Steven P. Reiss
Brown University

FIELD

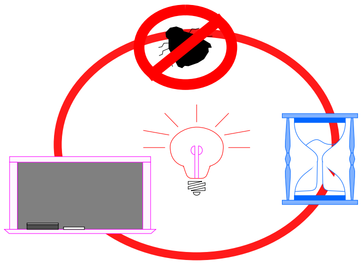
❖ A ❖

- ✔ **F*riendly,***
- ✔ **I*ntegrated***
- ✔ **E*nvironment for***
- ✔ **L*earning and***
- ✔ **D*evelopment***

Overview of Talk

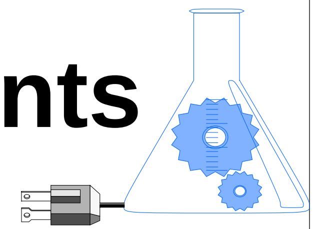
- ❖ **Previous work**
- ❖ **Motivation of FIELD**
- ❖ **The integration mechanism**
- ❖ **Data structure display**
- ❖ **Current research directions**

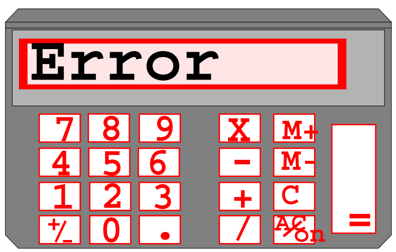
Programming Environments



1. Loose collections of tools

2. Closed, powerful environments





3. Student environments

We want the best of all these worlds—

Without the drawbacks.

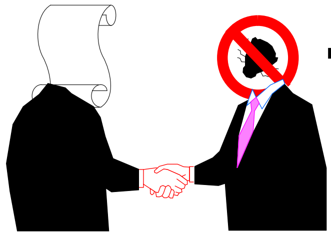
Motivations for FIELD

- **Graphical UNIX programming**
- **Non-toy Environment for teaching**
- **Showcase for research at Brown**
- **Basis for future research projects**
- **Front-end for algorithm animations**
- **Workstation-independent interface**
- **Distributed and threaded programs**
- **To do as little work as possible**

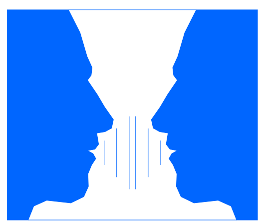
Overview of FIELD

- ☞ **Open collection of tools**
- ☞ **Selective Broadcasting**
- ☞ **Source annotations**
- ☞ **Emphasis on visualization tools**
- ☞ **Standard workstation interface**

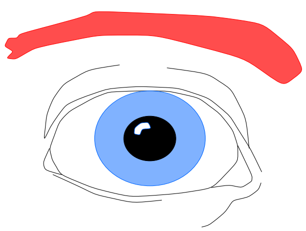
Integration Requirements



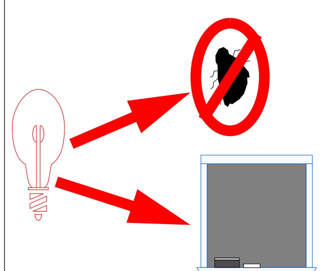
Tools must be able to interact with each other directly.



Dynamic information must be shared.



All source action should be through a single view.



Specialized information must be available to all tools.

Our Requirements

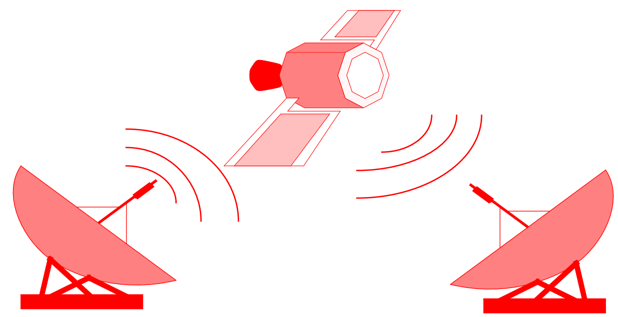
- **Extensible**
- **Simple enough for novices**
- **Rich enough for research**
- **Multiple Processes & machines**
- **Excellent student environment**
- **Inexpensive to build and maintain**

Integration Alternatives

- **Graft Tool Functionality**
- **Program Database**
- **Single System**
- **Selective Broadcasting**

Selective Broadcasting

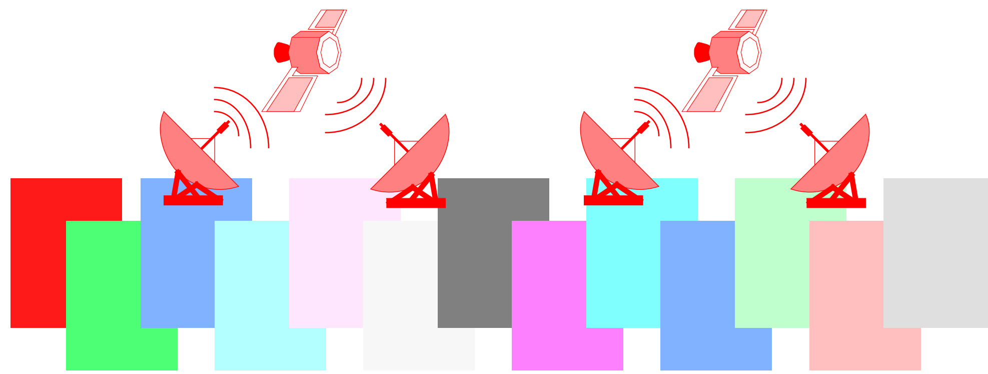
☞ **Central Message server (MSG)**



☞ **Clients**

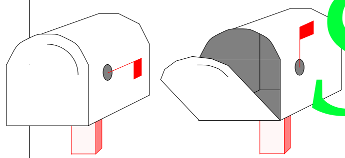
- **Send and receive messages**
  - ✗ **Command requests**
  - ✗ **Information**
- **Can be added or removed**

Message Patterns

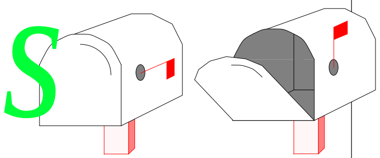


- Messages are strings

- Conventions for encoding
  - WHO CMD system args...
  - ID WHAT system info...



