# Designing Systems Using Rosetta

Dr. Perry Alexander
The University of Kansas
2291 Irving Hill Road
Lawrence, KS 66044
(785) 864-7741
alex@ittc.ukans.edu

David L. Barton
AverStar, Inc.
1593 Spring Hill Rd.
Vienna, VA 22182
(703) 852-4254
dlb@averstar.com

Danny C. Davis
AverStar, Inc.
1593 Spring Hill Rd.
Vienna, VA 22182
(703) 852-4031
ddavis@averstar.com

# What is Systems Engineering?

- Managing and integrating information from multiple domains when making design decisions
- Managing constraints and performance requirements
- Managing numerous large, complex systems models
- Working at high levels of abstraction with incomplete information
- …Over thousands of miles and many years

"…the complexity of systems… have increased so much that production of modern systems demands the application of a wide range of engineering and manufacturing disciplines. The many engineering and manufacturing specialties that must cooperate on a project no longer understand the other specialties. They often use different names, notations and views of information even when describing the same concept. Yet, the products of the many disciplines must work together to meet the needs of users and buyers of systems. They must perform as desired when all components are integrated and operated."

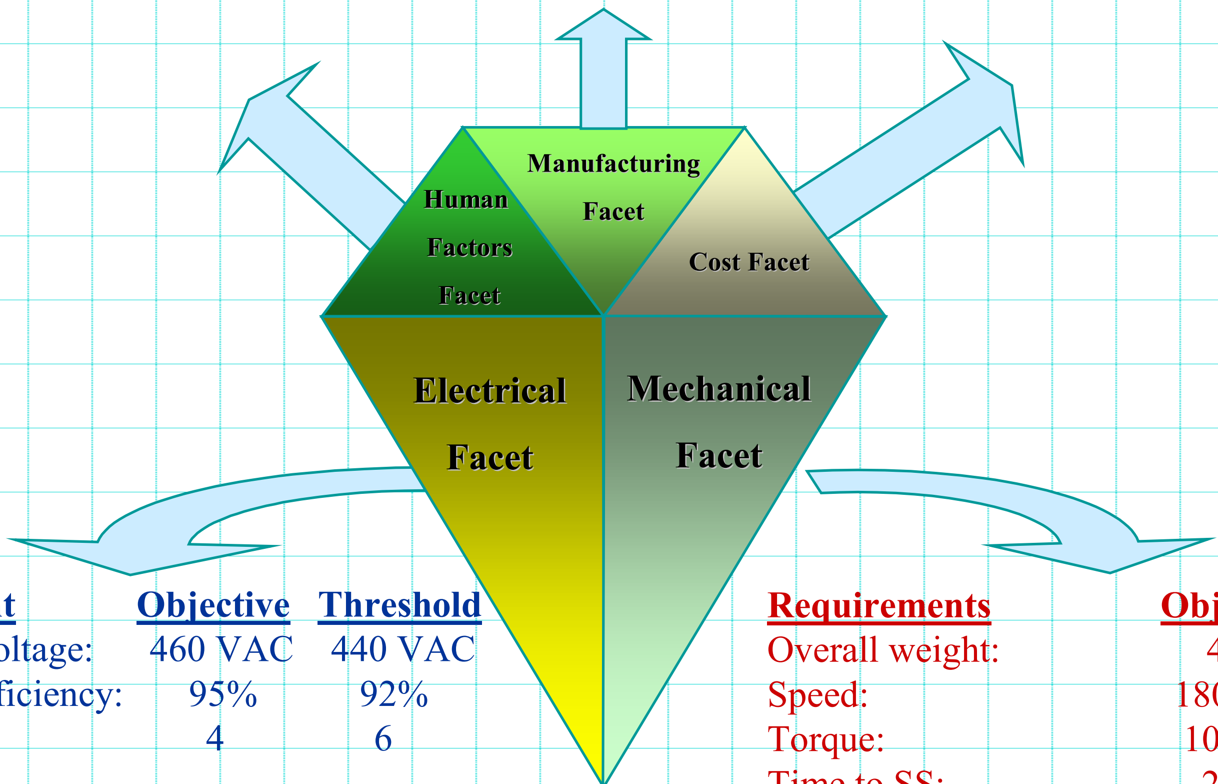
*D. Oliver, T. Kelliher, J. Keegan, Engineering Complex Systems, McGraw-Hill, 1997.*

# The Systems Level Design Problem

*The cost of systems level information is too high…*

- Design goals and system components interact in complex and currently unpredictable ways
- Interrelated system information may exist in different engineering domains
- Information may be spread across the system specification, in separate parts of the system (intellectually distant)
- Representation and analysis of high level systems models is difficult and not well supported
- Representation and analysis of interactions between system elements is not supported at all

