CS466 – SOFTWARE PROCESS

AGILE & ITERATIVE DEVELOPMENT (CHAPTER 6)

A MANAGER'S GUIDE BY: CRAIG LARMAN

WEEK 6: EVIDENCE

BY: JOSEPH MARTINAZZI

EVIDENCE

WHAT ARE THE MOST EXCITING, PROMISING SOFTWARE ENGINEERING IDEAS OR TECHNIQUES ON THE HORIZON?

I DON'T THINK THAT THE MOST PROMISING IDEAS ARE ON THE HORIZON. THEY ARE ALREADY HERE AND HAVE BEEN FOR YEARS BUT ARE NOT BEING USED PROPERLY. – DAVID L. PARNAS

TODAY'S LECTURE FOCUSES ON WHY ITERATIVE AND INCREMENTAL DEVELOPMENT (IID) HAS A HIGHER PROBABILITY OF SUCCESS COMPARED TO THE WATERFALL MODEL. TOPICS THAT WILL BE COVERED INCLUDE:

1. RESEARCH EVIDENCE – STUDIES THAT PROVE PROGRAMS THAT USE AN IID APPROACH HAVE LOWER RISK, ARE MORE EFFICIENT, AND PRODUCES A HIGHER QUALITY PRODUCT.
2. EARLY LARGE PROJECT EVIDENCE – EXAMPLES OF LIFE-CRITICAL SYSTEMS THAT HAVE SUCCESSFULLY BEEN DEVELOPED USING AN IID APPROACH.
3. STANDARDS-BODY EVIDENCE – DESCRIBES HOW THE DOD ADOPTED MIL-STD-498 IN 1987 THAT UTILIZES ITERATIVE AND EVOLUTIONARY METHODS.
4. EXPERT THOUGHT LEADER EVIDENCE – EXAMPLES OF PROMINENT SOFTWARE ENGINEERS AND THEIR RECOMMENDATION TO ADOPT AN IID APPROACH TO SOFTWARE DEVELOPMENT.
5. A BUSINESS CASE – A COMPARISON OF AN IID APPROACH TO SOFTWARE DEVELOPMENT VS. A SERIAL WATERFALL APPROACH.
6. WATER FALL PROBLEMS AND WHY IT IS STILL PROMOTED – COMPANIES LIKE THE IDEA OF "REQUIREMENT DEVELOPMENT IS COMPLETE" PRIOR TO BEGINNING SOFTWARE DEVELOPMENT.

1. RESEARCH EVIDENCE

THE AUTHOR POINTS TO VARIOUS STUDIES THAT SHOW EVOLUTIONARY DEVELOPMENT RESULTS IN A HIGHER PROBABILITY OF SUCCESS COMPARED TO PROGRAMS THAT FOLLOW A WATERFALL MODEL.

A STUDY LEAD BY ALAN MAC CORMACK [MACCORMACK01] IDENTIFIED 4 PRACTICES THAT WERE COMMON ACROSS THE MOST SUCCESSFUL PROGRAMS. THESE PROGRAMS:

1. FOLLOWED AN IID PROCESS WHICH EMPHASIZED AN EARLY RELEASE OF THE PRODUCT TO THE STAKEHOLDERS FOR REVIEW AND FEEDBACK. [COMMON AMONG ALL IID] – *"SOFTWARE DEVELOPMENT BEST PRACTICE"*
2. DAILY INCORPORATION OF NEW SOFTWARE ONTO A REGRESSION TESTED BUILD. [COMMON AMONG ALL IID]
3. A TEAM EXPERIENCED IN SHIPPING MULTIPLE PROJECTS.
4. EARLY ATTENTION TO SYSTEM ARCHITECTURE AND COUPLING OF MAJOR COMPONENTS [UP PRACTICE]

DEFECT DENSITY IS THE NUMBER OF DEFECTS FOUND IN THE SOFTWARE/MODULE DURING A SPECIFIC PERIOD OF OPERATION OR DEVELOPMENT DIVIDED BY THE SIZE OF THE SOFTWARE/MODULE. IT ENABLES ONE TO DECIDE IF A PIECE OF SOFTWARE IS READY TO BE RELEASED. DEFECT DENSITY IS COUNTED PER THOUSAND LINES OF CODE ALSO KNOWN AS KLOC.

