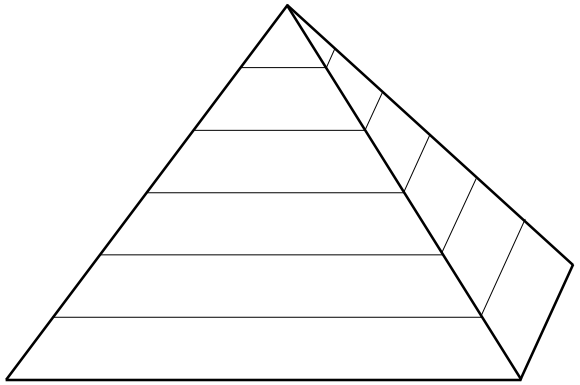
Automated Rearchitecting of Avionics Mission Software

Ira D. Baxter
Semantic Designs, Inc.
12636 Research Blvd, Suite C214
Austin, Texas 78759
idbaxter@semdesigns.com

OMG ADM Modernization Workshop
October, 2005

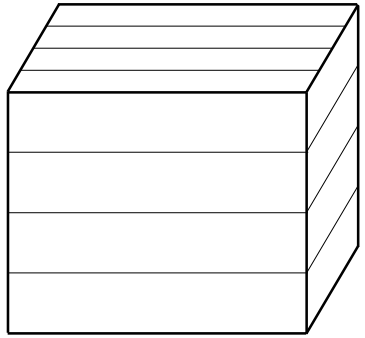
An Egyptian Legacy System



30 years to build
10,000 slaves

An impossible change: How can we reengineer this?

… Modernized



Requires massive reengineering

➡ ***TOOLS!!!***

Tools change your definition of architecture!

How to Modernize Real Code?

Converting Legacy Components into Modern Components

Avionics Application
Legacy Bold Stroke
6000 components
3M SLOC C++

1990s
Distributed Architecture

?

Modern Bold Stroke
6000 components
3M SLOC C++

MoBIES architecture
"PRiSm" 2004

!

- **Apply a Program Transformation System**
  DMS operating on C++
- **Using Custom Transforms to Revise Architecture**
  Restructure components into new form

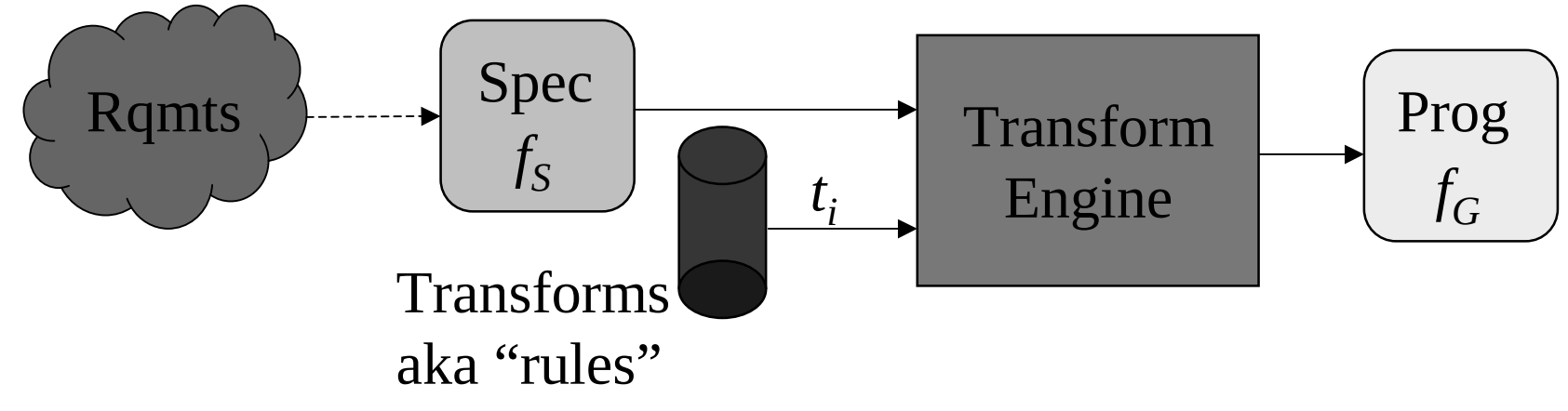
implement BMT, a practical conversion tool

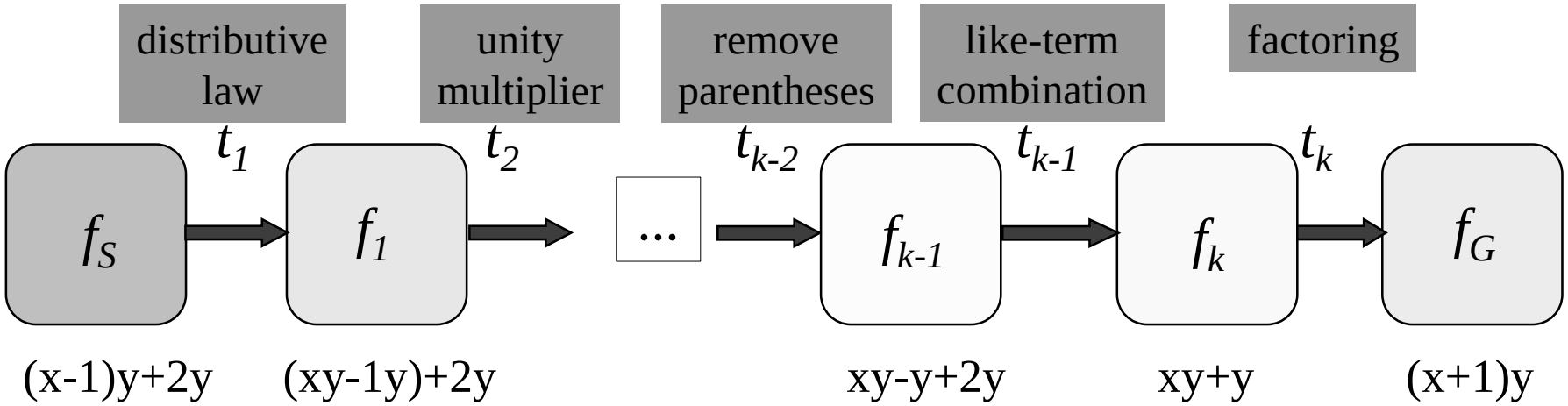
Software Mass Change Tools Need Infrastructure Too

- To build a *conventional application*
  - Define specifications
  - Implement application
  - Use *Operating System* to provide standard services
    - File/Terminal I/O, CPU management, Security
- To build an *automated software change tool*
  - Define specifications
  - Implement tool
  - Use …?… to provide standard services
    - Lexing/Parsing
    - *Life after Parsing*: Tree/Symbol table building, Analysis management, Transformation, PrettyPrinting
- Our goal: show how such Software Change tools work
  - **DMS**: a practical change-tool infrastructure
  - **BMT**: An application to legacy BoldStroke system

