Methodologies for Rapid Prototyping of Tools for Visualizing Performance of Parallel Systems

James Arthur Kohl
Thomas L. Casavant
Parallel Processing Laboratory
Department of Electrical and Computer Engineering
University of Iowa
Iowa City, IA 52242

OUTLINE:

I. Introduction & Motivation

II. Overview of PARADISE

III. User View

IV. Meta-Tool Designer View

V. Toolkit View

VI. Directions for Future Work

INTRODUCTION

Selected Questions:

- What performance data would you like to see from a 1K-10K PE MIMD system?
- What performance data would you like to see from a large SIMD system?
- What is the relative importance of system performance data versus application data?
- Do we need time-dependent data or are averages sufficient?
- How do we compare application performance across multiple architectures?
- Can we get more detailed, portable data than just execution times?
- What kind of performance tools do we need?

Answers:

- Don't know. Maybe this, but what about that? Not sure.

$\rightarrow$ Too early to tell yet?

MOTIVATION

- Complexities of Parallel Software Development
- Standard Sequential Methods Inadequate
- Visualization Techniques: "Program Visualization"

PROBLEMS

- Visualization is Special Purpose in Nature - Custom Tools
- Expensive Development of Visualization Tools
- Requires Working Knowledge of Graphics System
- What to Visualize?

  $\Rightarrow$ Every User Has Unique View and Intuition

  $\Rightarrow$ Is the Effort Actually Worth the Benefit?

APPROACHES

- User View

    $\rightarrow$ What Does the User Want?

- Meta-Tool Designer View

    $\rightarrow$ What Can a Meta-Tool Designer Provide?

- Toolkit View

    $\rightarrow$ Are We Making It All Too Complicated?

$\Rightarrow$ Why Do Users Do It The Hard Way?