Sample Messages



DDT VIEW ***sys file fct line cnt stkdelta***

DDTR RUN ***sys args in out new***

DEBUG AT ***sys file fct line***

DEBUG VALUE ***sys file line var value***

XREF QUERY ***sys args conditions***

XREF FLOW SET ***file line button***

BUILD MAKE ***system***

FORM CONFIG ***dir file*** **CHECKIN** ***mode***

MSG Operation

☞ **Clients register patterns**
- **Scanf- based pattern matching**
- **Normal text matches exactly**
- **%s, %d, %f — argument match**

☞ **Tools send messages to server**
- **Rebroadcast selectively**
- **Printf formatting for send calls**

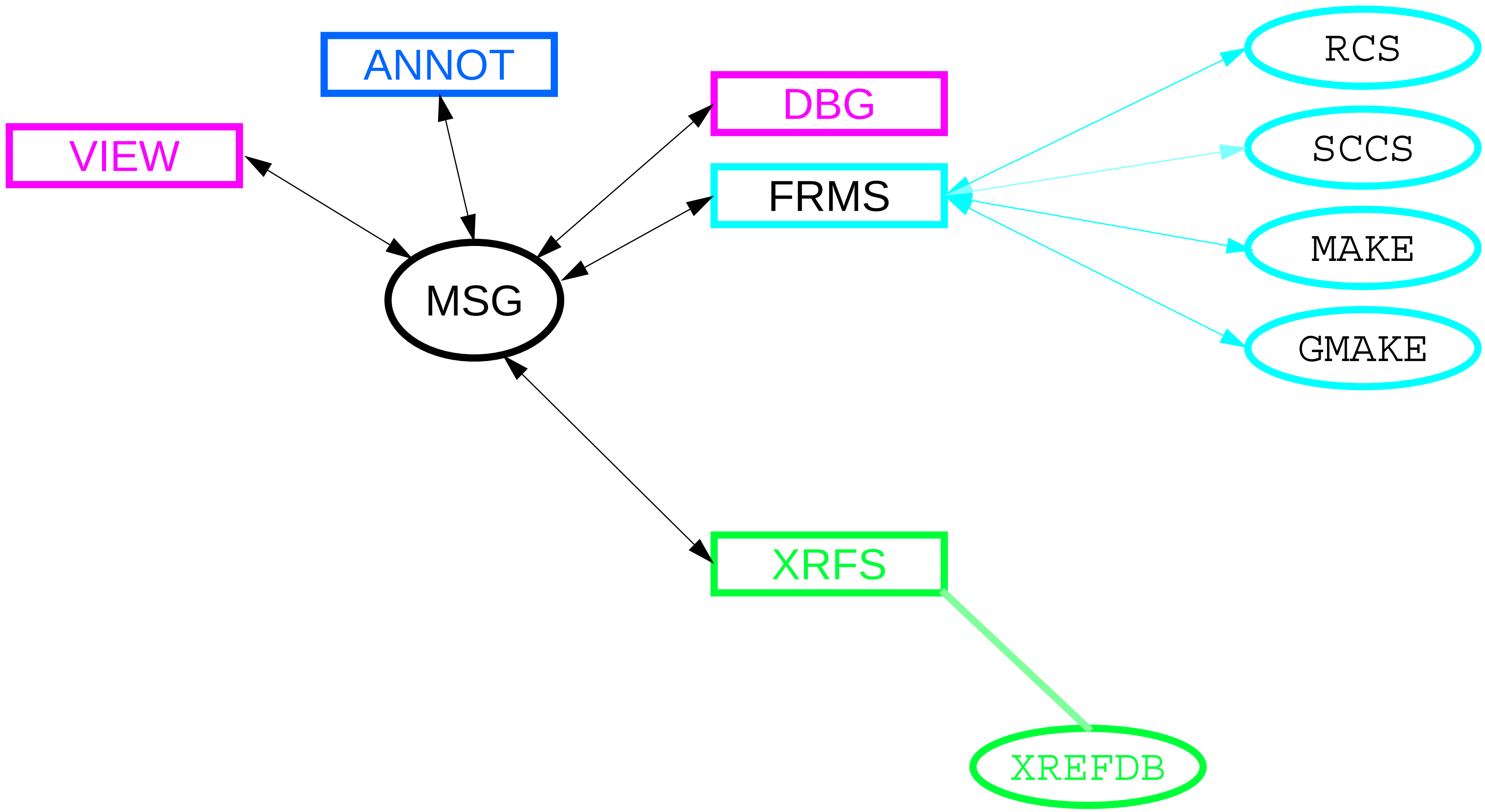
MSG: *Synchronization*

- **Messages can be asynchronous**
  - **Sender continues immediately**
- **Messages can be synchronous**
  - **Wait for 1st non-null reply**
  - **Wait for all receivers**
- **Messages are decoded by MSG**
  - **Receiver gets args & reply id**

