Supplementary Materials

Governing the Recovery of Stalled Systems-of-Systems:
ADAPT, a Dependency-Anchored Methodology, with Multi-Case Feasibility Illustrations

Elias Calboreanu

This document provides the supporting estimates and methods for Section 5.7 of the main text and the external-validation portfolio. Every value below is produced by the scripts in the `external_validation/` folder from the frozen public datasets; see S5 for the exact commands. The four formal propositions are stated and proved in the main text (Appendix A) and are not reproduced here.

S1. Per-project dependency-load estimates

Table S1 reports the per-link effect of dependency load on time-to-resolution, estimated separately (unadjusted Cox) in each of the fourteen Apache projects; it is the tabular form of the forest plot in the main text. A hazard ratio below one denotes slower resolution.

Table S1: Per-project unadjusted Cox hazard ratio for the per-link effect of dependency load on resolution (HR < 1 = slower). Random-effects pooled estimate and heterogeneity in Table S2.

Project	n	HR (95% CI)	p	Direction
Cassandra	6,000	0.679 (0.649, 0.709)	5.6×10^{-66}	slower
Flink	6,000	0.819 (0.787, 0.853)	2.3×10^{-22}	slower
Spark	5,711	0.834 (0.812, 0.858)	1.3×10^{-37}	slower
HBase	6,000	0.864 (0.843, 0.885)	1.4×10^{-31}	slower
BookKeeper	1,095	0.871 (0.805, 0.941)	4.7×10^{-4}	slower
Kafka	5,950	0.899 (0.869, 0.931)	1.5×10^{-9}	slower
Hadoop	6,000	0.912 (0.893, 0.932)	1.5×10^{-16}	slower
Curator	724	0.960 (0.870, 1.060)	0.42	n.s.
Mesos	6,000	0.963 (0.943, 0.984)	5.6×10^{-4}	slower
HDFS	6,000	0.987 (0.971, 1.004)	0.13	n.s.
YARN	6,000	0.997 (0.983, 1.011)	0.68	n.s.
Hive	6,000	1.000 (0.987, 1.012)	0.94	n.s.
ZooKeeper	4,862	1.017 (0.991, 1.044)	0.20	n.s.
MapReduce	6,000	1.040 (1.021, 1.058)	2.1×10^{-5}	faster

S2. Assumption-light and pooled estimators

Because the Cox proportional-hazards (PH) assumption is violated for dependency load (PH test $p < 10^{-50}$), the inference in the main text rests on the assumption-light estimators in Table S2; the pooled and Cox rows are reported for completeness. The 95% *prediction* interval is the range expected for a new, previously unseen project and includes unity, the quantitative counterpart of the $I^2 \approx 98\%$ heterogeneity.

Table S2: Assumption-light and pooled estimators for the dependency-load effect, cleaned corpus ($n = 72,342$ issues). KM, Kaplan–Meier; RMST, restricted mean survival time; AFT, accelerated failure time; PH, proportional hazards.

Estimator	Result	Assumption
Median resolution time (resolved), load 0/1/2–3/4+	11, 32, 70, 219 days	none (descriptive)
Restricted mean time unresolved over 365 d, load 0/1/2–3/4+	126, 147, 172, 228 days (+102 at 4+)	none (RMST)
Pooled per-link hazard ratio (random effects)	0.913 (0.873, 0.955); I^2=98%	random-effects
95% prediction interval (new, unseen project)	0.76, 1.10 (includes 1)	random-effects
Weibull AFT time ratio per link, unadjusted	1.21 (1.18, 1.24)	AFT (no PH)
Weibull AFT time ratio per link, adjusted	1.09 (1.07, 1.11)	AFT (no PH)
Cox hazard ratio per link, adjusted	0.966 (0.960, 0.972); PH $p < 10^{-50}$	Cox (PH violated)

S3. Structural measures across the public datasets

Table S3 gives the structural dependency measures ADAPT defines (trace completeness, branchless fraction, critical-path depth, change blast radius) computed on the three further public datasets. These characterize dependency structure; they are not outcome tests.

Table S3: Structural dependency measures on the public datasets. "Branchless" is the fraction of artifacts that trace to no validated parent; "max blast radius" is the largest downstream reachable set per change; for Dronology/SAFA it is the maximum impact localized across the measured V0→V1 changes (cf. main-text Table 8).

