Interactive models as a system design tool: Applications to system project management

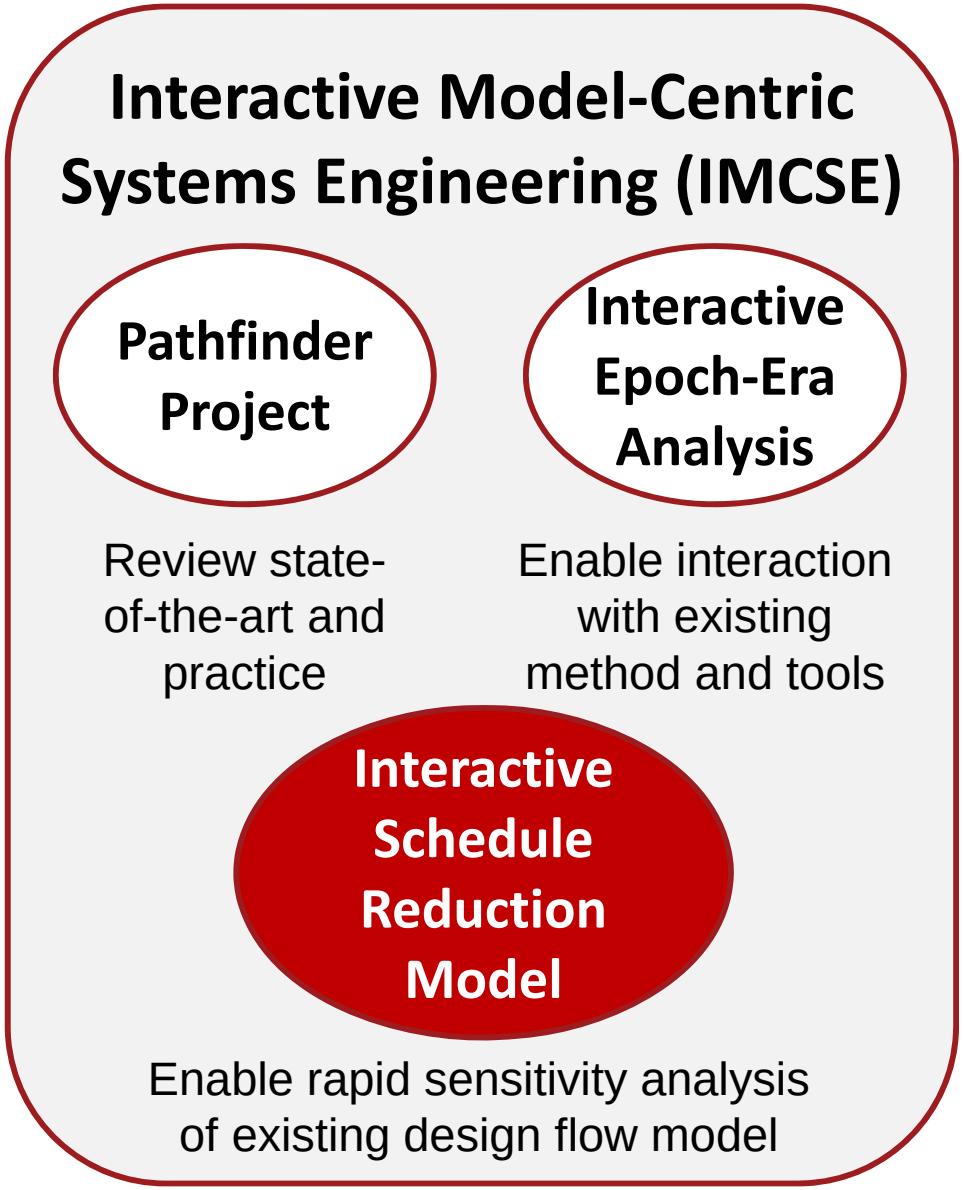
By
Paul T. Grogan*, Olivier L. de Weck, Adam M. Ross, and Donna H. Rhodes
13th Annual Conference on Systems Engineering Research (CSER)