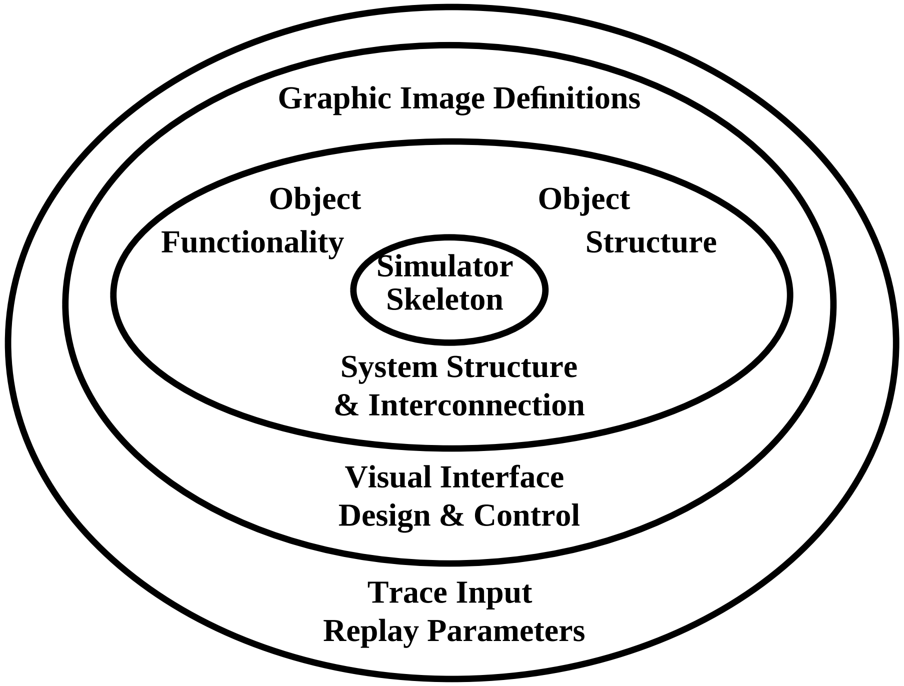
PARADISE

PARallel Animated DebuggIng and Simulation Environment

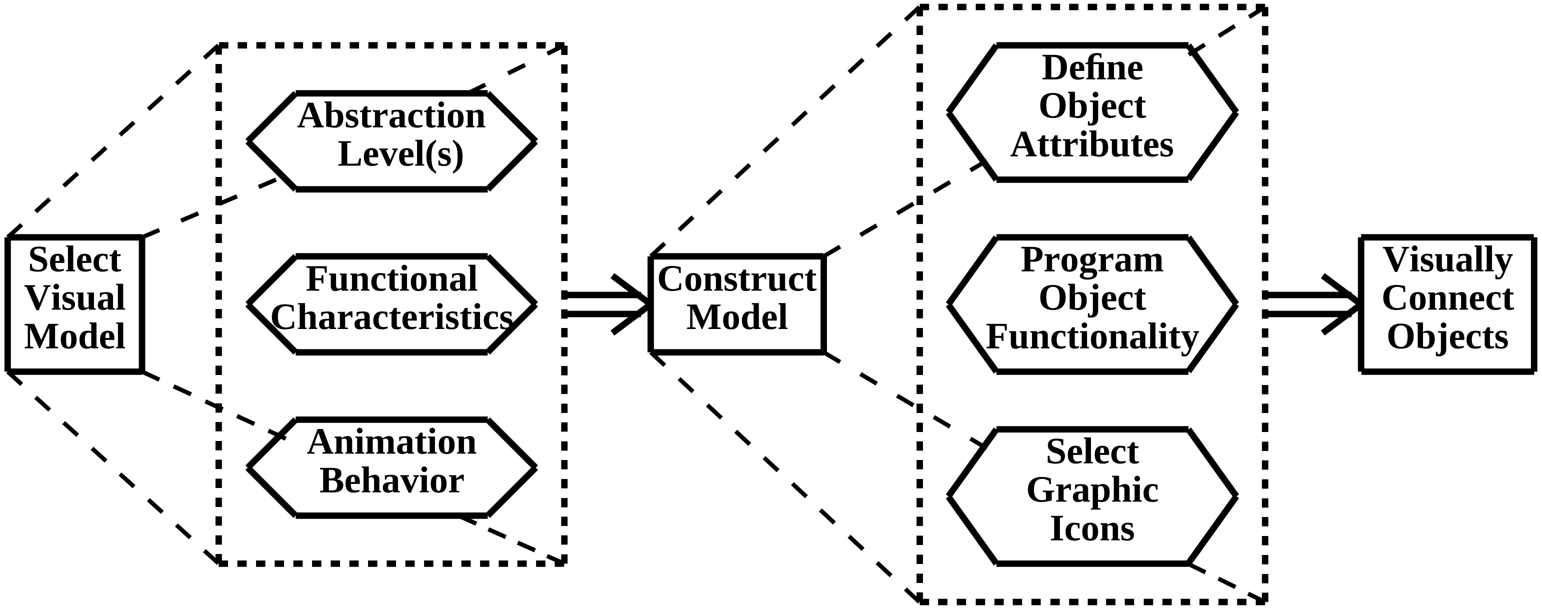
- Meta-Tool for Constructing Visual Analysis Tools
- Custom Visual Models
  - $\longrightarrow$ Object-Oriented, Modular
  - $\longrightarrow$ Animated Simulations
- Interactive Visual Interface
  - $\longrightarrow$ Model Construction
  - $\longrightarrow$ Simulation Control
  - $\longrightarrow$ Manipulate Objects in Model Visually

PARADISE INTERFACE

PARADISE LAYERED STRUCTURE



SCENARIO OF USE:



- User Constructs Visual Model
- PARADISE Animates Overall System Behavior
  - → Individual Object State
  - → Interaction Among Objects
  - → Global System Dynamics

USER VIEW

- Custom Views / Configurability

    → Individual User Preferences & Perspectives

    → "Law of Unlimited Options"
    Users will continue to request new options throughout development and usage of tools. The number of options expands without bound and the tool is never completed.

    → User Never Completely Pleased Unless They Can Customize

⇒ PARADISE Provides Design of Arbitrary Objects

  → User Chooses Object Images and Functionality

  → Model Caters to Individual User Intuition

⇒ Pablo/Hyperview & Triplex Configurability Via Filters

  → User Creates Configurable Processing Modules

  → Arbitrarily Arrange Filters to Produce Views

⇒ Tango, Voyeur, BALSA, Q+ - Arbitrary Animations

USER VIEW

- Flexibility and Expressive Power of Interface

    $\rightarrow$ Want Power to Construct Highly Custom Analyses

    $\rightarrow$ Want Simple and Automatic Alternatives As Well

    $\Rightarrow$ e.g. C Language - Low & High Level Capabilities

$\Rightarrow$ PARADISE Supports Abstract Visual Interface

$\rightarrow$ Animations as Complex as Desired (Advantage?)

$\rightarrow$ Automated Simulations

$\rightarrow$ Visual Model Specification Not Automatic

$\rightarrow$ Needs More Fundamental Graphics Capabilities

$\Rightarrow$ ParaGraph & TOPSYS Highly Automatic

$\rightarrow$ Less Flexible?

