Connecting Software Reliability Growth Models to Software Defect Tracking

Maskura Nafreen[1], Melanie Luperon[1], Vidhyashree Nagaraju[1], Ying Shi[2], and Lance Fiondella[1]

University of Massachusetts Dartmouth[1]

NASA Goddard Space Flight Center[2]

NASA Requirements for Defect Management

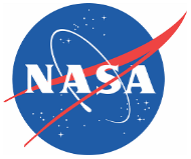
NPR 7150.2C Requirements			STD 8739.8 SA Tasking (New Draft)
5.5.1	201	**The project manager shall track and maintain software non-conformances** (including defects in tools and applicable ground software).	1. Confirm that all software non-conformances are recorded and tracked to resolution. 2. Independently verify closure of non-conformances; confirm that the same non-conformance does not exist elsewhere in the system. 3. Confirm that accepted non-conformances include the rationale for the non-conformance.
5.5.2	202	**The project manager shall define and implement clear software severity levels for all software non-conformances** (including tools, COTS, GOTS, MOTS, OSS, reused software components, and applicable ground systems).	1. Confirm that all software non-conformances severity levels per are defined. 2. Assess the application of the defined definitions as applied to identify software non-conformances. 3. Confirm that the project assigns severity levels to non-conformances with tools, COTS, GOTS, MOTS, OSS, reused software components, and applicable ground systems.
5.5.3	203	**The project manager shall implement mandatory assessments of reported non-conformances for all COTS, GOTS, MOTS, OSS, or reused software components.**	1. Confirm the necessary evaluations of reported non-conformances for all COTS, GOTS, MOTS, OSS, or reused software components are occurring throughout the project lifecycle. 2. Assess the impact of non-conformances.
5.5.4	204	**The project manager shall implement process assessments for all high severity software non-conformances** (closed loop process).	1. Perform a root cause analysis on all identified high severity nonconformances. 2. Confirm that the project assesses the processes for all high severity software non-conformances. 3. Independently track corrective actions from assessments to closure.4. Assess opportunities for process improvement.

A DCR Example

ID	Submitted Date	Title	DCRType	DCR SubType	Document Type	Originator	Phase Found	BuildFound	Subsystem	ProcessorTask	Severity	Symptoms	AssignedTo	BuildTarget	State	Priority
19	17-Jan-03	"Fix spelling of sp_default_parameter_tbl.c filename."	DEFECT			Lenny Becraft (lbecraft)		NA	COMMON FSW		High	Cosmetic	Peter Kutt (pkutt)	COMMON-2.0.0	Closed	Routine

Description	Solution	Related DCRs	Reproducibility	Resolution	Closure Notes	Additional Product Affected	Tester	TCC TestProcsUsed	TCC Tester Comments	TCC Log Folder	TCC Log File	TCC Test Outcome	TCC Verification Method	Test Lead's Comment
"Fix spelling of sp_default_parameter_tbl.c filename."	""	0	always	fixed										Per IRB Decision 03/31/2005, this DCR can be closed.

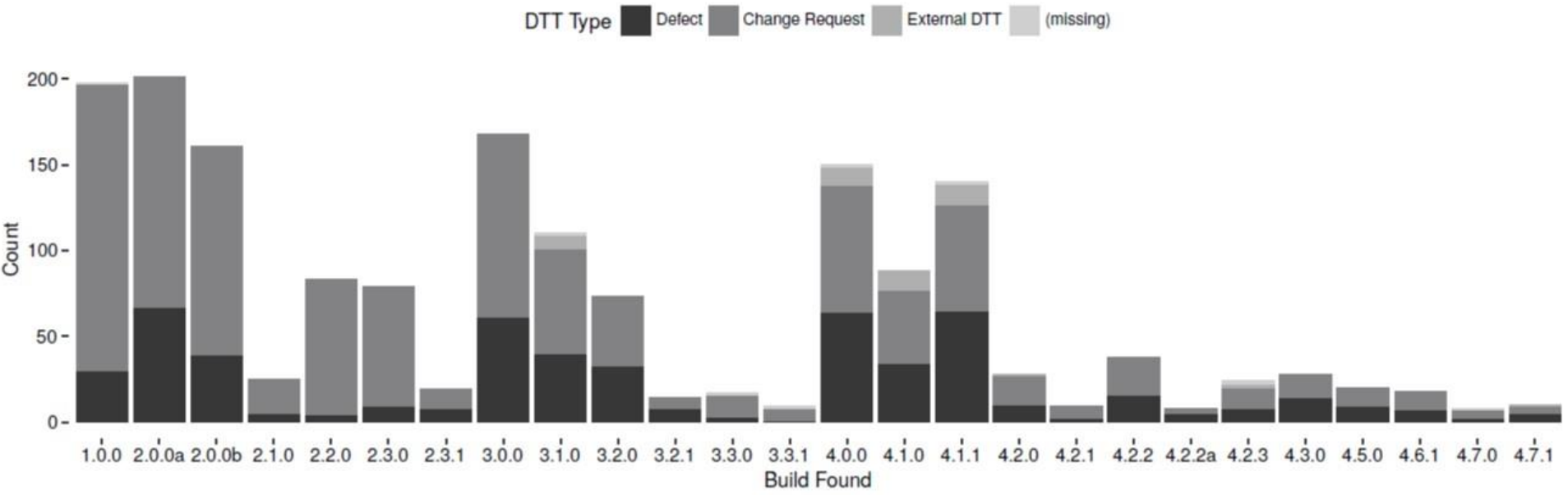
Modules	Workflow	Date Assigned	Date InWork	Date WorkCompleted	Date TestCompleted	Date Closed	Date Ready For Build Promotion	Date In Test	Date Ready For Closure	Date Closed With Defect	Date On Hold	Date Cancelled	Date Ready For Test	ref
				10-Mar-05	20-Apr-05	20-Apr-05	20-Apr-05	20-Apr-05	20-Apr-05				20-Apr-05	