March 18, 2015

Stevens Institute of Technology
Hoboken, NJ

www.stevens.edu/sse/CSER2015org

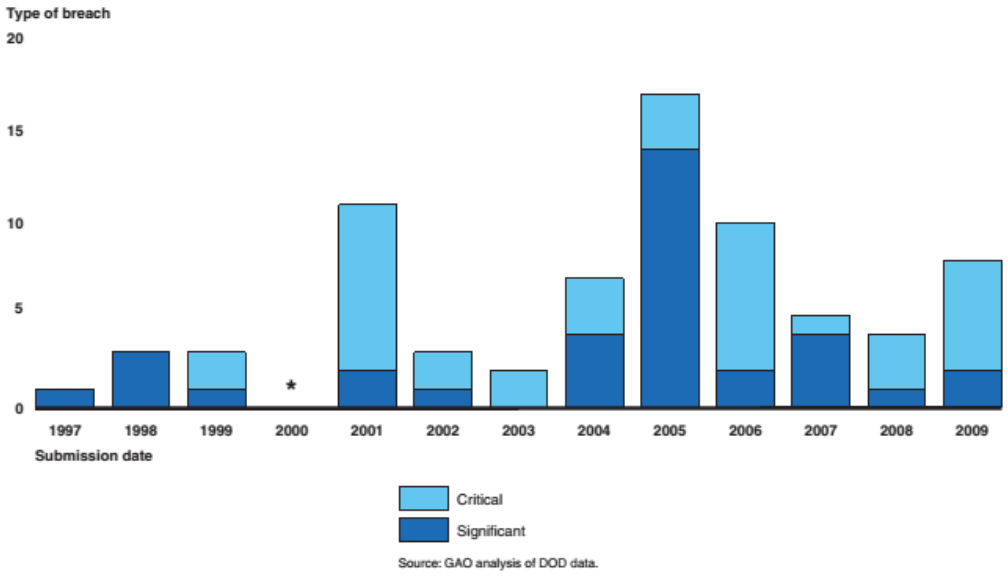
Overview



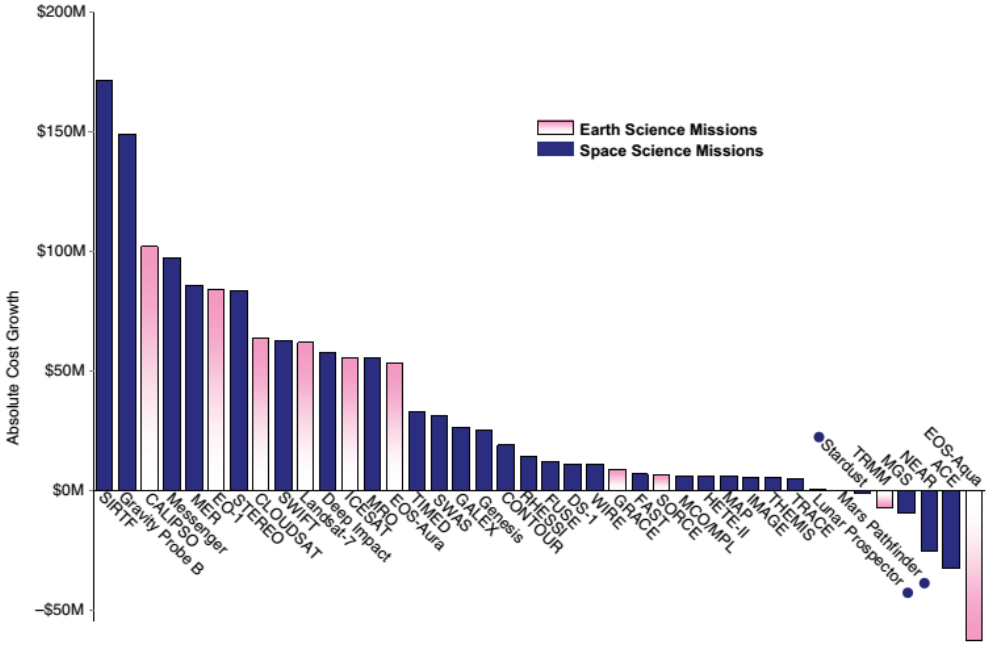
- Problem Statement
- Theoretical Framing
- Applied Objectives
- Prototype Design Tools
  - Standalone Tool
  - Service-based Application
- Conclusion

Cost and Schedule Overruns

- Major DoD acquisition programs (GAO, 2011)
  - —74 cost breaches in 47 of 134 major acquisition programs
  - —Sources: engineering/design, schedule, quantity changes



- NASA Earth and space missions (NRC, 2010)
  - —Excessive cost growth in 14 of 40 recent missions, avg. ~20%
  - —Sources: unrealistic estimates, instability, development issues



What drives effort overruns and how can it be avoided?

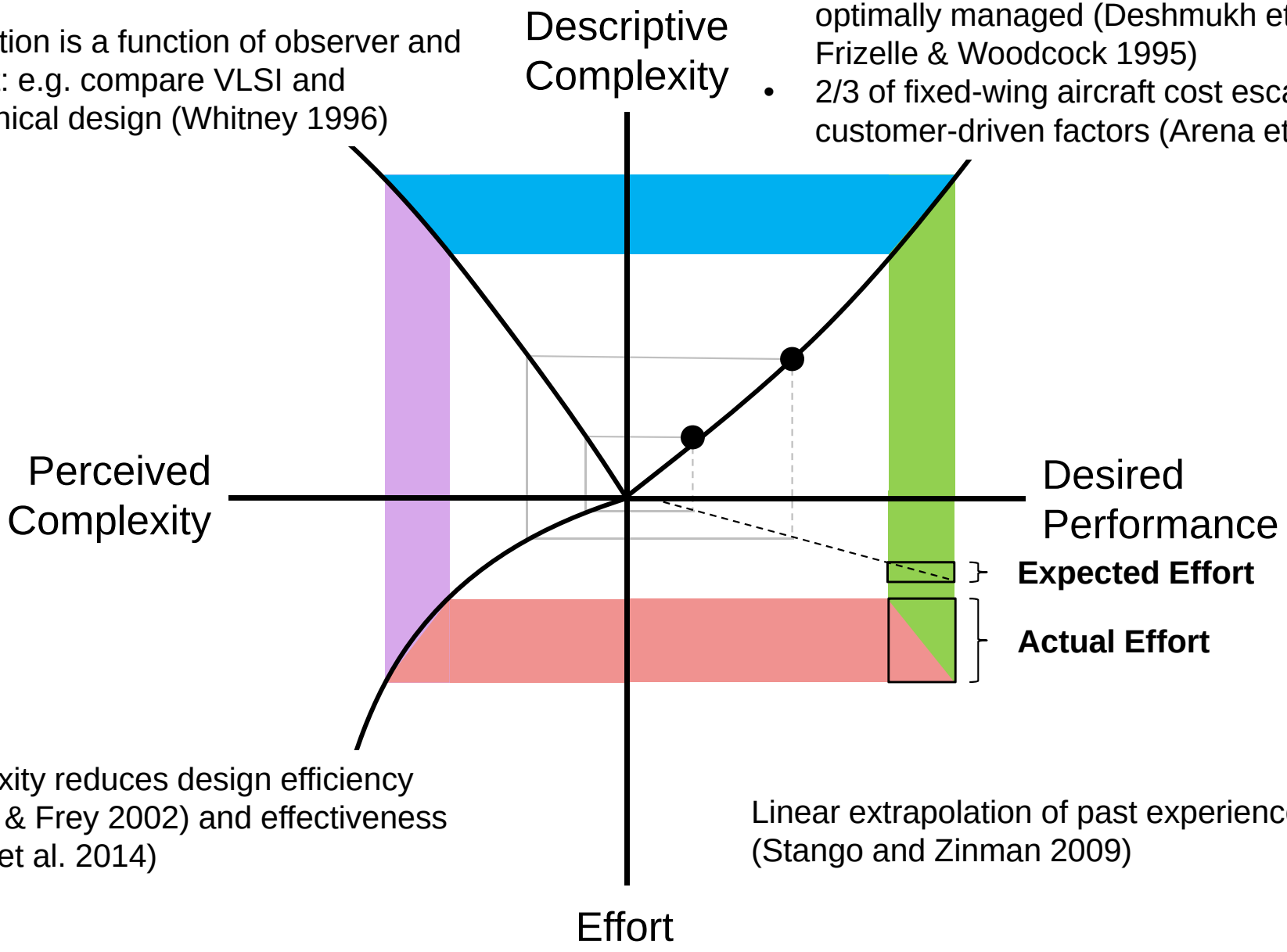
Complexity and Effort Overruns

- Misinformation causes overruns (Flyvbjerg et al. 2009)
  - Internal (delusion): optimistic estimates of cost and schedule
  - External (deception): strategic misrepresentation in contracting
- Complexity: uncertainty in meeting functional requirements (Suh 1999) within cost/schedule constraints
- Descriptive and perceived complexity (Schlindwein & Ison 2004)
  - Descriptive: objective measure of information content
  - Perceived: subjective measure of uncertainty in knowledge

Explanatory Model of Complexity in Design

- Complexity can improve performance if optimally managed (Deshmukh et al. 1998, Frizelle & Woodcock 1995)
- 2/3 of fixed-wing aircraft cost escalation from customer-driven factors (Arena et al. 2010)