Key Tool Technology: Transformation Systems

Motivation: Automatic Conversion of Specs to Code





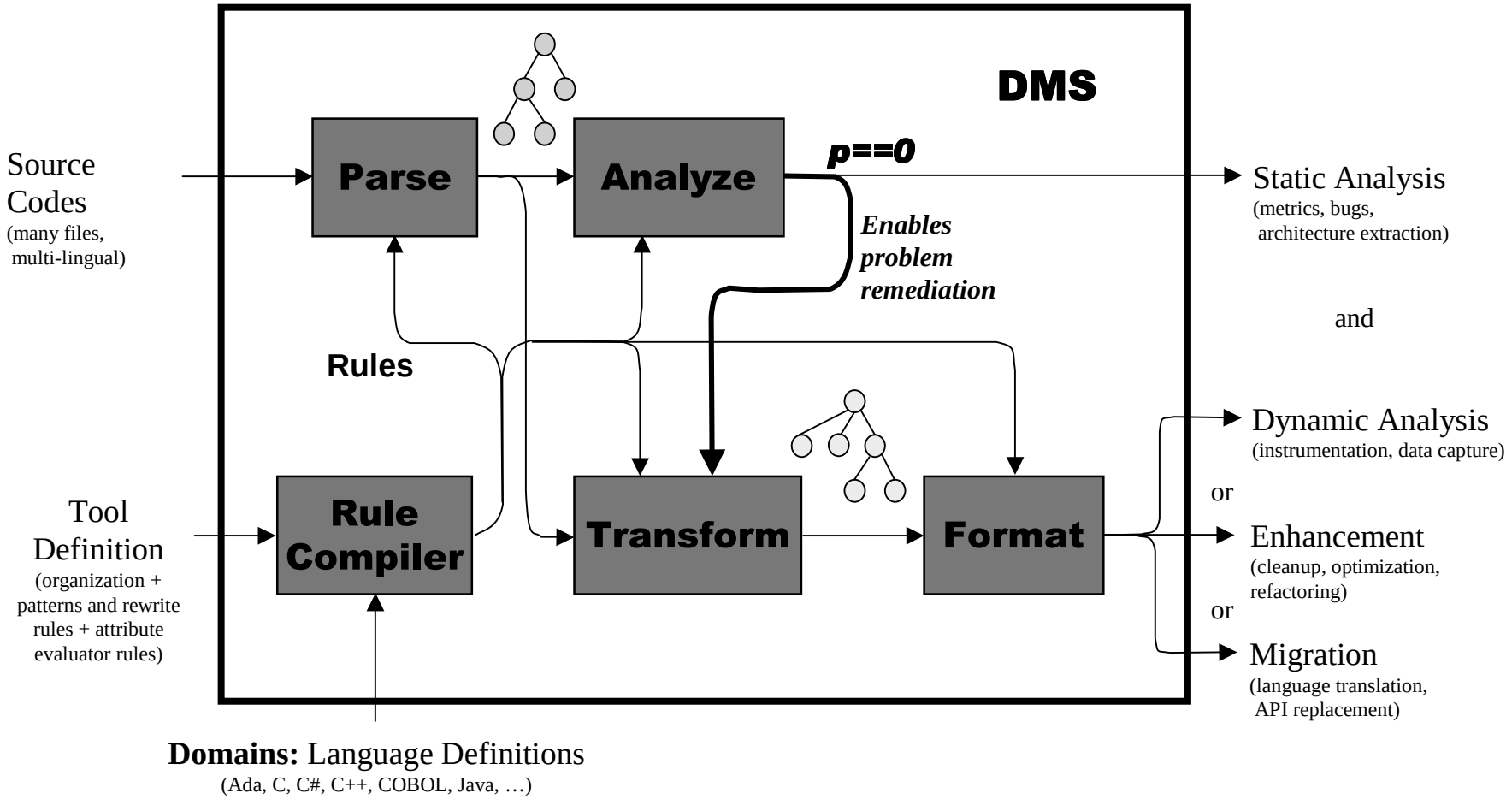
Expertise == Number of *Rules*

- Mathematics
  - Novice (9th grade algebra): x+0 ⇒ x
  - Amateur (HS Senior): sin^2(x)+cos^2(x) ⇒ 1
  - Journeyman: (Frosh Calculus) integrals
  - Craftsman: (B.S. Math) Linear Algebra, Group Theory
  - Expert: (Ph.D. Mathematics) Category Theory, Topology, …
- Transformation Engines
  - Toy: several rules
  - Useful: 50 rules (simplification/optimization)
  - Powerful: 250 rules (testing, code generation)
  - Indispensable: 2000 rules (massive program translation)

… + "*how to use rules*" knowledge = MetaProgram

- Mathematics
  - Solve for variable, Simplification (novice)
  - Simultaneous equations
  - Change of domain (expert)
  - …
- Transformation Engines
  - Tactic
    - Apply rules till exhaustion (useful)
  - Strategy
    - Metaprograms (sophisticated): rule sequencing

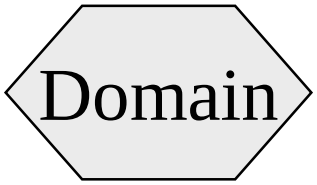
DMS: generalized compiler technology



Fundamental Issue: *Scale*

- Engineering hard but straightforward
  - Ugly details: C preprocessors, etc.
- Legacy Systems are huge
  - MSLOC + tens of thousands of files
    - Careful design of hypergraph to conserve space
  - Arbitrary languages => robust parsing technology
    - Need for ***domain agility***: fast domain/dialect definition
    - *Use domain notations to define knowledge*
  - Applications use multiple languages
    - reasoning and transforms must work with mix
- Reasoning/Analysis costs
  - "Incremental" Knowledge capture => domains
  - Computers do Symbolic computation slowly => *Parallel foundations*

Knowledge Capture:

DMS  Parts

- Syntax (domain notation)
  - External Form (what you can say: string or graphical)
  - Internal Form (How DMS stores it)
  - Parser (how to convert external form to internal form)
  - PrettyPrinter (how to display the Internal Form)
- Semantics (what the domain *means*)
  - Analyzers (how to analyze in the domain)
  - Optimizations (how to optimize in the domain)
  - Refinements (how to transform one domain to another