Outline

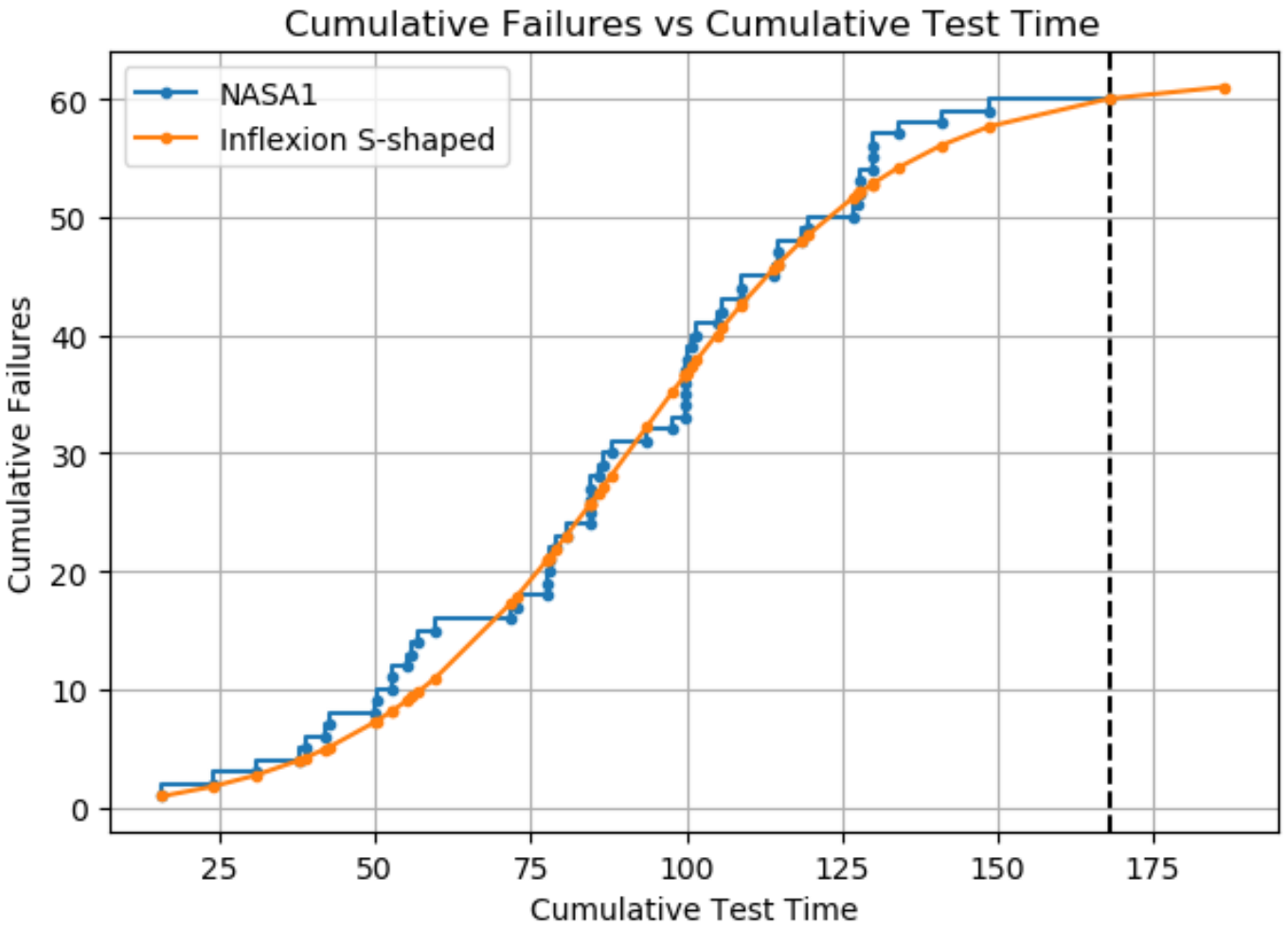
- Case study with application to NASA data
- Defect Discovery/Resolution Modeling
- Defect Lifecycle Modeling