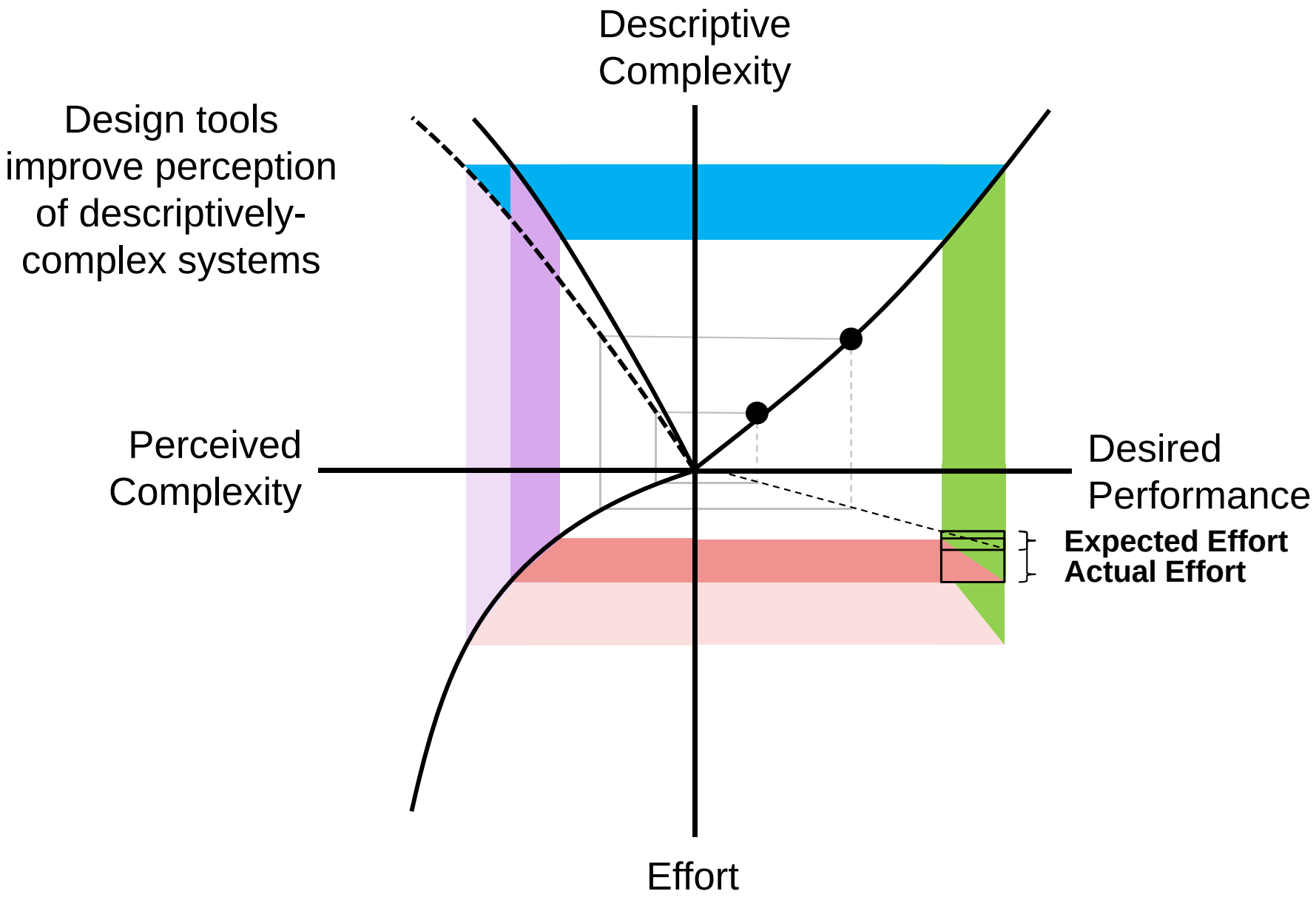
Perception is a function of observer and context: e.g. compare VLSI and mechanical design (Whitney 1996)



Complexity reduces design efficiency (Hirschi & Frey 2002) and effectiveness (Flager et al. 2014)

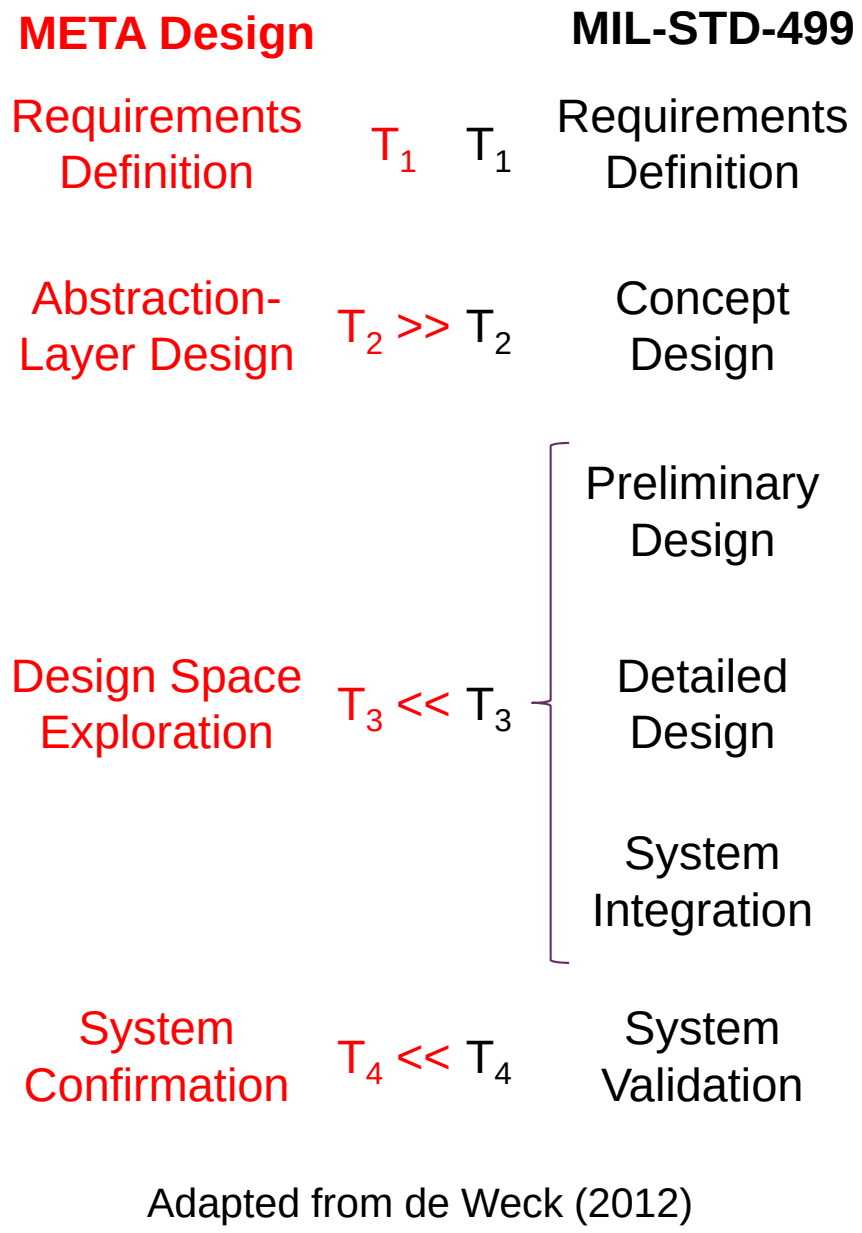
Linear extrapolation of past experiences (Stango and Zinman 2009)