1. RESEARCH EVIDENCE

A FOLLOW-UP STUDY LEAD BY MAC CORMACK [MKCC03] IDENTIFIED 2 DRIVING IID FACTORS THAT IMPACTED DEFECT DENSITY.

- BY RELEASING THE SYSTEM EARLY (E.G. WHEN 20% OF THE FUNCTIONALITY WAS COMPLETE VS. 40%), THE ESCAPING DEFECT RATE DECREASED BY 10 DEFECTS/MONTH PER MILLION LINES OF CODE.
- BY CONTINUOUSLY INTEGRATING CODE ONTO A REGRESSION TESTED DAILY BUILD, THE ESCAPING DEFECT RATE DECREASED BY 13 DEFECTS/MONTH PER MILLION LINES OF CODE.

THIS IMPLIES THAT MORE IN-PHASE DEFECTS WERE DETECTED DURING CODE/UNIT TEST MAKING THEM CHEAPER TO FIX!

(ON PAGE 79 OF THE TEXT, THE AUTHOR STATES SEVERAL CASE STUDIES REPORT LOWER DEFECT DENSITIES ARE ASSOCIATED WITH IID METHODS [MANZO02], HOWEVER THEY ARE NOT STATISTICALLY RELIABLE.)

THE SAME STUDY ALSO IDENTIFIED THAT THESE SAME FACTORS IMPACTED PRODUCTIVITY.

- BY RELEASING THE PRODUCT EARLY (E.G. WHEN 20% OF THE FUNCTIONALITY WAS COMPLETE VS. 40%), 8 ADDITIONAL LINES OF SOURCE CODE WERE PRODUCED BY EACH PERSON DAILY.
- BY CONTINUOUSLY INTEGRATING CODE ONTO A REGRESSION TESTED DAILY BUILD, 17 ADDITIONAL LINES OF SOURCE CODE WERE PRODUCED BY EACH PERSON DAILY.

1. RESEARCH EVIDENCE

ANOTHER LARGE STUDY CONDUCTED BY THE STANDISH GROUP [STANDISH98] ANALYZED 23,000 PROJECTS. THIS STUDY FOUND THAT 4 OF THE TOP 5 FACTORS IN SUCCESSFUL PROJECTS WERE RELATED TO IID METHODOLOGIES.

- HIGH USER INVOLVEMENT – WITH SHORT ITERATIONS, DEMOS, REVIEWS, EVOLUTIONARY REQUIREMENT REFINEMENT, AND CLIENT DRIVEN ITERATIONS
- EXECUTIVE SUPPORT – FOCUSED ON TANGIBLE RESULTS
- CLEAR BUSINESS OBJECTIVES – DRIVEN BY CLIENT-DRIVEN PLANNING
- EXPERIENCED PROJECT MANAGER
- SMALL MILESTONES – ARE AT THE HEART OF THE IID METHODOLOGY

