Software Development Risk Analysis (SDRA)

Final Brief

Derek Eichin
Operations Research Analyst
ODASA-CE, NISEC

Trevor Jaskot
Eric Jones
Leo Pacatan

Executive Summary

Bottom Line Up Front

ODASA-CE NISEC does not have sufficient methods for validating Program Office Estimates for cost and schedule required by software development projects. This project delivers documented baseline analytic model and accessible platform

Background	• Cost and schedule overruns cause delay in capability delivery o $402 billion in cost overrun, average 22 months schedule overrun per project[1] o 92 of 98 completed major defense acquisition programs overran and 12 cancelled in one year (FY2010)[1] • Strategic and operational impacts on contingency response and deterrence
Approach	• Discrete event simulation model • Accessible across platforms
Results	• Software platform accessible through access controlled website • Analytic model to assess cost and schedule of software development projects • Next steps to expand on stochastic considerations and integration into platform

1. Hofbauer, 2

Agenda

- Project Overview
- Products
- Insights and Recommendations
- Conclusion

Project Overview

- Background
- Stakeholders
- Objective
- Analytic Process

Background

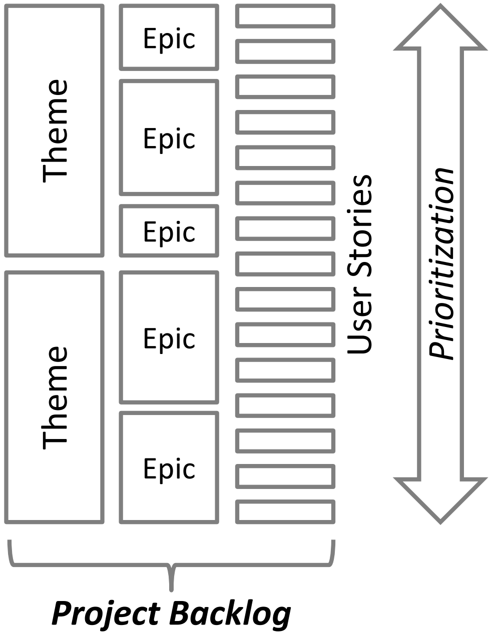
- **Problem.** Software development programs in the Department of Defense are plagued with cost and schedule overruns that drive delay in capability delivery
- **Impact.** Downtraces to delayed integration and deployment to operational forces, strategic and operational implications for contingency response and deterrence
  - In FY 2010, 98 Major Defense Acquisition Programs (MDAPs) collectively overran cost by $402 billion and schedule by an average of 22 months each[1]
  - Delays in 92 of 98 MDAPs and 12 cancelled programs in one year[1]
- What has been done?
  - ***Transition to Agile Software development***
  - Establishment of Defense Innovation Board in 2016
  - Predictive Regression Models
  - Varied Studies
    - Qualitative factors that improve software development cost estimation
    - COCOMO II – Constructive Cost model
    - ***Agile Project Dynamics from MIT***

1. Hofbauer, 2

Agile Software Development

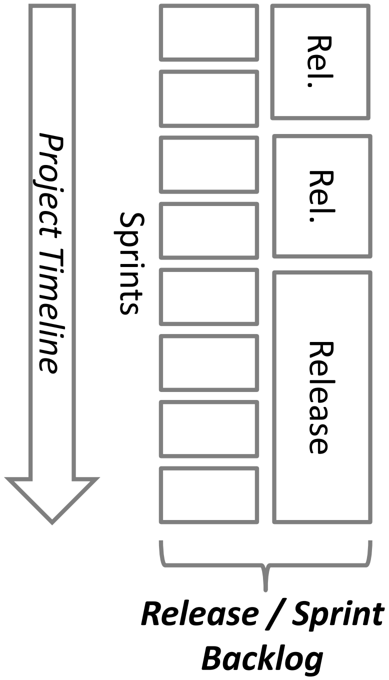
Software Elements

- Client requirements
- Breakdown features



Program Management

- Client priorities
- Scheduling
- Project Control



Client Reviews / Deliverables

- Regular interaction
- Functioning elements

- User stories are characterized by story points to represent work effort required
- Project managers estimate the number of story points that can be completed in a sprint based on staff level and experience
- Founded on regular client involvement to review and guide functional, interim products

Stakeholder Analysis

Validate | *Plan* | *Execute* | *Deliver* | *Operate*

Project Impact

NISEC (Sponsor)

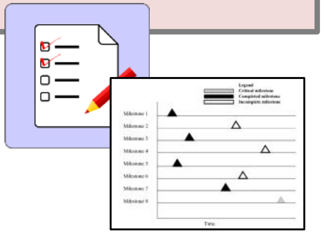
Analyze, Validate, Recommendations

← Submit Plans & Req.