Explanatory Model of Complexity in Design



SE Design Tools

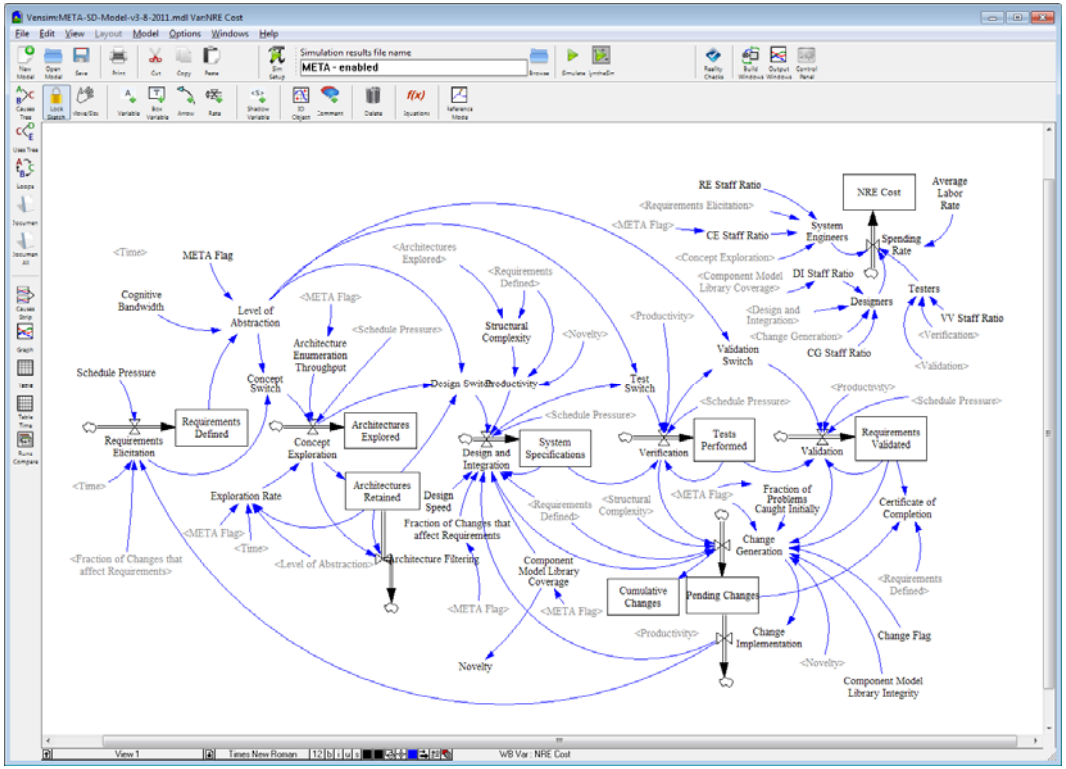
- Develop new methods and tools for systems engineering
  - —Improve perception of complexity
  - —Focus on early design stages
  - —Leverage computation and memory of software systems
- META II Complex Systems Design and Analysis (CODA) (Murray et al. 2011)
  - —Design at multiple layers of abstraction (coarse-to-fine)
  - —Develop and use a trusted component model library
  - —Virtual re-design cycles



Adapted from de Weck (2012)

Design Flow Model

- Project management as a secondary design problem
  - —Design Flow Model (de Weck 2012)
  - —Evaluate feasibility of 5X speedup in META development time
  - —Assess alternative resource allocations
- Design flow activities:
  - —Requirements elicitation
  - —Concept exploration and filtering
  - —Design and integration
  - —Verification and validation
  - —Change generation and implementation



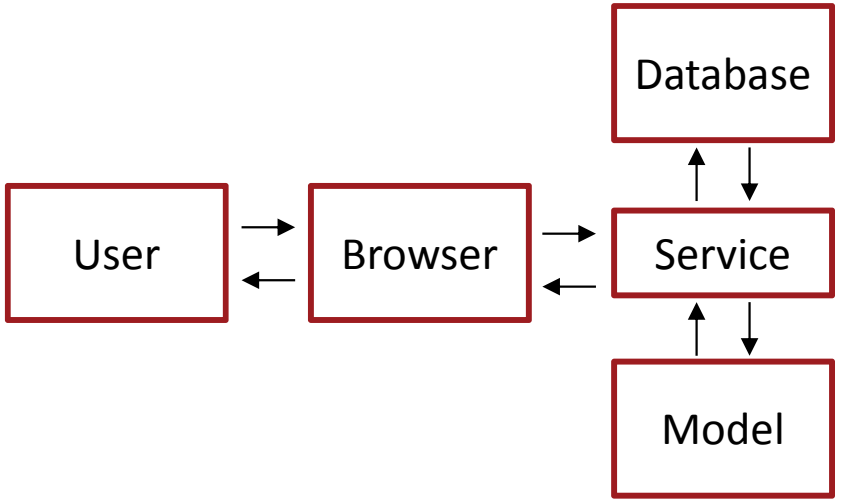
- Initial DFM results:
  - —Optimistic: 42.25 mo., $27.9M
  - —Realistic: 70 mo., $51.9M
  - —META: 15.75 mo., $31.5M

Project Objectives and Approach

- Extend DFM with rapid sensitivity analysis of factors
- Limited data aggregation and visualization in Vensim
  - —Closed source, proprietary
  - —Command line use licensed
  - —DLL integration licensed
- Develop browser-based methods to replicate and extend Vensim capabilities
- Application case for broader modeling methods and tools

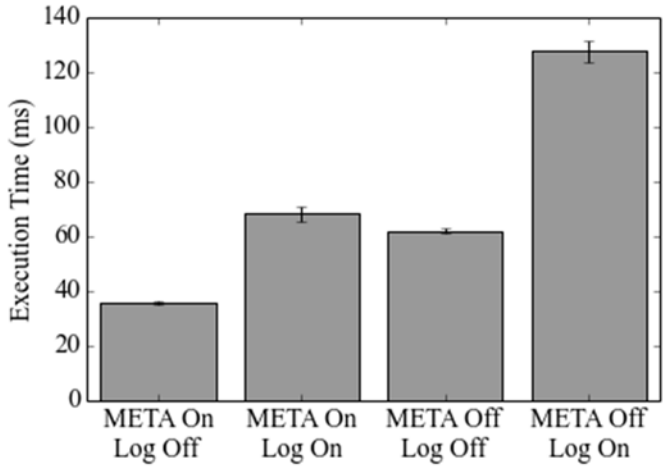
1. Standalone tool
   - —Execute and visualize single model runs via web browser