Summary of frequencies by DTT type and build found



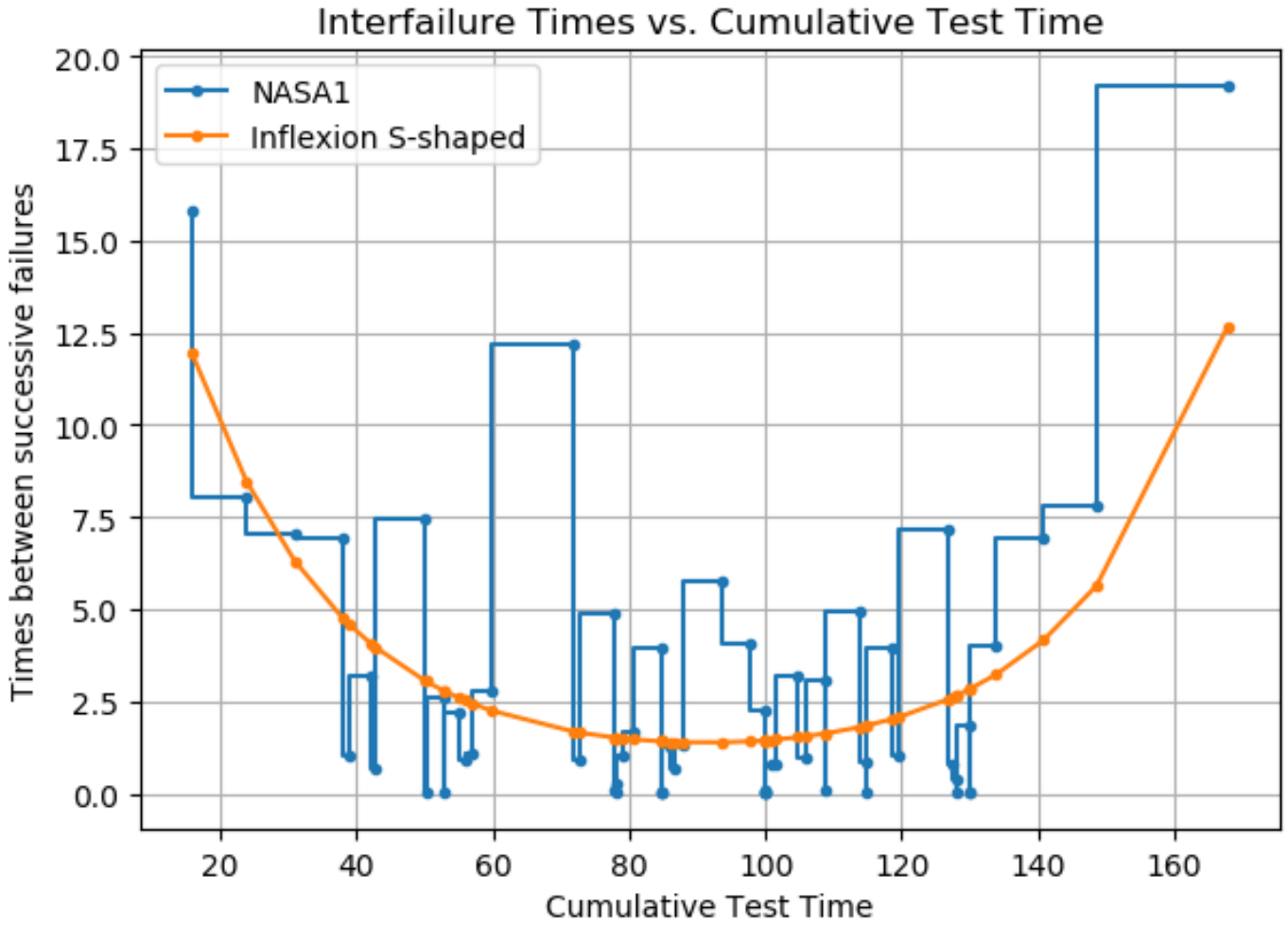
Extracted defects and change requests from major and minor versions exhibiting a large number of events

Cumulative failures



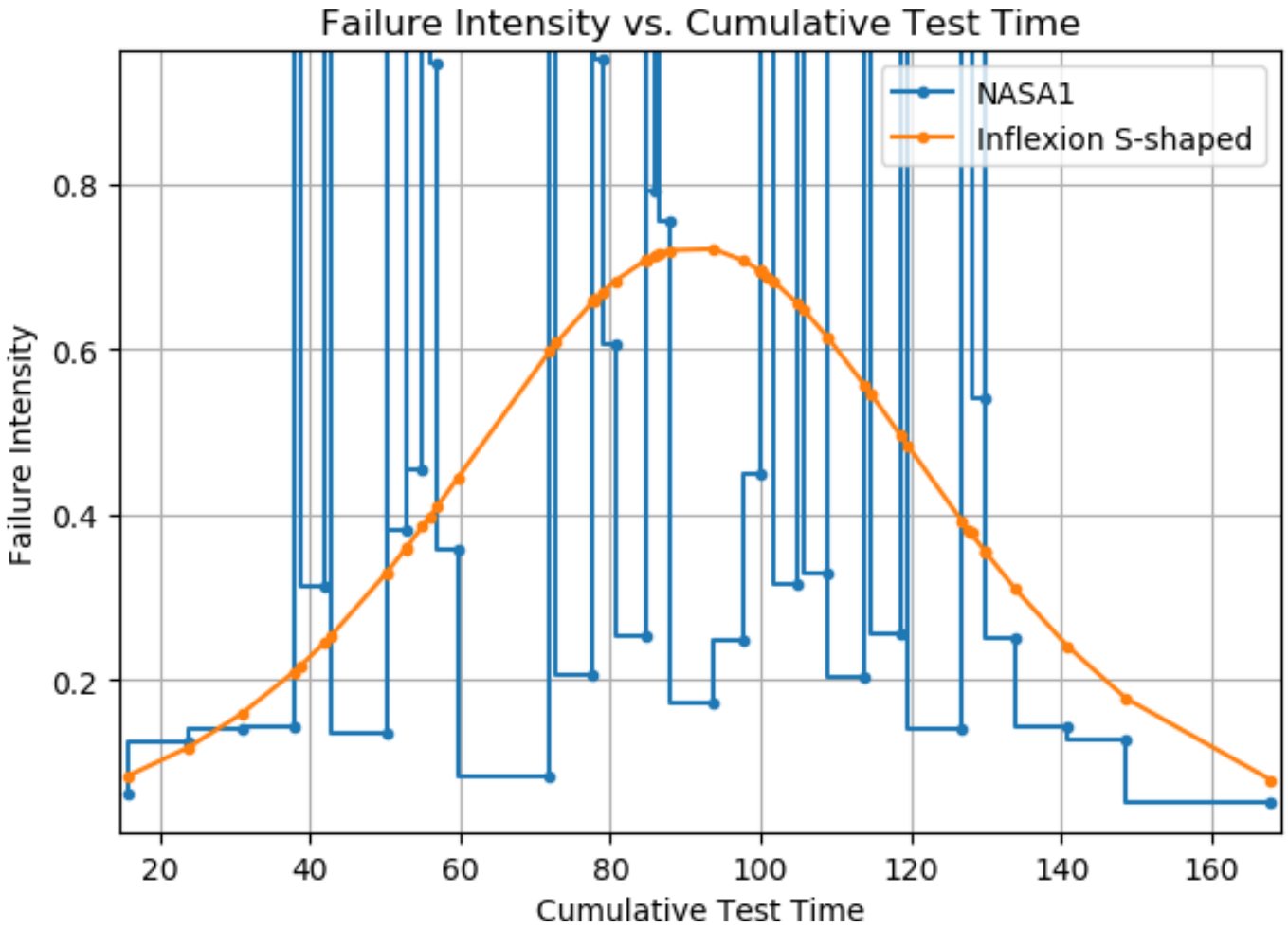
Plot enables comparison of data and model fits