Review & Advise →

Program Management Office

Plan, Build, Develop, Deliver, Sustain, Retire



Deliver & Field →

← Performance

Operational Units

Field, Integrate, Train, Deploy

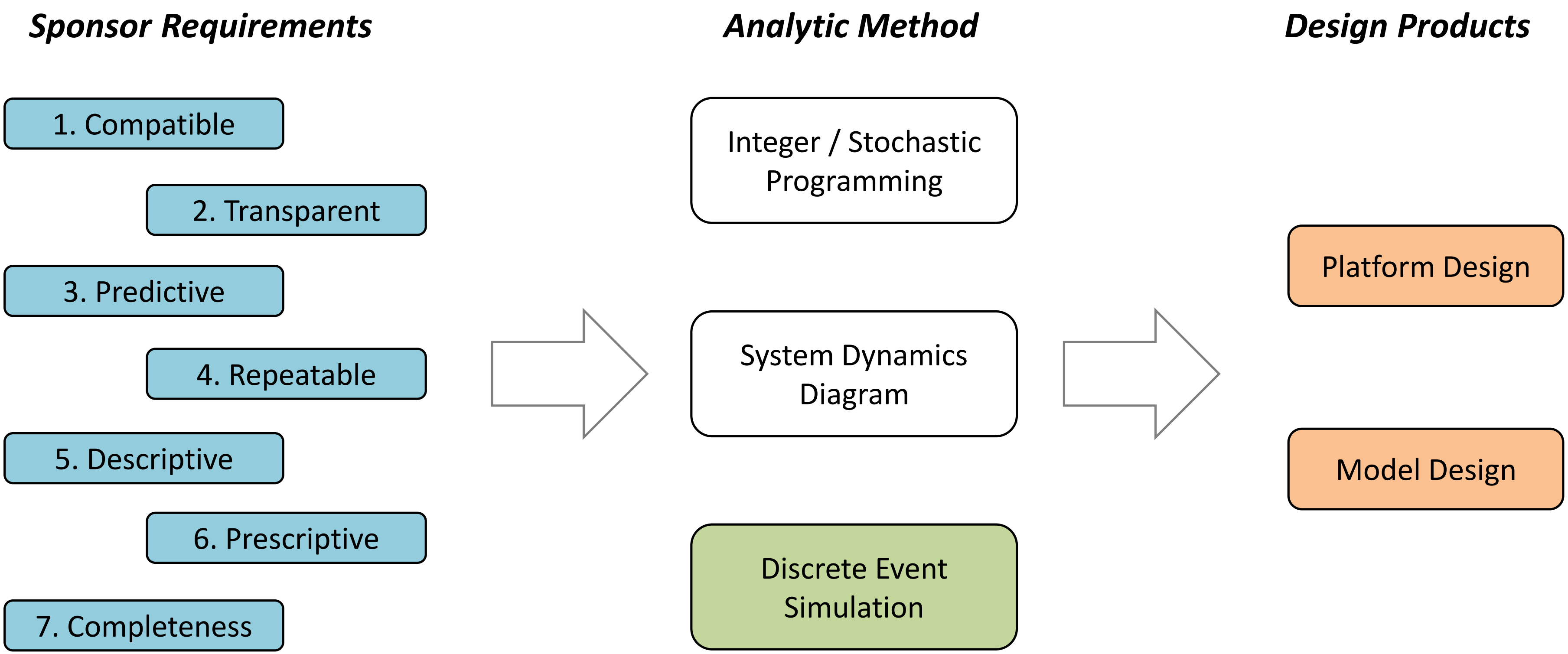
Objective

Develop, document, and deliver baseline model of Department of the Army (DA) agile software development teams for validating cost and schedule estimates.

- For use by ODASA-CE operations research analysts
- On an accessible platform and prepared for further development (beyond baseline process)

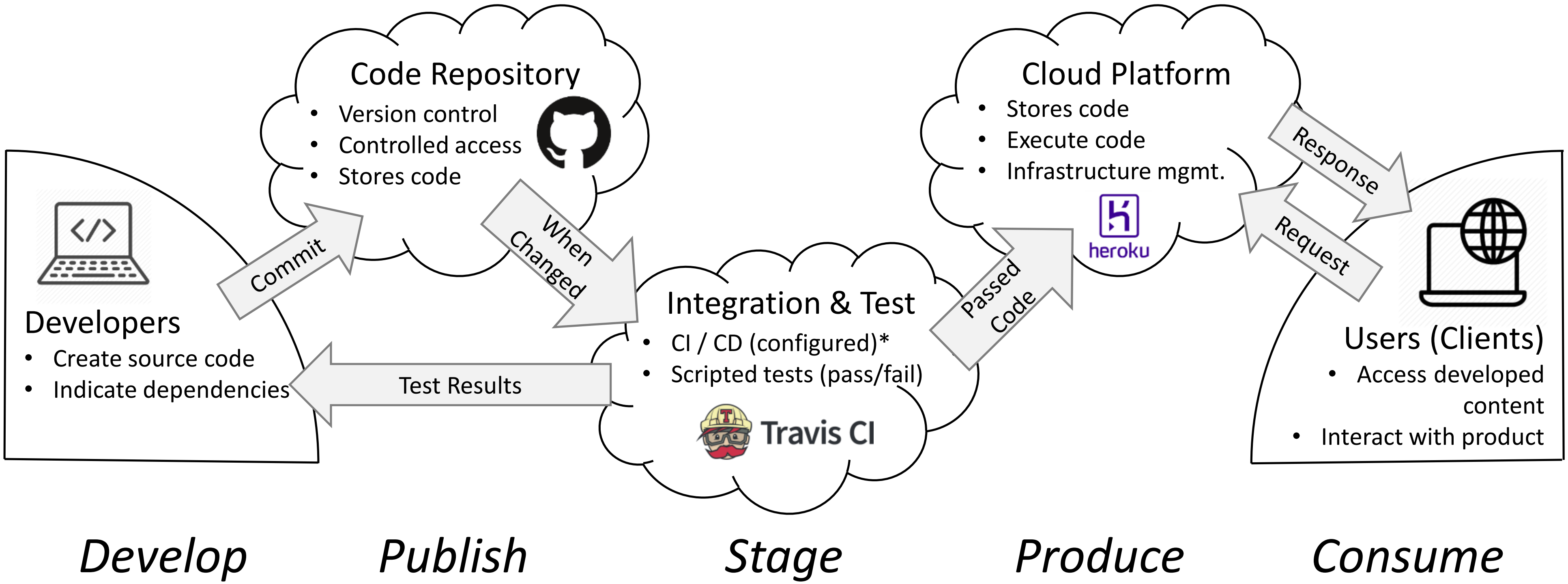
- Desired End State
  - Common understanding of software development teams and their processes
  - Baseline platform for hosting the delivered model
  - Baseline analytic model for evaluating cost and schedule
  - Identified path forward for development and implementation