2. Service-based application
   - —Compose and query multiple model runs via HTTP services

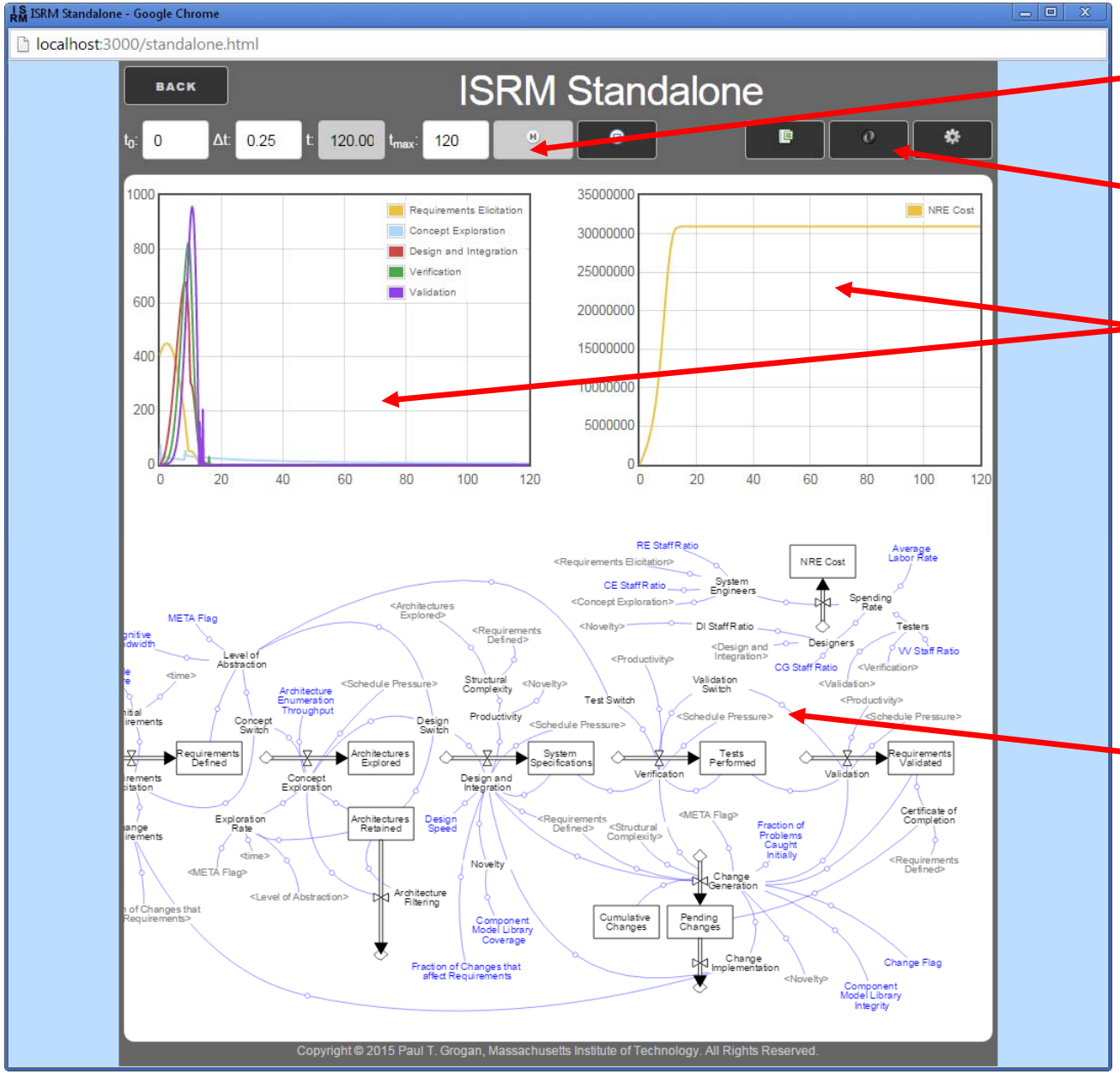


JavaScript Modeling & Simulation

- Few existing computational libraries for JavaScript
- New JavaScript Modeling and Simulation (MAS) library/API
  - —Implements a portion of the system dynamics (SD) formalism
  - —Stocks, flows, parameters
  - —Delay, smoothing functions
  - —Simulators and logging facilities
- Validated with classic SD cases
  - —Predator-prey model
  - —Diffusion of innovation model
- JavaScript port of DFM
  - —25 stocks and 25 flows
  - —~650 lines with documentation
- Validated with Vensim results
  - —Small differences due to numerical precision
  - —~100 ms execution time on consumer hardware



Standalone Tool



Simulation settings

Data export (CSV, JSON)

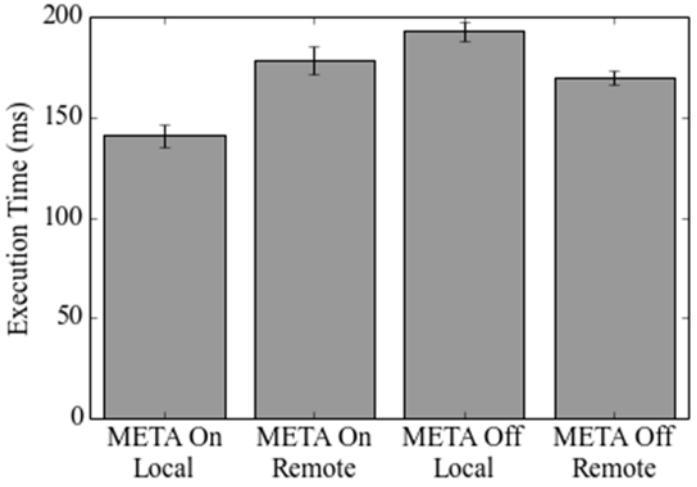
Customizable Plots

Interactive Stock and Flow Diagram

Runs completely in a browser with no external dependencies

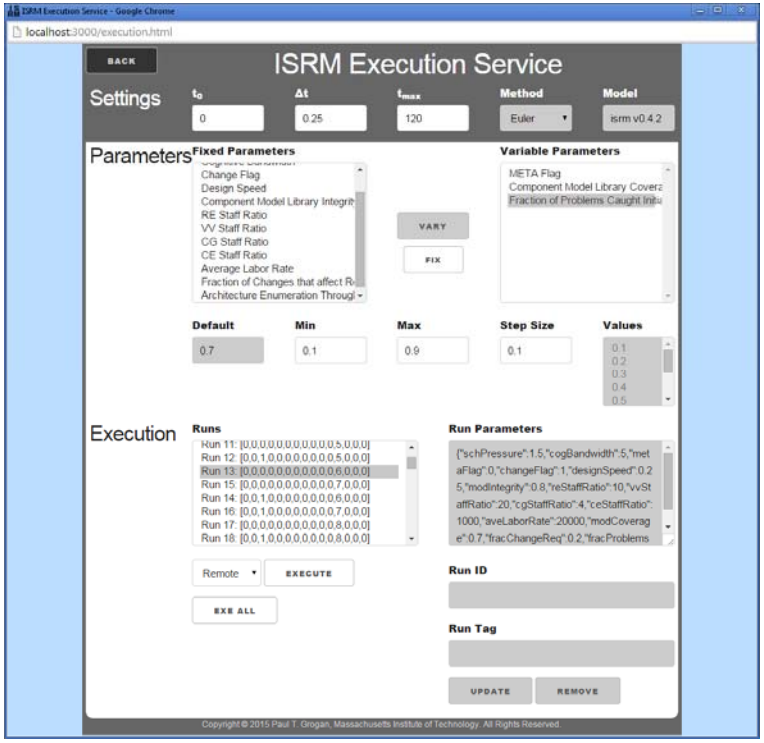
Service-based Application

- HTTP Data Services API
  - GET /results/:id Query individual result
  - GET /data Query all results
  - POST /results Submit execution results
  - POST /execute Remote model execution