Dataset	Scale	Trace completeness / branchless	Max depth	Max blast
Microservice corpus (20 systems)	199 services, 373 deps	8.0% branchless (16/199)	7	12
CoEST Albergate	55 code, 17 reqs	94.5% traced, 3 branchless	–	–
Dronology/SAFA V0	240 artifacts, 234 links	100% trace-complete, 0 branchless	6	–
Dronology/SAFA V1	360 artifacts, 355 links	100% trace-complete, 0 branchless	8	3

S4. Methods

Structural measures. On the trace/dependency graph $D = (V, A)$: *trace completeness* is the fraction of artifacts with a path to a capability-need root; an artifact is *branchless* if it has no such path; *critical-path depth* is the longest directed path length; the *blast radius* of a change to v is the set of vertices reachable from v, computed by graph search (see Propositions A1 and A4 in Appendix A of the main text).

Survival and meta-analysis. Time-to-resolution is right-censored at the snapshot date for unresolved issues. Per-project effects are Cox proportional-hazards coefficients on link count; the PH

assumption was tested via scaled Schoenfeld residuals and rejected, so the analysis additionally reports Kaplan–Meier curves, restricted mean survival time at $\tau = 365$ days, and Weibull accelerated-failure-time time ratios, none of which assume proportional hazards. Per-project estimates were combined by DerSimonian–Laird random-effects meta-analysis; I^2 quantifies between-project heterogeneity, and the prediction interval uses $\hat{\mu} \pm t_{k-2}\sqrt{\hat{\tau}^2 + \mathrm{Var}(\hat{\mu})}$.

Data cleaning. Of 72,681 pulled issues, 339 were dropped (289 with out-of-range-year timestamps and 50 with unparseable or non-positive durations), leaving 72,342.

Blocking-link restriction. The main-text load counts all issue-link types. Restricting it to blocking and blocked-by relations (the prerequisite dependencies ADAPT sequences) reproduces the same monotone, sharper association. Median resolution time of resolved issues rises 17, 48, 105, and 547 days across 0, 1, 2–3, and 4+ blocking links (vs. 11, 32, 70, 219 for all links); the per-project unadjusted Cox estimate pools by DerSimonian–Laird to a hazard ratio of 0.934 (95% CI 0.890–0.980; $I^2 = 88\%$), slower in six projects, faster in one, null in seven. The pooled effect stays significant and the gradient steepens; individual-project significance falls only because blocking links are far rarer (present on about 3,900 issues vs. about 23,800 with any link). Reproduce with `blocking_reanalysis.py` (stdlib + `statsmodels`; validated to reproduce the all-link medians and pooled HR exactly), output in `blocking_results.txt`.

S5. Reproduction

From `Supplementary/external_validation/`: `jira_pull_large.py` re-pulls the corpus; `rebuild_figs.py` cleans and produces the Kaplan–Meier figure and the per-project forest (Table S1); `render_v2.py` renders the forest; `adj_check.py` fits the adjusted Cox model and the PH test; `math_audit.py` computes the prediction interval, RMST, and the Weibull AFT time ratios (Table S2); `micro_analysis.py` and the CoEST/Dronology scripts produce Table S3; `verify_figures.py` independently re-derives and cross-checks the figure inputs; `blocking_reanalysis.py` re-runs the analysis restricted to blocking links (and validates the pipeline by reproducing the all-link medians and pooled HR exactly). Figures 4 and 5 as printed are produced by `regenerate_figures.py` (product-limit Kaplan–Meier with Greenwood log-log bands, and per-project Cox via `statsmodels`), which reproduces the same medians, per-project hazard ratios, and pooled estimate as the `lifelines`-based pipeline. The frozen datasets ship alongside the scripts.

Independent re-coding materials (Sections 3.4 and 6.4 of the main text). The blind re-coding protocol and its coding sheet ship in the `Supplementary/` folder as `independent_recoding_protocol.pdf` and `recoding_sheet_template.csv`. The protocol specifies a neutral auditor with no role in the engagements, the P1–P5 coding definitions, the redaction bound (main-text Section 3.6), and the agreement statistics (percentage agreement and Cohen's κ) to be reported on completion. The re-coding had not been executed as of submission; inter-rater agreement will be reported when it is.

