Modeling and Quantifying IT System Configuration Complexity

Aaron B. Brown, Alexander Keller, Joseph L. Hellerstein
Introduction

- Focus of this work: human labor cost of IT systems management

- Key source of labor cost is complexity of configuration procedures
- Goal: create quantitative framework for measuring and addressing complexity of IT system configuration
Why Focus on Quantification?

- **Sustained progress comes only with quantitative measurement**
  
  “Benchmarks shape a field, for better or worse”
  
  – *Dave Patterson*

  - Case in point: past two decades of database performance improvement largely driven by quantitative TPC benchmarks

  - How do quantitative measures wield this power?
    - Provably demonstrate benefits of new technology
    - Provide targets, incentives, measures for improvement
    - As standards, enable competitive positioning of ideas and implementations

- **Challenge: Can we develop quantitative complexity measures for IT system configuration?**
Outline

- Introduction
- Quantifying Configuration Complexity
  - Methodology
  - Case Study
- Next Steps & Conclusions
Options for Quantifying Configuration Complexity

- **Static complexity of an existing configuration**
  - Sample metrics
    - Number of configuration controls
    - Degree of coupling between components
  - Sample techniques
    - Design Structure Matrix analysis
    - Dependency analysis
    - Graph complexity of system model
  - Drawbacks
    - Not representative of actual usage and labor cost

- **Dynamic complexity of creating or changing a configuration**
  - Sample metrics:
    - Number of steps to make a change
    - Number of configuration controls altered
  - Examples
    - HCI user studies
    - Model Human Processor analysis
    - Activity-Based Costing analysis
  - Drawbacks
    - Cost and time to collect experimental data
Quantifying Dynamic Configuration Complexity

- **An analytic approach to dynamic configuration complexity**
  - Quantification achieved through analysis of captured configuration procedures
  - Provides insight without costly user studies

- **Leveraging expert human intelligence**
  - Procedures captured from authoritative sources starting with configuration goals

- **Focused on complexity perceived by expert users (like sysadmins)**
  - Based on fundamental structural complexity of procedure, not UI details
Complexity Analysis of Configuration Procedures

Example Process: DB2/WAS configuration for 2-node SPECJAppServer

**Design creates process**

**DB2 Server Context**

**Quality defined by resulting configuration**

**Memory is parameter flow across a cut**

**Execution**

### Parameters
- user/pass
- db_name
- db2host

### Execution Parameters
- db2_port
- instance
- svc_name
- env_var
- app cmd (cli)
- cfg file (1 line)
- spec params
- cfg file
- env_var
- app cmd

### WAS Server Context
- J2EE App
- EJBs
- DBMS Client
- OS: WinXP
- node alias

- Context model used to define step boundaries
- Nested contexts abstract UI details, allowing fair cross-system comparison
SPECjAppServer simulates business process for a fictitious Car Manufacturer
  - Manufacturing, supply-chain management, order/inventory tracking, external dealers

J2EE 1.3 Application
  - Servlets, EJBs, JSPs to implement Transactions, Persistence, Messaging, Security
  - Web, EJB, DB and Messaging tiers
Complexity Analysis: Manual Provisioning of SPECjApp

- **Manual configuration procedure:**
  - Execution complexity
    - 59 steps, 27 context switches
  - Parameter complexity
    - 32 parameters used 61 times, 18 outside of source context
    - Source score: 125
  - Memory complexity (LIFO stack model)
    - Size: max 8, avg 4.4
Could Automated Provisioning Help?

CHAMPS System*

Stakeholder | Requirement
---|---
Customer | Request for Change
Developer | Deployment Descriptors
Customer | Service Level Agreements

Service Provider

Administrator | Policies and Best Practices

Automated Change Workflow

- **SPECjAppServer provisioning workflow**
  - Generated by CHAMPS system
  - BPEL language
  - Executable in workflow engine
Complexity Comparison: Provisioning SPECjAppServer

- **Manual configuration:**
  - Execution
    - 59 steps, 27 context switches
  - Parameter
    - 32 parameters used 61 times, 18 outside of source context
    - Source score: 125
  - Memory (LIFO stack model)
    - Size: max 8, avg 4.4

- **Automated provisioning (CHAMPS):**
  - Execution
    - 5 steps, 1 context switch
  - Parameter
    - 17 parameters used 17 times, 0 outside of source context
    - Source score: 94
  - Memory
    - Size: max 0, avg 0

**CHAMPS significantly reduces complexity across-the-board**
Outline

- Introduction
- Quantifying Configuration Complexity
  - Methodology
  - Case Study
- Next Steps & Conclusions
Next Step: Interpreting the Metrics

- **Raw metrics require interpretation**
  - E.g., how does a 10% reduction in memory complexity affect configuration time? error rate?

- **Mapping to be achieved via user study** (planned)
  - Measure time and error rate for procedure variants with different complexity levels
  - Synthetic web-based configuration environment based on PO system
Next Step: Design Complexity

- **Design complexity captures difficulty of figuring out how to achieve a systems management goal**
  - Reducing design complexity:
    - Improves consumability of systems management technology
    - Reduces labor cost and required skill

- **Design complexity arises from decisions in a configuration procedure**

- **Possible aspects to quantify:**
  - Decisions along a single path
  - Overall complexity over all paths in design space
  - Tradeoffs between simplicity and flexibility
Conclusions

“Benchmarks shape a field, for better or worse.”
— Dave Patterson

- Complexity of configuration procedures drives labor costs for systems management
  - Complexity is a key challenge for systems management technology
- Quantification is essential if we are to effectively reduce complexity
  - And virtually no prior art for quantifying configuration complexity
- We have developed an initial approach to analytic modeling of configuration complexity
  - Based on analysis of configuration procedures, not user studies
  - System-independent
  - Focused on expert users
- Case study of CHAMPS shows the potential value of the approach
- But much fundamental work remains – calibration, design complexity, ...
  - A fertile new research domain in modeling & quantifying human impacts of system management technology