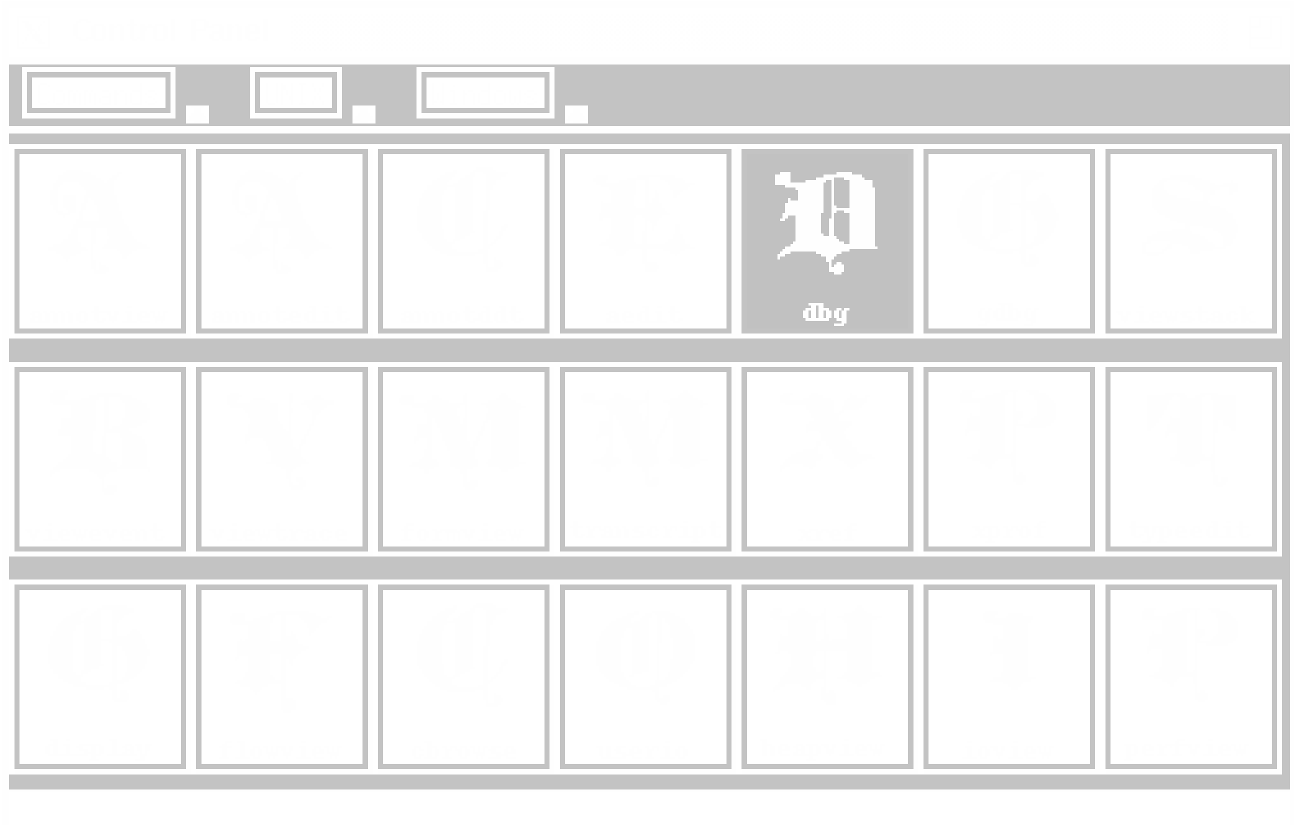
MSG Utility Services

- **MSG is responsible for services**
  - **Central to the environment**
  - **Every tool talks to it**
- **Working Directory**
- **Check and start services**
- **Environment**

Basic Organization



Control Panel



Overview Screen

DDT Program Control

- ☞ ***dbx* or *gdb* as low-level interface**
- ☞ **MSG-based debugger back end**
- ☞ **Textual debugger front end**
- ☞ **Window interface to debugger**
- ☞ **Separate interface for I/O, Views**

Debugger Extensions

- ***dbx* + *gdb* command languages**

- **Programmability**

- **C++, Pascal, Modula 3 Support**
  - **Full name mapping**
  - **Dynamic type information**

- **Extended commands**
  - **Stop in all <pattern>**

Debugger Interface

Annotation Editor

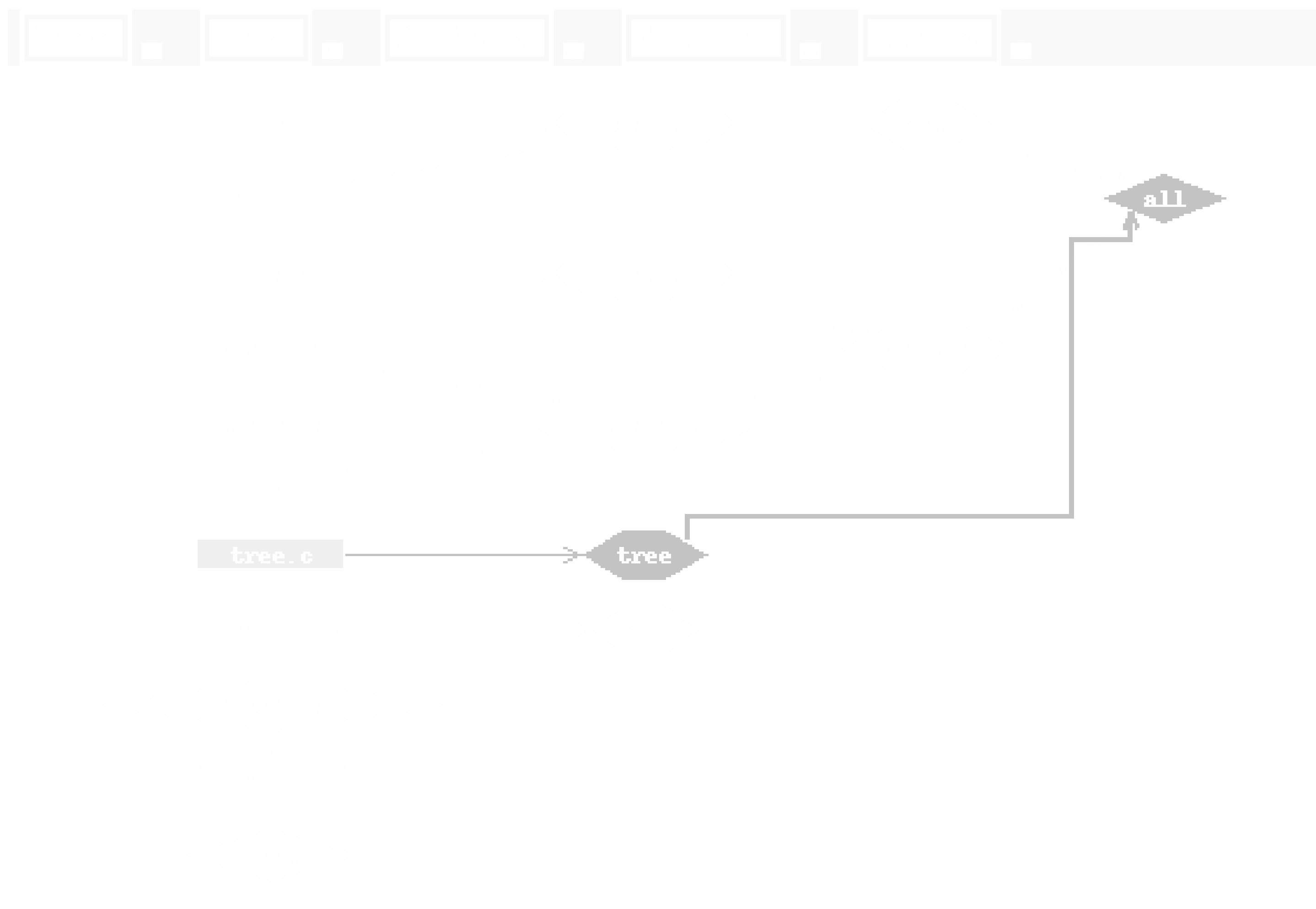
Annotations

- ☞ **Annotations relate source to tools**
- ☞ **Integrated through messages**
- ☞ **Defined in resource files**
- ☞ **Can be permanent**
  - **Original file, annotations saved**
  - ***Fixannot* uses *diff* & heuristics**