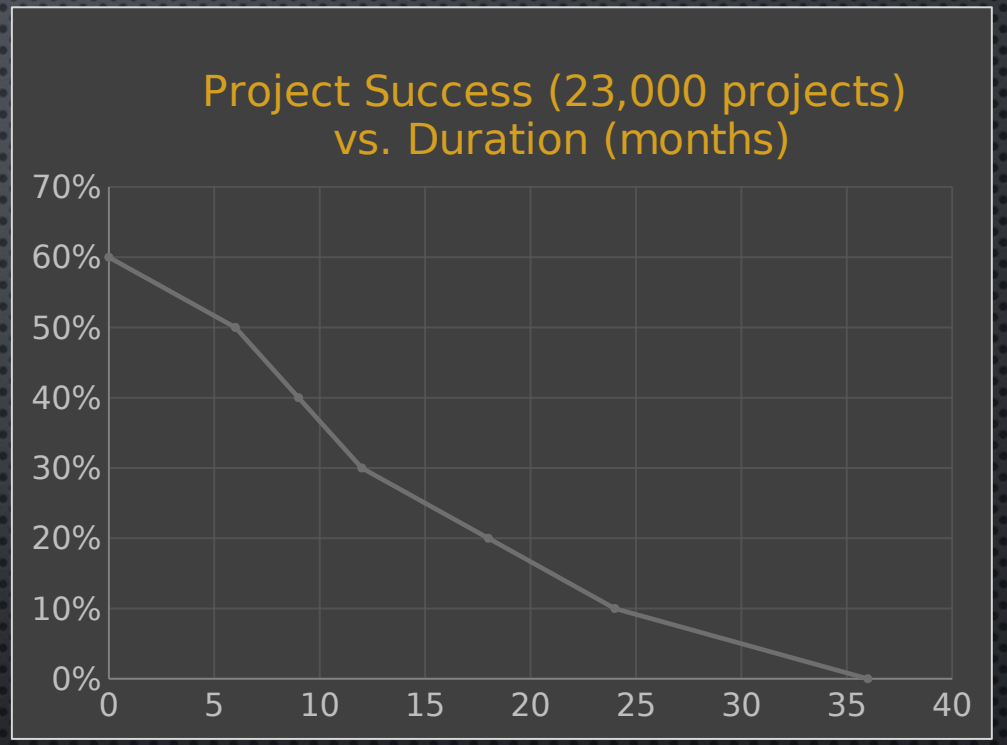
1. RESEARCH EVIDENCE – *SIZE RESEARCH*

THIS SAME STUDY [STANDISH98] ALSO ANALYZED PROJECT SUCCESS, BASED ON THE PROJECT COMPLETING WITHIN COST/SCHEDULE AND CONTAINING ALL THE SPECIFIED FUNCTIONALITY, IN RELATIONSHIP TO DURATION. AS SHOWN IN THE GRAPH TO THE LEFT; SMALLER PROJECTS THAT COMPLETED IN SEVERAL MONTHS EXPERIENCED A HIGHER SUCCESS RATE THAN LARGER PROJECTS LASTING 36 MONTHS.

- SMALL PROJECTS ARE LESS COMPLEX AND TAKE LESS TIME TO COMPLETE.
- FOR A LARGE PROJECT TO BE SUCCESSFUL, IT MUST BE BROKEN DOWN INTO SMALL (LESS COMPLEX) ITERATIONS.

THIS TREND WAS CONFIRMED BY A FOLLOW-UP STUDY SPANNING 35,000 PROJECTS [STANDISH00] THAT FOCUSED ON COST.

- SMALL PROJECTS ARE LESS COSTLY TO COMPLETE.
- FOR A LARGE PROJECT TO BE SUCCESSFUL, IT MUST BE BROKEN DOWN INTO SMALL (LESS COMPLEX) ITERATIONS.



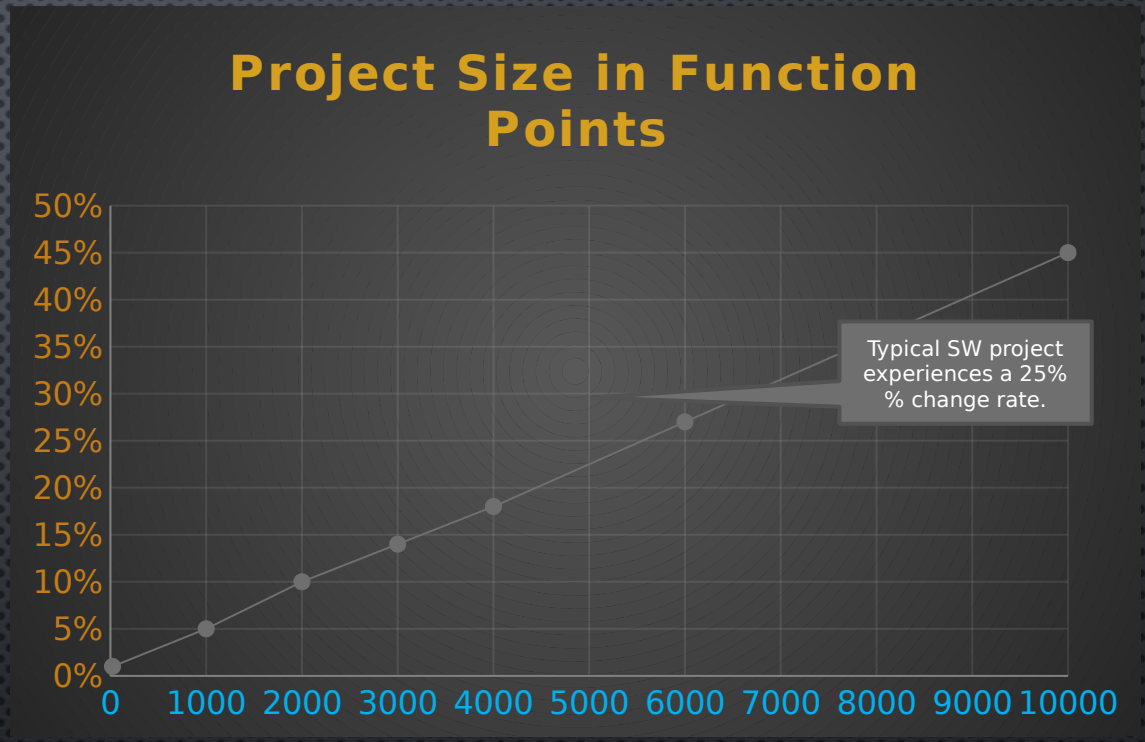
Cost	<0.5 M	0.5M-3M	3M - 6M	6M-10M	>10 M
Success	68%	22%	9%	1%	0%