USER VIEW (cont.)

- Fast and Efficient Use

  $\rightarrow$ General Functionality as Automatic as Possible

  $\rightarrow$ Concise Specification of User Preferences

  $\rightarrow$ Natural Interface for Intuitive Interaction

  $\rightarrow$ Accumulation of User Configurations and Style

  $\Rightarrow$ One-Time Setup for Years of Analyses

  $\Rightarrow$ What Information?

$\Rightarrow$ PARADISE Visual Model Construction and Simulation Control

$\rightarrow$ Reusability of Specification Through Libraries

$\rightarrow$ Still Some Textual Specification

$\Rightarrow$ ParaGraph & TOPSYS - Automatic Views From Visual Menus

$\Rightarrow$ Pablo/Hyperview Visual Filter Configuration & Defaults Manager

USER VIEW (cont.)

- Multiple Granularity Control
  - $\rightarrow$ Visualization Speed Control
    - $\Rightarrow$ Adjustable Speed for Focusing
    - $\Rightarrow$ Flash Quickly Through Uninteresting Portions
    - $\Rightarrow$ Slow Down for Close Examination of Important Details
  - $\Rightarrow$ PARADISE, SHMAP, Others - Support Variable Speed Controls
  - $\Rightarrow$ Many Tools Support Start/Stop & Single Step
  - $\Rightarrow$ Real Time vs. Post Mortem?
    - $\rightarrow$ Control Speed in Real Time?
    - $\rightarrow$ Post Mortem Flexible, But Cumbersome Data Collection & Storage

USER VIEW (cont.)

- Multiple Granularity Control (cont.)
  - $\rightarrow$ Visualization Direction Control
    - $\Rightarrow$ Forward and Reverse Playback
    - $\Rightarrow$ Causality Problem - When to Stop?
    - $\Rightarrow$ Inherent Human Inaccuracy - Oops, Rewind !!
  - $\Rightarrow$ Difficult With PARADISE $\rightarrow$ Reverse Functionality? (Histories)
  - $\Rightarrow$ Easy With Timeline Displays
  - $\Rightarrow$ Not Easy for Most Displays
    - $\rightarrow$ No Inverse for Display Functions
    - $\rightarrow$ Undrawing Graphics?

USER VIEW (cont.)

- Multiple Granularity Control (cont.)
  - → Abstraction Level & Display Size
    - ⇒ Skip Low Level Details During General Viewing
    - ⇒ Avoid Unnecessary Sensory Overload
    - ⇒ Examine Low Level Details of Interest
  - → Analogies
    - ⇒ Human Grasping - Quick Move, Then Slow Grasp
    - ⇒ Learning - High-Level Overview, Then Low-Level Details
    - ⇒ Far Side Cartoons - Punch Line is in the Detail
  - ⇒ PARADISE Paradigm Supports Abstraction Capabilities
    - → Model Hierarchy
    - → Abstraction Zooming

USER VIEW (cont.)

- Animations and Views
  - $\rightarrow$ What Level of Abstraction or Detail?
  - $\rightarrow$ Scalability?
    - $\Rightarrow$ Some Analyses On Small Scale Elude To Large Scale
    - $\Rightarrow$ Many Large Scale Analyses Require Large Scale Details
    - $\Rightarrow$ Triplex "Binning" of Equivalence Classes
  - $\rightarrow$ How to Specify Visual Details - Choices / Alternatives
    - $\Rightarrow$ Fixed vs. Dynamic Images
    - $\Rightarrow$ Movable Images
    - $\Rightarrow$ Discrete vs. Continuous
  - $\Rightarrow$ PARADISE - Dynamic Images (Discrete), Immobile
  - $\Rightarrow$ TOPSYS (VISTOP) - Fixed Images, Mobile
  - $\Rightarrow$ Tango - Continuous Animation

USER VIEW (cont.)

- Extra Assistance in Analysis
  - $\rightarrow$ Show Information To Guide User
    - $\Rightarrow$ User Doesn’t Always Know Where to Start
  - $\rightarrow$ Display Information in Views Which Stimulate User
    - $\Rightarrow$ Emphasize Similarities and Contrasts
    - $\Rightarrow$ Inherently Provided By "Good" Visualization

$\Rightarrow$ Unanswered Issue

- $\rightarrow$ Custom vs. General
  - $\Rightarrow$ What is the Best Visual Display?
  - $\Rightarrow$ Is There a Best At All?
- $\rightarrow$ How Much of the Burden is on User Intuition?
- $\rightarrow$ Can General Visualization Spark User Ideas?