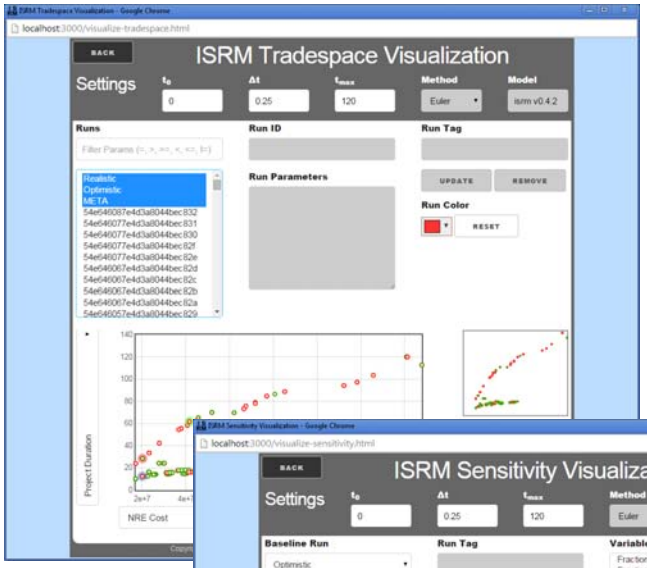


- GET services filterable by data structure
  - GET /results/54de7aae9565b5eeced138c9/settings
  - GET /data/outputs/nreCost
  - GET /data/finalOutputs/nreCost?params.changeFlag=0&params.metaFlag=0
- Implementation: Node.js/Express web server, MongoDB database
- JavaScript is a common language on client (browser), server (Node.js), and database (JSON documents)

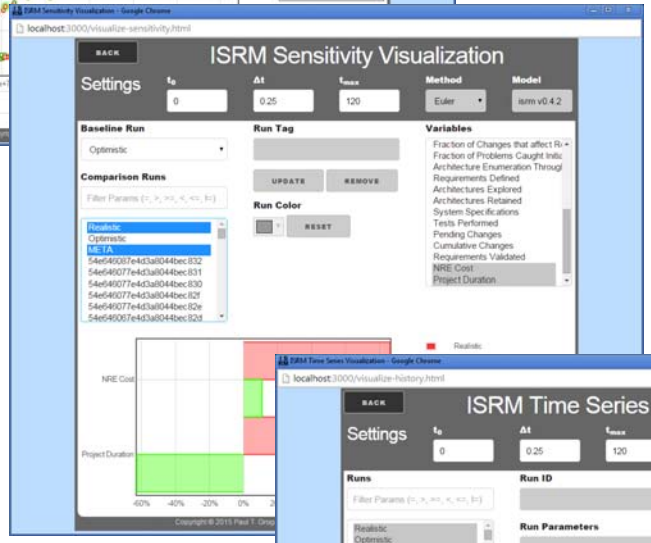
Service-based Application User Interfaces



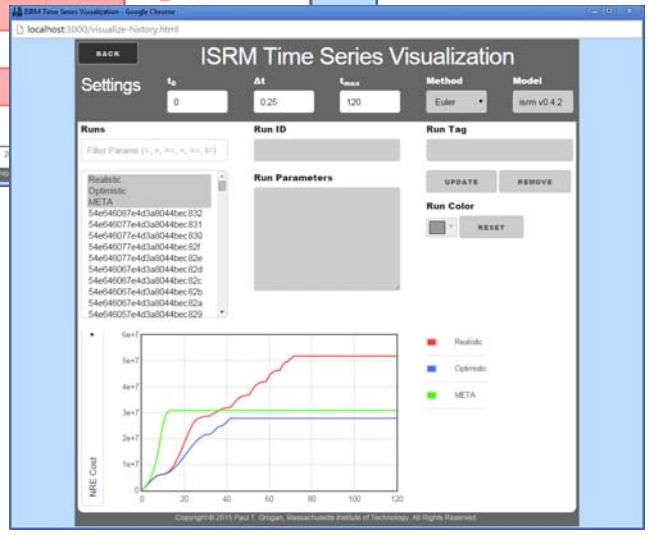
Run full-factorial design-of-experiments in batch mode by varying parameters of interest