UP, XP, & SCRUM

1. RESEARCH EVIDENCE – *CHANGE RESEARCH*

THE FOLLOWING GRAPH IS BASED ON RESULTS FROM MULTIPLE LARGE-SCALE SOFTWARE DEVELOPMENT PROJECTS. [JONES97]

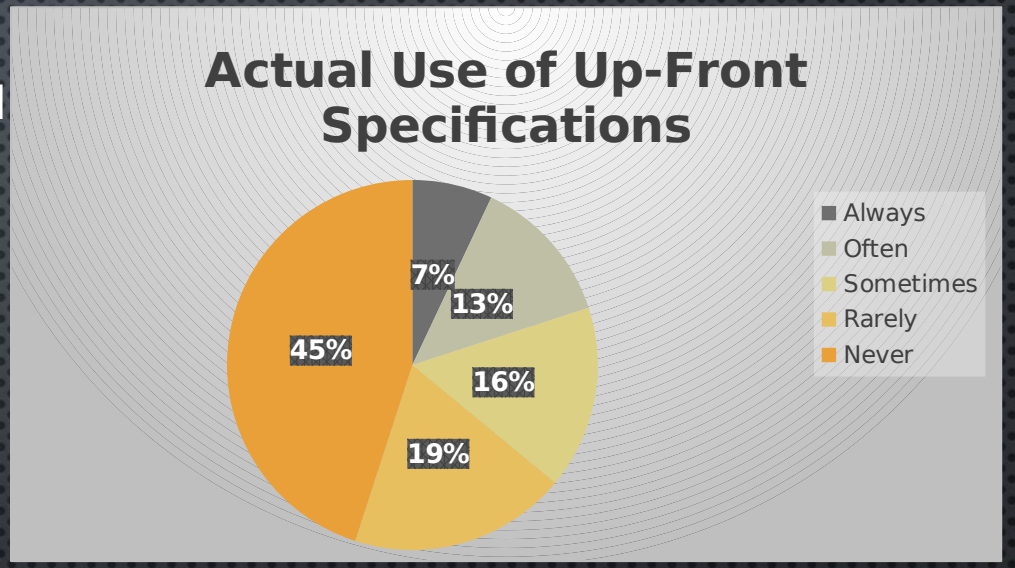
- IT ILLUSTRATES THAT AS THE COMPLEXITY OF THE PROJECT INCREASES (FUNCTION POINTS) THE AMOUNT OF REQUIREMENT CHANGE (OR CREEP) ALSO INCREASES.
- MEDIUM SIZE PROJECTS HAVE A CHANGE RATE OF 25%
- LARGE SIZE PROJECTS HAVE A CHANGE RATE OF 35%



THIS ENFORCES THE CONCEPT THAT AN ITERATIVE LIFECYCLE MODEL HAS A BETTER CHANCE OF SUCCESS THAN A SEQUENTIAL LIFECYCLE MODEL SINCE IT CAN ADAPT BETTER TO CHANGING REQUIREMENTS, FOCUSES ON ARCHITECTURE AND HIGH-RISK REQUIREMENTS EARLY, HAS A BETTER PRODUCTIVITY RATE, AND A PRODUCES A HIGHER QUALITY PRODUCT (ONE WITH FEWER DEFECTS).

