INSTRUMENTATION DYNAMIQUE POUR L'ADMINISTRATION ET L'ANALYSE D'APPLICATIONS A COMPOSANTS

Walter RUDAMETKIN
Master 2 Recherche – SL
Projet SARDES INRIA – LIG
Outline

• Introduction
• Existing solutions
• Our objectives
• Our approach
• Short summary
• Implementation
• Conclusions
• Future work
Outline

• Introduction
• Existing solutions
• Our objectives
• Our approach
• Short summary
• Implementation
• Conclusions
• Future work
SARDES Project

- Part of INRIA and LIG
- Main research subjects
  - Distributed systems
  - Operating Systems
  - Middleware
  - ...
- Our area of interest
  - Dependable and adaptable software infrastructures
    - Component Model
    - Reflective programming
Problems

- Applications are complex
  - Hundreds of thousands of lines of code
- Code is not reusable
  - Mix of functional and non-functional concerns
- Hard to Debug and Optimize
- Analysis is insufficient or too low-level
  - Thousands of events to comprehend
- Administration capabilities are lacking
  - Interaction points ineffective
Overview of our approach

• Two large groups of tools
  • Application analysis tools
  • Application management tools
  • Tools are independently developed to solve individual problems

• Proposition
  • Unify management and analysis
    • Provide shared, dynamic and fine-grained instrumentation
    • Create an application management infrastructure
    • Provide application analysis interaction points
Outline

- Introduction
- Existing solutions
- Our objectives
- Our approach
- Short summary
- Implementation
- Conclusions
- Future work
Analysis VS. Management

- **Analysis**
  - Study the application

- **Techniques**
  - Profiling
  - Workload analysis
  - Performance debugging

- **Management**
  - Control the application

- **Techniques**
  - Administer non-functional concerns
  - Interposition
  - Meta-applications
Application Analysis

• Software tracing
  • Obtain information from the application
• Profiling
  • Investigate programs behavior
  • Optimize specific parts
• Workload analysis
  • Analyse responses to varying workloads
Example: Magpie

- Fine-grain analysis of events
- Extracts requests from low-level event analysis
- *Analysis* requires complicated event schemas
- *Cannot* control execution, only analyse events
Application Management

• Interposition
  - Simple techniques for modifying execution
  - Problem specific solutions (e.g., DoS, QoS)
  - Integrated into the application

• Meta-applications
  - High-level concept for *non-functional* concerns
  - External to the application
  - Control execution of the application
Example: Causeway

- Executes user-specified code at interception points
- Automates metadata propagation
- Very coarse-grain – System call interception
Outline

- Introduction
- Existing solutions
- **Our objectives**
- Our approach
- Short summary
- Implementation
- Conclusions
- Future work
Motivations

- *Management should use Analysis*
  - “Control requires understanding”
  - Granularity of analysis and management are inconsistent
    - Some analysis tools are very fine-grain
    - Management interaction points are very coarse-grain
  - Developers are forced to interpret results from multiple independent tools
Goals

• Bridge the conceptual gaps
  • Identify individual tasks across different tools
  • Synchronize “understanding” with “control”

• Integrate Analysis in Management
  • Use information obtained from analysis for decisions in management
  • Improve software
    • Self-optimizations
    • Flexibility
    • Reusability
Requirements

• Unified instrumentation
  • Fine-grain — useful for both analysis and management
  • Dynamic — avoid overhead when not needed

• Abstract entities
  • Granularity understandable by developers, useful to tools

• Integrate analysis into management
  • Meta-application infrastructure with analysis based on same granularity
Outline

• Introduction
• Existing solutions
• Our objectives
• Our approach
• Short summary
• Implementation
• Conclusions
• Future work
Our approach

• Provide common granularity
  • Requests
• Integrate management and analysis
  • Management requires Analysis
• Build Application Management Infrastructure
  • Take meta-application approach
  • Share instrumentation and request abstraction
  • Additional requirement
    • Metadata and context propagation
• Base solution on Component Model
Based on Component Model

- Improvements on modularity
- **Dynamic reconfigurations**
- Introspection
- Well defined interactions (through bindings)
Application Management