Visualize a complete 2-D tradespace of all model results

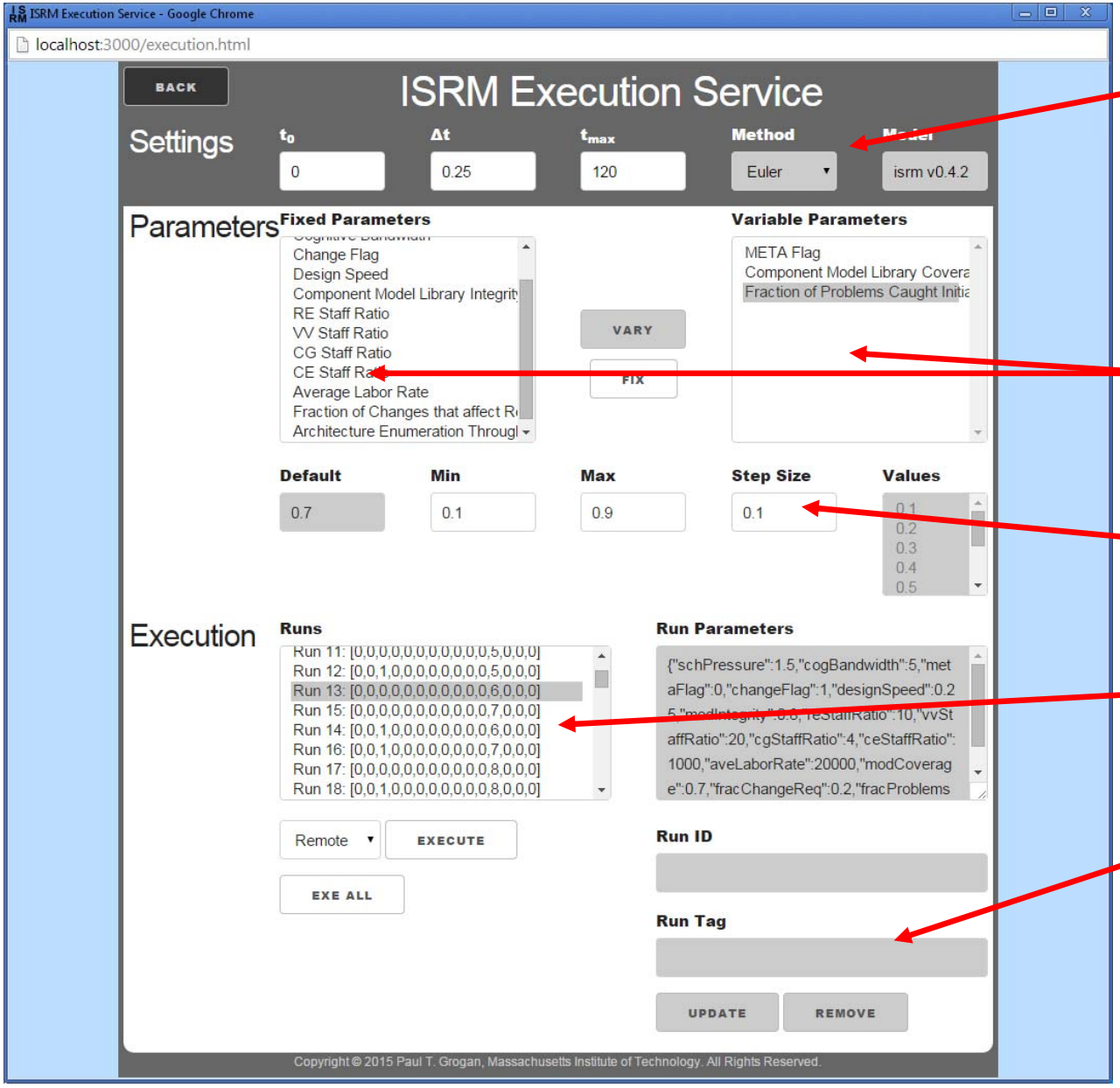


Visualize sensitivity of results to various factors



Visualize time series data across any model results

Batch Execution Tool



Simulation settings

Lists of fixed vs. variable parameters

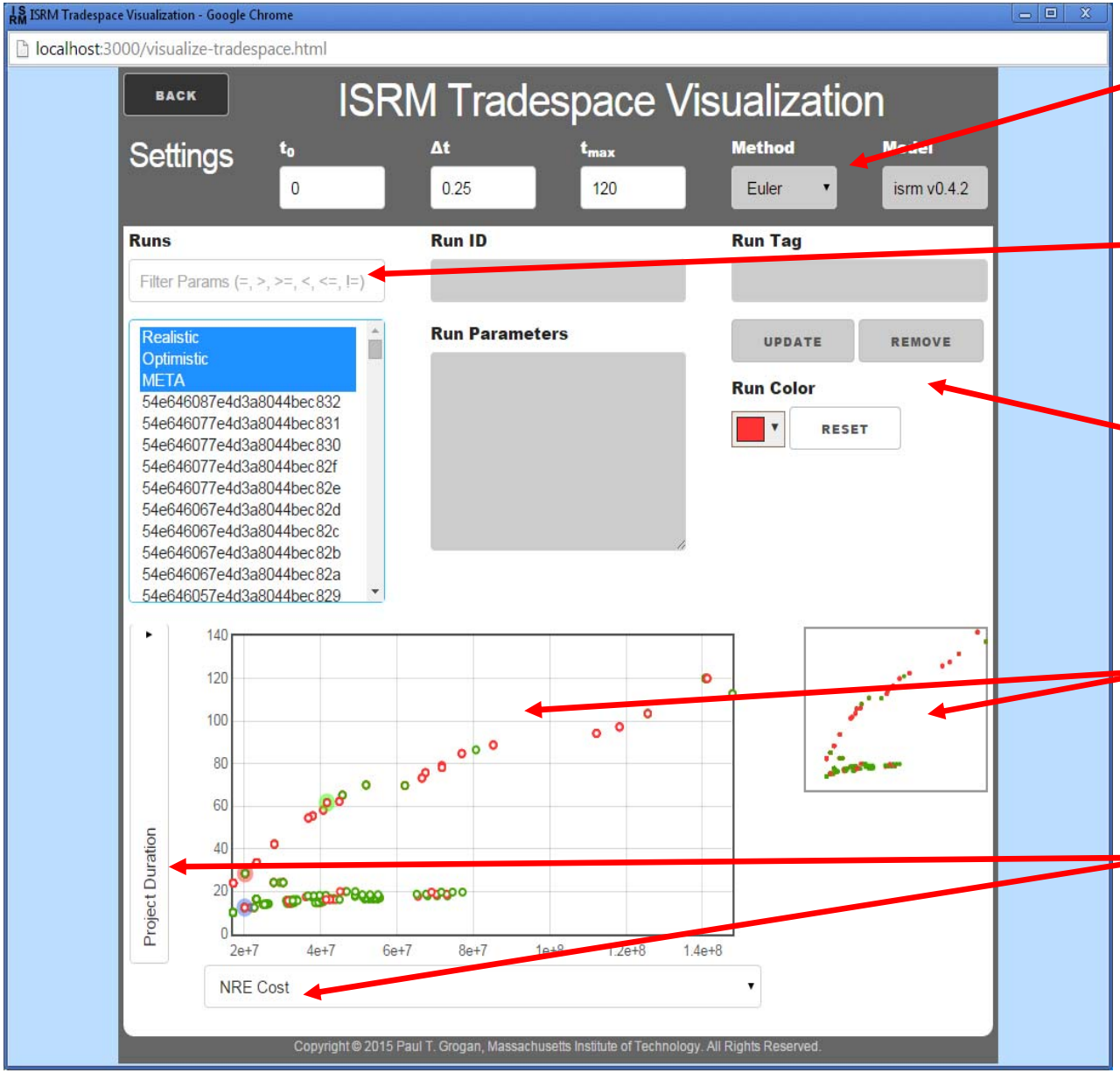
Select parameter values

Generate full-factorial experiment and execute (local or remote)

Customize tags

Requires access to data services (local OK)