META-TOOL DESIGNER VIEW

- Mask Unnecessary Detail From User
  - → Hide Internal Graphics & Simulation Details
  - → Utilize Abstractions in User Interface - Visual Semantics
    - ⇒ Control of Simulation - "Visual Programming" of Analysis Control
    - ⇒ Specification of Visualization - Abstract Visual Language
      - Solves Visual Programming Problem?
      - More Difficult Than Natural Language - Universal Language
      - Possible Within Limited Context of Visualization?

⇒ PARADISE Hides Internal Simulation and Graphics Details

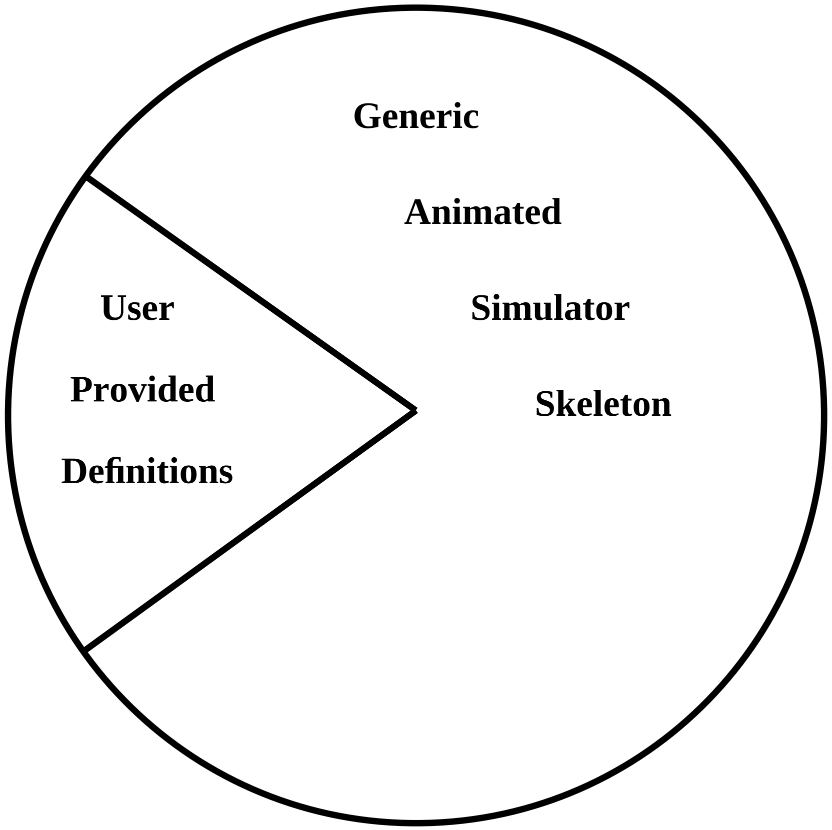
- → Plug-In Simulator Concept (See Figure)
- → Specification of Model Functionality → Cumbersome
- → Need Better, More Abstract, Visual Semantics

⇒ Tango Uses Abstract Visual Manipulations

- → Highly Integrated With Source Program - Need Separation of Concerns

Plug-In Simulator Concept

User generates a module of fundamental system definitions of event types and objects, including structure and functionality, and then "plugs in" these definitions, into a generic animated simulator structure, to produce a customized animated simulation environment.



→ Complete Customized Animated Simulator

META-TOOL DESIGNER VIEW (cont.)

- Provide Customizability in Views and Options
  - $\rightarrow$ Generalize and Formally Classify Fundamental Parameters
    - $\Rightarrow$ What Are These Parameters?
    - $\Rightarrow$ What Operations On These Parameters?
  - $\rightarrow$ Construct Means for Specifying Parameters of Classes
    - $\Rightarrow$ Abstract, Probably Visually-Oriented, Language
  - $\rightarrow$ Design Environment Without Static Semantics
    - $\Rightarrow$ Minimal Predefined Parameters and Configurations
    - $\Rightarrow$ Generic Skeleton - Details To Be Filled In By User
  - $\Rightarrow$ PARADISE, Tango, Pablo/Hyperview, Triplex
    - $\rightarrow$ Various Levels of Customizability
    - $\rightarrow$ Assortment of Visual "Languages" or Interfaces

META-TOOL DESIGNER VIEW (cont.)