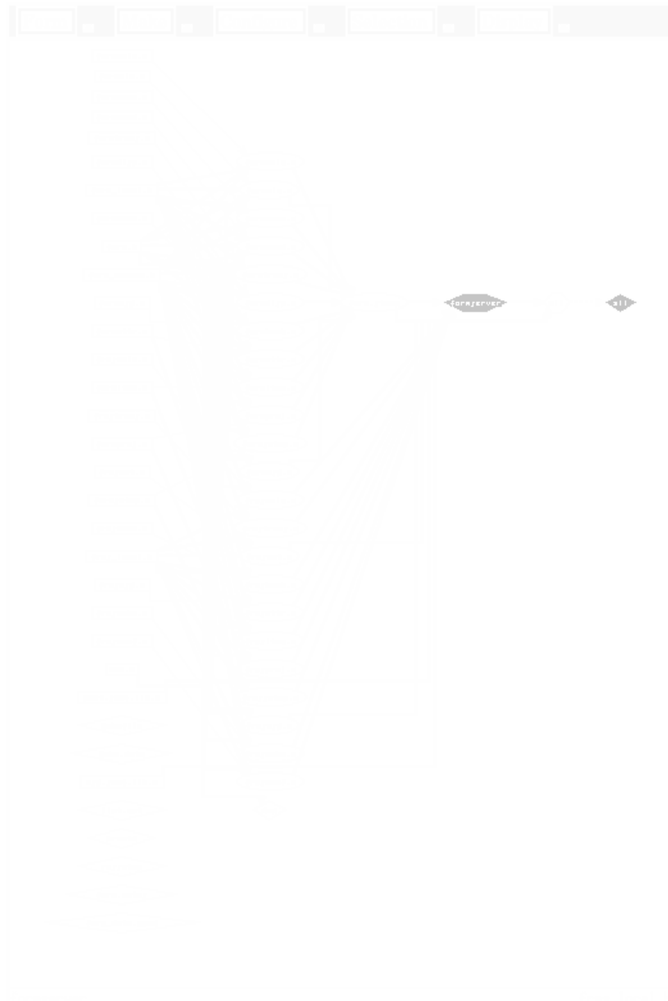
Configuration Management

- **Formview offers interactive MAKE**
- **MSG interface**
  - **Handle: BUILD, COMPILE, CONFIG**
  - **Generate: ERROR, WARNING**
- **Graphical dependency Display**
- **RCS support**

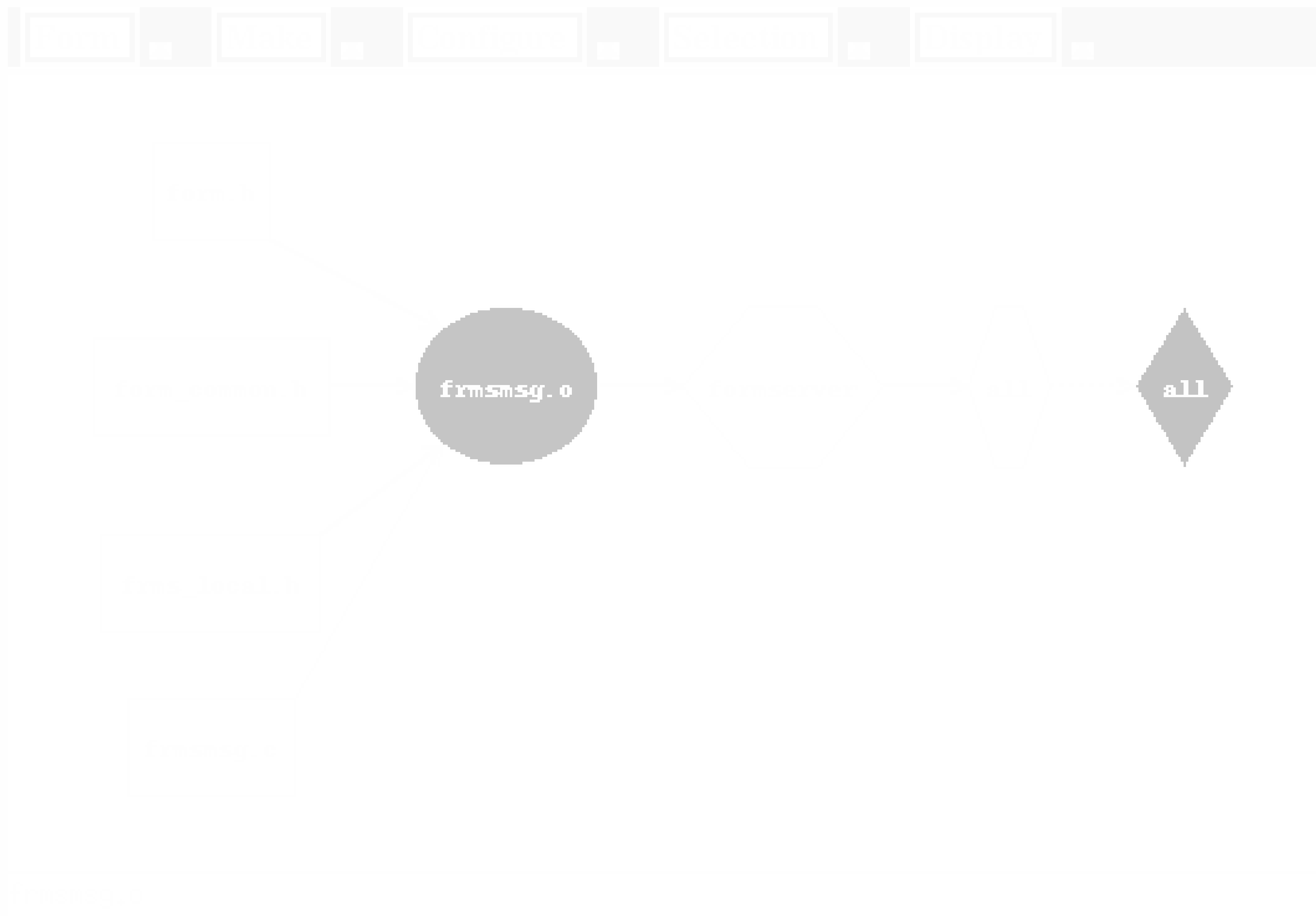
FORMVIEW Example



FORMVIEW Example



FORMVIEW Example



Cross Referencing

- **XREF offers interactive database**
  - **Definable QBE-like queries**
  - **Full relational database**
- **XREFDB provides services**
  - **Relational calculus queries**
- **MSG interface**
  - **Handles: XREF QUERY**
  - **Generates: XREF SET**

XREF Example

Profiling Utility

- **Visual front end to UNIX profiling**
  - ***prof, gprof, pixie, iprof, fgprof***
  - **Selective viewing -- file, fct, line**
  - **Incremental profiling capability**

- **MSG interface**
  - **Generates: XREF PROF SET**

XPROF Example

Flowgraph Visualization

☞ **Call Graph Display**

- **Use XREFDB to get information**
- **Hierarchical display; Grouping**

☞ **Browsing Capabilities**

- **Using hierarchy**
- **Selective include and exclude**
- **Local graph for a node**

☞ **Tracing Program Execution**

FLOWVIEW Example

FLOWVIEW Example

Class Visualization

☞ **Graphical view of class hierarchy**
- **Full browsing capabilities**

☞ **Shows wealth of information**
- **Class relationships**
- **Member information**
- **Member inheritance & props**
- **Information window & dialogs**