1. RESEARCH EVIDENCE – *CHANGE RESEARCH*

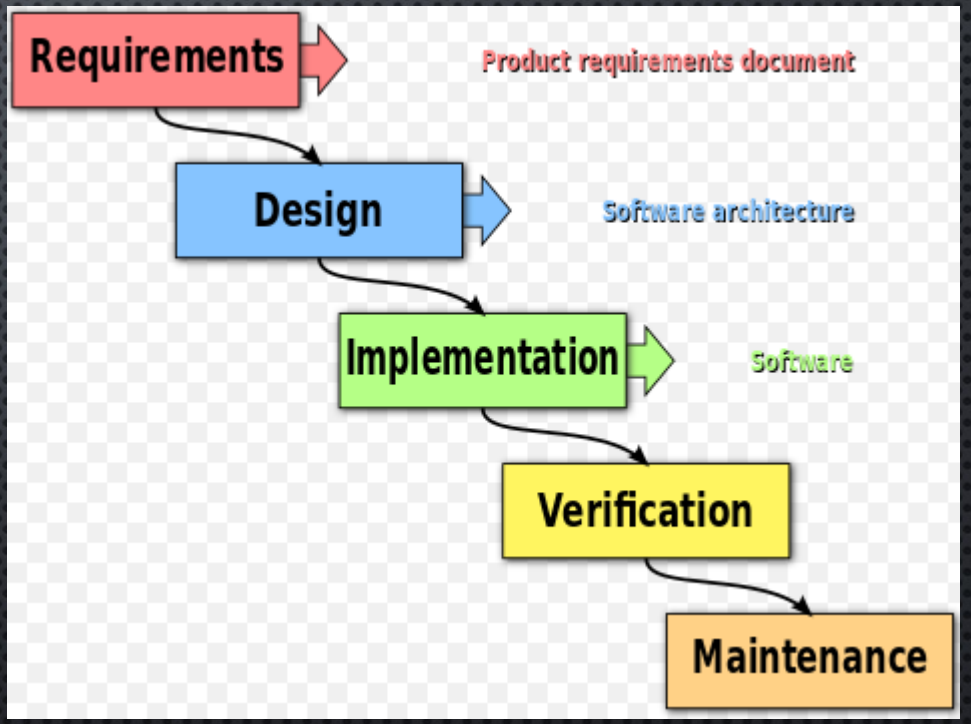
- IN ANOTHER STUDY THE AUTHOR STATES THAT UP-FRONT SPECIFICATION WITH A SIGN-OFF CAN NOT BE SUCCESSFULLY CREATED AND THAT A STUDY SHOWED THAT 45% OF THE FEATURES CREATED FROM EARLY SPECIFICATION WERE NEVER USED, WITH AN ADDITIONAL 19% RARELY USED [JOHNSON02]
- THE AUTHOR THEN PROCEEDED TO SAY "AVOID PREDICTIVE PLANNING BECAUSE YOU CAN NOT SIMPLY PLAN THE WORK AND WORK THE PLAN" WHEN DOING ITERATIVE SOFTWARE DEVELOPMENT.
- THIS WILL ONLY WORK IF YOU PROJECT IS NOT FIRM FIXED PRICE OR IF YOUR CUSTOMER HAS BOUGHT INTO THE IDEA OF YOU DELIVERING A SYSTEM WITH ONLY 75%-95% OF THE FEATURES THEY CONTRACTED!



1. RESEARCH EVIDENCE – *WATERFALL FAILURE RESEARCH*

THE AUTHOR PROVIDED NUMEROUS EXAMPLES OF STUDIES SHOWING HOW MOST PROGRAMS FOLLOWING THE WATERFALL LIFE-CYCLE MODELED FAILED.