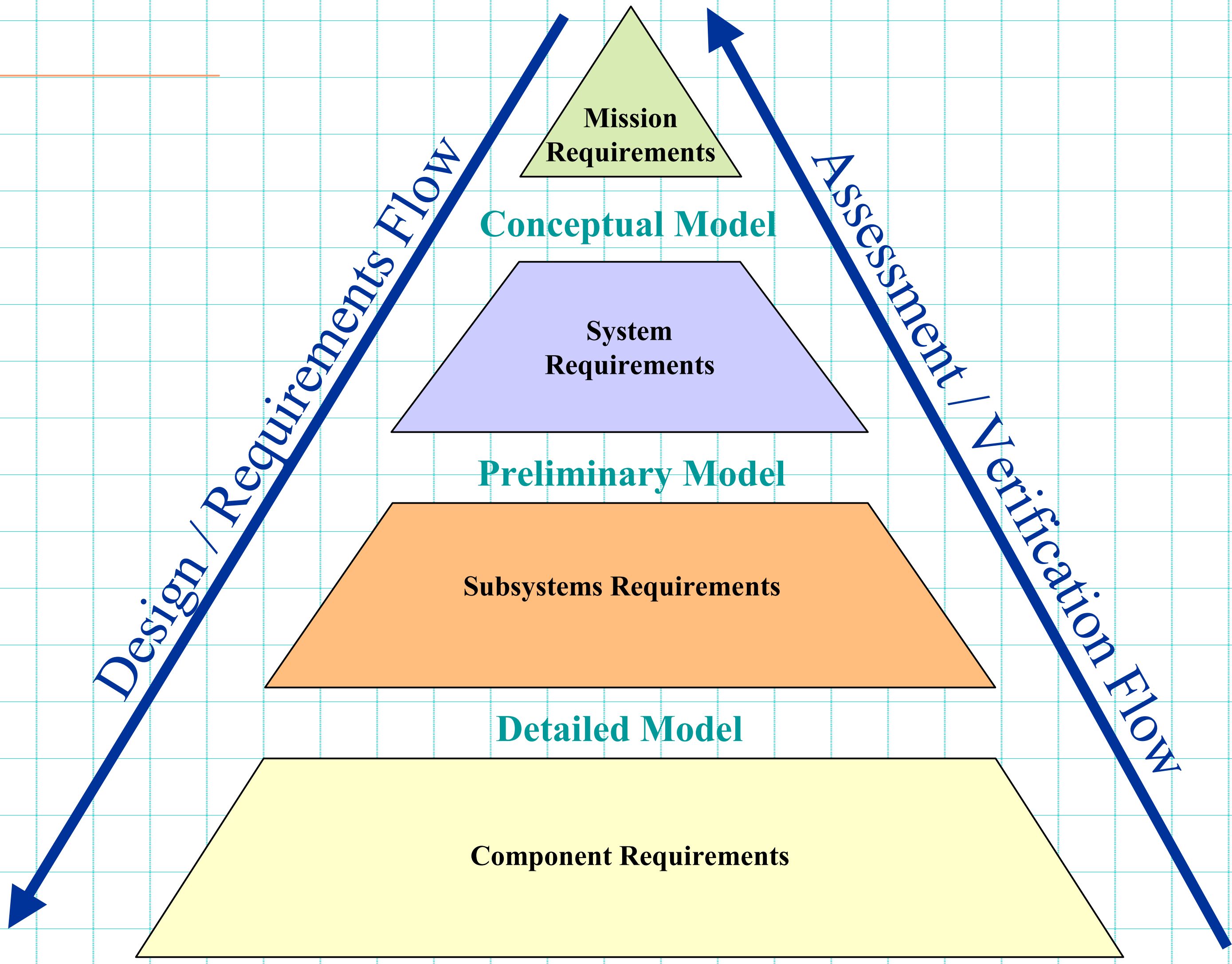
# Multi-Domain Design



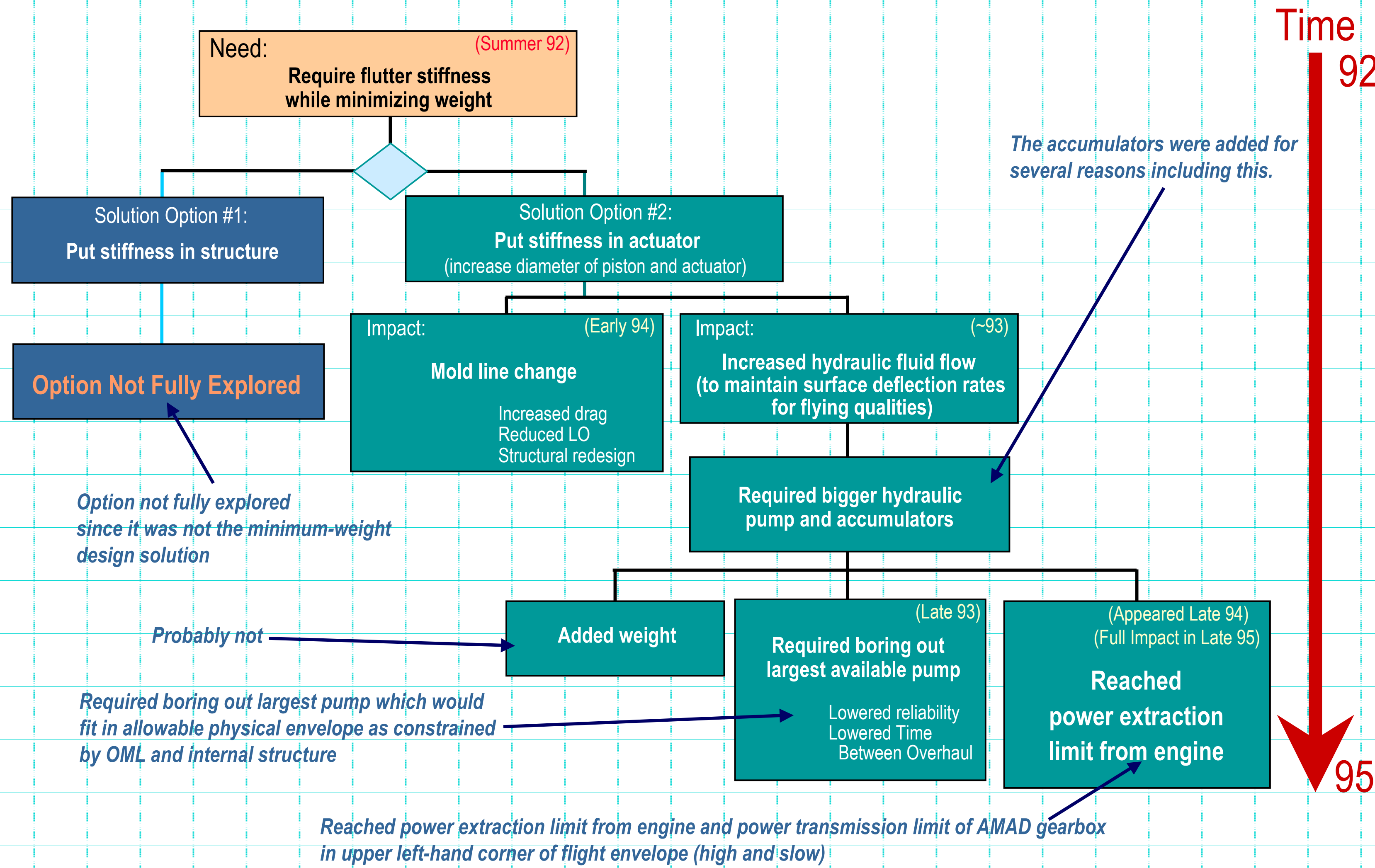

| Requirement | Objective | Threshold |
| --- | --- | --- |
| Operating Voltage: | 460 VAC | 440 VAC |
| Electrical Efficiency: | 95% | 92% |
| Rotor Poles: | 4 | 6 |

| Requirements | Objective | Threshold |
| --- | --- | --- |
| Overall weight: | 40 lbs | 50 lbs |
| Speed: | 1800 rpm | 1750 rpm |
| Torque: | 100 ft-lb | 95 ft-lbs |
| Time to SS: | 25 sec | 30 sec |
| Critical Frequency: | 3000 rpm | 24000 rpm |
| Mechanical Efficiency: | 92% | 88% |

# Abstraction and Refinement



# Prior Airplane Design Experience with Altering an Existing Design



# What is Missing?

*Systematic support for whole systems design throughout the system lifecycle…*

- Languages and notations for integrated modeling of system domains
- Tools for integrated analysis of interactions between system domains
- Tools supporting predictive analysis early in the design process
- Tools supporting systems level analysis throughout the system lifecycle

*…A systems level design language and support environment to reduce the cost of information*

# Supporting Systems Level Design

*The cost of systems level information can be reduced by providing a language and toolset for systems level design*

- A standard methodology, language and tool infrastructure for
  1. *Describing, analyzing and integrating systems models in different engineering domains*
  2. *Describing and analyzing interactions between systems models in different engineering domains*
  3. *Performing predictive analysis at and between varying abstraction levels*

*Rosetta* is a systems level design language being developed explicitly to address these needs

# What is Rosetta?

Rosetta *is an emerging high-level description language for specifying systems that explicitly provides*:

- Support for domain specific modeling
- Support for modeling cross domain interactions
- Support for defining and combining models of systems and system components
- Support for modeling and analysis at high levels of abstraction
- Support for specifying constraints and performance requirements for the system and system components in a top-down manner
- Support for requirement verification as the system description evolves