DMS Domain for Java
Parser + Pretty Printer

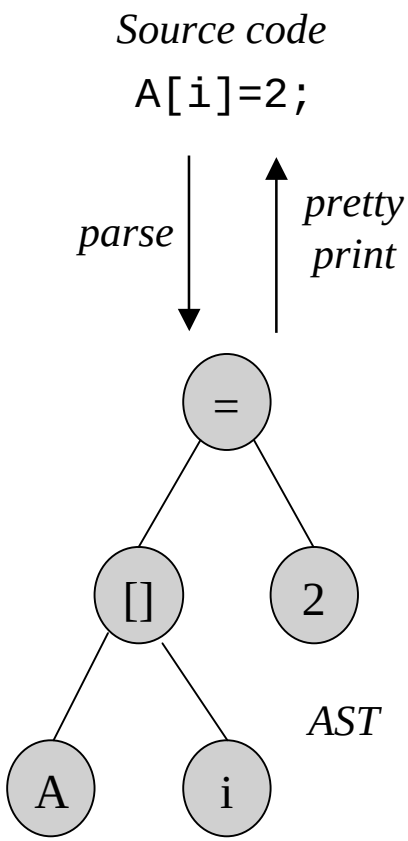
```
nested_class_declaration = nested_class_modifiers class_header class_body ;
  <<PrettyPrinter>>:  { V(H(nested_class_modifiers,class_header),class_body); }

class_header = 'class' IDENTIFIER ;
  <<PrettyPrinter>>:  { H('class',IDENTIFIER); }
class_header = 'class' IDENTIFIER 'implements' name_list ;
  <<PrettyPrinter>>:  { H('class',IDENTIFIER,'implements',name_list); }
class_header = 'class' IDENTIFIER 'extends' name;
  <<PrettyPrinter>>:  { H('class',IDENTIFIER,'extends',name); }
class_header = 'class' IDENTIFIER 'extends' name 'implements' name_list ;
  <<PrettyPrinter>>:  { H('class',IDENTIFIER,'extends',name,'implements',name_list); }

class_body = '{' class_body_declarations '}' ;
  <<PrettyPrinter>>:  { V(H('{',STRING(" "),class_body_declarations),'}'); }

nested_class_modifiers = nested_class_modifiers nested_class_modifier ;
  <<PrettyPrinter>>:  { H(CH(nested_class_modifiers[1]),nested_class_modifier); }
```

… + 300 more rules…(COBOL is 3500!)

Parsing/PrettyPrinting Abstract Syntax Trees (AST)

A Program Representation used by Compilers

- Use DMS grammar domain to define language syntax
- DMS generates parser/prettyprinter automatically
- *Reengineering* parser reads source file(s)
  - Carries out lexical conversions (FP text -> IEEE binary)
  - Captures comments and formats of literals
  - Builds Abstract Syntax Tree
  - Records Position of every node (file, line, col)
  - Constructs Symbol Table if needed
- PrettyPrinter regenerates source text using lexical clues
  - Preserve formatting where possible
  - Generate new formatting



Winner: Obfuscated “C” Contest

The Maintenance Programmer’s Nightmare

```
#include                                    <math.h>
#include                                   <sys/time.h>
#include                                   <X11/Xlib.h>
#include                                  <X11/keysym.h>
                                          double L ,o ,P
                                         ,_=dt,T,Z,D=1,d,
                                         s[999],E,h= 8,I,
                                         J,K,w[999],M,m,O
                                        ,n[999],j=33e-3,i=
                                        1E3,r,t, u,v ,W,S=
                                        74.5,l=221,X=7.26,
                                        a,B,A=32.2,c, F,H;
                                        int N,q, C, y,p,U;
                                       Window z; char f[52]
                                    ; GC k; main(){ Display*e=
 XOpenDisplay( 0); z=RootWindow(e,0); for (XSetForeground(e,k=XCreateGC (e,z,0,0),BlackPixel(e,0))
; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400,
0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=sin(O)){ struct timeval G={ 0,dt*1e6}
; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_*P; r=E*K; W=cos( O); m=K*W; H=K*T; O+=D*_*F/ K+d/K*E*_; B=
sin(j); a=B*T*D-E*W; XClearWindow(e,z); t=T*E+ D*B*W; j+=d*_*D-_*F*E; P=W*E*B-T*D; for (o+=(I=D*W+E
*T*B,E*d/K *B+v+B/K*F*D)*_; p<y; ){ T=p[s]+i; E=c-p[w]; D=n[p]-L; K=D*m-B*T-H*E; if(p [n]+w[ p]+p[s
]== 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T-a *E)> K)N=1e4; else{ q=W/K *4E2+2e2; C= 2E2+4e2/ K
 *D; N-1E4&& XDrawLine(e ,z,k,N ,U,q,C); N=q; U=C; } ++p; } L+=_* (X*t +P*M+m*l); T=X*X+ l*l+M *M;
  XDrawString(e,z,k ,20,380,f,17); D=v/l*15; i+=(B *l-M*r -X*Z)*_; for(; XPending(e); u *=CS!=N){
                                   XEvent z; XNextEvent(e ,&z);
                                       ++*((N=XLookupKeysym
                                         (&z.xkey,0))-IT?
                                         N-LT? UP-N?& E:&
                                         J:& u: &h); --*(
                                         DN -N? N-DT ?N==
                                         RT?&u: & W:&h:&J
                                          ); } m=15*F/l;
                                         c+=(I=M/ l,l*H
                                         +I*M+a*X)*_; H
                                         =A*r+v*X-F*l+(
                                         E=.1+X*4.9/l,t
                                         =T*m/32-I*T/24
                                          )/S; K=F*M+(
                                          h* 1e4/l-(T+
                                          E*5*T*E)/3e2
                                          )/S-X*d-B*A;
                                          a=2.63 /l*d;
                                          X+=( d*l-T/S
                                           *(.19*E +a
                                           *.64+J/1e3
                                           )-M* v +A*
                                           Z)*_; l +=
                                           K *_; W=d;
                                           sprintf(f,
                                           "%5d  %3d"
                                           "%7d",p =l
                                          /1.7,(C=9E3+
                                  O*57.3)%0550,(int)i); d+=T*(.45-14/l*
                                 X-a*130-J* .14)*_/125e2+F*_*v; P=(T*(47
                                 *I-m* 52+E*94 *D-t*.38+u*.21*E) /1e2+W*
                                 179*v)/2312; select(p=0,0,0,0,&G); v-=(
                                  W*F-T*(.63*m-I*.086+m*E*19-D*25-.11*u
                                   )/107e2)*_; D=cos(o); E=sin(o); } }
```