Analytic Approach



Products

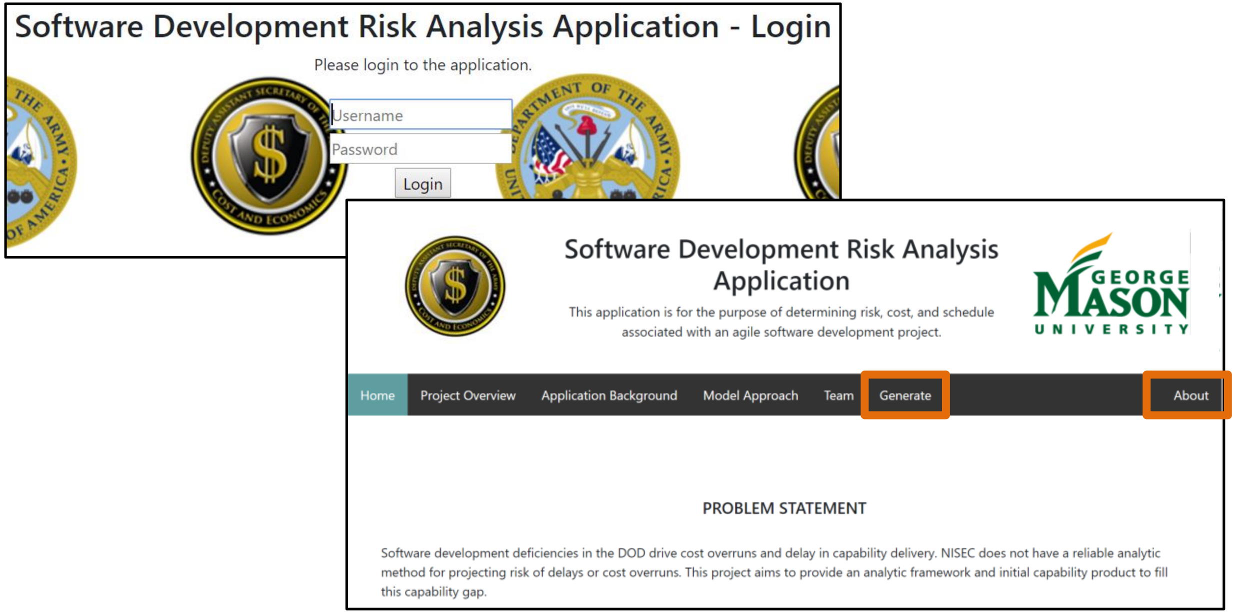
Software Platform Architecture



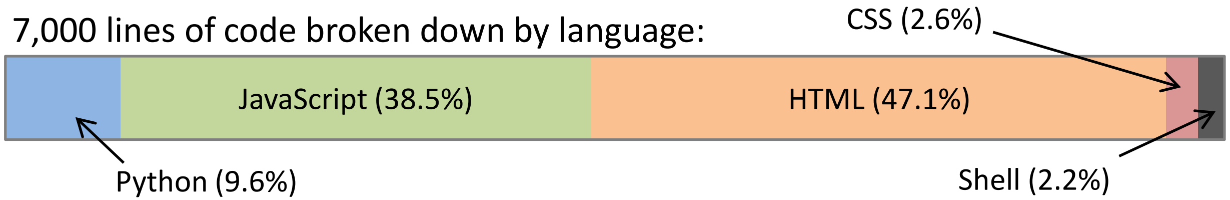
* Continuous Integration / Continuous Delivery

Platform Design

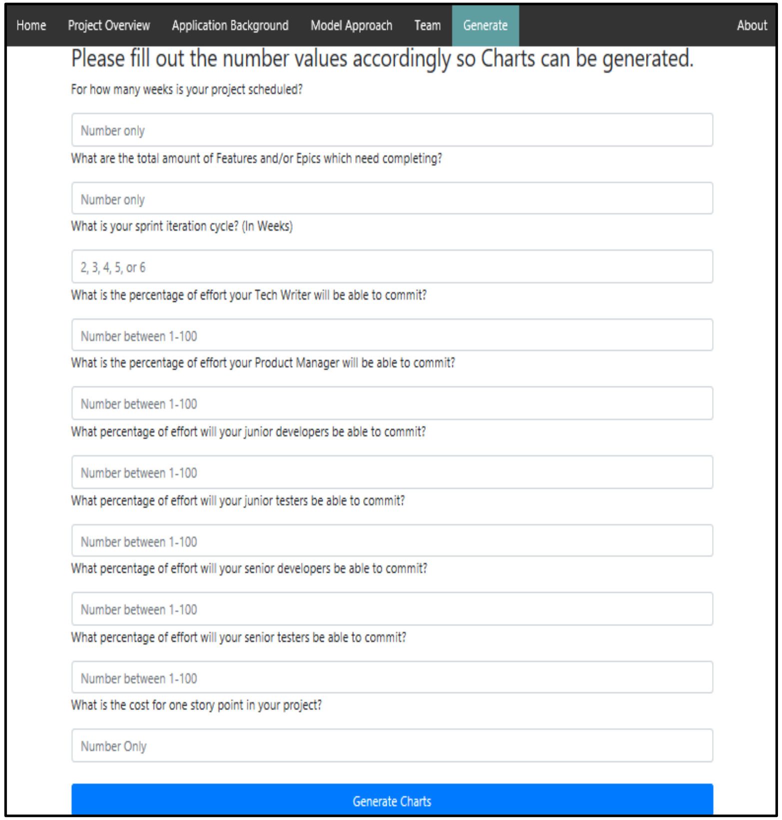
- Web access controlled through username and password
- Project information co-located with analytic model (documentation of assumptions)
- 'Generate' tab designed to host analytic model
- 'About' tab documents path forward and developer notes