Non Functional Concerns

Meta-application Infrastructure

Fine-grained Interaction-points

Instrumented Application

Security

QoS

Priority

...
Layer one
- Provide expression of non-functional concerns

Layer two
- Analyse instrumentation and provide Request entity
- Interpret and execute user specified code
  - Non-functional concerns
  - At interaction points
- Propagate contexts

Layer three
- Provide instrumentation
  - Well defined, consistent, fine-grained interception points
We describe a request as
- The sequence of application components involved in its processing
- Asynchronous links between different tasks of the request
Instrumentation

- Dynamic tracers for synchronous execution
- Asynchronous annotation toolkit for asynchronous execution
Dynamic tracers

• Inserted into application bindings
  • Automated creation of dynamic tracer
  • Transparent insertion into application

• Detect thread execution
  • When a call is made
  • When a call returns
  • When an error occurs from the call

• Uses
  • Create *thread execution path* for request tracking
  • Provide interaction points for *meta-application*
Asynchronous Annotation Toolkit

- Annotations inserted into code
- Identify asynchronous execution
  - Thread creation/Thread pools
  - Message passing
  - Data streams and files (not yet implemented)
- Uses
  - Identifying dependency between thread execution paths
  - Context propagation points
    - Contexts must follow data across communication points
Context Propagation

- **Metadata**
  - Meta-applications store and access information

- **Propagation**
  - Must follow communication paths expressed by the *asynchronous execution annotation toolkit*
  - Automated propagation

- **Improvements over existing solutions**
  - Two novel types of metadata
    - *request context* and *message context*
  - Respect causal information pathways
    - Handle multiple contexts
Request consumer interface

- External applications solicit *request execution paths* for analysis
  - Performance debugging
    - Why did *this* request take much longer than *that* request?
  - Fault detection
  - Configuration (application tuning)
- Performance prediction
  - Realistic workload models for capacity planning
  - Obtain automatically on a “live” system
Outline

• Introduction
• Existing solutions
• Our objectives
• Our approach
• Short summary
• Implementation
• Conclusions
• Future work
Summary of the meta-application infrastructure

- **Support non-functional concerns**
  - This request is more important than that one
  - Request is taking too long, cancel it or increase priority?
  - Load-balancing
  - Quality of service
  - ...

- **Support for application analysis**
  - Provides a *Request Consumer Interface*
    - Profiling
    - Workload analysis
    - Performance debugging
Outline

- Introduction
- Existing solutions
- Our objectives
- Our approach
- Short summary
- **Implementation**
- Conclusions
- Future work
Implementation

- Fractal Component Model
  - Modular and extensible component model
    - Various implementations using different platforms
    - Multi-purpose model
  - Heavily uses the separation of concerns design principle
    - Separate into distinct pieces of code different functionality
  - Open-source
  - Chosen implementation
    - Julia, Java and reference implementation
**Architecture**

Meta-Application Manager

Trace Administrator

Trace Creator

Trace Logger

Trace Pool

T1

T2

Tn

Trace Manager

Request Manager

Request Analyzer

Request Recorder

Request Tracker

Context Handler

Callback Administrator

Callback Creator

Callback Logger

Callback Pool

C1

C2

C3

Cn

Callback Manager
Architecture(2)

CONCEPTUAL VIEW
Outline

- Introduction
- Existing solutions
- Our objectives
- Our approach
- Short summary
- Implementation
- Conclusions
- Future work
Conclusions

• Unified instrumentation
• Improvement for both analysis and management
• Fine-grain interception points make meta-applications more useful than before
• Fully dynamic solution
• Generic solution
Outline

- Introduction
- Existing solutions
- Our objectives
- Our approach
- Short summary
- Implementation
- Conclusions
- Future work
Future work

● Quantify overhead (i.e., execution, memory)
● Create a profiling application to prove unification
● Create a DSL (Domain Specific Language) to improve meta-application creation
  ● Specify non-functional concerns
  ● Specify interaction points
  ● Simply interaction between users and the meta-application infrastructure
Questions