… Parsed then Pretty Printed

```
#include <math.h>
#include <sys/time.h>
#include <X11/Xlib.h>
#include <X11/keysym.h>
double L, o, P, _ = dt, T, Z, D = 1, d, s[999], E, h = 8, I, J, K, w[999],
       M, m, O, n[999], j = 3.3e-2, i = 1e3, r, t, u, v, W, S = 7.45e1,
       l = 221, X = 7.26, a, B, A = 3.22e1, c, F, H;
int N, q, C, y, p, U;
Window z;
char f[52];
GC k;
main()
{
  Display * e = XOpenDisplay(0);
  z = RootWindow(e, 0);
  for (XSetForeground(e, k = XCreateGC(e, z, 0, 0), BlackPixel(e, 0));
  scanf("%lf%lf%lf", y + n, w + y, y + s) + 1; y++);
  XSelectInput(e, z = XCreateSimpleWindow(e, z, 0, 0, 400, 400,
              0, 0, WhitePixel(e, 0)), KeyPressMask);
  for (XMapWindow(e, z);; T = sin(O))
    {
      struct timeval G = { 0, dt * 1e6 };
      K = cos(j);
      N = 1e4;
      M += H * _;
      Z = D * K;
      F += _ * P;
      r = E * K;
      W = cos(O);
      m = K * W;
      H = K * T;
      O += D * _ * F / K + d / K * E * _;
      B = sin(j);
      a = B * T * D - E * W;
      XClearWindow(e, z);
      t = T * E + D * B * W;
      j += d * _ * D - _ * F * E;
      P = W * E * B - T * D;
      for (o += (I = D * W + E * T * B, E * d / K * B + v + B / K * F * D) * _; p < y;)
        {
          T = p[s] + i;
          E = c - p[w];
          D = n[p] - L;
          K = D * m - B * T - H * E;
          if (p[n] + w[p] + p[s] == 0 | K < fabs(W = T * r - I * E + D * P) | fabs(D = t * D + Z * T - a * E) > K)
            N = 1e4;
          else
            {
              q = W / K * 4e2 + 2e2;
              C = 2e2 + 4e2 / K * D;
              N - 1e4 && XDrawLine(e, z, k, N, U, q, C);
              N = q;
              U = C;
            }
          ++p;
        }
      L += _ * (X * t + P * M + m * l);
      T = X * X + l * l + M * M;
      XDrawString(e, z, k, 20, 380, f, 17);
      D = v / l * 15;
      i += (B * l - M * r - X * Z) * _;
      for (; XPending(e); u *= CS != N)
        {
          XEvent z;
          XNextEvent(e, & z);
          ++ * ((N = XLookupKeysym(& z.xkey, 0)) - IT ? N - LT ? UP - N ? & E : & J : & u : & h);
          -- * (DN - N ? N - DT ? N == RT ? & u : & W : & h : & J);
        }
      m = 15 * F / l;
      c += (I = M / l, l * H + I * M + a * X) * _;
      H = A * r + v * X - F * l + (E = 1e-1 + X * 4.9 / l, t = T * m / 32 - I * T / 24) / S;
      K = F * M + (h * 1e4 / l - (T + E * 5 * T * E) / 3e2) / S - X * d - B * A;
      a = 2.63 / l * d;
      X += (d * l - T / S * (1.9e-1 * E + a * 6.4e-1 + J / 1e3) - M * v + A * Z) * _;
      l += K * _;
      W = d;
      sprintf(f, "%5d  %3d"
                 "%7d", p = l / 1.7, (C = 9e3 + O * 5.73e1) % 0550, (int) i);
      d += T * (4.5e-1 - 14 / l * X - a * 130 - J * 1.4e-1) * _ / 1.25e4 + F * _ * v;
      P = (T * (47 * I - m * 52 + E * 94 * D - t * 3.8e-1 + u * 2.1e-1 * E) / 1e2 + W * 179 * v) /
2312;
      select(p = 0, 0, 0, 0, & G);
      v -= (W * F - T * (6.3e-1 * m - I * 8.6e-2 + m * E * 19 - D * 25 - 1.1e-1 * u) / 1.07e4) * _;
      D = cos(o);
      E = sin(o);
    }
}
```

Rule Specification Language
A way to write Program Transformation *Rules*

```
default base domain Cpp;

rule unit_quotient(e: expression) =
   "\e / \e"  -> "1"
    if is_integral(e)
       /\ no_side_effects(e).

ruleset simplify =
    { unit_quotient, plus_0, times_1, equal_self  }
    + boolean_simplify.

pattern getter_method(t: typespecifier,
                      attribute: identifier) =
  "\t \get\(\attribute\)
      { return \attribute; }"
   with side-effect add_new_symbol(get(attribute)).
```