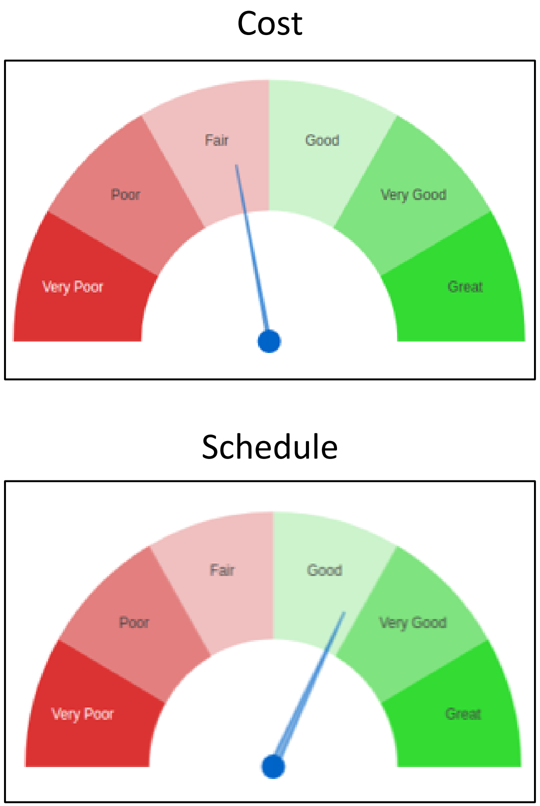


7,000 lines of code broken down by language:



Example Model (current on website)



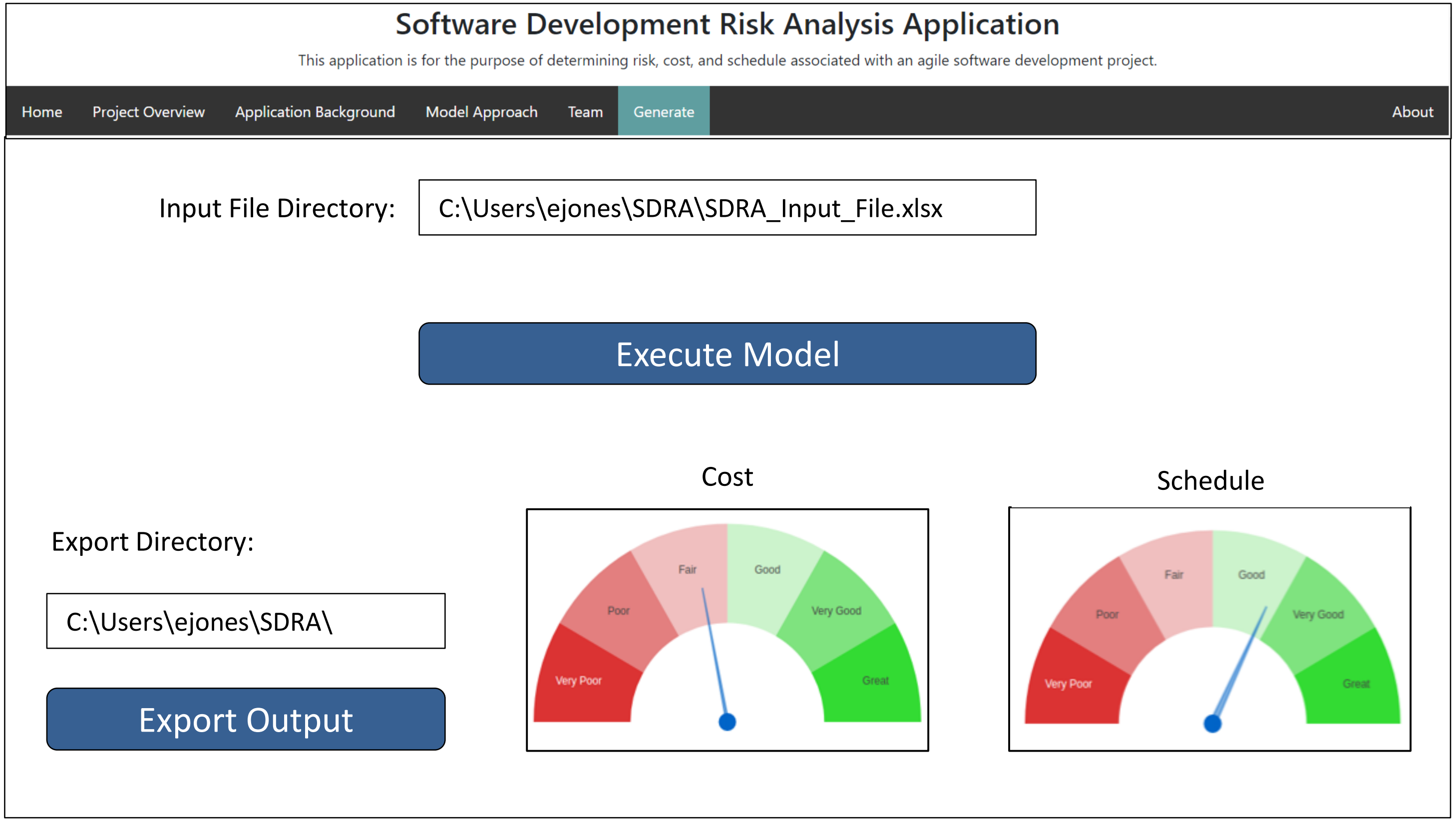


Analytic Model (to be integrated)