Time between failures



Times between failures should increase (indicates reliability growth)

Failure intensity



Failure intensity should decrease (indicates reliability growth)

Failure Predictions

	Model	Time to achieve R = 0.9 for mission of length 10	Expected # of failures for next 10 time units	Nth failure	Expected times to next 1 failures
	All	All	All	All	All
1	Delayed S-Shape	342.43212	2.916884	1	3.349118
2	Goel-Okumoto		3.57428	1	2.797767
3	Inflexion S-Shape	207.73900	0.639518	1	2.12015

Time to achieve target reliability can help identify potential schedule overruns

Published results in peer-review open access journal

SoftwareX 10 (2019) 100357

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Practical software reliability engineering with the Software Failure and Reliability Assessment Tool (SFRAT)

Vidhyashree Nagaraju [a], Venkateswaran Shekar [a], Joshua Steakelum [a], Melanie Luperon [a], Ying Shi [b], Lance Fiondella [a,*]

[a] Department of Electrical and Computer Engineering, University of Massachusetts, Dartmouth, MA, 02747, USA
[b] Goddard Space Flight Center, National Aeronautics and Space Administration, USA

ARTICLE INFO

Article history:
Received 20 August 2019
Received in revised form 3 November 2019
Accepted 4 November 2019

Keywords:
Software reliability engineering
Software reliability growth model
Software Failure and Reliability Assessment Tool
Open source
R programming language
GitHub

ABSTRACT

Many large software projects struggle to achieve their reliability targets. Software reliability growth models quantify reliability from failure data collected during software testing. This paper presents the Software Failure and Reliability Assessment Tool (SFRAT), which implements several software reliability growth models as a free and open source application. The open source nature of the tool enables users to integrate the methods into their organization's workflow and researchers to contribute additional statistical methods. The tool is presented in the context of a NASA project.

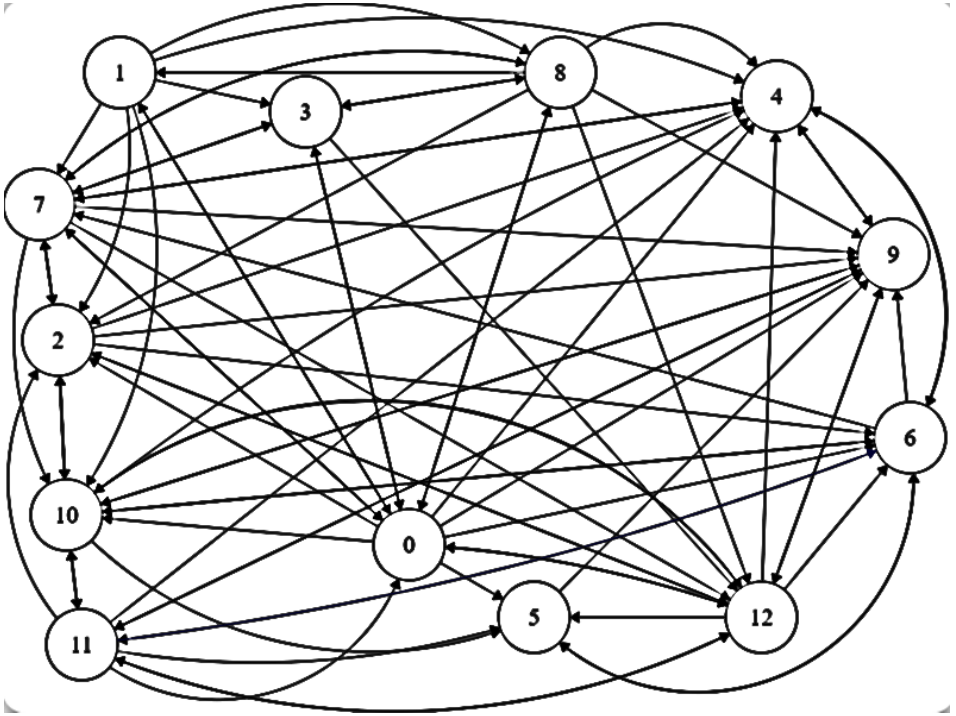
Defect Lifecycle Modeling

- 2000 defects
- Times between thirteen possible stages of the defect lifecycle from creation to closure

State	Description	State	Description
0	Created	7	In Work
1	Assigned	8	On Hold
2	Build Integration	9	Ready for Closure
3	Canceled	10	Ready for Test
4	Closed	11	Test Complete
5	Closed with Defect	12	Work Completed
6	In Test		