Domain Name

Quoted domain Syntax

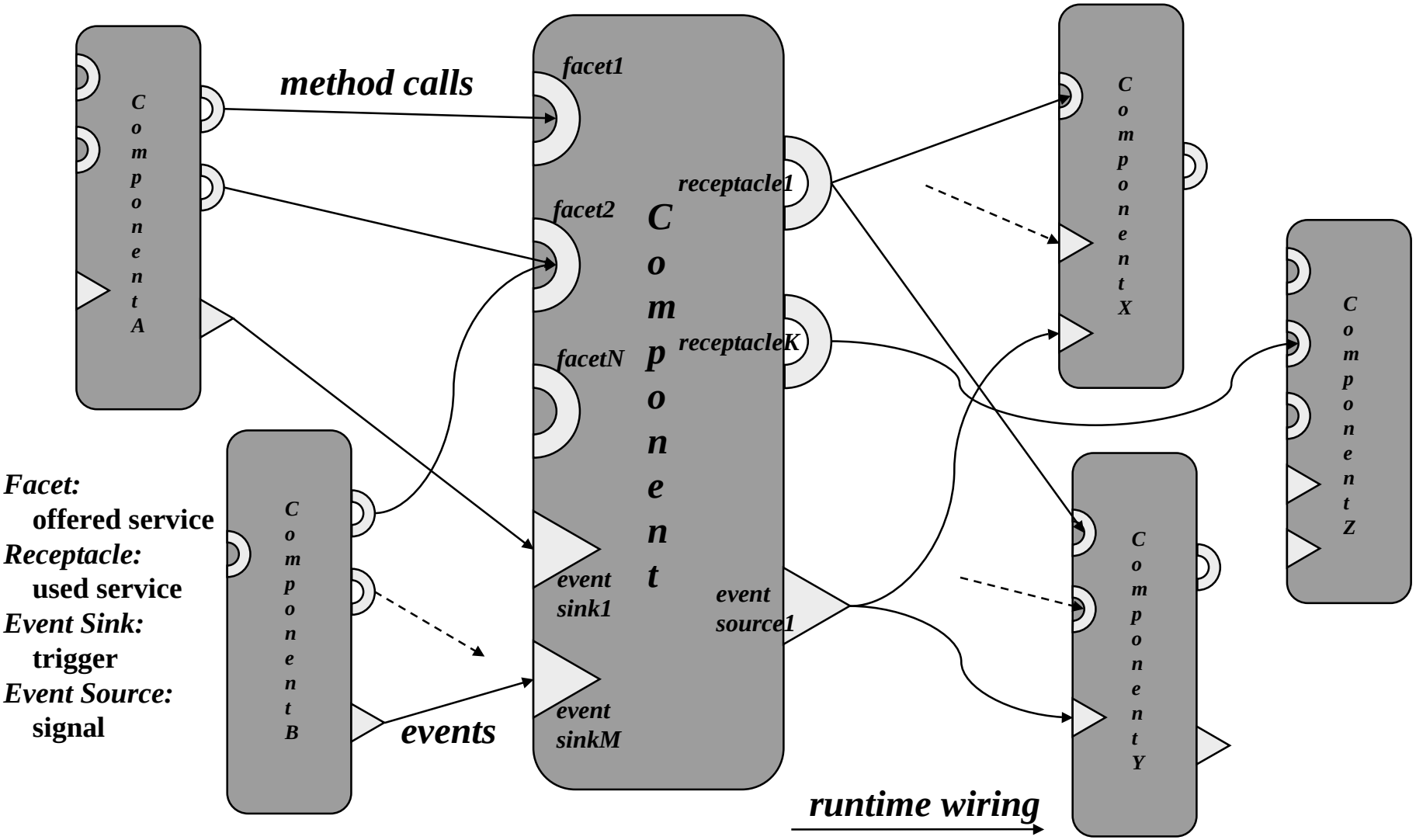
Semantic condition

Meta function

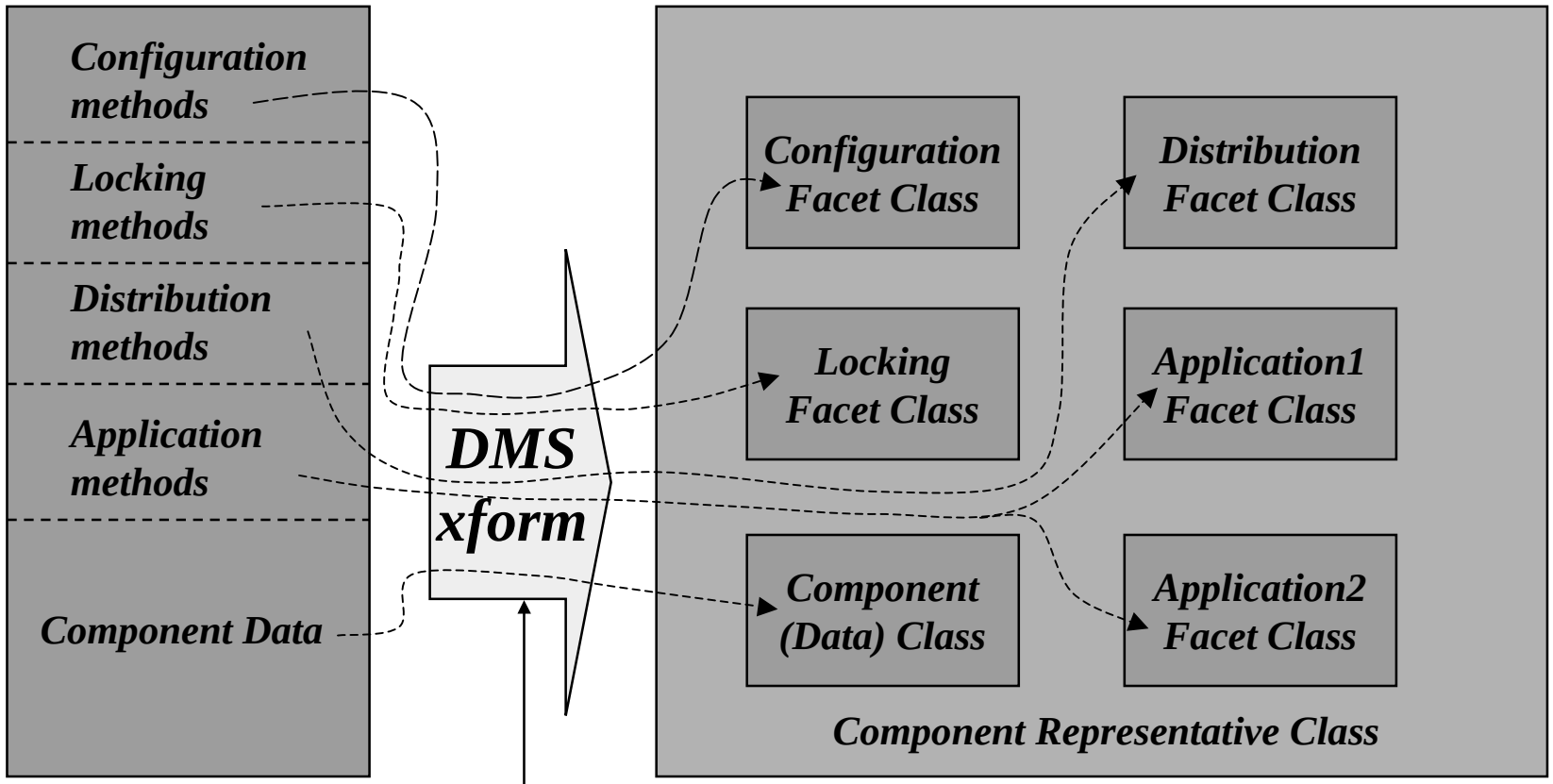
BMT: Rearchitecting distributed Avionics Software

- Boeing BoldStroke: 6000 C++ modules
  - Avionics Mission Software (~~ 1M SLOC/plane)
    - Product Line: Used in several different military airframes
  - Distributed across multiple CPUs in airplane
  - ***Architected 1992***
    - as components with monolithic APIs
    - use custom distributed data protocols
- Goal: Rearchitect into more reusable parts
  - Restructure APIs into conceptually clean groups
  - Move towards CORBA/RT component model
  - Change communication to use ORBs
  - Use automated tool

CORBA
Distributed Component Architecture



Bold Stroke Component Conversion



Legacy Bold Stroke Component (single C++ class and support classes)

Engineer Specified Facetization

PRiSm Bold Stroke Component:
Always an Interface
(Not shown: any needed support classes)

FACET (Aspect) Specification
Describes desired component structure

```
COMPONENT GPS
        FACET GPS_State
        FACET GPS_UpdateState
        EVENTSINK GPS_changed
        PORT EXECUTION Basic
        PORT EVAPORATION Basic
        PORT PERSISTANCE Basic
        PORT LOCK Basic
END COMPONENT
```

```
FACET GPS_UpdateState
      void Set_GPS();
      void Move_GPS ();
END FACET
```

```
EVENTSINK GPS_change_changed
      // Optional… only needed if standard push convention not used
      void Update_GPS(Events& event);
END EVENTSINK
```

- *A DMS domain*
- *Used to lookup methods*
- *Uses C++ overload name resolution*

```
PORT EXECUTION Basic  (… EVENT SINK)
// Standard methods for execution need not be mentioned.
      void Push (const Events& myEventSet);
END FACET

DELETED Basic  // signatures in this section are removed
   PushSupplier_ptr GetEventSupplierReference ();
END FACET

PERSISTANCE PORT Basic
// Standard methods for execution need not be mentioned.
void MapState (Elements& GPS_Elements);
      void SetPersistenceAdapter (PersistenceAdapter& anAdapterPtr);
     ACE_Lock& GetLock () const;
END FACET

PORT LOCK Basic
// The lock facet is accessible to the public.
      ACE_Lock& GetLock () const;
END FACET
```