# Rosetta Modeling Approach

***Decompose a system description into multiple, interacting models***

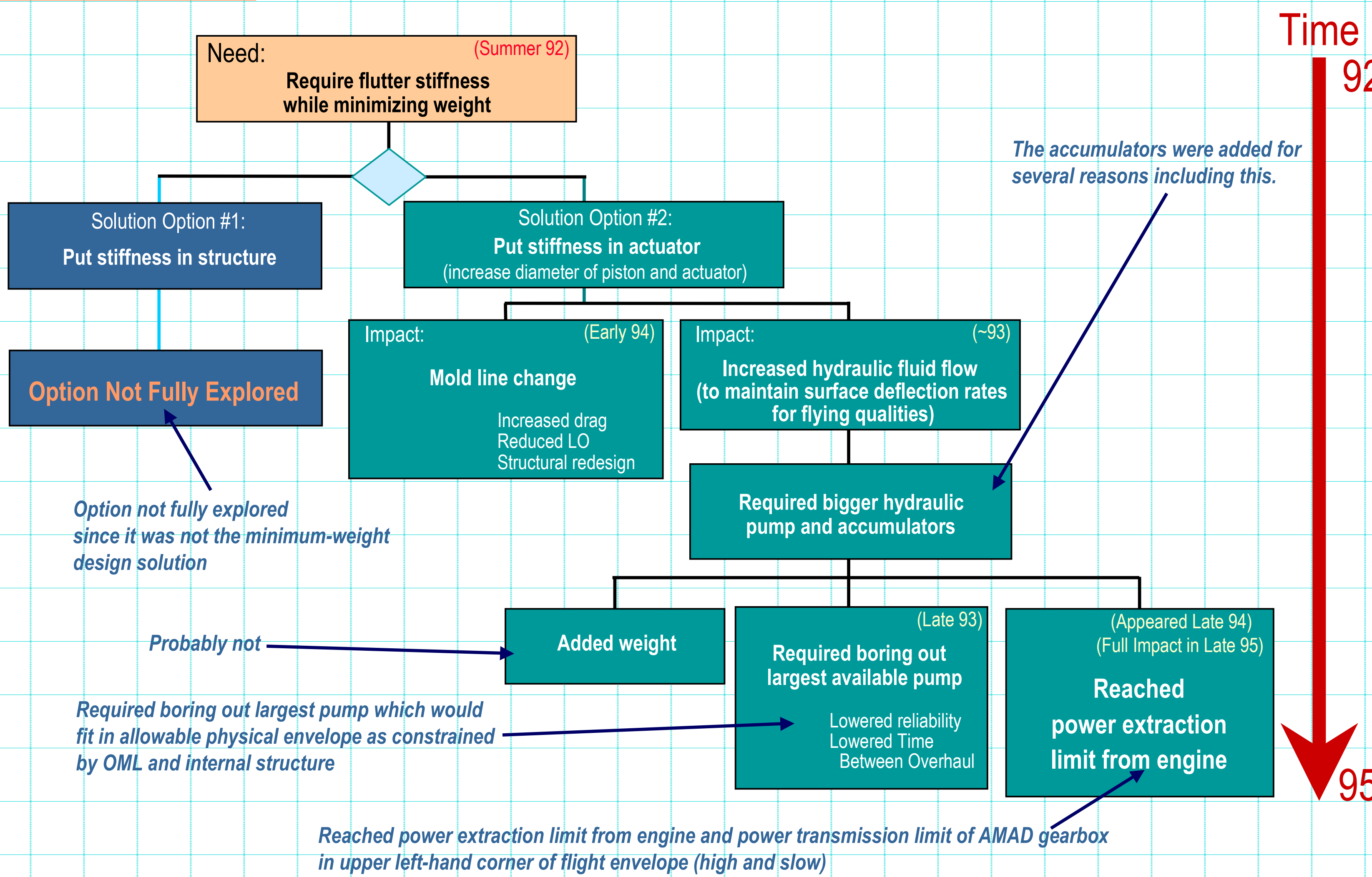
- Models describing system structure
  - Describe the interconnection of constituent components
  - Describe the characteristics of required components
- Models describing system characteristics
  - Describe each perspective, or *facet*, of the system
  - Use a domain model appropriate for each system facet
  - Use an interaction model to describe how domain models interact
  - Combine characteristic models and interaction to provide a complete composite system model

# Rosetta Analysis Approach

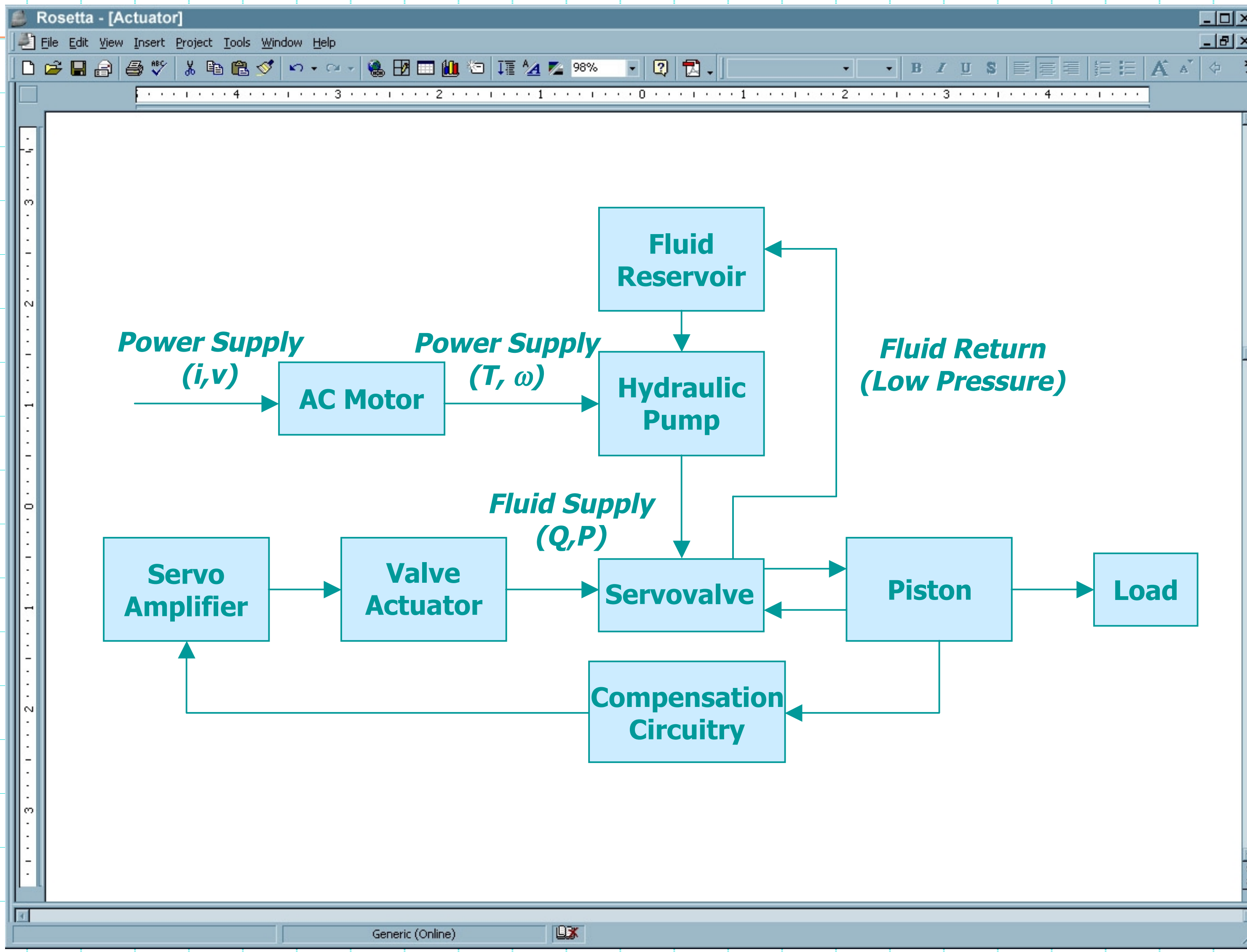
*Analyze models throughout system development to detect errors earlier in the lifecycle*

- Analyze models at each stage of the design lifecycle
  - Determine the consistency of individual models
  - Determine the consistency of interactions between models
  - Detect model inconsistency and interaction errors early
- Analyze design iterations with respect to systems level goals to assure correctness
  - Determine if the design iteration is correct
  - Detect design errors early