☞ **Tied to rest of system thru MSG**

CBROWSE Example

CBROWSE Example

CBROWSE Example

Data Structure Display

- ☞ **Graphical Display of arbitrary data**
  - **Dynamic updating**
  - **User definable displays**
- ☞ **DISP -- interface to GELO**
  - **Boxes, Tilings, Layouts**
  - **MSG interface to DDT**
- ☞ **TYPEEDIT -- interface to APPLE**

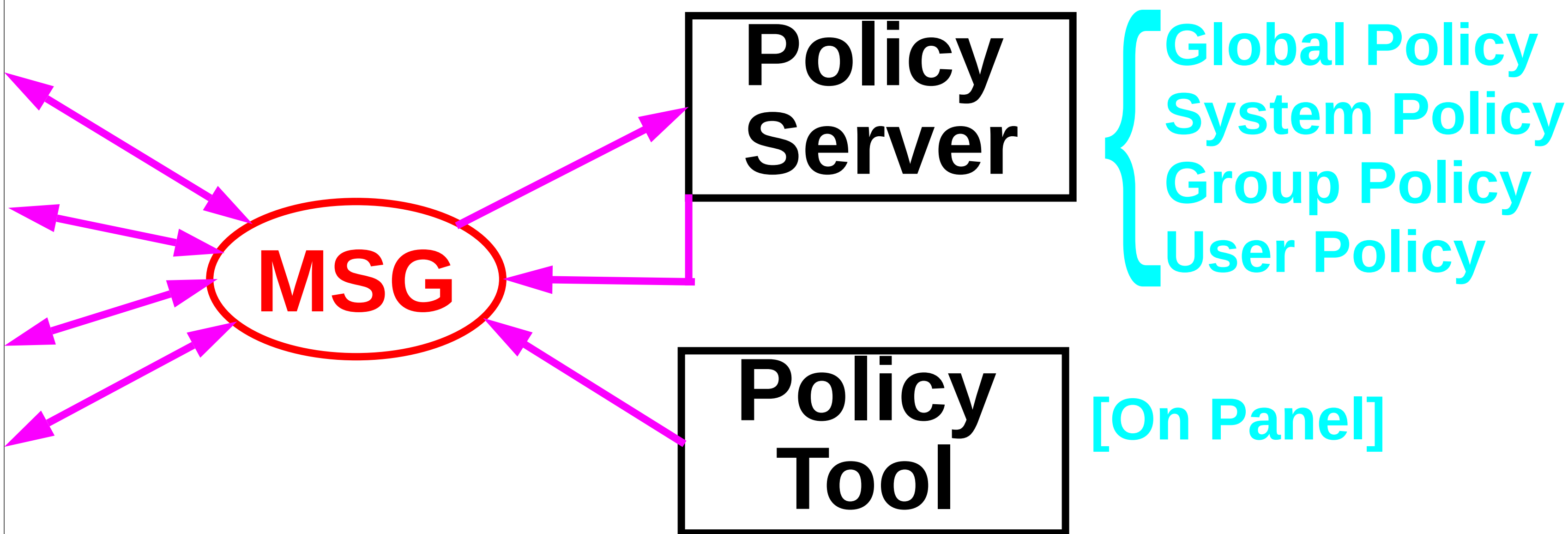
DISP Example

TYPEEDIT Example

DISP Example

DISP Example

FIELD Policy Tool



☞ **Policy Programs are used**

- **FOREST: Garlan, Ilias**
- **Mediators: Sullivan, Notkin**

Policy Programming

☞ **Declarations**

- **Levels, users, tools, patterns, variables, load requests**

☞ **Simple actions**

- **WHEN pattern IF cond DO action**
- **Done on a tool Basis**

☞ **Actions -- conditional; required**

- **Send, Setenv, Set, Call, Null**

Policy Programming

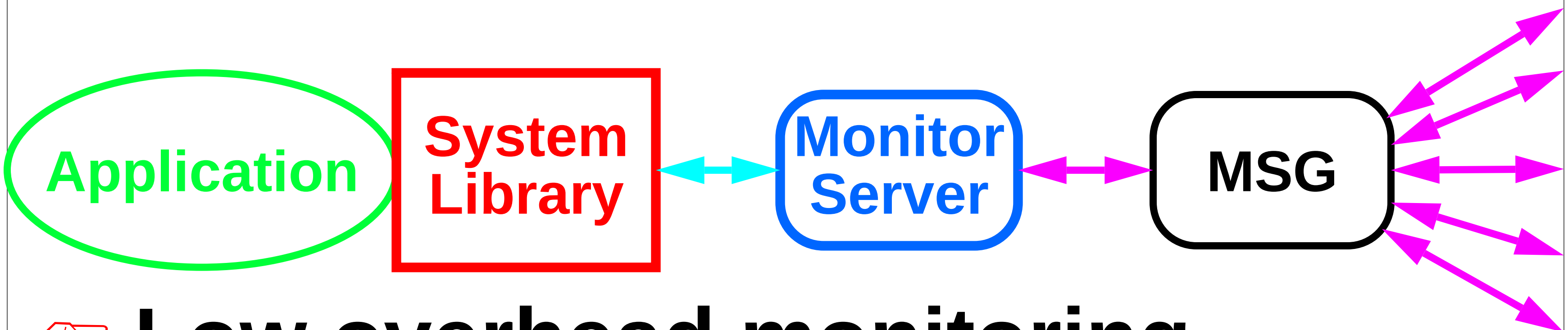
- **Complex Actions**
  - **Sophisticated policy programs**
  - **Based on path expressions**

- **FOR patterns [WITH locals] DO**
  WHEN path_expr : actions
  ...
  END

- **Multiple actions can occur at once**

Program Monitoring

- Modified System (Shared) Library



- Low overhead monitoring

- Monitor
  - Memory usage (malloc, free)
  - File I/O (open, close, read, write, ...)
  - Performance (getrusage)