Several kinds of change subtasks

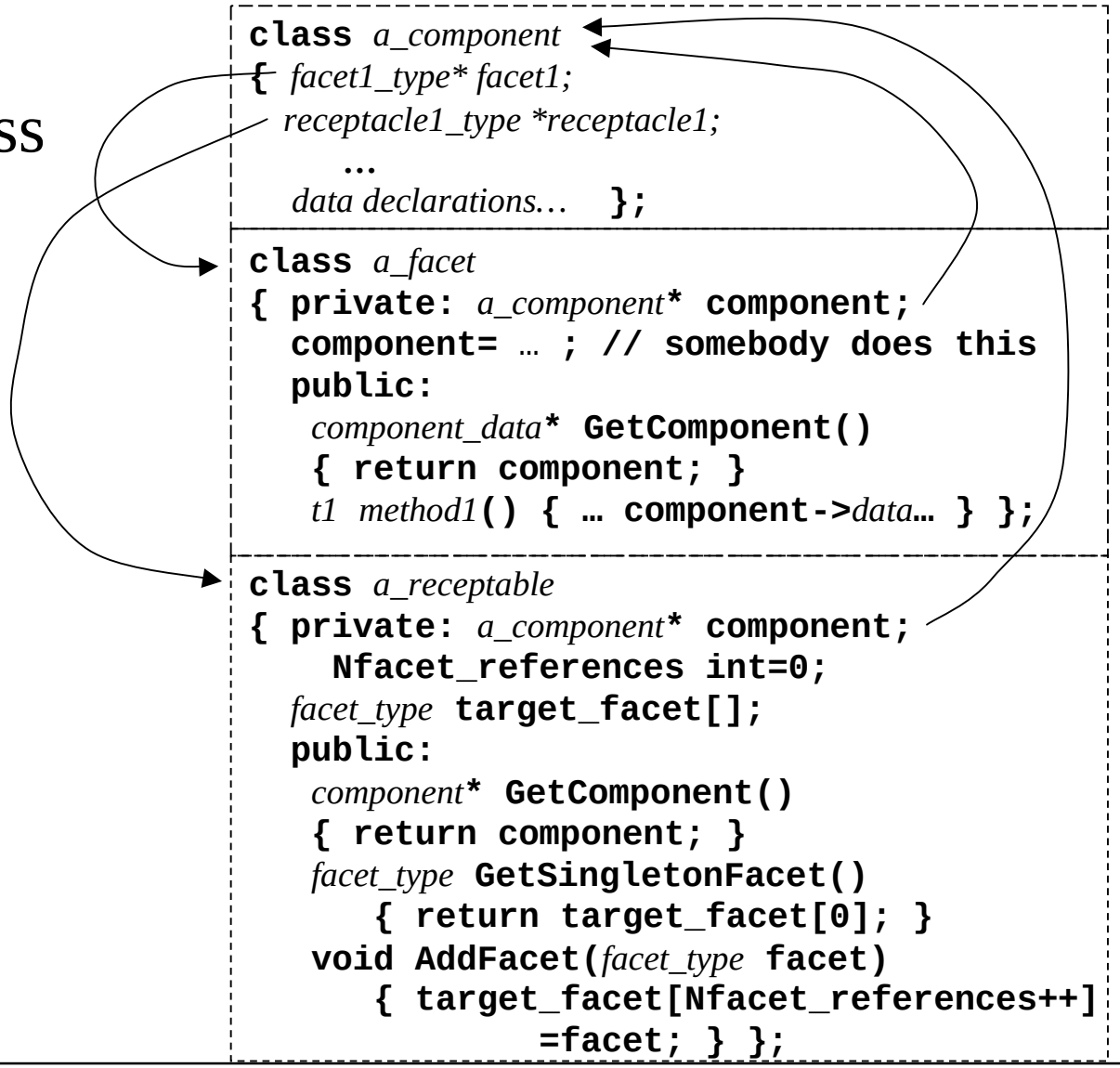
- Creation of new classes with predetermined structure
  - facets, wrappers, EquivalentInterface
    - Drawn from facet specifications and the legacy code
  - Event sinks and receptacles
    - From legacy code fragments recognized by idiom
    - Some related legacy code to be deleted
- Modification of reference access paths in moved methods
  - in legacy classes
  - in code relocated to facets and wrappers
- Extension and modification of name space
  - to support new and relocated code

Doing all this requires…

- Parsing and name and type resolution of C++ code
  - preprocessing directives
    - processed (for semantic analysis)
    - preserved (as modified source code)
- Parsing facet specifications
  - For targeted component and all of component's neighbors
  - Looking up facet method signatures in application code
- Constructive patterns
  - for new classes and their constituents
- Recognizer patterns
  - for recognizing idiomatic forms
- Rewrite rules
  - for access path modification
- Prettyprinters to display re-engineered component
- Sequencing code to wire these pieces together

PRiSm Component Sketch

- Central class
- Facet
- Receptacle



PRiSm idiom as DMS pattern

ComponentEI_impldothFile(ComponentName,ApplicationFacets)

The detailed truth
Used to generate
PRiSm component
instance

```
pattern ApplicationFacets(members: identifier_list)=tag.

pattern GenericComponentEI_Impldoth(ComponentName:identifier,
                                    ApplicationFacets: tag): compilation_unit =

"\ComponentNameComment\(\ComponentName\)
#pragma once
#ifndef \ComponentHFileName\(\ComponentName\)
#define \ComponentHFileName\(\ComponentName\)

#include "GenericComponent.h"

\ApplicationFacetClassDeclarations\(\ApplicationFacets\)

class \ComponentEI_ImplName\(\ComponentName\): public GenericComponent
{ public:
    // Called during first pass of the configurator
    \ComponentEI_ImplName\(\ComponentName\)
        (const UUIdentifier& id, // every created component has an object id
         GenericComponentParameters& parameters, // and some initialization parameters
         \FacetPointerArgumentDeclarations\(\ApplicationFacets\) // One for every facet:
        );
    ~\ComponentEI_ImplName\(\ComponentName\)();
    virtual bool IsNil () const;
    virtual const UUIdentifier& GetId () const;

    \ProvideFacetDeclarations\(\ApplicationFacets\)

  protected:
     // Protected Declarations... none needed for MOBIES components

  private:
     \ComponentEI_ImplName\(\ComponentName\)(); // parameterless constructor; NEVER CALLED
     \ComponentName(const \ComponentName &right); // copy constructor; DO NOT USE
     UUIdentifier id_; \\ component instance name

     \FacetPtrMemberDeclarations\(\ApplicationFacets\)
};
#endif".
```

Rough procedural metaprogram
"How to use rules"

For each faceted component specification
Create central class (instantiate pattern)
Populate with application data
For each specified facet
Create class for facet
Add facet data member, getter to central class
Add central class member, getter to facet class
Copy methods for facet into facet class
Modify all legacy class members accesses:
"\name -> \which_facet_class\(\name\)->\name"
For each specified receptable
Create class for receptacle
Add receptable data member, getter to central class
Add central class member, getter to receptacle class
Process other PRiSm APIs…

BMT Toy Example: Spec

Example "legacy code"

FACET specification

```
COMPONENT GPS
  FACET GPS_This
  FACET GPS_Other
END COMPONENT

FACET GPS_This
  "GetStatus";
  "ThisFacetStatus";
END FACET

FACET GPS_Other
  "OtherFacetStatus";
END FACET
```

