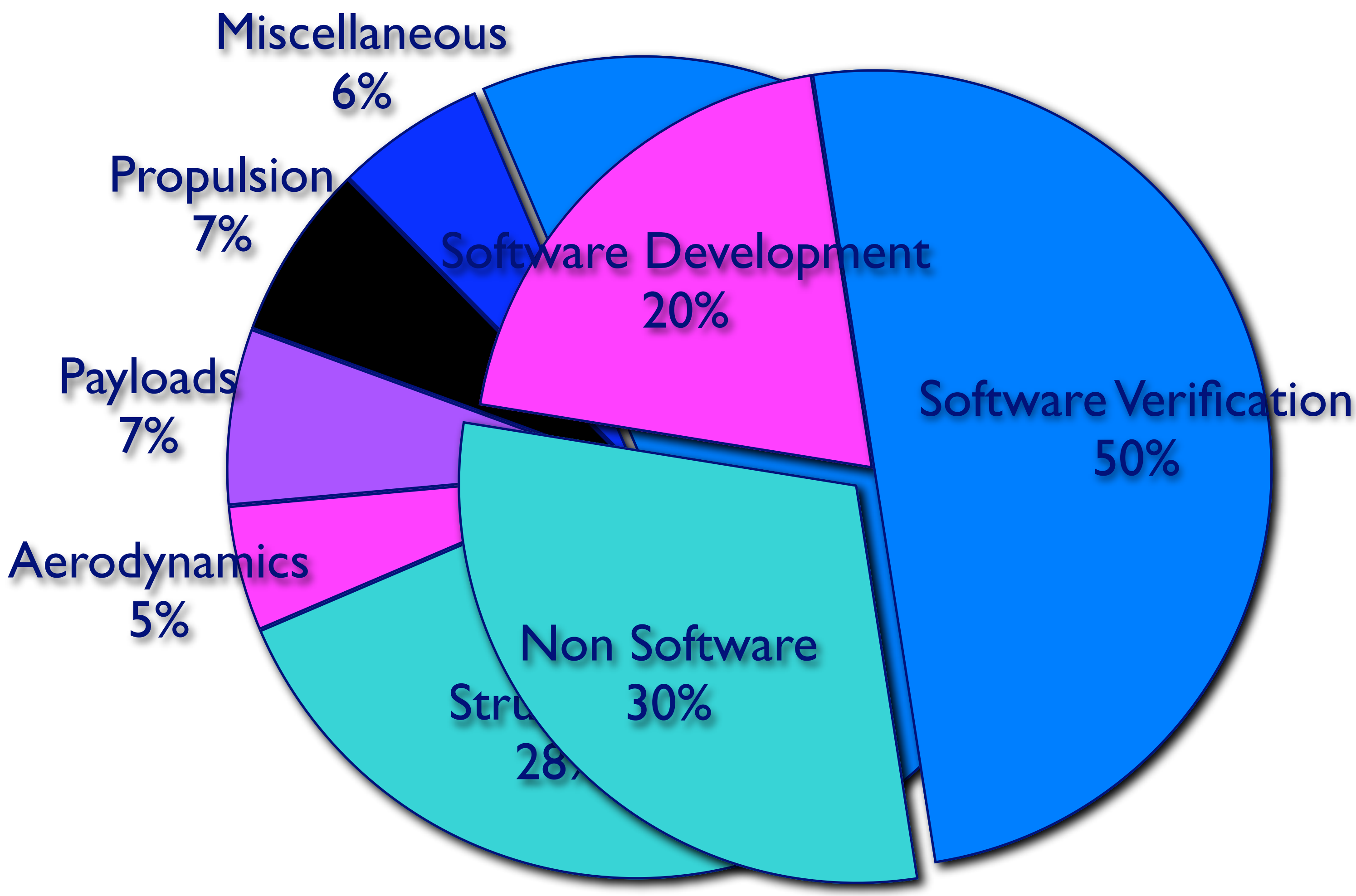
Newport News, VA

April 2008

Commercial Air Transport



Air Transport Development Costs



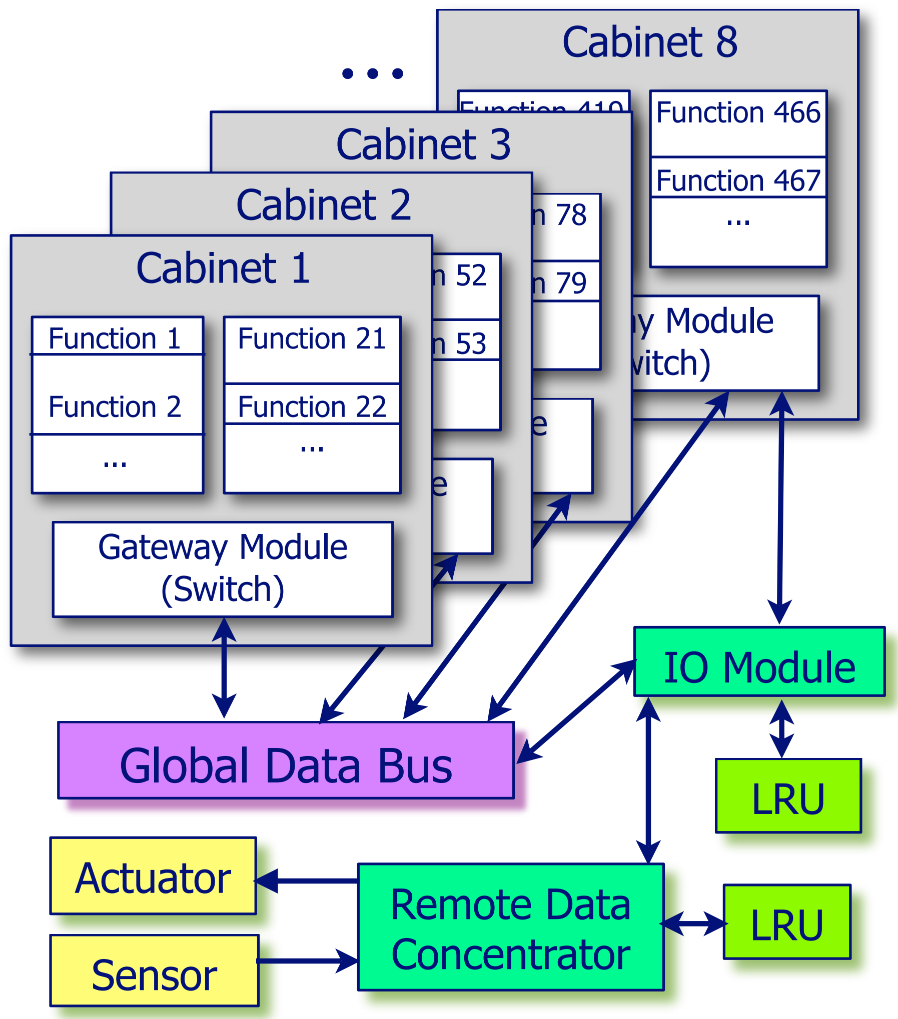
Is Boeing a Software Company?

- Software development and verification account for 1/3 cost
  - Important to build reliable, dependable commercial avionics systems
  - The industry is heavily regulated by the FAA
- The military side is also very dependent on software
  - 1960 - 8% F4 fighter capability came from software
  - 2000 - 85% F22 fighter capability provided by software
  - Even more now

- Boeing's core competence is system integration
  - New business model
  - Dependent on large network of suppliers, globally distributed

Integrated Modular Avionics

- Past: federated systems
- IMA: shared resources
- COTS components
- Multiplexed communication
- Smaller, lighter, cost-effective components
- Powerful computer processing modules handle multiple apps
- Cabinets are connected to global data bus, IO modules, LRUs, sensors, actuators, etc.
- Integration, configuration, assembly?



Component-Based System Design

- Goals of CBSD
  - Construction of systems from independent components
  - Use of commercial-off-the-shelf (COTS) components
  - Separation of concerns
  - Decrease risk, system complexity, development time & cost
  - Increase reliability, malleability, and flexibility
- Domain-specific challenges
  - System architecture
  - Interface definitions
  - Trusted infrastructure
  - Problem domain decomposition
  - …