- Provide Flexibility for User Specification Interaction
  - $\rightarrow$ Allow Dynamic, Interactive Specification
    - $\Rightarrow$ Interpretation for Dynamic Characteristics & Parameters
    - $\Rightarrow$ Dynamic Compilation for Static Characteristics & Parameters
    - $\Rightarrow$ Interactive Modification of Specification

$\Rightarrow$ PARADISE, Pablo/Hyperview, Triplex, ParaGraph

- $\rightarrow$ Various Levels of Dynamic Specification

META-TOOL DESIGNER VIEW (cont.)

- Provide Fast and Efficient Interface
  - → Minimize User Overhead
    - ⇒ Minimal Number of Steps for Operations
    - ⇒ Reduce Keystrokes, Mouse Clicks, etc.
  - → Automatic Capabilities
    - ⇒ Intrinsic Functions For Standard (Generic) Analysis Functions
  - → Specification Reuse
    - ⇒ Simple Storage and Retrieval Interface
    - ⇒ Libraries for Application Sets
    - ⇒ Replication Schemes / Specification "Macros"
  - → Default User Preferences
    - ⇒ Categorize Candidate Parameters
    - ⇒ Organize Information Using Database
    - ⇒ Concise Yet Simple Interface

META-TOOL DESIGNER VIEW (cont.)

- Providing Animations and Views
  - $\rightarrow$ How to Drive An Animation - State Machines?
    - $\Rightarrow$ Moore Approach - Large Collections of States Control Output
    - $\Rightarrow$ Mealy Approach - Fewer, More Complicated States
  - $\rightarrow$ Specify Animations?
    - $\Rightarrow$ Sequences of Discrete Images
    - $\Rightarrow$ Initial States & Functions for Continuous Modulation
    - $\Rightarrow$ Expressions for Relating Performance Data to Animation

TOOLKIT ALTERNATIVE

- Provide Limited Set of General Tools
- User "Makes Tools Work" for Desired Analysis

  $\Rightarrow$ e.g. Unix Hacker, "Hammer Fixes Anything" Approach.

- Larger the Tool Kit, Better the Solution
- Adding New Tools to Kit - Cost / Performance Issue

Drawback:

- Even When "Good" Solution Exists, High Overhead to Create

  $\rightarrow$ User Experience With Tool Kit Reduces Overhead

  $\rightarrow$ Feasible Alternative - To What Extent?

$\Rightarrow$ TOPSYS, ParaGraph, IPS-2, Faust

  $\rightarrow$ Lean Towards Tool Kit Perspective

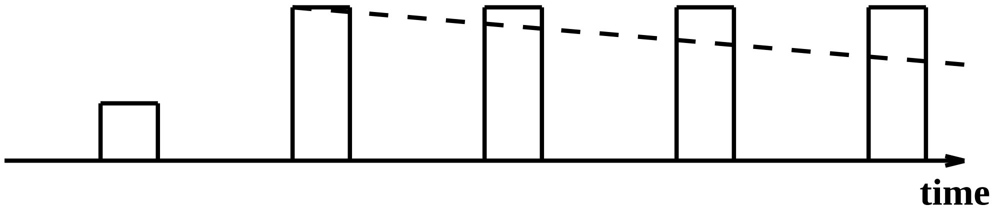
EFFECTS OF EXPERIENCE AND LEARNING

Time for First Analysis = Learn_Time + Design_Time(Analysis.1)

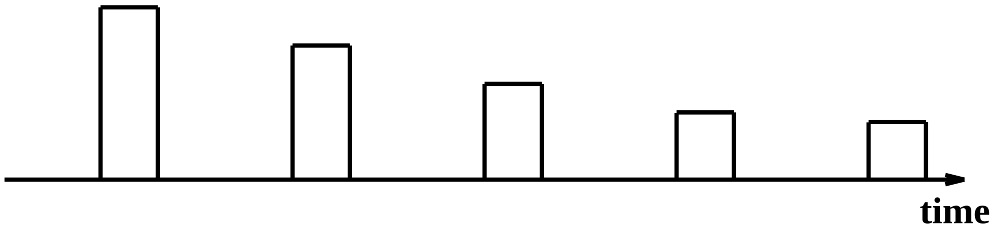
Time for Subsequent Analyses = Design_Time(Analysis.N) - f(Experience)

$\rightarrow$ Pay the "Dues" of Learning, Benefit From Experience

- Tool Kit Approach



- Meta-Tool Approach



Directions for Future Work

Δ Consideration of True User Needs and Perspectives

Δ Development of Abstract, Natural, Visual Specification Language

→ Construction of Specifications for Analysis

→ Visual Programming in General

→ Manipulation of Graphics in Animation

Δ Range of Capability → Simple Power

Δ More Flexible User Preference Integration

Δ Visualization Control Issues - Focusing & Scalability