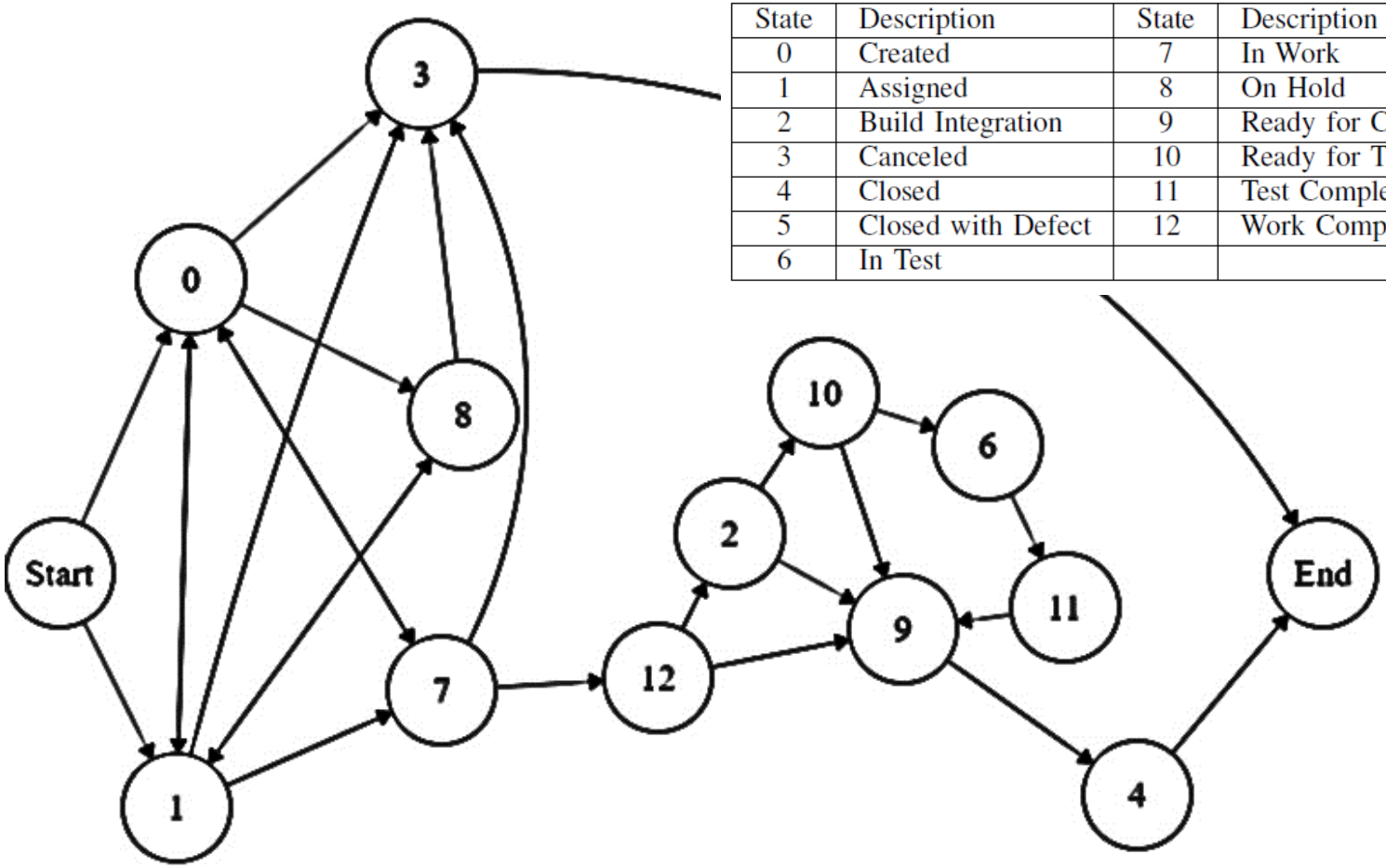
Defect Lifecycle Modeling: Data before cleaning

- Initial impressions
  - Individuals logging data may not
    - See value in activity
    - Receive consistent training
    - Perform when events occur
  - May benefit from models that inform progress