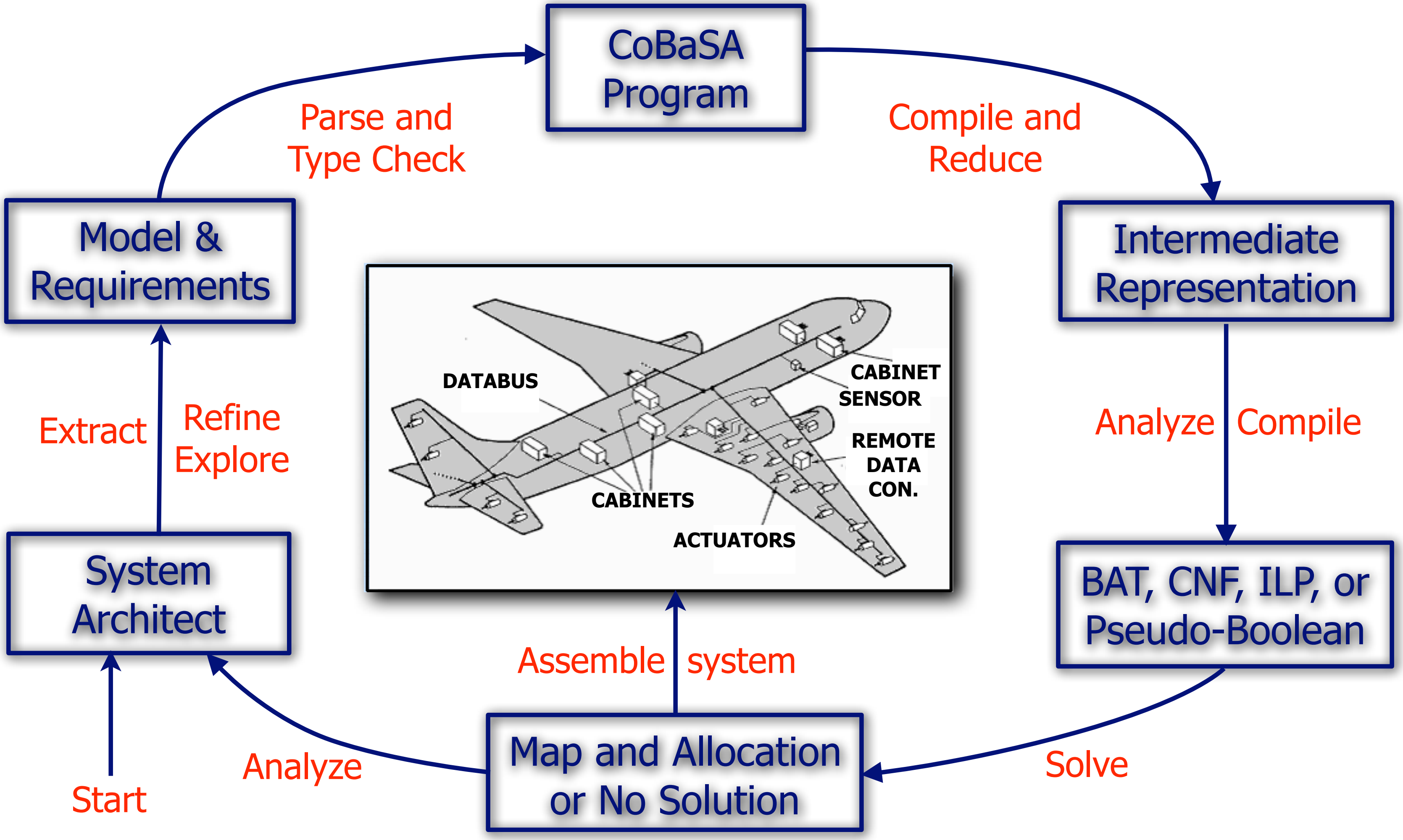
System Assembly

- The general challenge is the system assembly problem:
  - From a pool of available components,
  - Which should be selected &
  - How should they be connected, integrated, assembled
  - So that system requirements are satisfied?
- Currently this is application specific and labor intensive
- Our focus is on automation
  - Algorithmically find optimal solutions directly from requirements
  - Insight: We can reduce system assembly to a satisfiability question
  - Does there exist a way of selecting & assembling components that satisfies the system requirements?