Step 1: Define Input

Step 2: Execute Model

Step 3: Analyze Output



Step 1: Define Input

- Model input is a *.xlsx file named ‘SDRA_Input_File’ that is located in accessible directory
  - Global inputs
  - Project backlog
  - Staff

Global Inputs

- Sprint length
- POE for project cost
- POE for project makespan
- Probability of identifying defects (in sprint or after release)

Project Backlog

- Themes (ID, name)
- Epics (ID, name, parent, release, predecessors)
- Stories (ID, name, parent, predecessors, category, divisible, story points)

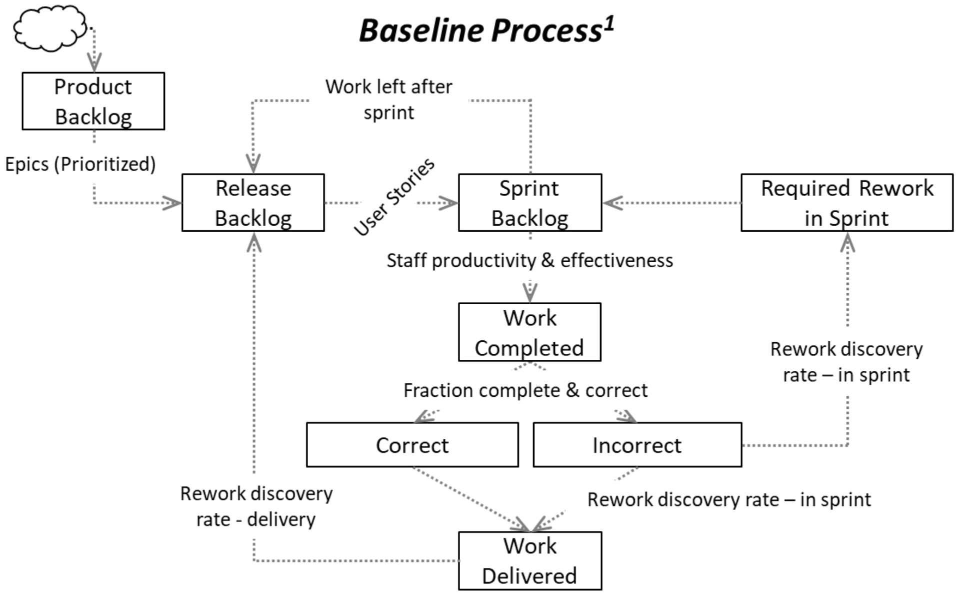
Staff

- ID, Name, Category, Cost
- Planned velocity per sprint
- Distribution of actual velocity
- Reliability factor

Step 2: Execute Model

- Analytic model simulates baseline process
- Discrete event simulation with each step the length of one sprint
  - If beginning of release, load all user stories from assigned epics to release backlog
  - Assign tasks to staff members
  - Stochastic completion of tasks
    - Correct and incorrect work separated
  - If discovered, incorrect work initiates additional tasking in backlog
  - Record model state
    - If release complete, go to next release
    - If project complete, end and output data
  - Go to next sprint

Baseline Process[1]

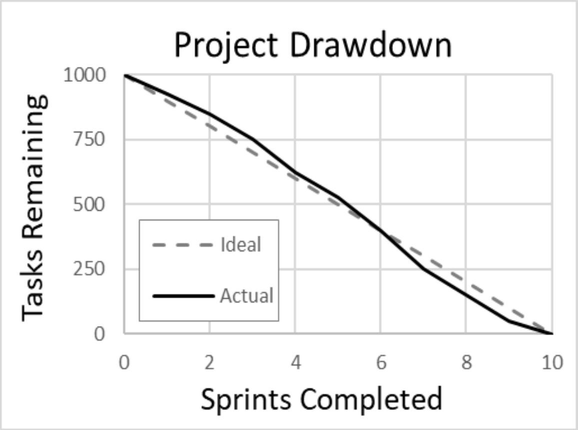


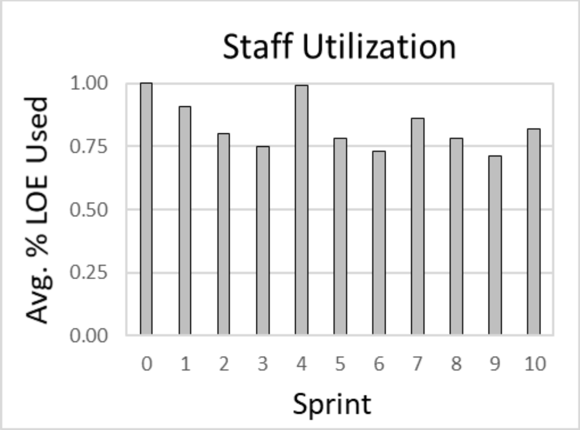
Six Major Agile Functions[1]