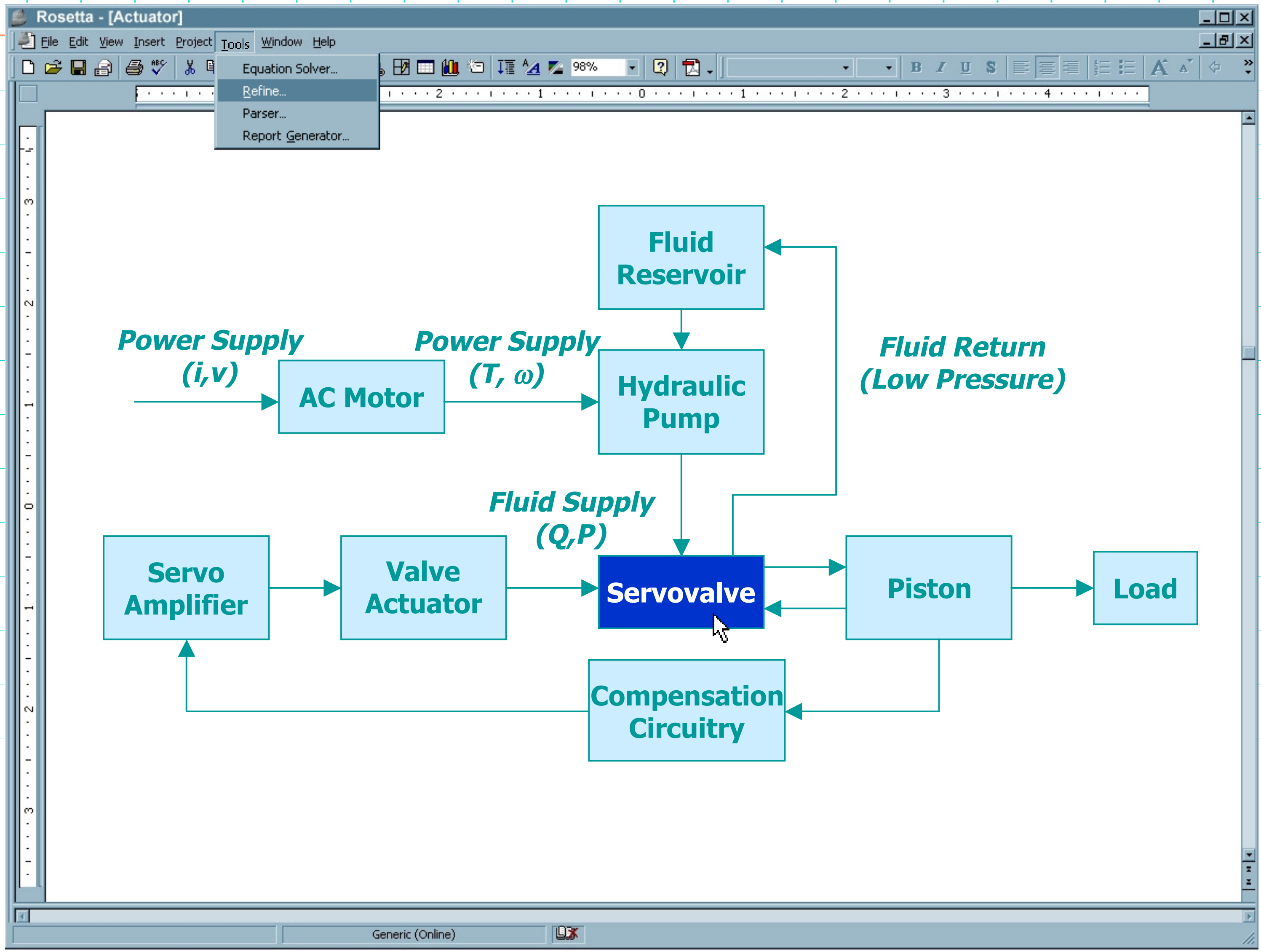
# Prior Airplane Design Experience with Altering an Existing Design