1. IN A STUDY OF 1,027 IT PROJECTS IN THE UK [THOMAS01]THAT USED THE WATERFALL METHODOLOGY; 87% OF THE PROJECTS FAILED. OF THESE FAILED PROJECTS, 82% CITED THE NUMBER ONE PROBLEM WAS DEVELOPING ALL THE REQUIREMENTS UP FRONT.
2. PREVIOUSLY THE DEPARTMENT OF DEFENSE (DOD) REQUIRED PROJECTS TO ADHERE TO STANDARD DOD-STD-2167 WHICH REQUIRED THE USE OF THE WATERFALL LIFECYCLE MODEL. THIS RESULTED IN 75% OF THE DOD PROJECTS FAILING OR NEVER BEING USED.
3. ONE STUDY [JARZOMBEK99] FOUND THAT EVEN THOUGH 46% OF THE SYSTEMS DEVELOPED FOR THE DOD MET THE SPECIFICATIONS, THEY FAILED TO MEET THE REAL NEEDS OF THE CUSTOMER AND WERE NEVER SUCCESSFULLY USED.
4. ANOTHER STUDY IDENTIFIED THAT THE INABILITY TO DEAL WITH CHANGING REQUIREMENTS AND LATE INTEGRATION WERE ALSO SIGNIFICANT CONTRIBUTORS TO FAILED PROJECTS [JONES95].



Wikibooks Creative Commons

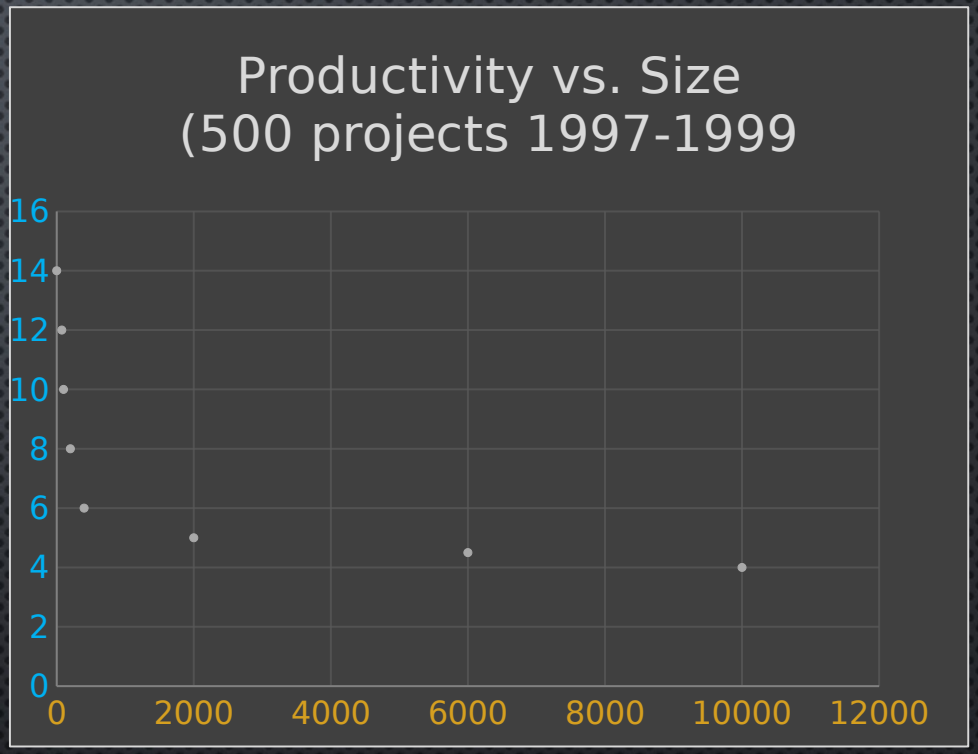
1. RESEARCH EVIDENCE – *PRODUCTIVITY RESEARCH*

- IN A STUDY [JONES00] COMPARED 500 PROJECTS FROM 1997-1999 AND FOUND THAT AS THE SIZE OF FUNCTION POINTS IN A PROJECT INCREASES, THE MONTHLY PRODUCTIVITY OF THE STAFF DECREASES.

  THIS MEANS THAT PROJECTS WITH 1,000 OR FEWER FUNCTION POINTS ARE THE MOST PRODUCTIVE AS SHOWN IN THE GRAPH TO THE LEFT.

- IN A STUDY[MARTIN91] FOUND THAT TIMEBOXING ITERATIONS ALSO SIGNIFICANTLY INCREASED PRODUCTIVITY.