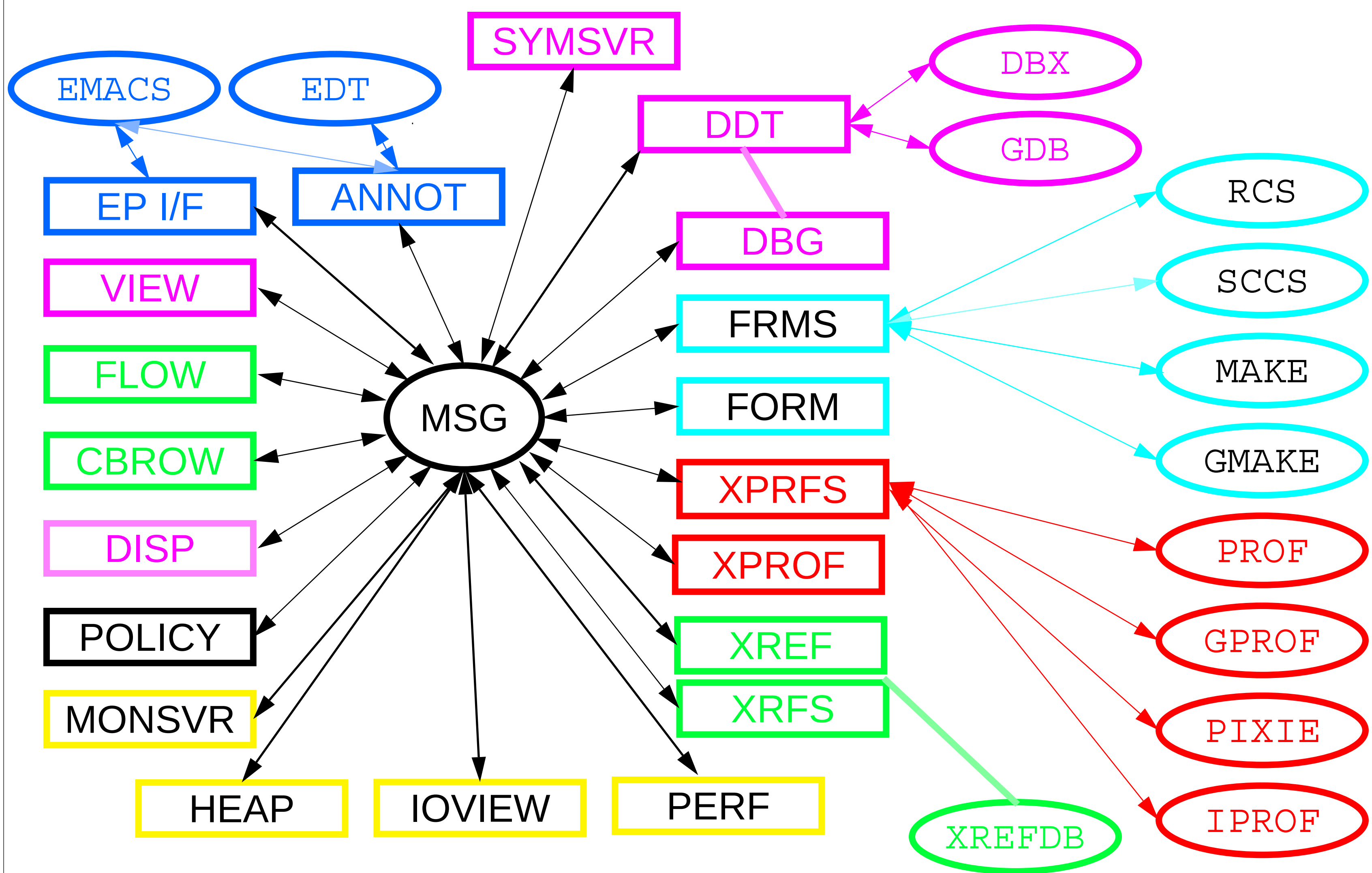
HEAPVIEW Example

IOVIEW Example

IOVIEW Example

PERFVIEW Example

FIELD Architecture

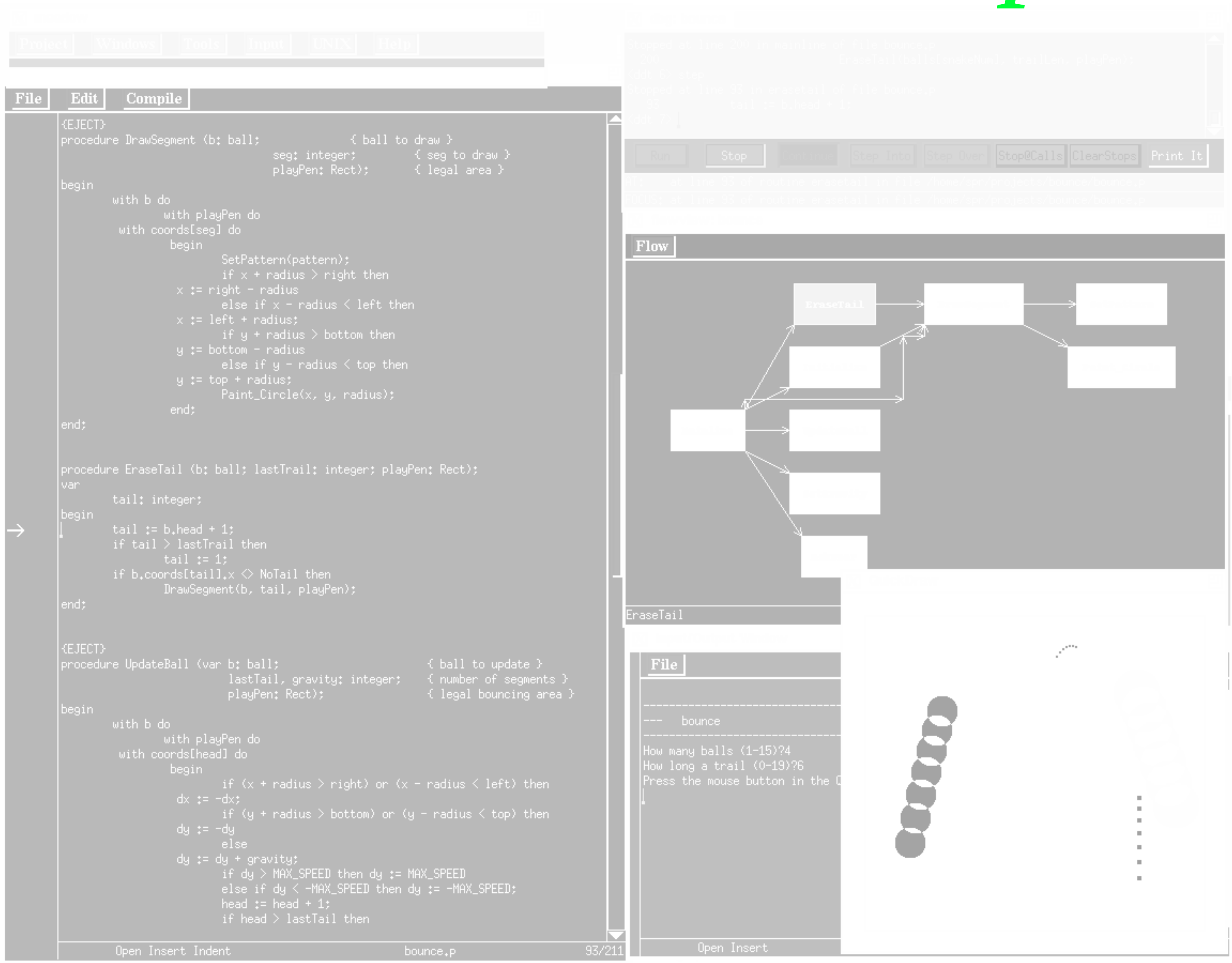


MEADOW

☞ **FIELD is an adaptable system**
- **Resource files used extensively**
- **Ability to dynamically load code**