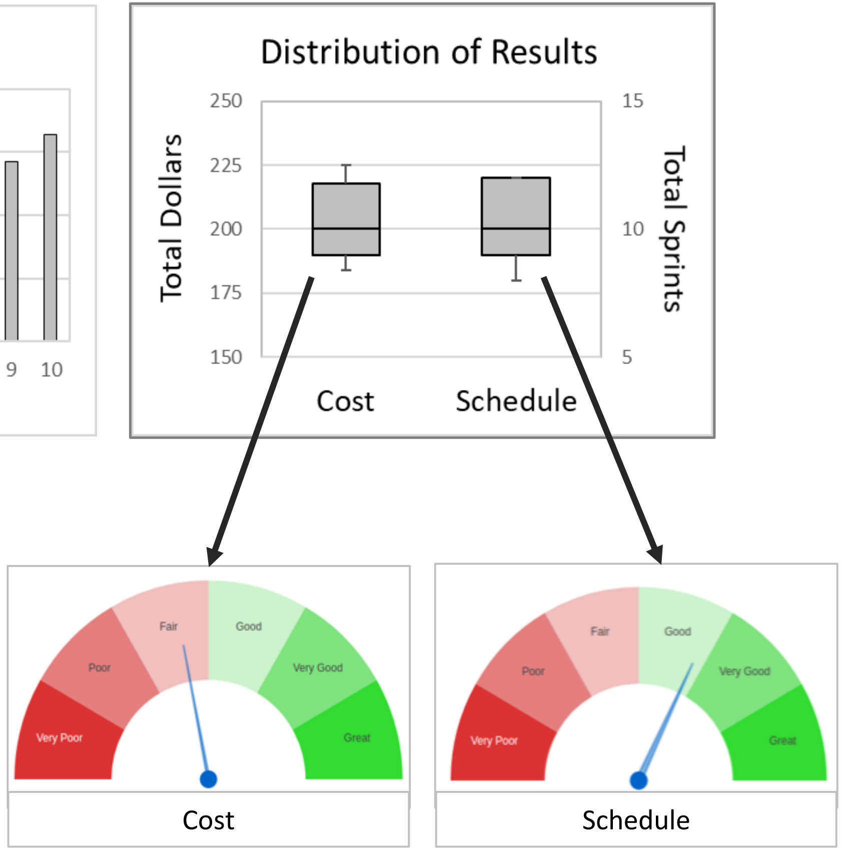
✓ Feature driven processing	○ Micro-optimizing
✓ Iterative, incremental delivery	○ Customer involvement
○ Refactoring	○ Team dynamics

1. Glaiel, 9-11

Step 3: Analyze Output







- Model provides output as MS Excel output file and high-level assessments in model
  - Project progress
  - Tasking
  - Defects
- Distributions of cost and schedule feed red-green assessment, displayed on website

Insights and Recommendations

- Insights and Recommendations
- Path forward
- Conclusion

Insights and Recommendations

- Predicting employee performance using a model is risky
  - Hard to tune to specific team dynamics (predictive modeling)
  - No data to support estimation of abstract variables like percent of work with defects (discrete event simulation)

- Best application of this method is for trade-off studies for staffing, prioritization of tasks, or project breakdown
  - Indicate bottlenecks (staff or task related)
  - Advise best policies
  - Stochastic nature will show a distribution of results which helps assess risk

Path Forward

- Complete integration of the analytic model into the platform
  - Add field for directory for input file
  - Load analytic model code into GitHub
  - Provide template for output data

- Expand baseline model to include added capabilities
  - Metaheuristics for task prioritization
  - Include additional agile functions (refactoring, micro-optimization, etc.)
  - Stochastic analysis of critical path as scheduled

1. Glaiel, 9-11

Conclusion

- Cost and schedule overruns cause delay in capability delivery
  - $402 billion in cost overrun, average 22 months schedule overrun per project[1]
  - 92 major defense acquisition programs delayed and 12 cancelled in one year (FY2010)[1]
- Required to baseline a sufficient tool for validating cost and schedule estimates for software development projects:
  1. Compatible
  2. Transparent
  3. Predictive
  4. Repeatable
  5. Descriptive
  6. Prescriptive
  7. Complete
- Software platform and baseline analytic model delivered to NISEC analysts for baseline implementation and further development
- Applied in the appropriate fashion, this tool will be effective in informing NISEC decisions when reviewing and advising cost and schedule estimates for program management offices

1. Hofbauer, 2

Questions?

References

Glaiel, Moulton, Madnick (MIT). “Agile Project Dynamics: A System Dynamics Investigation of Agile Software Development Methods.” March 2013. CISL# 2013-05. PDF File.

Modigliani, Chang (MITRE). “Defense Agile Acquisition Guide.” March 2014. Release# 14-0391. PDF File.

Northern, Mayfield, Benito, Casagni (MITRE). “Handbook for Implementing Agile in Department of Defense Information Technology Acquisition.” December 2010. Release# 11-0401. PDF File.

Hofbauer, Sanders, Ellman, Morrow (CSIS). “Cost and Time Overruns for Major Defense Acquisition Programs.” April 2011. PDF File.

Defense Acquisition University (DAU) Press. “Systems Engineering Fundamentals.” January 2001. Supplementary Text Publication. PDF File.