CoBaSA System

- Developed CoBaSA: Component-Based System Assembly
- An object-oriented modeling language
- A declarative constraint language
- Assembly is solved using formal verification technology
- Used CoBaSA to solve actual Boeing problems

Assembly of Avionics Systems



Outline

Motivation

System Assembly

CoBaSA Language

CoBaSA Constraint Solving

Case Study

Conclusions and Future Work

CoBaSA Modeling Language

- Needed complete control of syntax and semantics
- Developed our own language
- Object-oriented language
- Functions as a target language
- This is what we did with the Boeing project

CoBaSA Data Types

- Basic data types:
  - Booleans
  - Strings
  - Integers & integer ranges
  - Enumerated data types
- Complex data types include:
  - Recursive data types
  - Entities (classes)
  - Multidimensional arrays

CoBaSA Language: Entities

```
entity server {
  ;id string
  ;ram-available int
  ;cpu-time-available 10000
  ;secure bool
}
```

```
entity process {
  ;id string
  ;ram-req int
  ;cpu-time-req int
  ;sec-req bool
}
```

```
entity linux-server extends server {
  ;max-num-procs int
  ;neighbor linux-server
}
```

CoBaSA Language: Maps

- Variable declarations

```
var linux-server[20] linux-servers =
      [ { ;120 ; ;"LS-001" ;1024 ; ;False}, …
        { ;80 ;ls[2] ;"LS-020" ;512 ; ;True}]

var process[500] processes = ...
```

- Objects can be assigned values, but write-once memory

```
assign linux-servers[0].neighbor = linux-servers[19]
```

- Map constraints: map consumers to resource providers

```
map proc-serve processes linux-servers
```

- Field constraints: specify dependence between consumer and resource fields

```
constraint proc-serve ((ram-req, cpu-time-req))
                      ((ram-available, cpu-time-available))
```

CoBaSA Language: Constraints

- Arbitrary Boolean & relational constraints
- Boolean expressions with map references
- Relational arithmetic expressions
- Quantification: universal and summation

```
For_all p in processes
   For_all s in linux-servers
      (proc-serve(p,s) and p.sec-req) implies s.secure
```

- Preprocessing w/ Lisp code: (let ((v1 a1) … (vn an)) <lisp code>)

```
For_all s in linux-servers
   Sum p in processes proc-serve(p,s)
   <= (let ((v1 s.max-num-procs))
          _(floor (* 0.75 v1))_)
```

More CoBaSA Constraints

- Optimization
  - An objective function can maximized or minimized
- Interdependent maps
  - Result of one map affects the result of another map
  - Arise from hierarchies of resource/consumer relationships
- Examples of generalized notion of maps
  - To express relation, r, over A, B: map >= 0 r A B
  - 2-function, f, from D to R: map = 2 f D R