S6. Background note: large-scale agile transformation failures and the quantified cost of technical debt

This note expands the evidentiary basis for two background claims the main text makes in Sections 1 and 2.3: that large-scale method adoptions frequently stall for governance-shaped rather

than technique-shaped reasons, and that unmanaged debt in engineering artifacts carries a large, quantifiable cost. Both literatures frame the stalled-SoS problem; neither is offered as evidence of ADAPT's effectiveness (the main text's evidence tiering, Section 6.5, applies).

S6.1 Large-scale agile transformations: failure and challenge evidence

The most comprehensive synthesis remains Dikert, Paasivaara, and Lassenius (2016), a systematic review that screened 1,875 papers and retained 52 publications describing 42 industrial large-scale agile transformations. Two of its findings matter here. First, the evidence base itself is thin: roughly 90% of the included publications are experience reports rather than controlled studies, so even the practitioner community's own record of what makes transformations succeed is weakly verified. Second, the content of the reported difficulties is organizational and governance-shaped: the review catalogues 35 challenges in nine categories, including coordination challenges in multi-team environments, hierarchical management and organizational boundaries, requirements-engineering challenges, and the integration of non-development functions, while the most salient success-factor categories are management support, choosing and customizing the agile model, training and coaching, and mindset and alignment. The challenge categories are close analogues of the governance functions whose loss the main text identifies in stalled SoS programs (shared meaning, cross-boundary coordination, connected requirements), and the success factors align with ADAPT's entry criteria (committed executive sponsorship; an explicitly diagnosed rather than assumed approach).

Edison, Wang, and Conboy (2022) extend the picture across methods: a systematic review of 191 primary studies covering 134 organizations (2003–2019) comparing SAFe, LeSS, Scrum-at-Scale, Disciplined Agile, and the Spotify model. Challenges recur across all five methods, and the review finds no empirical basis for the superiority of any one framework. The implication adopted in the main text is that framework choice is not the binding constraint: scaling frameworks presuppose a functioning coordination substrate (connected artifacts, decision authority, shared vocabulary) and do not themselves restore one. Recovery of that substrate is the gap ADAPT addresses.

S6.2 Quantification of technical debt

For the cost side, Besker, Martini, and Bosch (2019) provide the most direct quantification: in a longitudinal study following 43 professional developers, with follow-up interviews of 16 practitioners, developers reported wasting on average 23% of development time to the consequences of technical debt, and reported being frequently forced to introduce *new* debt because of already-existing debt. The second finding is the compounding mechanism the main text invokes: debt begets debt, consuming the margin meant to absorb risk.

At portfolio scale, the U.S. Software Engineering Institute's congressionally mandated study of technical debt in DoD software-intensive systems (CMU/SEI-2023-TR-003, produced under Section 835(b) of the FY22 National Defense Authorization Act) reached qualitatively matching conclusions for the defense domain from which the main text's cases are drawn: technical-debt management practices are inconsistent and deprioritized, and unmanaged debt erodes assurance, delays modernization, and remains largely invisible to program oversight. Invisibility to oversight is precisely the condition the main text's traceability component targets.

Two scope cautions. These quantifications concern software-implementation debt; the main text's argument transfers the *mechanism* (compounding, invisibility, interest paid at integration and review points) to SoS engineering artifacts, and that transfer is analogical, argued in Sections 1

and 2.3, not measured here. And none of this section bears on whether ADAPT reduces such debt; it establishes only that the problem ADAPT targets is real, large, and independently documented.

S6.3 Sources

- Dikert, K.; Paasivaara, M.; Lassenius, C. Challenges and Success Factors for Large-Scale Agile Transformations: A Systematic Literature Review. *Journal of Systems and Software* **2016**, *119*, 87–108.
- Edison, H.; Wang, X.; Conboy, K. Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review. *IEEE Transactions on Software Engineering* **2022**, *48*, 2709–2731.
- Besker, T.; Martini, A.; Bosch, J. Software Developer Productivity Loss Due to Technical Debt: A Replication and Extension Study Examining Developers' Development Work. *Journal of Systems and Software* **2019**, *156*, 41–61.
- Software Engineering Institute. *Technical Debt in Software-Intensive Systems: Report to Congress under NDAA FY22 Section 835(b)*; CMU/SEI-2023-TR-003; Carnegie Mellon University: Pittsburgh, PA, 2023.