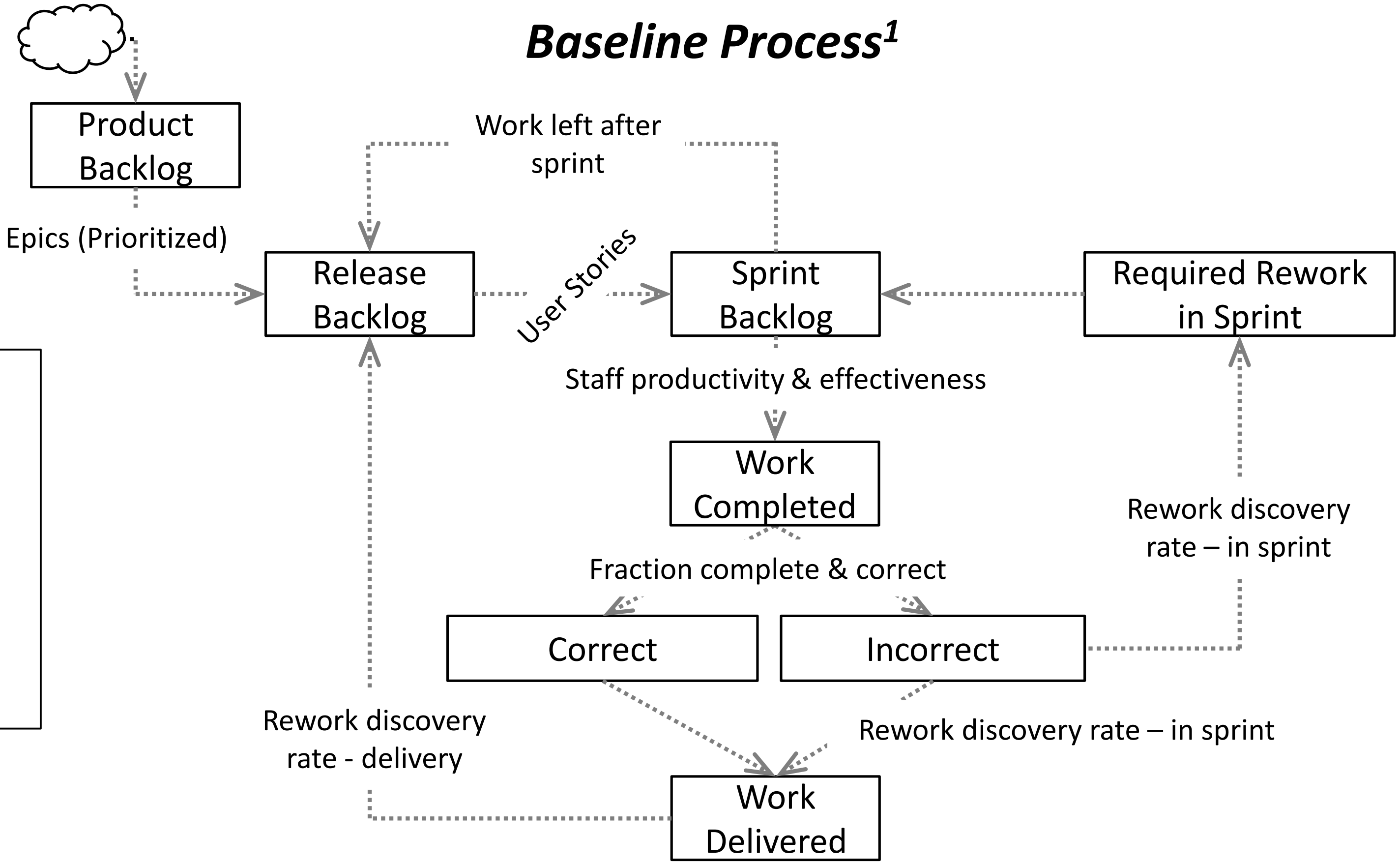
Sponsor

Provides Army decision-makers with cost, performance, and economic analysis in the form of expertise, models, data, estimates, and analysis

Networks, Information Systems, Software & Electronics Costing

Develops life cycle cost estimates for communications-electronics systems, major automated information systems (AIS), defense business systems (DBS) and software intensive systems