Not explicitly specified what to do

```
unsigned int GlobalStatus();

class GPS
  { public:
    unsigned int GetStatus();
    unsigned int ThisFacetStatus();

    unsigned int OtherFacetStatus();

    unsigned int ComponentStatus();

    unsigned int status_;
  };

unsigned int GPS::GetStatus()
  { return status_;
  };

unsigned int GPS::ThisFacetStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = ComponentStatus();
    unsigned int tmp3 = ThisFacetStatus();
    unsigned int tmp4 = OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }

unsigned int GPS::OtherFacetStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = ComponentStatus();
    unsigned int tmp3 = ThisFacetStatus();
    unsigned int tmp4 = OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }

unsigned int GPS::ComponentStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = ComponentStatus();
    unsigned int tmp3 = ThisFacetStatus();
    unsigned int tmp4 = OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }
```

BMT Toy Example: Component

FACET specification

```
COMPONENT GPS
  FACET GPS_This
  FACET GPS_Other
END COMPONENT

FACET GPS_This
  "GetStatus";
  "ThisFacetStatus";
END FACET

FACET GPS_Other
  "OtherFacetStatus";
END FACET
```

Example "legacy code"

```
unsigned int GlobalStatus();

class GPS
  { public:
    unsigned int GetStatus();
    unsigned int ThisFacetStatus();

    unsigned int OtherFacetStatus();

    unsigned int ComponentStatus();

    unsigned int status_;
  };

unsigned int GPS::GetStatus()
  { return status_;
  };

unsigned int GPS::ThisFacet
  { unsigned int tmp1 = Gl
    unsigned int tmp2 = Co
    unsigned int tmp3 = Th
    unsigned int tmp4 = Ot
    return tmp1 | tmp2 | t
  }

unsigned int GPS::OtherFac
  { unsigned int tmp1 = Gl
    unsigned int tmp2 = Co
    unsigned int tmp3 = ThisFacetStatus();
    unsigned int tmp4 = OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
```

```
unsigned int GPS::ComponentStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = ComponentStatus();
    unsigned int tmp3 = ThisFacetStatus();
    unsigned int tmp4 = OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }
```

Resulting Component.h, ...cpp

```
/*component.h*/
#include "BM__Component/BM__Component.h"
class GPS_Component : public BM__Component
  { public:
    GPS_Component(UUIdentifier componentId,
       class GPS_ThisFacet *GPS_ThisFacetPtr,
       class GPS_OtherFacet *GPS_OtherFacetPtr);
    ~GPS_Component();
    unsigned int ComponentStatus();
    class GPS_ThisFacet *GetGPS_ThisFacetPtr();
    class GPS_OtherFacet *GetGPS_OtherFacetPtr();
    const UUIdentifier &GetId() const;
    bool IsNil() const;
    private:
    UUIdentifier componentId_;
    class GPS_ThisFacet *GPS_ThisFacetPtr_;
    class GPS_OtherFacet *GPS_OtherFacetPtr_;
```

```
unsigned int GPS_Component::ComponentStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = ComponentStatus();
    unsigned int tmp3 = GPS_ThisFacetPtr_
        ->ThisFacetStatus();
    unsigned int tmp4 = GPS_OtherFacetPtr_
        ->OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }
```

```
class GPS_ThisFacet *GPS_Component::GetGPS_ThisFacetPtr()
  { return GPS_ThisFacetPtr_; }

class GPS_OtherFacet *GPS_Component::GetGPS_OtherFacetPtr()
  { return GPS_OtherFacetPtr_; }

const UUIdentifier &GPS_Component::GetId() const
  { return componentId_; }

bool GPS_Component::IsNil() const
  { return (componentId_ == UUIdentifier::GetNullId());
  }
```

BMT Toy Example: Facet1

FACET specification

```
COMPONENT GPS
  FACET GPS_This
  FACET GPS_Other
END COMPONENT

FACET GPS_This
  "GetStatus";
  "ThisFacetStatus";
END FACET

FACET GPS_Other
  "OtherFacetStatus";
END FACET
```

Not explicitly specified what to do

Example "legacy code"

```
unsigned int GlobalStatus();

class GPS
  { public:
    unsigned int GetStatus();
    unsigned int ThisFacetStatus();

    unsigned int OtherFacetStatus();

    unsigned int ComponentStatus();

    unsigned int status_;
  };

unsigned int GPS::GetStatus()
  { return status_;
  };

unsigned int GPS::ThisFacetStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = ComponentStatus();
    unsigned int tmp3 = ThisFacetStatus();
    unsigned int tmp4 = OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }

unsigned int GPS::OtherFacetStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = ComponentStatus();
    unsigned int tmp3 = ThisFacetStatus();
    unsigned int tmp4 = OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }

unsigned int GPS::ComponentStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = ComponentStatus();
    unsigned int tmp3 = ThisFacetStatus();
    unsigned int tmp4 = OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }
```

Resulting ThisFacet.h, …cpp

```
/*facet.h*/
#include "BM__Component/BM__Facet.h"
class GPS_ThisFacet : public BM__Facet
  { public:
    GPS_ThisFacet(const UUIdentifier &facetId,
        class GPS_Component *const component);
    ~GPS_ThisFacet();
    unsigned int GetStatus();
    unsigned int ThisFacetStatus();
    const UUIdentifier &GetId() const;
    bool IsNil() const;
    private:
    UUIdentifier facetId_;
    class GPS_Component *component_;
  };
```

```
/*facet.cpp*/
GPS_ThisFacet::GPS_ThisFacet(
   const UUIdentifier &facetId,
   class GPS_Component *const component) :
facetId_(facetId), component_(component)  { }

GPS_ThisFacet::~GPS_ThisFacet() {}

unsigned int GPS_ThisFacet::GetStatus()
  { return component_->status_; }

unsigned int GPS_ThisFacet::ThisFacetStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = component_
       ->ComponentStatus();
    unsigned int tmp3 = ThisFacetStatus();
    unsigned int tmp4 = component_
       ->GetGPS_OtherFacetPtr()->OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }

const UUIdentifier &GPS_ThisFacet::GetId() const
  { return facetId_; }

bool GPS_ThisFacet::IsNil() const
  { return (facetId_ == UIdentifier::GetNullId()); }
```

BMT Toy Example: Facet2

FACET specification

```
COMPONENT GPS
  FACET GPS_This
  FACET GPS_Other
END COMPONENT

FACET GPS_This
  "GetStatus";
  "ThisFacetStatus";
END FACET

FACET GPS_Other
  "OtherFacetStatus";
END FACET
```

Not explicitly specified what to do

Example "legacy code"

```
unsigned int GlobalStatus();

class GPS
  { public:
    unsigned int GetStatus();
    unsigned int ThisFacetStatus();

    unsigned int OtherFacetStatus();

    unsigned int ComponentStatus();

    unsigned int status_;
  };

unsigned int GPS::GetStatus()
  { return status_;
  };

unsigned int GPS::ThisFacetStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = ComponentStatus();
    unsigned int tmp3 = ThisFacetStatus();
    unsigned int tmp4 = OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }

unsigned int GPS::OtherFacetStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = ComponentStatus();
    unsigned int tmp3 = ThisFacetStatus();
    unsigned int tmp4 = OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }

unsigned int GPS::ComponentStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = ComponentStatus();
    unsigned int tmp3 = ThisFacetStatus();
    unsigned int tmp4 = OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }
```