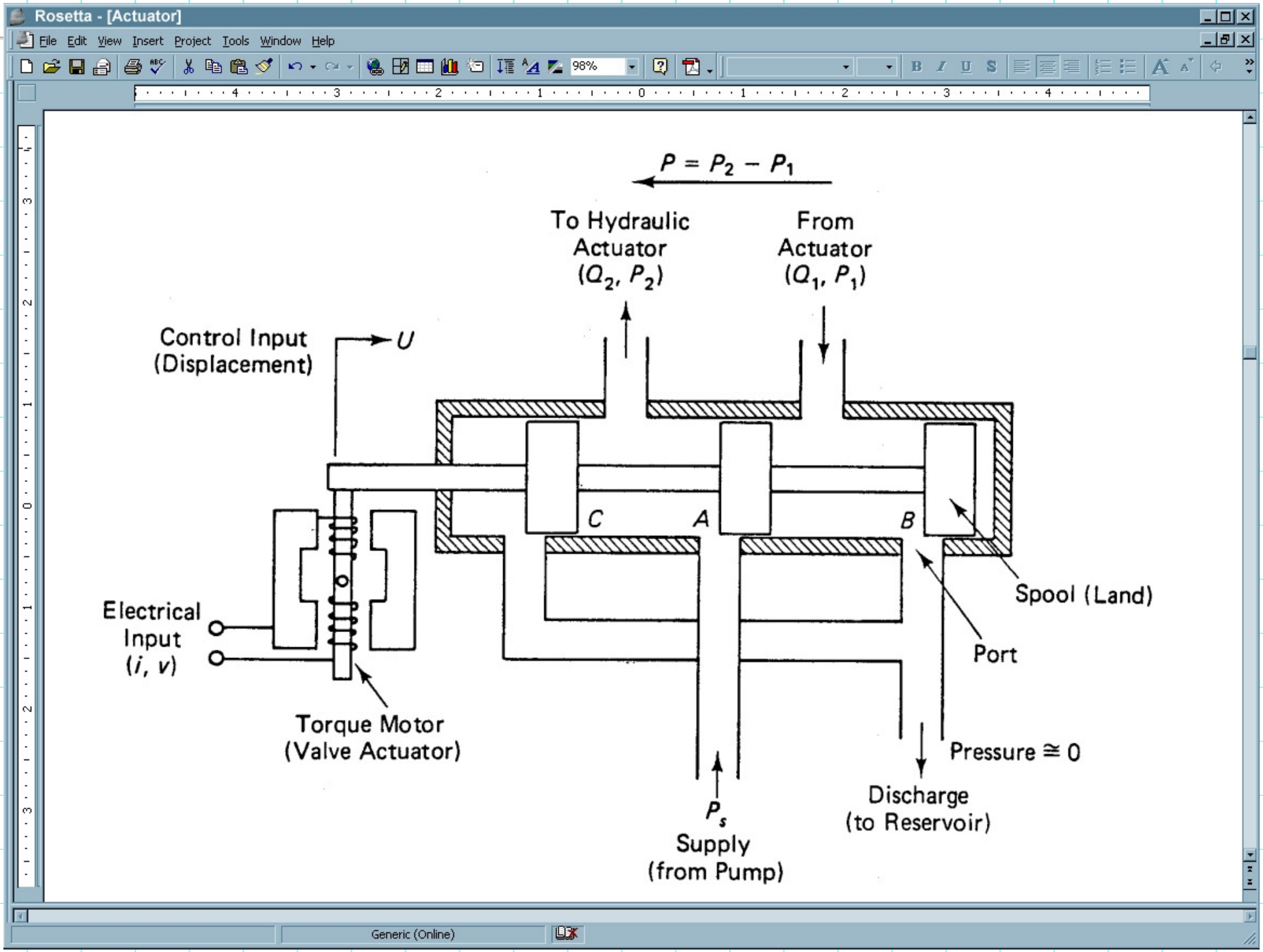
# Using Systems Design Tools –Actuator



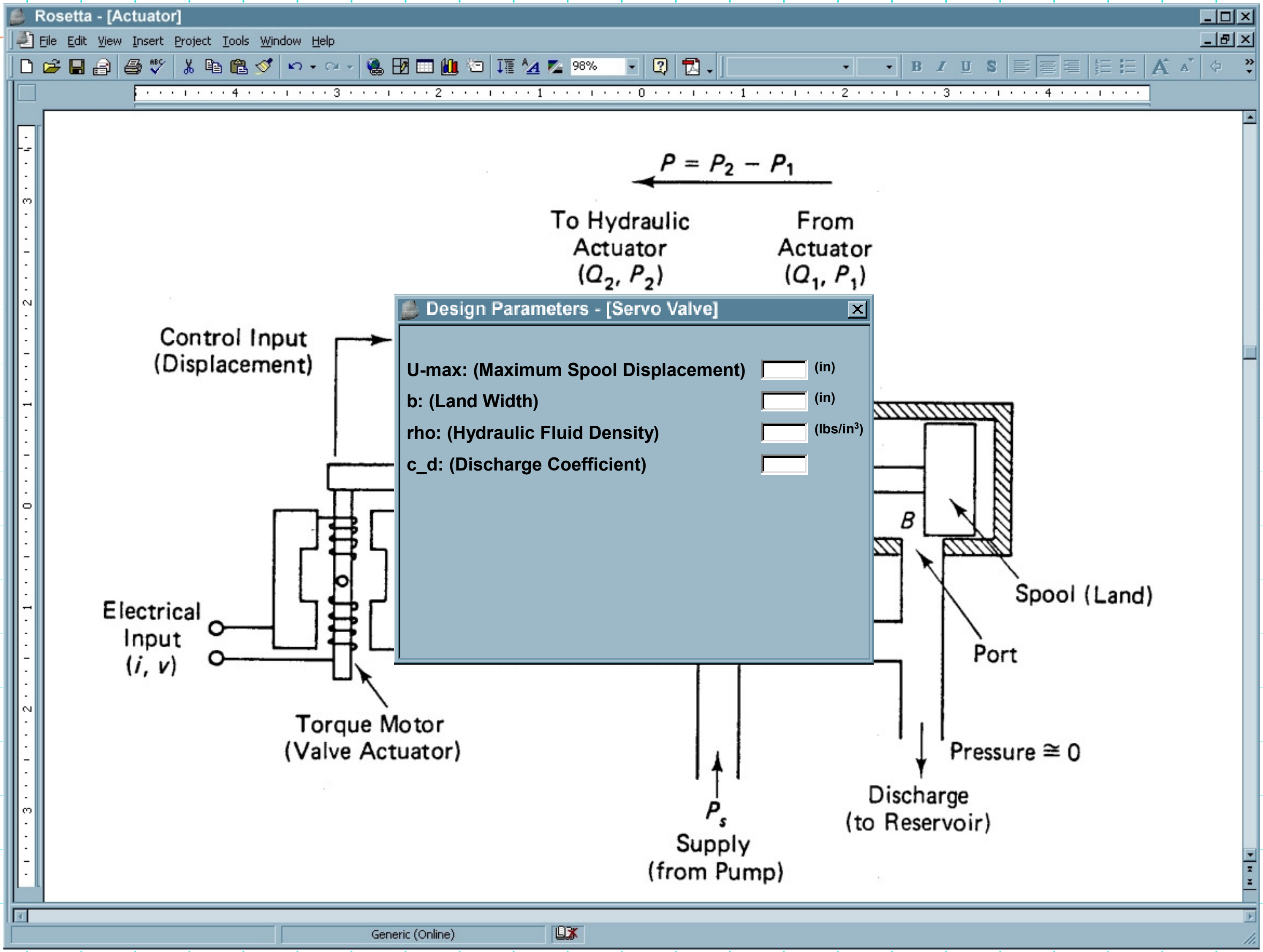
# Using Systems Design Tools – Actuator



# Refinement – The Servovalve



# Servo Valve Design Parameters



# Activity Refinement

- The user reaches the servovalve by selecting it on the upper level diagram and then selecting "refine" from the tool menu.
- Because an actuator has been defined as a standard part, the component diagram is displayed.
- The user can then fill in the appropriate parameters for the servovalve.
- For each diagram, with accompanying parameters, a Rosetta *facet* is created.
- The facet contains the parameter values and the mathematical relationships between them.

# Rosetta Concepts

- A *facet* is a model representing one perspective of a system
  - One particular point of view
  - One particular abstraction level
- A *domain* is a semantic system for defining facets
- A *component* is described by:
  - Defining and composing its various facets
  - Defining and composing its various sub-components

# Servovalve Facet Interface

- The facet interface defines design parameters (**blue**) and operational interface (**red**)

```
facet servovalve_fcn(
     U::real;   //Spool displacement (in).  U positive indicates
                //Q_2 is flow out of valve to hydraulic cylinder,
                //and Q_1 is flow from cylinder to valve.
                //U negative indicates the reverse is true.

     P_S::posReal;   //source pressure (lbs/in^2)
     Q_1, Q_2 :: real;    //flow rate to/from servo (in^3/sec)

     U_max::posReal; //maximum spool displacement (in)
     b::posReal;     // land width (in)
     rho::posReal;   // hydraulic fluid density (lbs/in^3)
     c_d::posReal;   // discharge coefficient at each port
) is
```

# Servovalve Facet Local Definitions

- Local definitions provide internal items such as local variables (red) and function definitions (blue)

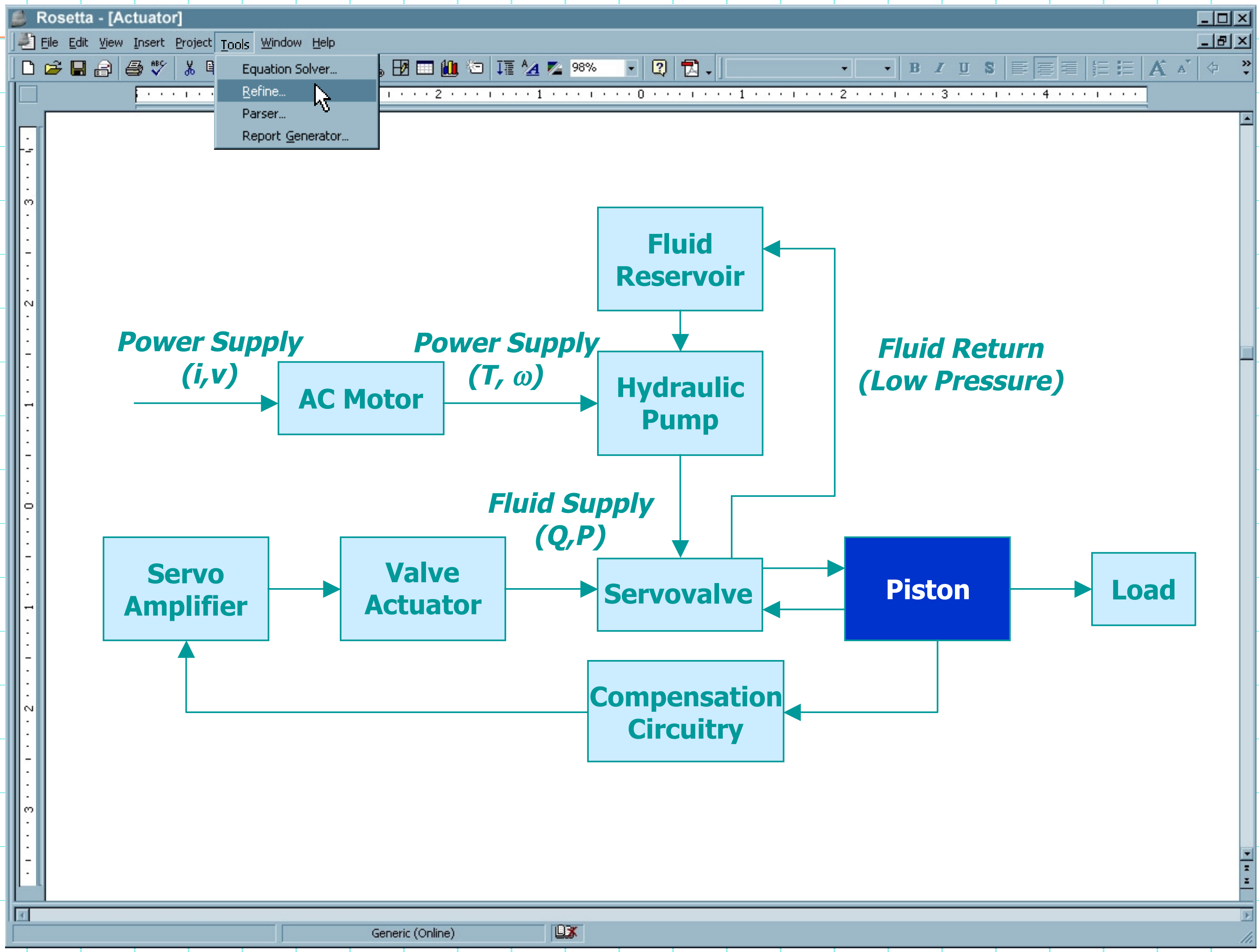
```
P :: real is P_2-P_1;       //differential pressure (lbs/in^2)
Q :: real is (Q_1+Q_2)/2;  //average flow rate (in^3/sec)
Q_max :: real is U_max * b * c_d * (P_s/rho)^0.5;
                            //maximum flow into
                            // servo valve (in^3/sec)

sgn(x::real)::real is
  if x /= 0 then x/abs(x)
  else 0 endif;
```