☞ **MEADOW is the student version**
- **Simplified menus**
- **Single file Pascal programs**
- **Help and documentation**

MEADOW Example



Status of FIELD

- **Used extensively at Brown**
  - **For research (C++ debugging)**
  - **For instruction**

- **Commercialized**
  - **DEC/FUSE**
  - **Basis for Softbench, Tooltalk, ...**

- **Being distributed from Brown**

Current Research

- **Better programming tools**
  - **Intelligent debuggers**
  - **Intelligent documentation**
  - **Configuration management**
- **Abstraction visualization**
- **Design-level programming**

Abstractions

☞ **Model how system is seen**

☞ **Provide basis for visualization**

☞ **Four basic types:**
- **Syntactic**
- **Semantic**
- **Event-based**
- **Data-based**

Syntactic Abstractions

- **Based on program syntax**

- **Examples:**
  - **Call graph**
  - **Class hierarchy display**
  - **Design language**

Semantic Abstractions

☞ **Based on program analysis**
- **More complex than syntactic**

☞ **Examples:**
- **Program slices**
- **Data flow graphs**
- **Library-based (e.g. threads)**
- **Potential class groupings**

Event Abstraction

☞ **Based on dynamic events**

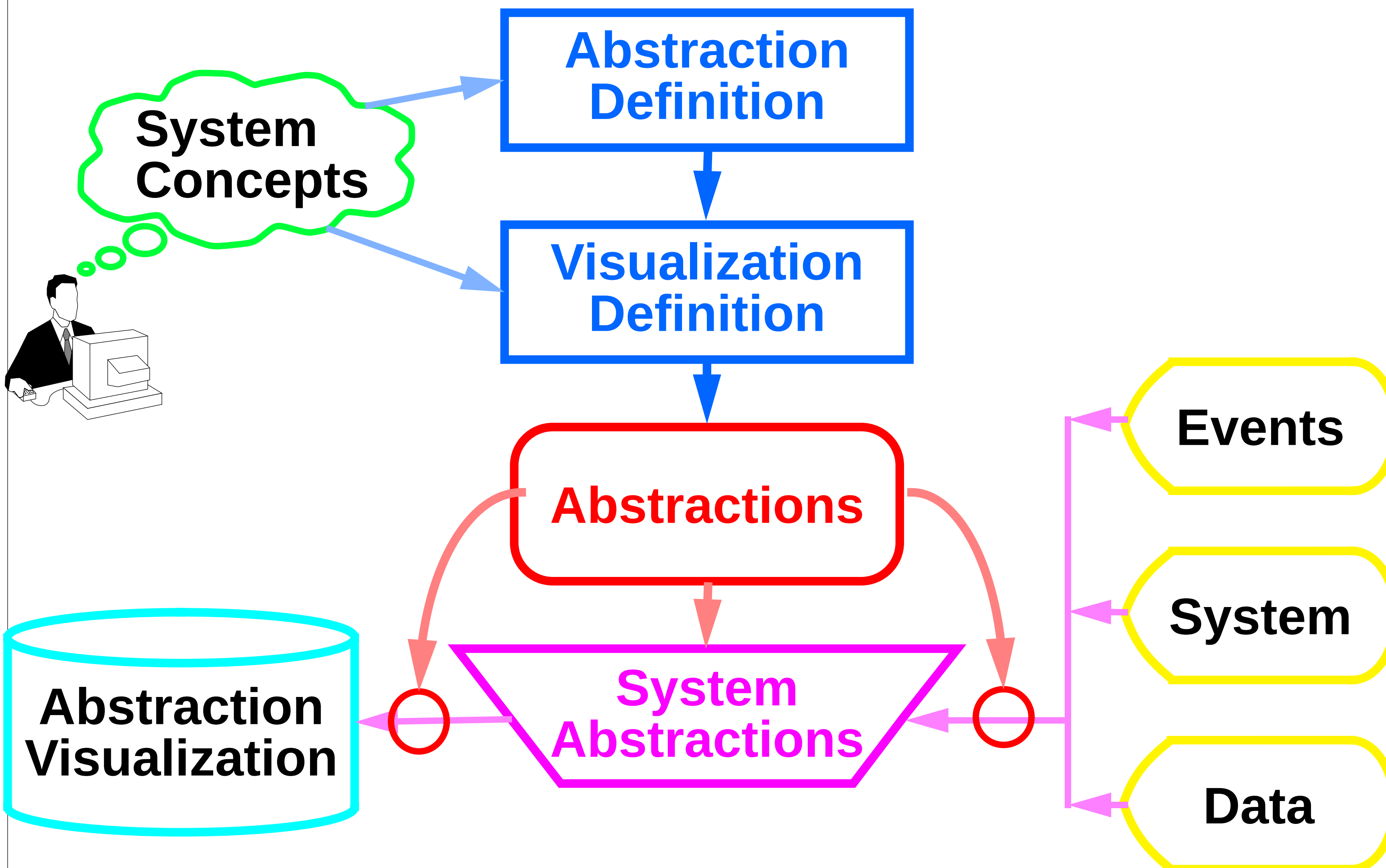
☞ **Examples:**

- **Dynamic call graph**
- **Synchronization viewer**
- **Algorithm animation**

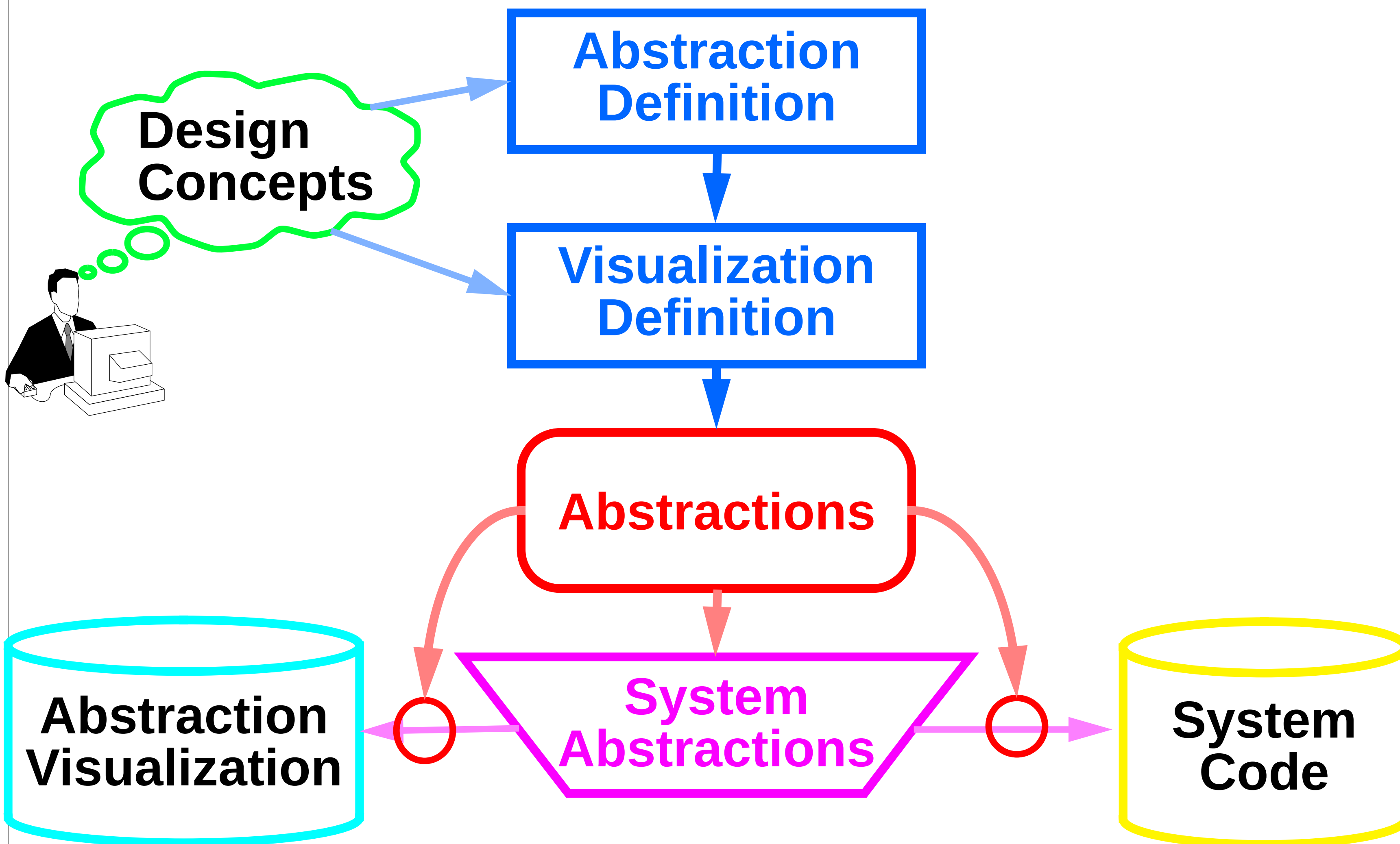
Data-based Abstractions

- **Based on program data**
- **Examples:**
  - **Data structure visualization**
  - **Make dependency graph**

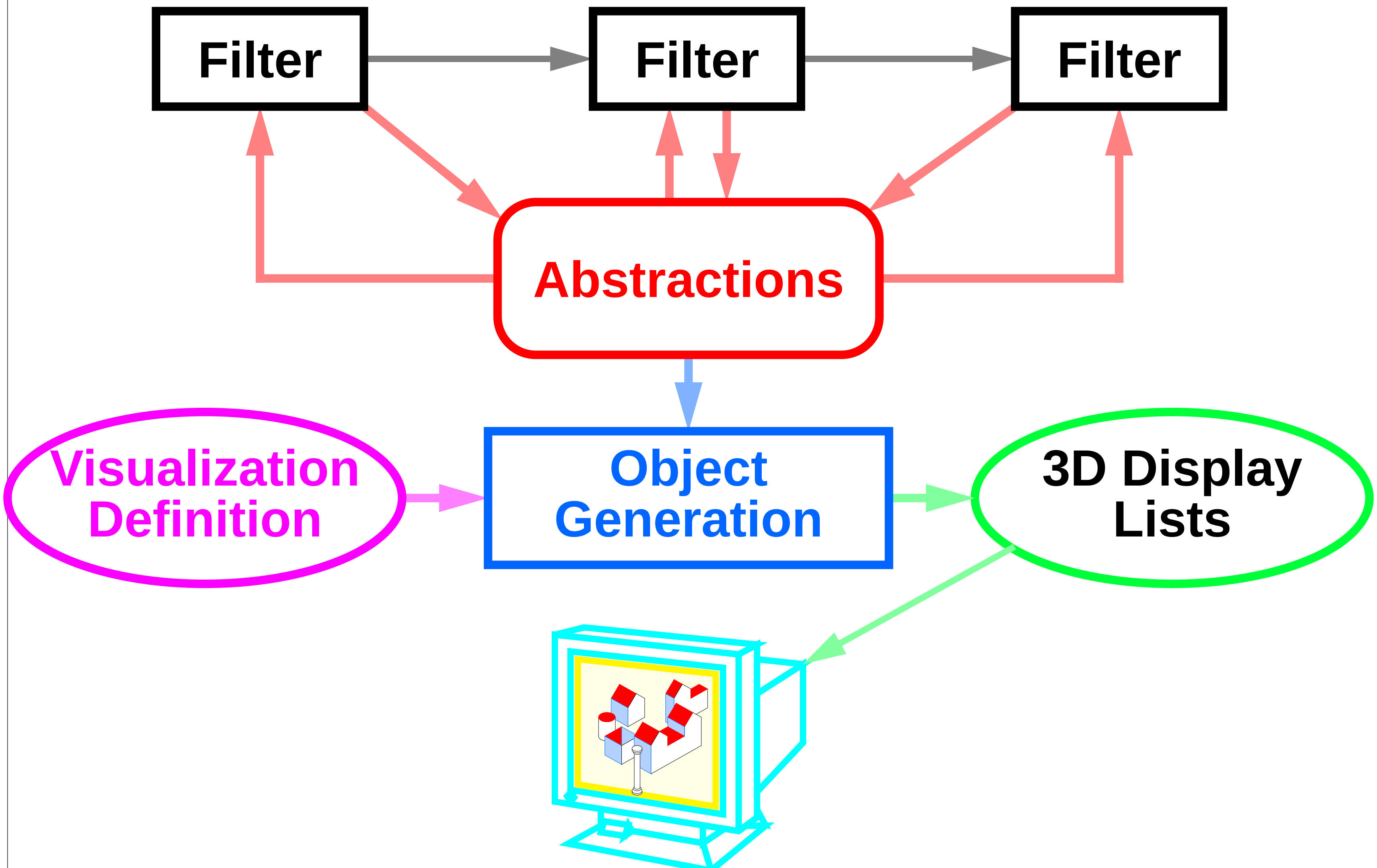
Abstraction Visualization



Abstraction Editing



Editing Abstractions



Questions

Comments

Suggestions ...

The End