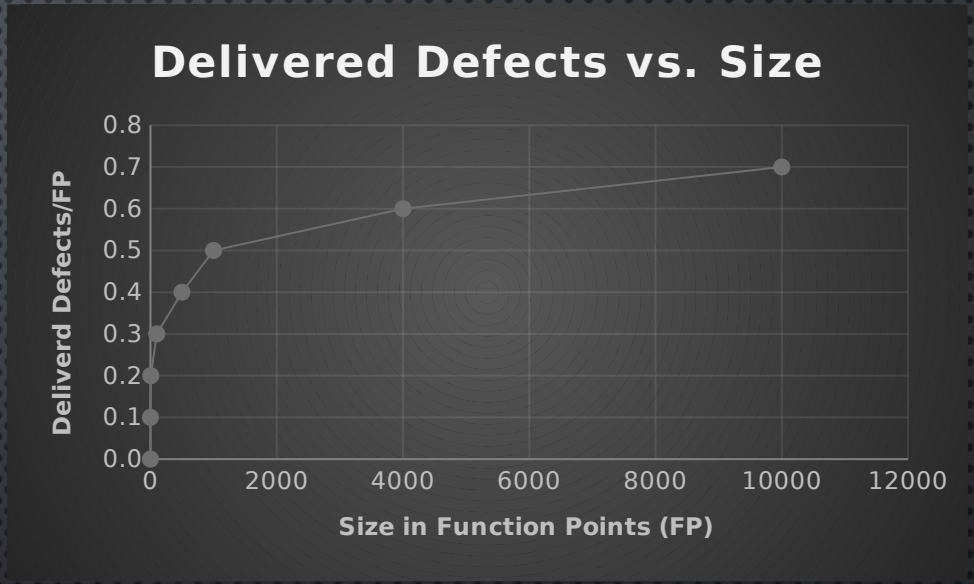
- IN ANOTHER STUDY [JONES00] FOUND THAT PRODUCTIVITY IS ALSO IMPACTED BY COMPLEXITY AS SHOWN IN THE TABLE TO THE LEFT.



	Low Complexity	High Complexity
Productivity	13%	-35%

1. RESEARCH EVIDENCE – *QUALITY & DEFECT RESEARCH*

- DEFECT REDUCTION COMES FROM AVOIDING DEFECTS BEFORE THEY OCCUR (DEMING'S TOTAL QUALITY MANAGEMENT PRINCIPAL) AND FROM FEEDBACK (PEER REVIEWS, TEST, DEMOS ETC.)
- THE AUTHOR POINT TO VARIOUS STUDIES TO SHOW THE BENEFITS OF IID:
  - [MKCC03] INDICATING IID WAS CORRELATED TO LOWER DEFECTS,
  - [MV101] INDICATING THAT DUE TO LESS TIME BETWEEN CODING AND TESTING, DEFECT RATES DECREASE,
  - [DECK94] SHOWS A STATISTICALLY SIGNIFICANT REDUCTION IN DEFECTS USING AN IID APPROACH.
- IID METHODOLOGIES:
  - ENCOURAGE CONTINUOUS PROCESS IMPROVEMENT BY MEASURING, REFLECTING, AND ADJUSTING EACH ITERATION.
  - EMPHASIZES EARLY DEVELOPMENT OF RISKY ITEMS, DEMOS, AND TEST-DRIVEN DEVELOPMENT.



The author then points to a large case study by [Jones00] that show defect rates increase non-linearly as the project size grows.
Finally, the author states several case studies report lower defect densities are associated with IID methods [Manzo02], *however they are NOT statistically reliable!*

2. EARLY HISTORICAL PROJECT EVIDENCE

PRE-1970:

1958 PROJECT MERCURY – USED ITERATIVE DEVELOPMENT TO SUCCESSFULLY BUILD THE SYSTEM INCREMENTALLY.

1970S:

THIS PROJECT LAID THE FOUNDATION FOR THE IBM FEDERAL SYSTEMS DIVISION (FSD) WHICH BUILT MANY AEROSPACE AND DEFENSE SYSTEMS THROUGHOUT THE 1970S INCLUDING:

THE US TRIDENT SUBMARINE (1972) WHICH USED 4 TIMEBOXED ITERATIONS OF 6 MONTHS IN DURATION.

THE TRW/ARMY SITE DEFENSE SOFTWARE PROJECT FOR BALLISTIC MISSILE DEFENSE WHICH DEVELOPED THE SYSTEM IN 5 ITERATIONS WITHOUT TIMEBOXING.