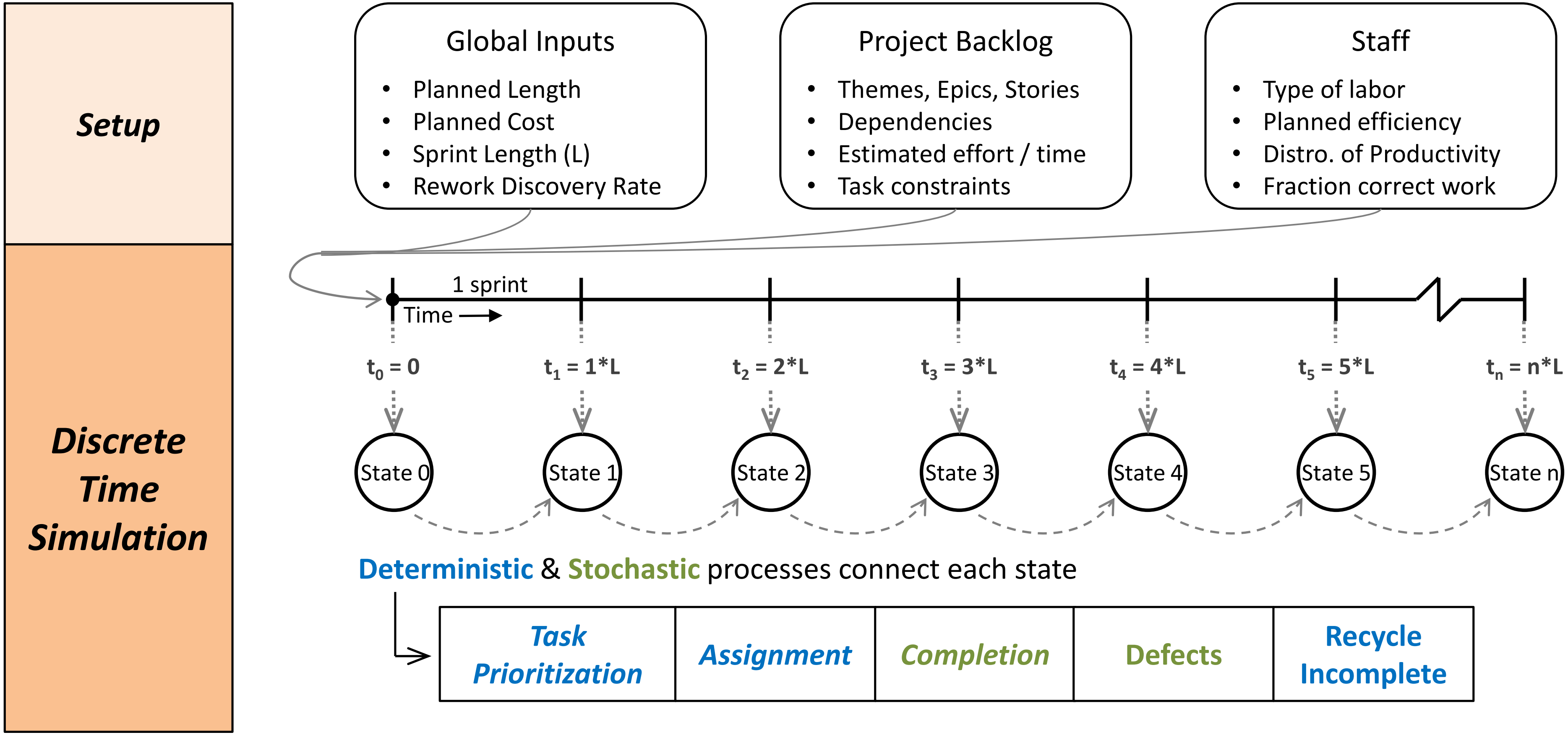
Agile Software Development CONOPs



- Six Major Agile Functions[1]
  - ✓ Feature driven processing
  - ✓ Iterative, incremental delivery
  - ○ Refactoring
  - ○ Micro-optimizing
  - ○ Customer involvement
  - ○ Team dynamics

1. Glaiel, 9-11

Analytic Model



Approach 1 – Integer / Stochastic Programming (I/SP)

- Formulate an integer program and wrap in code to generate stochastic influencers on constraints prior to optimization

Stochastic Input		Integer Program		Output
• Staff productivity by sprint • Generate task list (bugs, added tasks, etc.)	→	• Minimize makespan (# sprints) • Cost is derived result	→	• Feasibility • Schedule slack reduction • Sensitivity

Model Formulation

$$minmize\ z = \sum s_i$$

Subject to:

1. Every task has a sprint
2. Task points completed in sprint < staff capacity
3. Prerequisite tasks completed before start
4. Tasks that are bug fixes immediately follow task with bug
5. Tasks start after release date
6. s_i = 1 if sprint is used, 0 otherwise

Pros	Cons
• Gives optimal makespan solution for schedule and can be reprogrammed to do the same for milestones • I/SP formulation is intuitive and therefore transparent, even if the math is not • Custom inputs and code make this approach repeatable	• Staff available is fixed (with stochastic performance) • High resolution input on development tasks • Limited insight to interdependent processes • Unknown: Implementation of optimization software on government systems

Approach 2 – System Dynamics Diagram (SDD)

- Based on "Agile Project Dynamics: A System Dynamics Investigation of Agile Software Development Methods," March 2013.
  - Massachusetts Institute of Technology (MIT) professors define key processes (genes) that define agile software development and build systems dynamics model of software projects aimed at comparing waterfall development with different Agile development frameworks

- Begins with replicating the model from this paper, then attempting to expand and tailor the Agile modules
  - Baseline flow model represents basic Agile software development
  - Diagrams of other development processes (genes) are modular and adaptable

Pros	Cons
• Transparency – system is visualized through model building • Leverage effort already dedicated to dissect Agile and Software teams • MBSE approach becoming popular for Department of Defense analysis	• SDD modeling tools are not typically commutable (not sure which are available on government systems) • High resolution model, with little data to support inputs • Flexibility of task properties is lower than other approaches

Approach 3 – Discrete Event Simulation (DES)

- Independently develop discrete event simulation for the accomplishment of project tasks by sprint
  - Leverage research items in MIT Agile Project Dynamics model to inform simulation design
  - Flexible input, output, and architecture to be discussed and presented for concurrence
- To be designed with modular framework for tradeoff analysis of development processes
- Solution deployed with standalone executable and web based application

Pros	Cons
• Customized approach means DES is most easily deployed solution • Flexible inputs and outputs means DES is highly repeatable and could be implemented in distributed computing environment • Strong for tradeoff analysis of multiple cases	• Coded simulations are inherently less transparent / harder to update and require thorough documentation • Most difficult to deliver sophisticated product in a single semester • Little data on model inputs means less predictive value

Proposed Approach

- Decision Analysis
  - Ranked proposed approaches 1 (worst) – 3 (best) for each key criteria
  - Four highest priority criteria (bolded) are weighted twice as heavily as three lowest priority
  - Weighted sum of rankings provides relative score between approaches (higher is better)

- Documentation for rankings provided in backup.

$$score_a = \sum_{i=1}^{7} s_i r_{a,i} \;\; for\ a\ in\ \{1, 2, 3\}$$

- s_i = criteria weight of criteria i
- $r_{a,I}$ = rank of approach a under criteria i

	Compatible	**Transparent**	**Predictive**	**Repeatable**	Descriptive	Prescriptive	Completeness	**WEIGHTED SCORE**
1. Optimal Schedule	1	2	3	3	1	1	3	23
2. SDD Replicate	2	3	1	1	3	2	2	21
3. Flexible DES	3	1	2	3	2	3	2	25

Recommend Approach 3 due to its strength in compatibility and utility across analytic objectives