# Servovalve Facet Terms

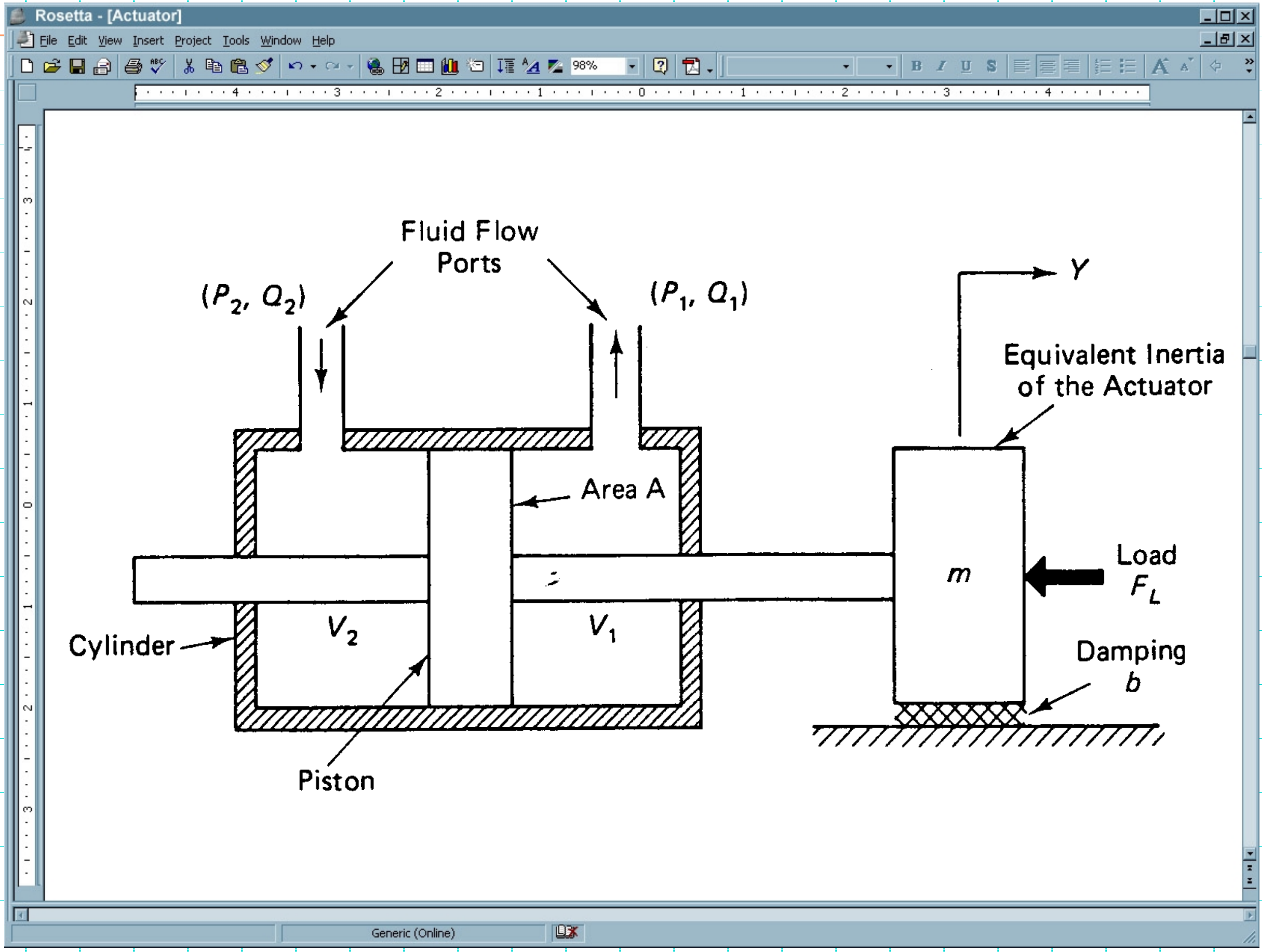
- Terms define the specification domain (red), functional properties (blue) and constraints (black)

```
begin continuous

  //Nonlinear steady state valve equation.
  F1: Q/Q_max=U/U_max * ((1 - P/P_S) * sgn(U/U_max))^0.5;

  //Flow, spool disp, diff pressure
  //cannot exceed max
  C1: abs(Q) =< abs(Q_max);
  C2: abs(U) =< abs(U_max);
  C3: abs(P) =< abs(P_S);
end servovalve_fcn;
```

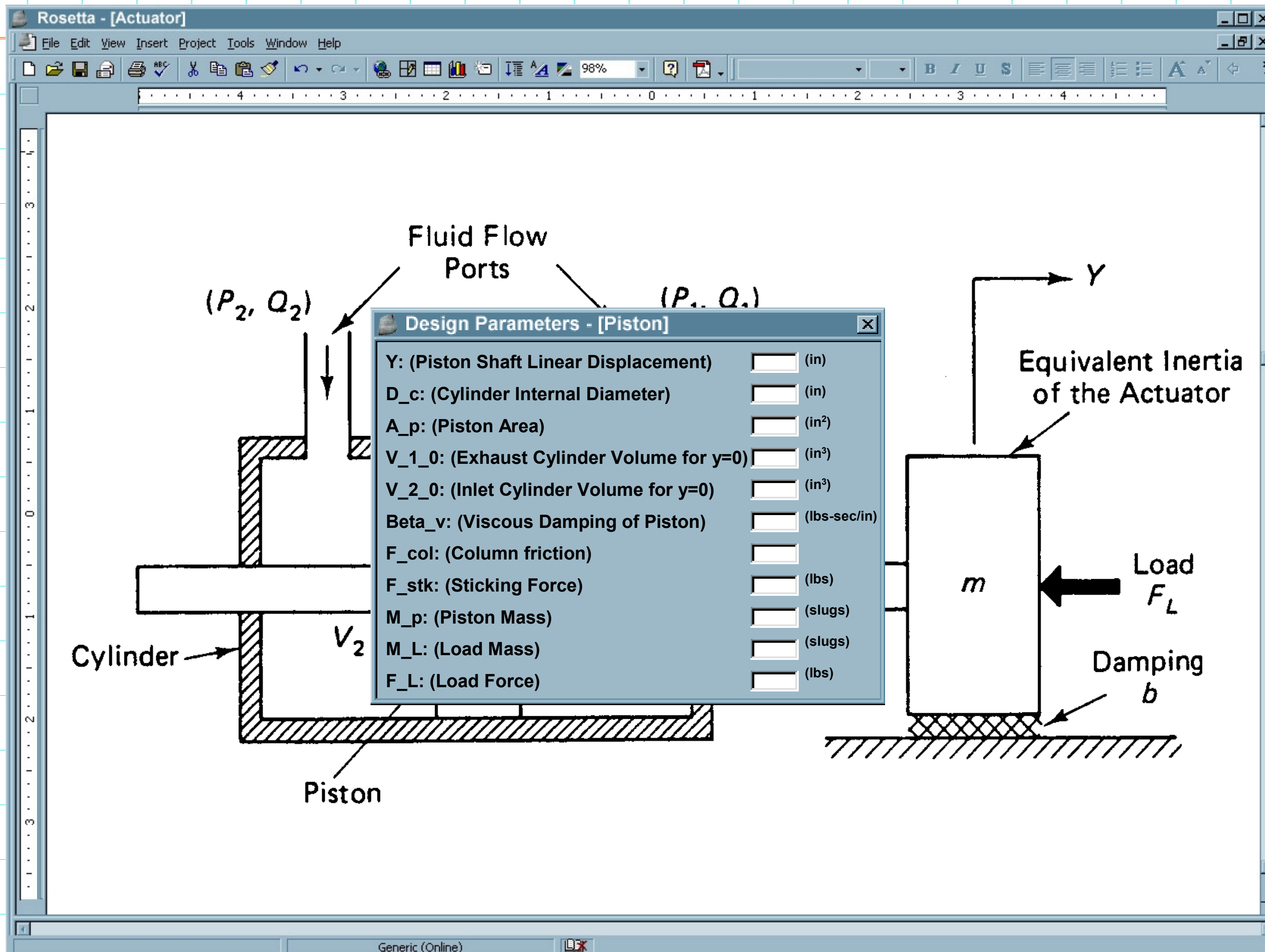
# Using Systems Design Tools –Actuator



# The Piston Model



# Piston Design Parameters



# Piston Functional Model Interface

```
facet cyl_piston_fcn(
    Q_1,Q_2 :: real; //exhaust & inlet volume flow rate (in^3/sec)
    P_1,P_2 :: real; //exhaust & inlet pressure (lbs/in^2)
    Y :: real; //piston shaft linear displacement (in)
    D_c :: posReal; // cylinder internal diameter (in)
    A_p :: posReal; // piston area (in^2)
    V_1_0, V_2_0 :: posReal; // exhaust & inlet side cylinder volumes for
                             // Y = 0 (in^3)
    Beta_v :: real; //viscous damping of piston (lbs-sec/in)
    F_col :: real;  // coloumb friction
    F_stk :: real; // "sticktion" force (lbs)
    M_p,M_L :: posReal; // Masses of piston and load (slugs)
    F_L :: real; // Load force (lbs)
    F_0 :: real; // Output force (lbs)
  ) is
```