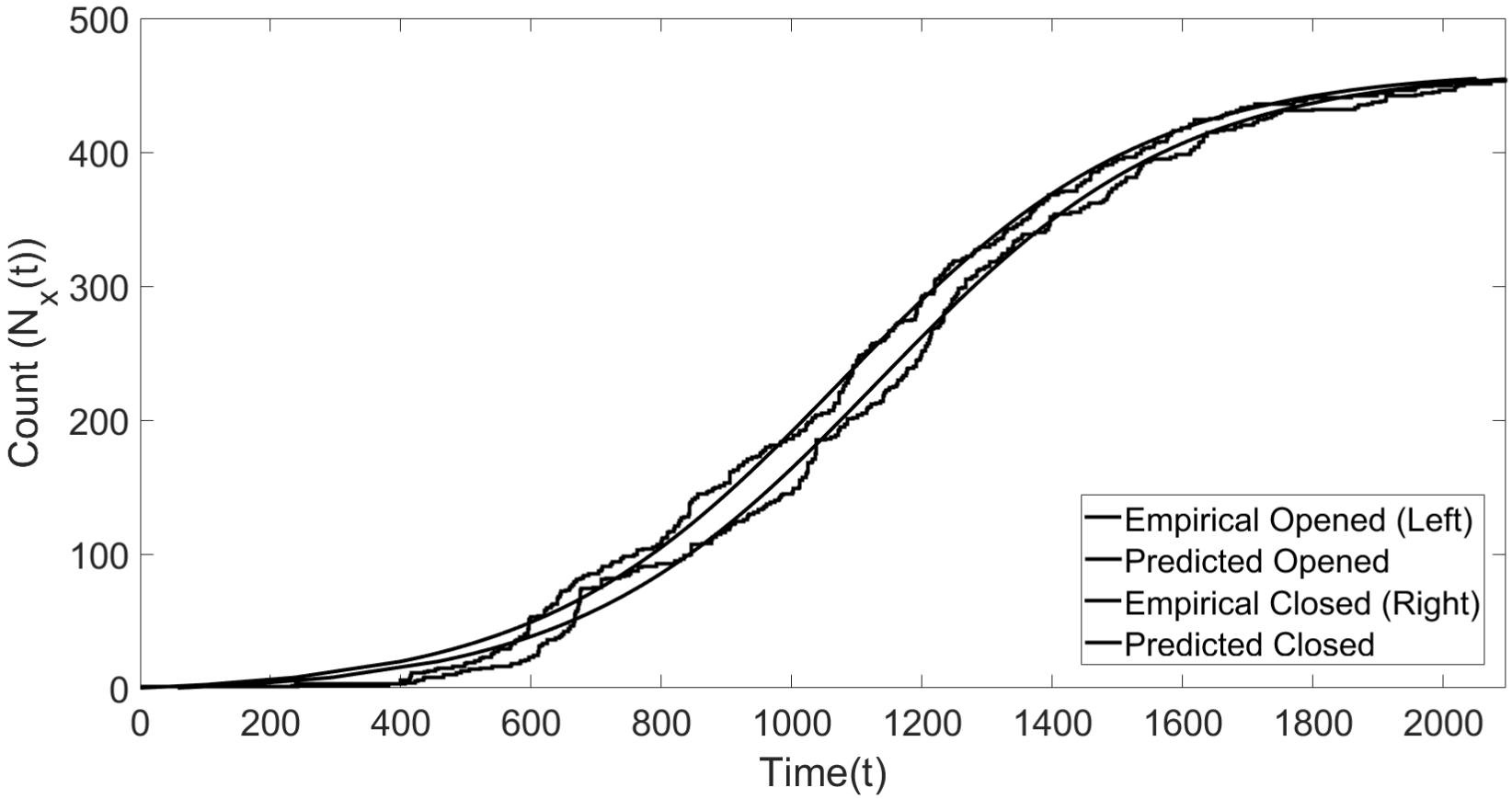


Defect Lifecycle Modeling: Data after cleaning

State	Description	State	Description
0	Created	7	In Work
1	Assigned	8	On Hold
2	Build Integration	9	Ready for Closure
3	Canceled	10	Ready for Test
4	Closed	11	Test Complete
5	Closed with Defect	12	Work Completed
6	In Test		

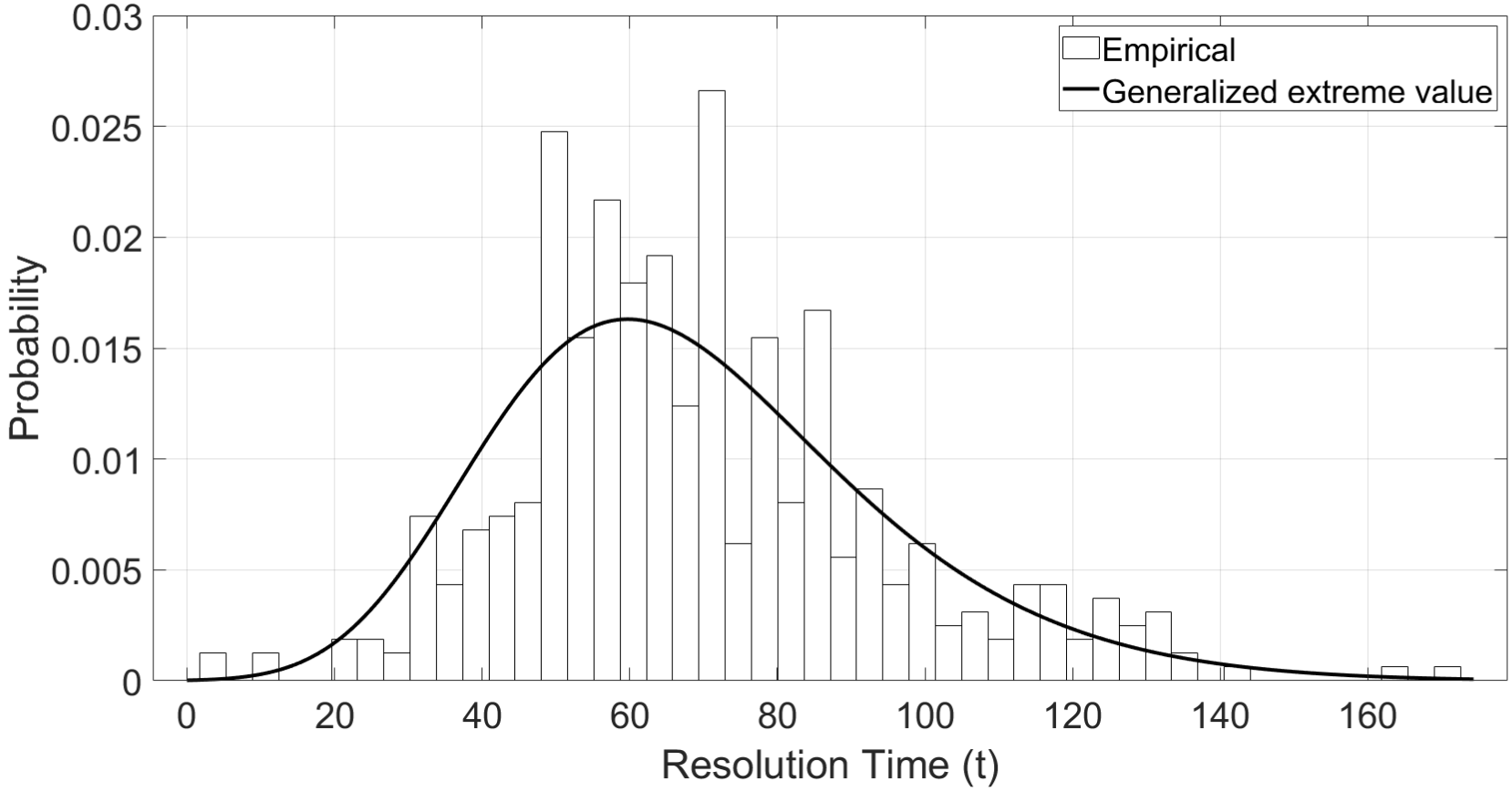


Defect Discovery/Resolution Modeling



- Mean time to resolve defects reasonable fit to closure curve
- More process oriented modeling and analysis will likely be more informative

Distribution of time between defect discovery and resolution



Not broken down by severity, number of defects in queue, or other process factors