Outline

Motivation

System Assembly

CoBaSA Language

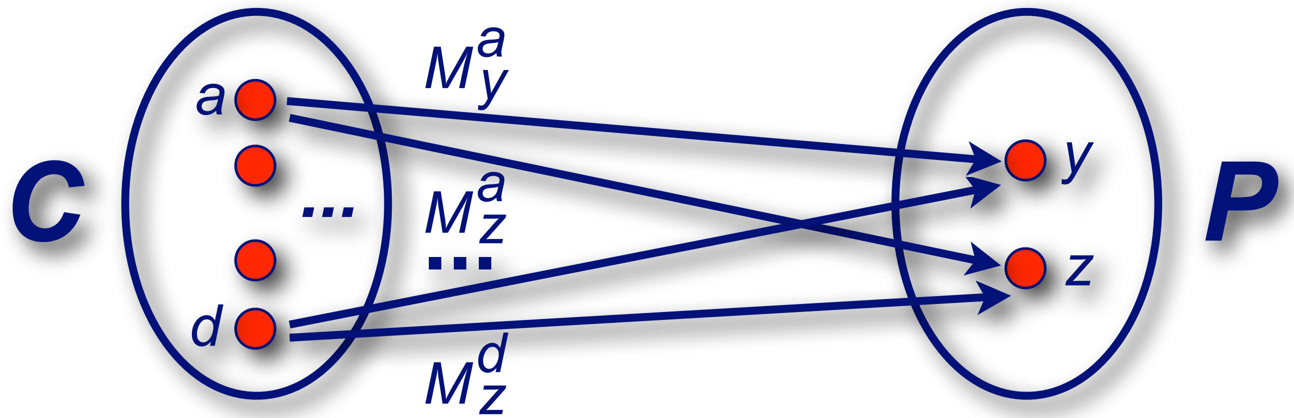
CoBaSA Constraint Solving

Case Study

Conclusions and Future Work

CoBaSA Constraint Solving

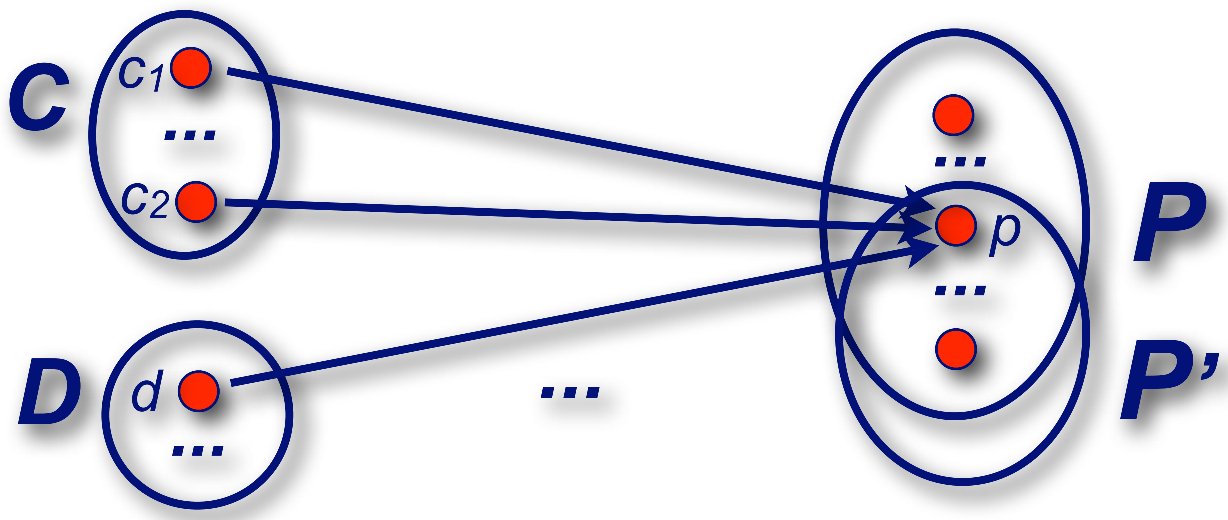
- CoBaSA programs are reducible to 0-1 integer programming
- Also known as pseudo-boolean SAT problems
  - Linear constraints of the form $\sum_{i=1}^{n} c_i x_i \ R \ c$
- For each map $M : C \rightarrow P$, we have an implicit constraint that elements of *C* map to elements of *P*



- For each *c* in *C*: $\sum_{p \in P} M_p^c = 1$

Solving Field Constraints

- We have to guarantee that *p* can provide resources for every consumer, *c*, mapped to *p*



- We express the above using pseudo-boolean constraints
- And we continue with a sequence of such transformations

Outline

Motivation

System Assembly

CoBaSA Language

CoBaSA Constraint Solving

Case Study

Conclusions and Future Work

Case Study: Boeing

- Models developed over several years
- The models are complex; they include:
  - I/O time
  - Latency
  - Network jitter
  - Context switching time
  - Cache flushing time
  - Memory latencies
  - Thousands of constraints
- Based on worst-case execution time
- Models are over 500K in size

Evaluation of Case Study

- Given collection of models from simple to complete
- No feasible solution was previously known
- Even the very simple, initial models:
  - Takes 3 person-weeks to describe problem & check solution
  - **Much** longer to solve with previous approaches
- We solve simple models in seconds
- We can solve the most complex models in minutes
- Allowed Boeing "to solve, in person-weeks, problems that were previously taking person-years"
- Flexible enough to accommodate what Boeing described as "serious architecture changes"

Outline

Motivation

System Assembly

CoBaSA Language

CoBaSA Constraint Solving

Case Study

Conclusions and Future Work

Summary

- Introduced the notion of system assembly
- Showed how to automatically solve system assembly problems [MSV'07,SAT'07,CAV'07]
- Developed CoBaSA system
  - Object oriented modeling language
  - Declarative constraint language
  - Decision procedure
- Showed the effectiveness and applicability of our work by solving problems arising in design of Boeing Dreamliner
- Can solve problems previously taking person-years

Future Work

- Algorithmic extensions
  - Hierarchical refinement (a component is a collection)
  - Better decision procedures
- Design support
  - If assembly is not possible, why not?
  - Threat analysis: what will drastically affect solution landscape?
- Adaptive assembly
  - Can we assemble & reconfigure in real time?
  - In response to system failure? account environmental factors?
  - Changes in mission priorities? response to invalid assumptions?
  - Under extreme conditions (low power, long latencies, …)
- Scheduling, power, weight, geometry, … .