# Piston Functional Model Declarations

```
use realTypes;

//internal vars

  A_c :: posReal; //cylinder internal cross-sectional area (in^2)
  V_1 :: nonNegReal; //exhaust-side cylinder volume (in^3)
  V_2 :: nonNegReal; //Inlet-side cylinder volume (in^3)
  V_p :: real is deriv(Y,t); //Piston velocity (in/sec)
  Q :: real is (Q_1+Q_2)/2; //average flow rate (in^3/sec)
  P :: real is P_2 - P_1; //differential pressure
  eff_piston :: real is 0.99; //piston efficiency, assumed to be
                                  // 99%
```

# Piston Functional Model Terms

```
begin continuous
   F1: A_c = pi*(D_c/2)^2;
//Calculation of continuous exhaust and inlet side
//volumes from initial value and displacement (Y)
   F2: V_2 = V_2_0 + A_p * Y;
   F3: V_1 = V_1_0 - A_p * Y;
//Total volume of the piston is constant over time
   C2: V_1 + V_2 = V_2_0 + V_1_0;
//Calculate the output force is ideal force minus
//stiction and friction
   F4: F_0 = A_p*P - F_stk - F_col - Beta_v*V_p;
//Newton says MA=F
   F5: (M_p + M_L)*deriv(V_p,t) = F_0 - F_L;
   F6: P = (Beta_v*V_p +  F_col + F_stk + F_L)/A_p;
   F7: Q_2 = A_p*deriv(Y,t);
   F8: Q_1 = A_p*deriv(Y,t);
end cyl_piston_fcn;
```

# Piston Power Model

```
facet cyl_piston_power(
    Q_1,Q_2 :: real; //exhaust & inlet volume flow rate (in^3/sec)
    P_1,P_2 :: real; //exhaust & inlet pressure (lbs/in^2)
    power :: posReal; //Cylinder power (ft-lbs/sec)
    PowOut_piston :: posReal; //piston output power (ft-lbs/sec)
    V_p :: real; //piston velocity (in/sec)
    A_p :: posReal;        //piston effective area (in^2)
    F_stk :: real;         //stiction force (lbs)
    F_col :: real;         //coloumb friction (lbs)
    F_L :: real; //load force (lbs)
  ) is
use realTypes;

begin continuous
  F1: power = Q_2*P_2 - Q_1*P_1;
  F2: PowOut_piston = V_p*(A_p*P - F_stk - F_col - F_L)*eff_piston;
end cyl_piston_power;
```

# Model Interaction

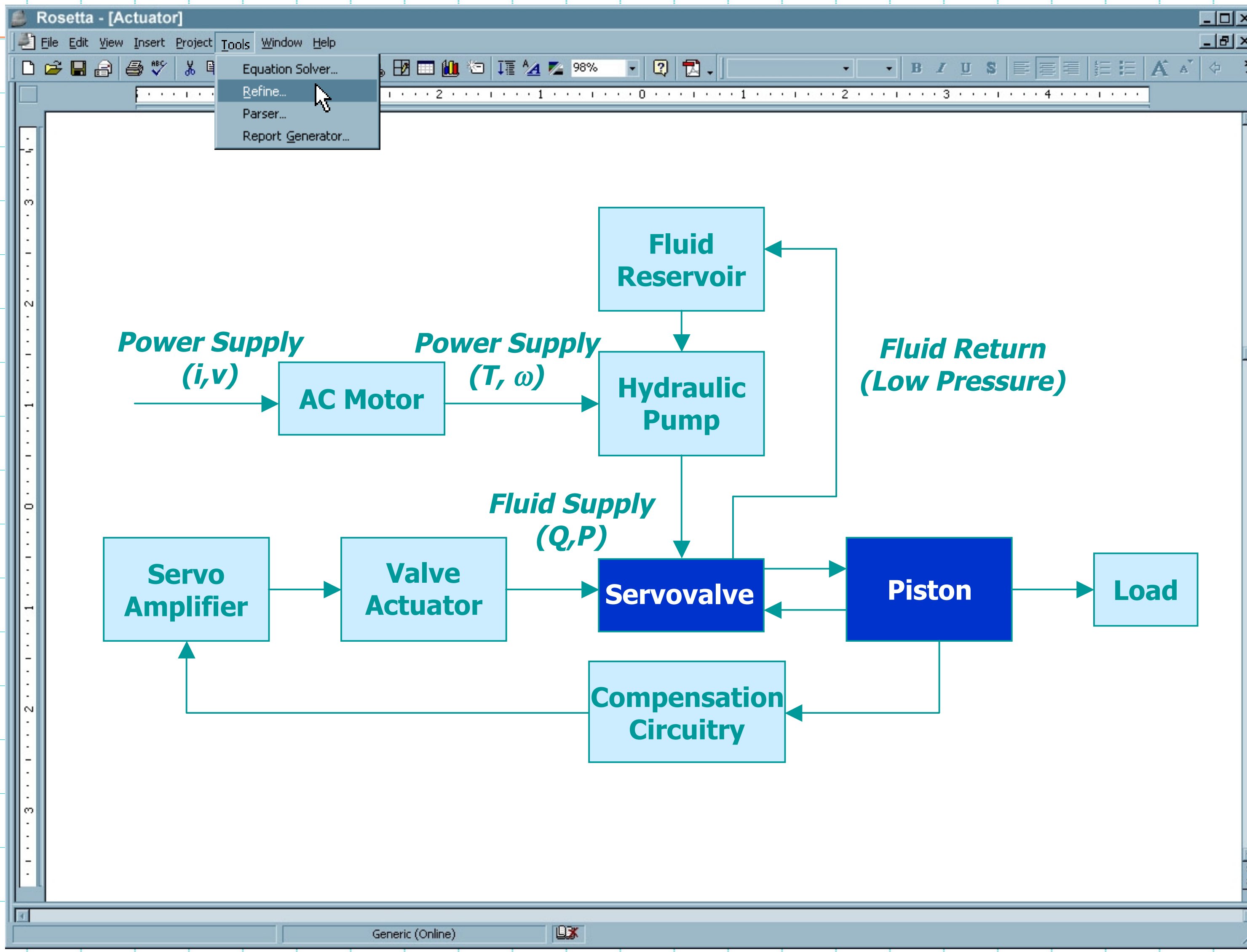
- Composing facets causes their associated system models to interact
- Model composition within the same component defines multiple, interacting component views
- Model composition between components provides structural representation

# Complete Piston Model

- Facet composition causes model interaction
  - The new model is has the properties of both original models
  - Identically named parameters represent different views of the same quantity
- The complete piston model consists of a functional model and a power model

```
facet cyl_piston is cyl_piston_fcn and
  cyl_piston_power;
```

# Using Systems Design Tools –Actuator



# The Actuator Interface

- Design parameters are propagated outward (red)
- Operational parameters are instantiated to connect components or propagated to the interface (green)

```
facet actuator (
    U::real;              //spool displacement (in).
    Q_S::posReal; //source volume flow rate (in^3/sec)
    P_S::posReal; //source pressure (lbs/in^2)

    b::posReal; // land width (in)
    rho::posReal; // hydraulic fluid density (lbs/in^3)
    c_d::posReal; // discharge coefficient at each port

    Y :: real; //piston shaft linear displacement (in)
    D_c :: posReal; // cylinder internal diameter (in)
    A_p :: posReal; // piston area (in^2)
```