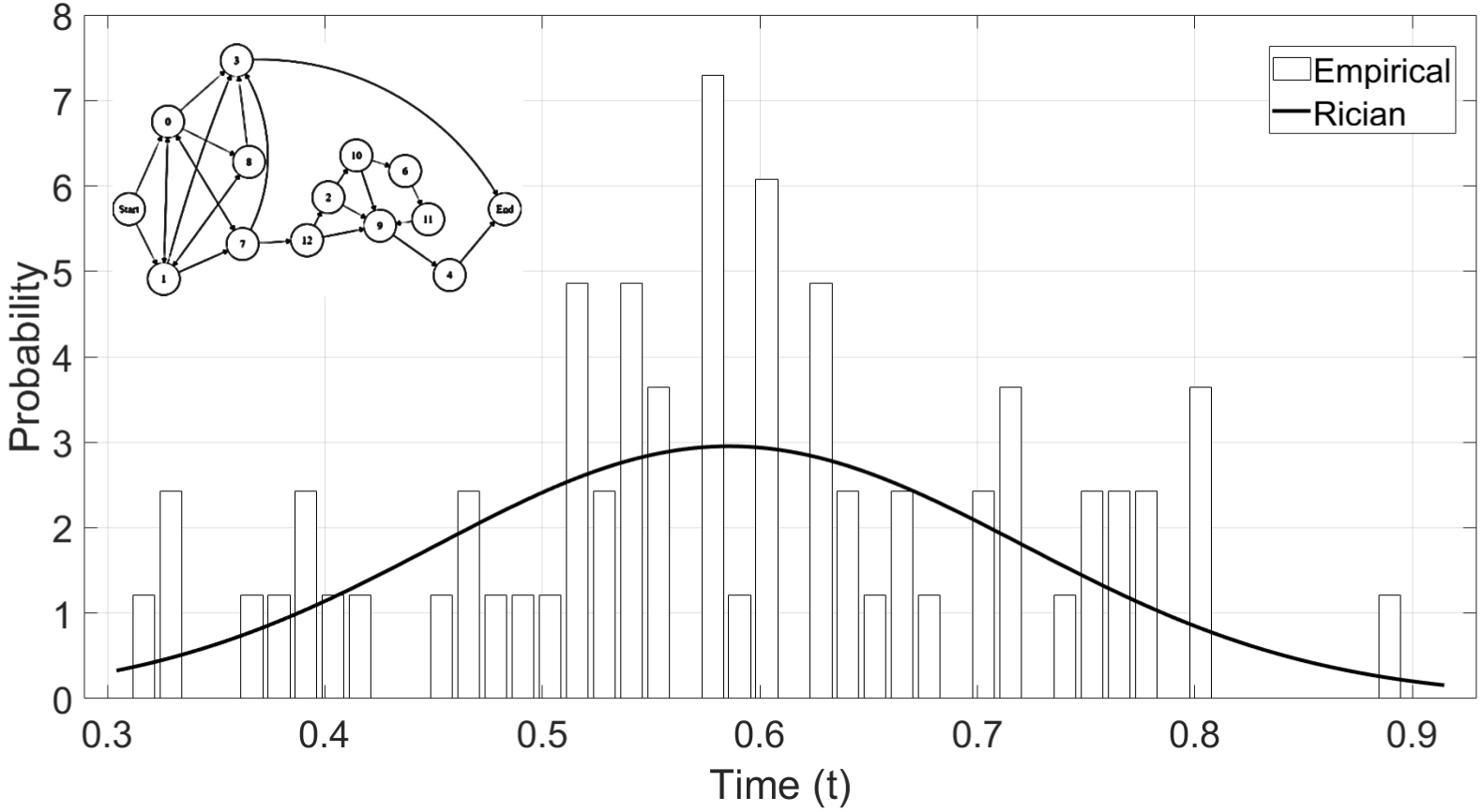
Maximum Likelihood Estimation for censored data

- Identify distribution to best fit defect resolution times, when only subset of defects discovered has been resolved

$$Lik = \prod_{i=1}^{k} f(t_{(i)}^{r} - t_{(i)}^{d}; \theta) \prod_{i=k+1}^{n} 1 - F(T - t_{(i)}^{d}; \theta)$$

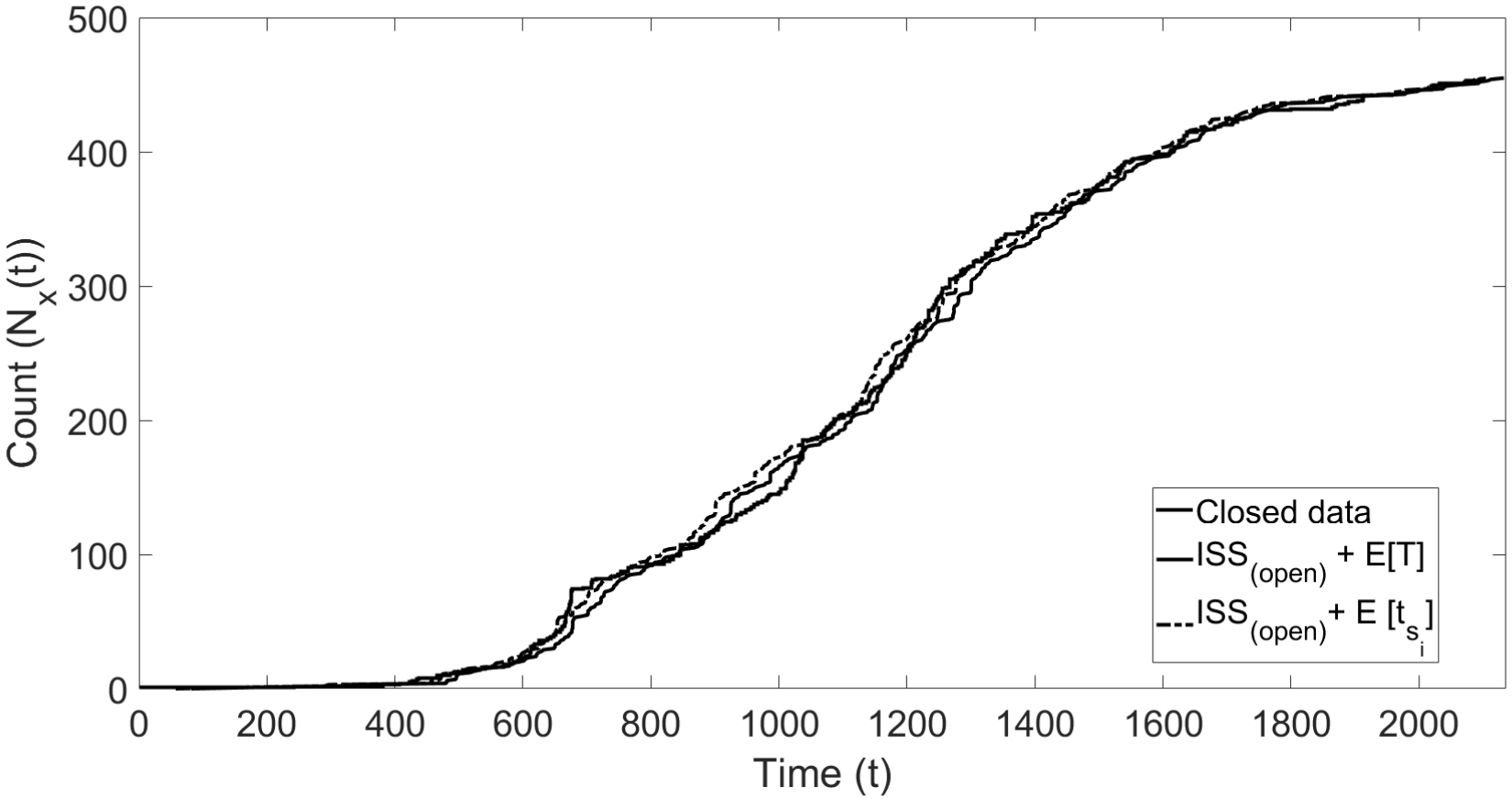
  - k – defects discovered and resolve
  - $(n-k)$ – defects discovered by not resolved
  - $t_{(i)}^{r} - t_{(i)}^{d}$ - time between discover and resolution
  - $T - t_{(i)}^{d}$ - time between discovery and present