Tradespace Visualization Tool



Simulation settings

Select results to display

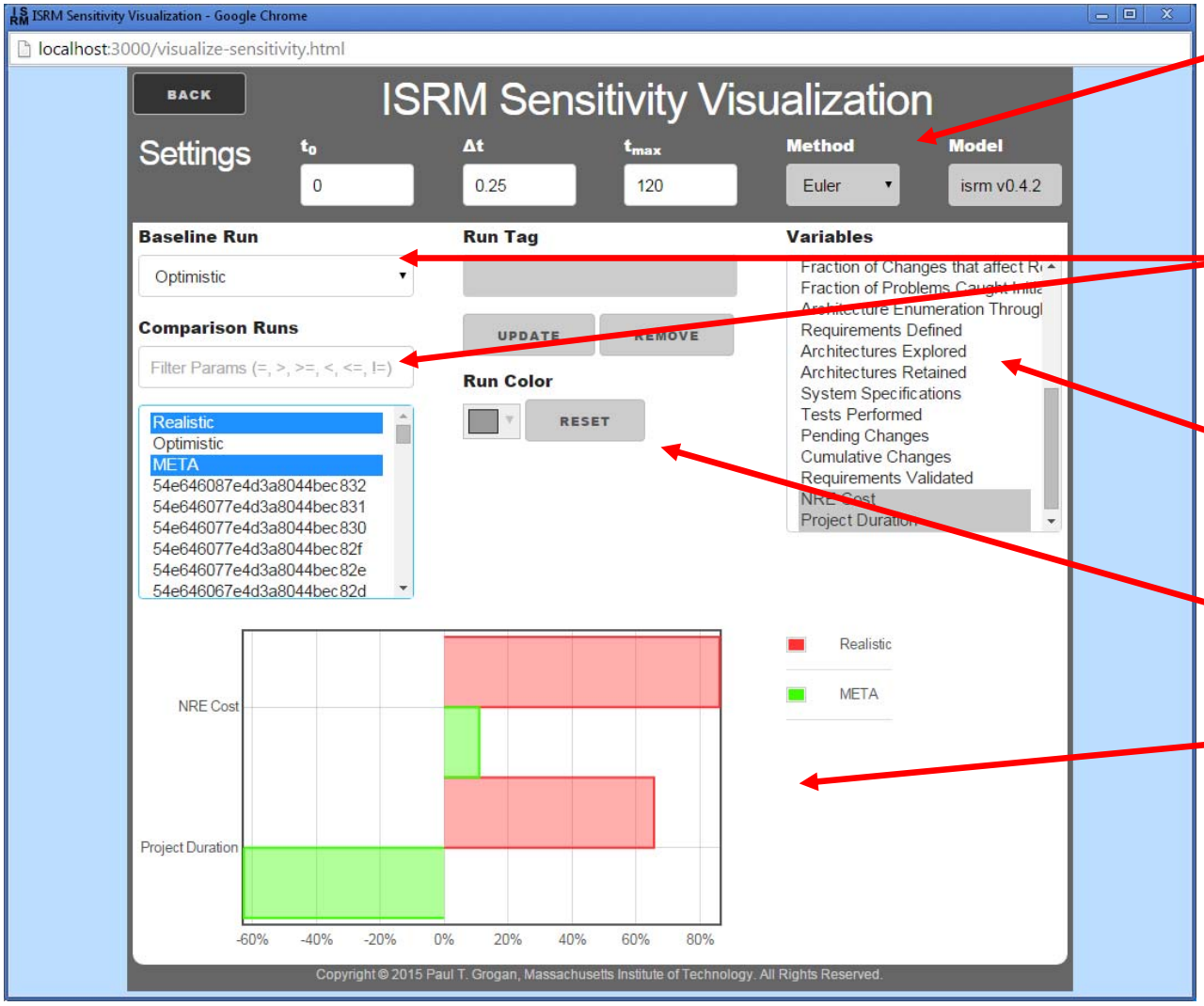
Customize tag and color

Tradespace plot (with overview)

Choose display dimensions

Requires access to data services (local OK)

Sensitivity Analysis Tool



Simulation settings

Select baseline and comparison results

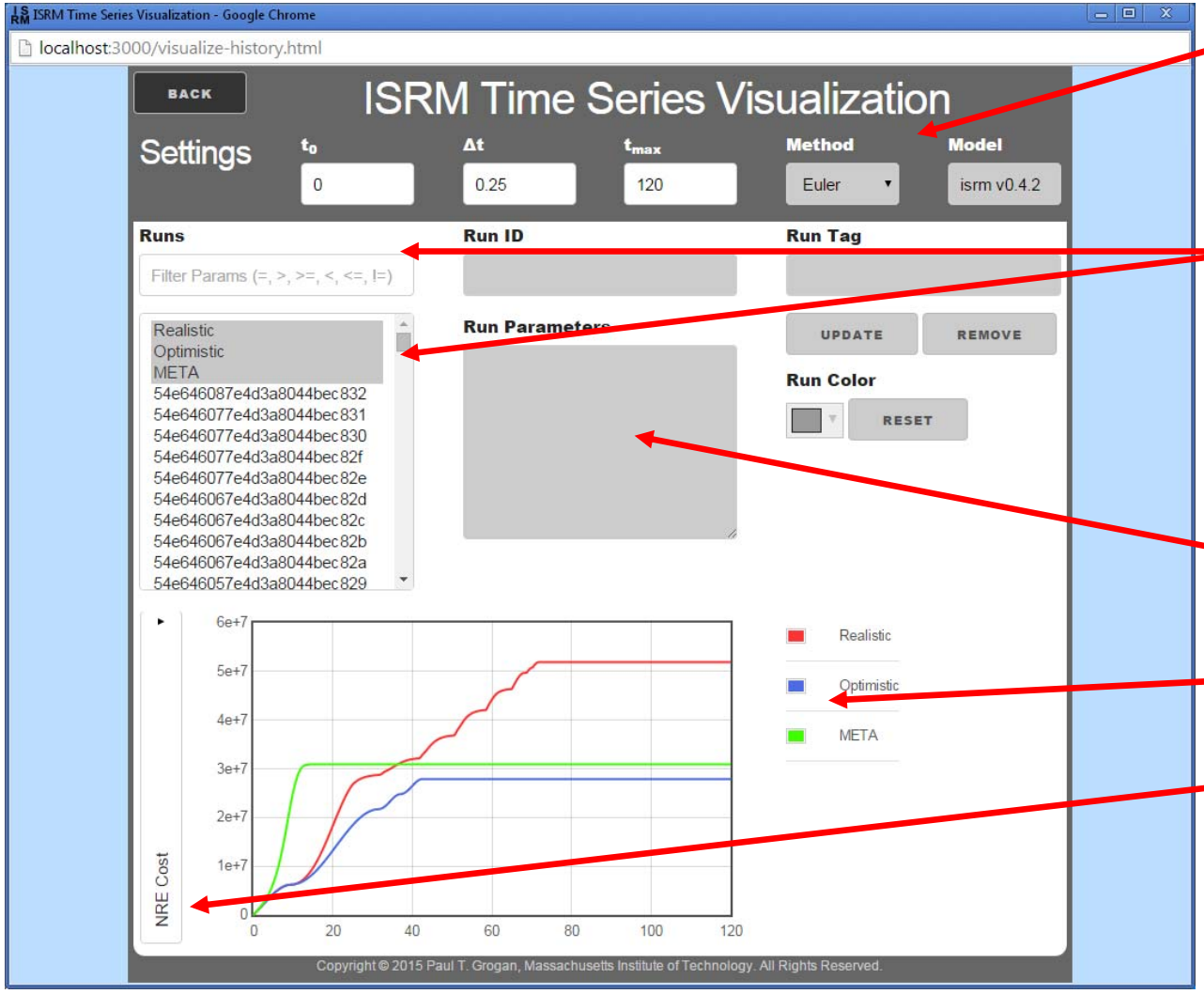
Choose display variables

Customize tag and color

Sensitivity diagram

Requires access to data services (local OK)

Time Series Tool



Simulation settings

Select baseline and comparison results

Customize tag and color

Sensitivity diagram

Choose display variable

Requires access to data services (local OK)

Conclusion

- ISRM is a case study which demonstrates:
  - Browser-based tool for model execution and visualization
  - Service-based application to aggregate and query model runs
  - Open source components; everything can be run locally
- Browser-based platforms are viable for modeling activities:
  - Low barrier to entry; accessible
  - High quality UI libraries
  - Limited computational libraries
- Future work:
  - Revisit DFM assumptions to assess project mgmt. decisions
  - Consider multi-user environments and network access security
  - Optimize browser performance
  - Extend MAS library to new formalisms and use cases
- Source code available:
  - ISRM: github.com/ptgrogan/isrm
  - MAS: github.com/ptgrogan/mas
- Node.js package available:
  - MAS: npmjs.com/package/mas

Thank you

References are included in the full paper. We acknowledge the many contributors to open software libraries including:

- JavaScript UI libraries: JQuery, JQueryUI, Flot, Kinetic.js
- JavaScript platform: Node.js, Express, RequireJS
- Document-based database: MongoDB, Mongoskin

This work is supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract HQ0034-13-D-0004. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.