Resulting OtherFacet.h, ...cpp

```
/*facet.h*/
#include "BM__Component/BM__Facet.h"
class GPS_OtherFacet : public BM__Facet
  { public:
    GPS_OtherFacet(const UUIdentifier &facetId,
             class GPS_Component *const component);
    ~GPS_OtherFacet();
    unsigned int OtherFacetStatus();
    const UUIdentifier &GetId() const;
    bool IsNil() const;
    private:
    UUIdentifier facetId_;
    class GPS_Component *component_;
  };
```

```
/*facet.cpp*/
GPS_OtherFacet::GPS_OtherFacet(
    const UUIdentifier &facetId,
    class GPS_Component *const component) :
    facetId_(facetId), component_(component) { }

GPS_OtherFacet::~GPS_OtherFacet()
  { }

unsigned int GPS_OtherFacet::OtherFacetStatus()
  { unsigned int tmp1 = GlobalStatus();
    unsigned int tmp2 = component_
       ->ComponentStatus();
    unsigned int tmp3 = component_
       ->GetGPS_ThisFacetPtr()->ThisFacetStatus();
    unsigned int tmp4 = OtherFacetStatus();
    return tmp1 | tmp2 | tmp3 | tmp4;
  }

const UUIdentifier &GPS_OtherFacet::GetId() const
  { return facetId_; }

bool GPS_OtherFacet::IsNil() const
  { return (facetId_ == UUIdentifier::GetNullId());
  }
```

BMT Status

- DMS Definitions
  - Domains: C++, FACET
    - C++ parser, name resolver, prettyprinter
      - Reads 150K SLOC of MSVC6 include files
    - FACET parser, attribute evaluator to extract facets
  - Rules/Patterns: ~400
    - organized in sets ("access path update", …)
  - Metaprogram
    - Exhaustive application of rule sets
    - 11K SLOC PARLANSE code
- Applied to multiple Boeing components
  - Does >95% of hand-work required
  - Could have done more… limited by project organization, not technology!
  - Hand touchup in an afternoon

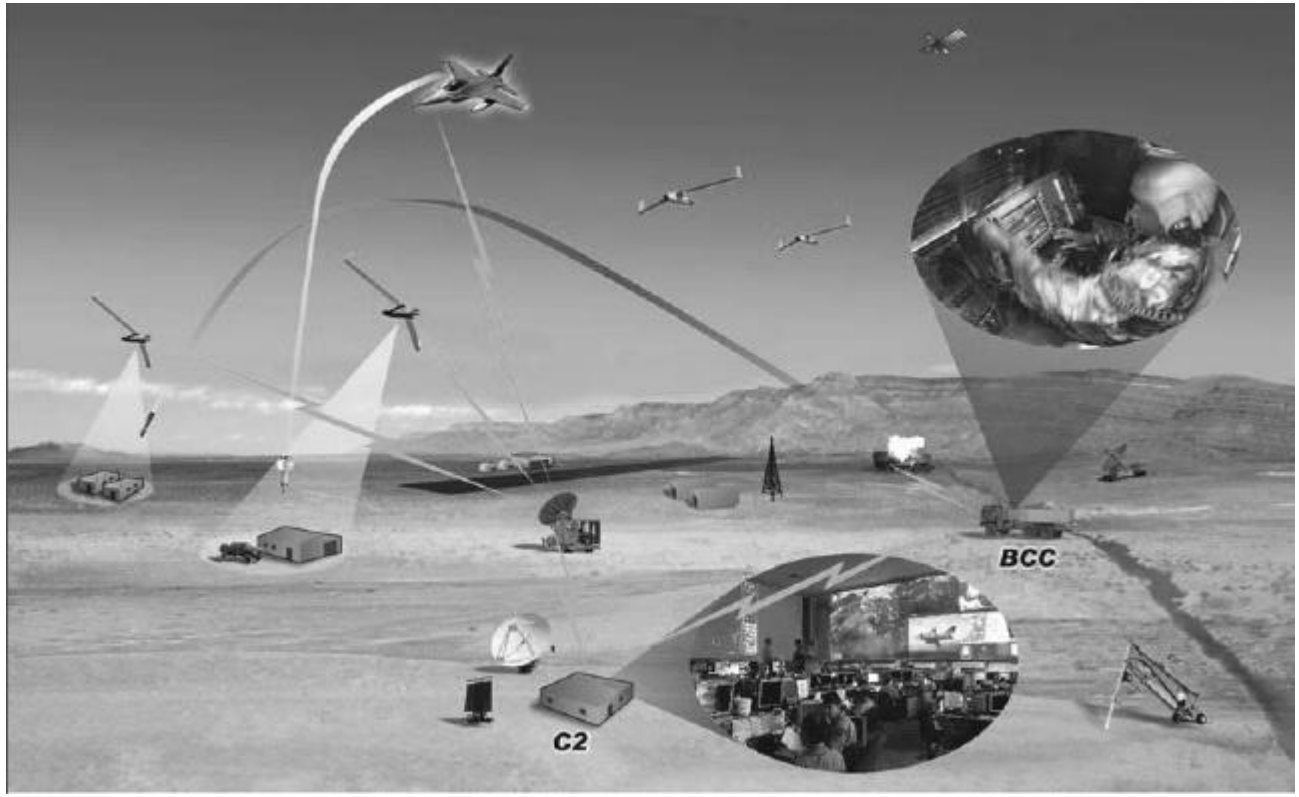
BMT Effort and Effect

- BMT contains
  - ~1.5 M lines of code
  - Most of this is DMS machinery
  - BUT only ~ 13K lines are BMT specific
  - → Huge leverage of DMS infrastructure for task

- Example component
  - Medium Sized
  - Estimated hand-conversion effort ~~ 1 man-month
  - Automated conversion time ~~ 8 minutes on 1 Ghz PC
    - Legacy classes retained in 10 source files
      - 2222 of 7347 lines of code changed
    - 34 new files
      - 2109 new lines of code and comments

White Sands Capstone Demo 4/14/2005

Boeing ScanEagle UAVs with PCES Software



- Multiple UAVs finding/targeting tactical targets
- *Live Fire*: F-16+JDAM and HIMARS+ATACMS

- Demonstrated PCES-funded results from a number of research teams, including:
  - End-to-end adaptive quality-of-service management for real time video data
  - Tools for composing and modeling component based software systems
  - *BMT: Several automatically converted BoldStroke mission components flew in UAVs*
- Contact Joe Cross (jcross@darpa.mil) for more details

Conclusion

- Useful to automate analysis/modification of programs
  - Many possible custom reengineering possibilities
  - A key technology for software quality improvement
- Need generalized compiler-like infrastructure
  - Definable parsers, prettyprinters, transforms
  - Must *scale* to application systems with MSLOCs
  - These are *hard* tools to build
- Such tools can carry out massive changes
  - They will become part of toolbox of corporate Software Engineering
  - If change is “easy”, is it architecture?
  - OMG standards for KDM, ASTM may enable tool integration
- Paper on BMT:
  `http://www.stsc.hill.af.mil/crosstalk/2005/05/0505akers.html`