Distribution of time between defect creation and assignment



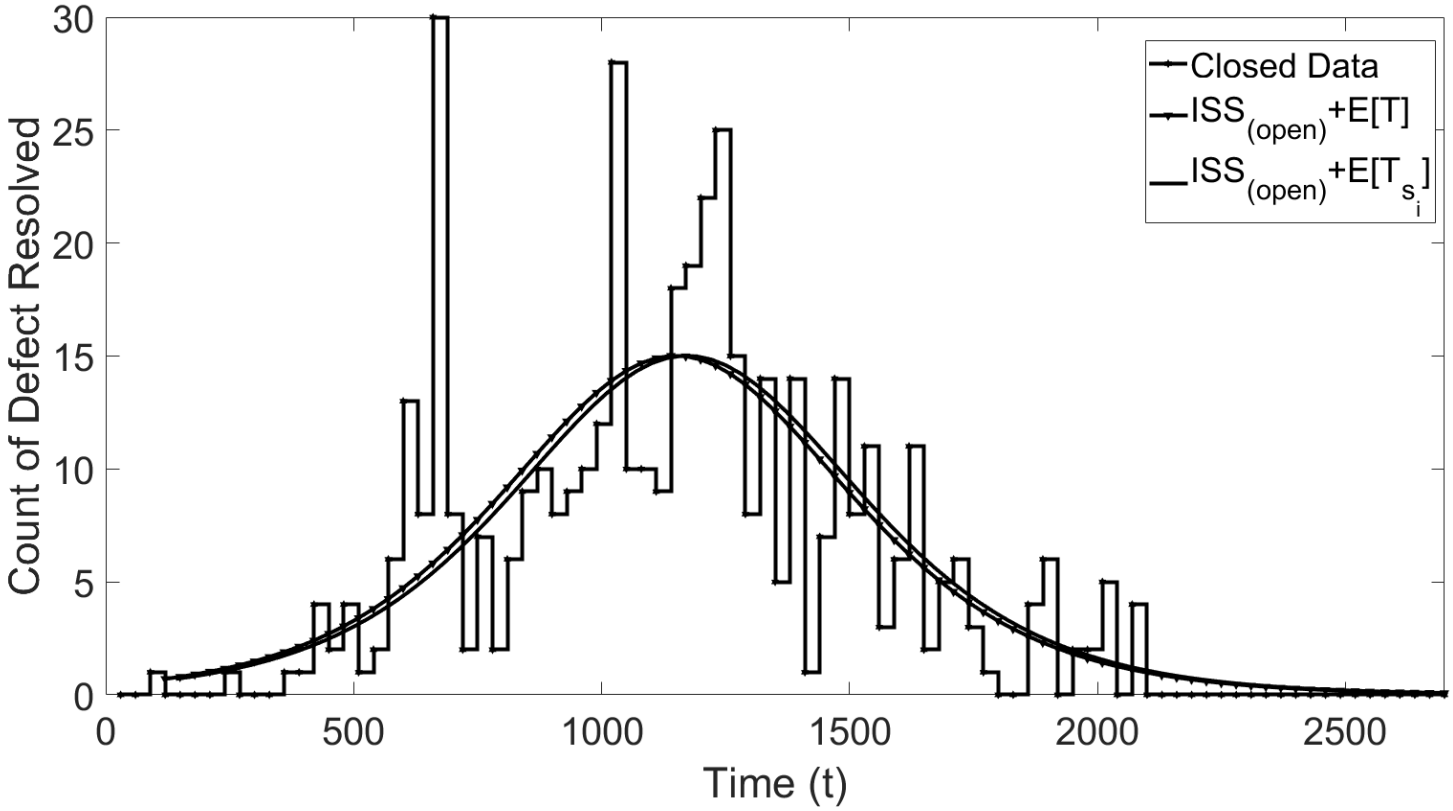
- Can model defect resolution as semi-Markov process
  - Nonidentical and non-exponentially distributed transitions
- Can break down by severity, number of defects in queue, or other process factors
- Need help identify practical process related issue to model and if present data would support

Online defect resolution prediction



- Simpler distributional model predicts nearly as well as state model
- Hypothesis test suggest state transitions follow first order Markov process

Inferences enabled by defect resolution model



- Models predict intensity of defects resolved well
- Covariate models could do even better

Acknowledgements

- This work was supported by the National Aeronautics and Space Administration (NASA) under Contracts (#80NSSC18K0154 and 80NSSC20K0276) and NSF Award #1749635.