# The Actuator Interface (cont)

```
    V_1_0, V_2_0 :: posReal // exhaust & inlet side cylinder
                            // volumes for Y = 0
    Beta_v :: real; //viscous damping of piston (lbs-sec/in)
    Beta :: posReal; //bulk modulus of hydraulic fluid (lbs/in^2)
    F_stk :: real; // "sticktion" force (lbs)
    M_p,M_L :: posReal; // Masses of piston and load
    F_L :: real; // Load force (lbs)
    F_0 :: real; // Output force (lbs)
) is
```

# The Actuator Definition

- Facets are instantiated to define local components (blue)
- Shared internal variables define connections (red)
- Instantiating component parameters with facet parameters propagates parameters to the facet interface (black)
- The actuator power consumption is the sum of the component power consumption values (green)

```
  Q_1,Q_2 :: posReal; //exhaust & inlet volume flow rate (in^3/sec)
  P_1,P_2 :: posReal; //exhaust & inlet pressure (lbs/in^2)
  actPower :: posReal; // actuator power consumption

begin logic
  valve: servovalve(Q_1,Q_2,P_1,P_2,U,Q_S,P_S,b,rho,c_d)
  piston: cyl_piston(Q_1,Q_2,P_1,P_2,Y,D_c,A_p,V_1_0,V_2_0,Beta_v,
                     Beta,F_stk,M_p,M_L,F_L,F_0);
  p: actPower = valve.power + piston.power;
end actuator;
```

# Mechanical Experiment Case Studies

- Determine Rosetta feasibility in mechanical domains
- Experiment 1 – Dual Spring
  - Goal: End-to-end analysis of spring system using existing Rosetta tools
- Experiment 2 – Actuator Redesign
  - Goal: Simulated re-creation of real-life actuator redesign problem

# Dual Spring System

- Goal: End-to-end analysis of spring system using existing Rosetta tools
- Achievements:
  - Developed Rosetta design model of simple spring
  - Developed Rosetta structural model of dual spring system
  - Automatically transformed Rosetta models into MATLAB system representations
  - Used MATLAB model to support parametric design for specific operational parameters
- Status:
  - Parsing and automatic translation achieved
  - Functional MATLAB models produced for demonstration

# Actuator Redesign

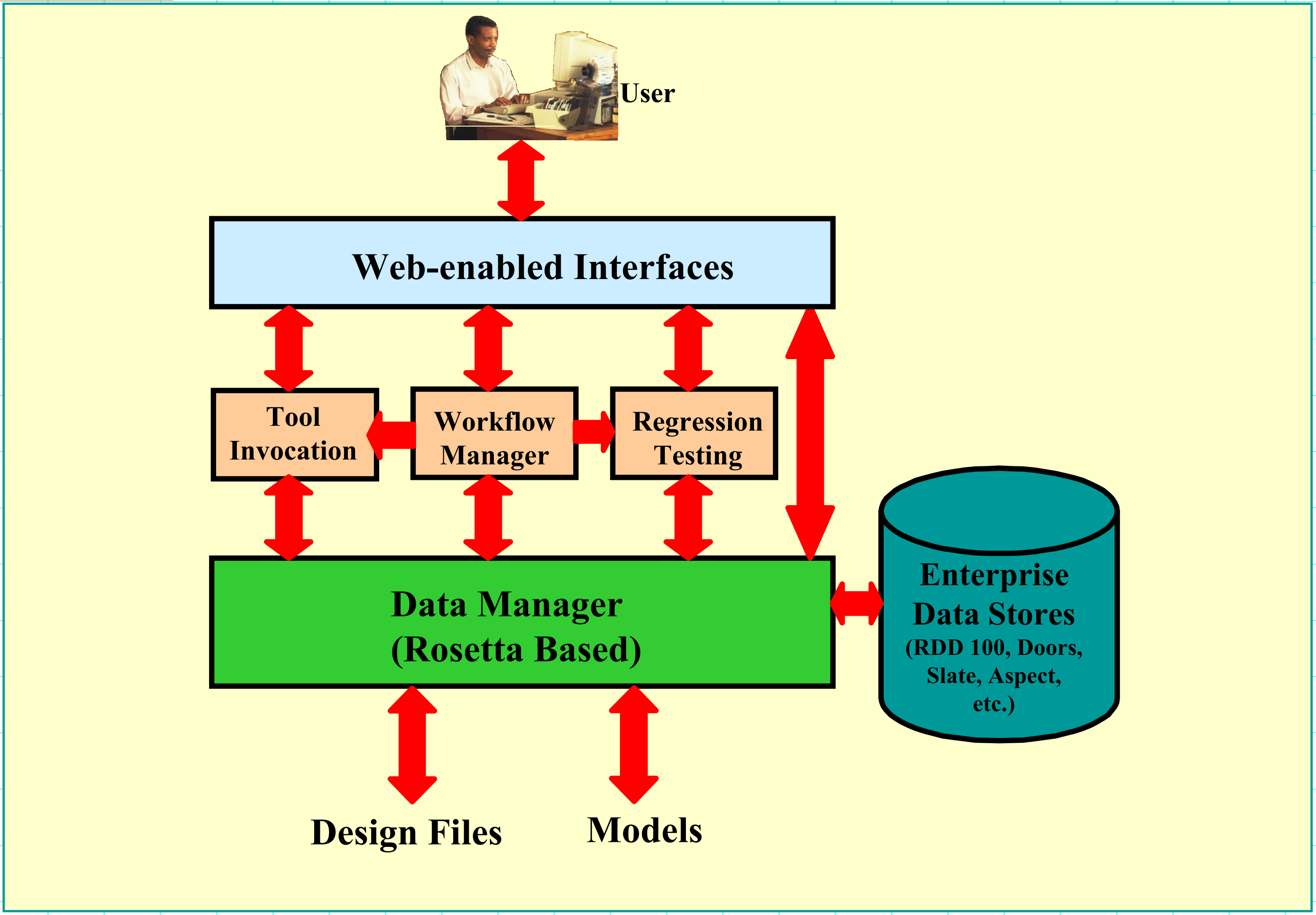
- Goal: Simulated re-creation of real-life actuator redesign problem
- Achievements:
  - Developed Rosetta design model of servovalve, cylinder and actuator
  - Developed power constraint and functional models
  - Hand translated Rosetta models into MATLAB system representations
  - Used interactions to represent constraint and functional model interaction
  - Used MATLAB model to demonstrate early detection of constraint violation
- Status:
  - Actuator problem analyzed and Rosetta models written
  - Generated interaction result between power and functional models
  - Analytically predicted power constraint violation based on MATLAB models

# Reducing The Cost of Information

**By reducing the cost of information, the Rosetta methodology can deliver on the elusive promises of:**

- ***Faster***
  - *Errors are avoided through early predictive analysis*
  - *Errors are discovered earlier in the system lifecycle*
- ***Better***
  - *High level analysis supports better systems level design*
  - *Analysis utilizes interaction information otherwise unavailable*
- ***Cheaper***
  - *Understanding system interaction supports faster systems integration*
  - *Discovering errors early reduces the cost of mitigation*
  - *Understanding the impacts of design changes across systems decreases the cost of component upgrade and replacement*

# POET Design Integration Architecture



# Status (1)

- Rosetta is being developed under the auspices of the Systems Level Design Language (SLDL) committee of VHDL International
- Initial focus was for supporting the development of Systems on Chip (SoC) with funding from AFRL/IF
- SLDL requirements document and representative examples are available on the SLDL web site

  http://www.intermetrics.com/sldl
- Initial definition of the language syntax, semantics and base domains is nearing completion
  - Version 0.4 of the Rosetta language definition and tools are available on the Rosetta web site:

    http://www.ittc.ukans.edu/Projects/SLDG/Rosetta

# Status (2)

- Prototype tool development is commencing
  - Version 0.4 of the Rosetta parser is available for download

  http://www.ittc.ukans.edu/Projects/SLDG/Rosetta
  - Extractor to Matlab for analysis of mathematical models completed
  - Other analysis and proof tools under development
- Second phase of AFRL/IF effort will include a series of demonstrations of capability and benefit
  - An industrial prototype of SoC produced using Rosetta is planned in early 2001
- Analysis of mechanical domain applicability, funded under a separate AFRL/ML contract, successfully completed