US NAVY HELICOPTER-SHIP SYSTEMS LAMPS THAT USED 45, 1-MONTH ITERATIONS TO SUCCESSFULLY DEVELOP THE SYSTEM.

1977-1980 - PRIMARY AVIONICS SOFTWARE SYSTEM FOR THE SPACE SHUTTLE WAS BUILT IN 17 ITERATIONS OVER 31 MONTHS AVERAGING 8 MONTHS/ITERATION.

EVERYONE OF THESE SYSTEMS WERE DEVELOPED ON TIME AND UNDER BUDGET.

2. EARLY HISTORICAL PROJECT EVIDENCE

1980S:

1984-1988 – MAGNAVOX ELECTRONIC SYSTEMS ARTILLERY COMMAND AND CONTROL SYSTEM FOR THE US ARMY WAS BUILT IN 5 ITERATIONS.

1983-1994 – US AIR TRAFFIC CONTROL (ATC) WAS RUN USING THE TRADITIONAL WATERFALL MODEL. IT FAILED DUE TO LACK OF STAKEHOLDER FEEDBACK, ANALYSIS PARALYSIS, COMPLEXITY OVERLOAD, ETC. THIS PROJECT WAS RESTARTED USING ITERATIVE DEVELOPMENT AND SUCCEEDED.

1990S:

THE CANADIAN AIR TRAFFIC CONTROL (CAATS) PROJECT IS ANOTHER EXAMPLE OF A FAILED PROGRAM THAT WAS RE-STARTED USING A UNIFIED PROCESS APPROACH WITH 6-MONTH ITERATIONS, A STAFF OF SEVERAL HUNDRED DEVELOPERS, AND OVER 1-MILLION LINES OF CODE (ADA). THE PROGRAM WAS SUCCESSFUL.

- THE PROGRAM WAS DEVELOPED BY A TEAM OF ENGINEERS THAT ORIGINATED AT HUGHES AIRCRAFT COMPANY, FULLERTON CA. AFTER WINNING THE CONTRACT THE TEAM WAS RE-LOCATED TO CANADA TO FORM HUGHES CANADA.
- THIS COMPANY WAS BOUGHT BY RAYTHEON AND TURNED INTO RAYTHEON CANADA.

3. STANDARDS-BODY EVIDENCE

TRANSITION OF US DOD STANDARDS FROM WATERFALL (1980) TO ITERATIVE AND EVOLUTIONARY (TODAY)

1980 – DOD-STD-2167 REQUIRED SOFTWARE DEVELOPMENT TO USE A WATERFALL LIFE-CYCLE MODEL AND FOLLOW A DOCUMENTATION DRIVEN APPROACH.

1988 – DOD-STD-2167A REVISED DOD-STD-2167 TO ENCOURAGE IID ALTERNATIVES TO THE WATERFALL LIFE-CYCLE MODEL. HOWEVER, SINCE THE STANDARD STILL FOCUSED ON A DOCUMENT DRIVEN APPROACH TO DEVELOPMENT, MANY CONTRACTS STILL INTERPRETED IT AS IMPLYING THEY SHOULD CONTINUE USING THE WATERFALL LIFE-CYCLE MODEL.

1994 – MIL-STD-498 SUPERSEDED DOD-STD-2167A. THIS STANDARD PROMOTED AN EVOLUTIONARY REQUIREMENTS AND DESIGN APPROACH FOR ALL INCREMENTAL ITERATIONS.

2002 THE US FOOD AND DRUG ADMINISTRATION (FDA) ALSO UPDATED THEIR STANDARDS TO ELIMINATE THE REQUIREMENT OF FOLLOWING THE WATERFALL LIFE-CYCLE MODEL AND REPLACED IT WITH ITERATIVE DEVELOPMENT.

ALTHOUGH MANY EUROPEAN STANDARDS STILL REQUIRE THE USE OF THE WATERFALL LIFE-CYCLE MODEL; NATO MADE THE LEAP TO EVOLUTIONARY DEVELOPMENT IN 2002.

4. EXPERT AND THOUGHT LEADER EVIDENCE

THE AUTHOR IDENTIFIES EXPERTS IN THE FIELD IDD INCLUDING:

HARLAN MILLS – WHO WORKED AT IBM FSD IN 1970. MILLS WAS A MAJOR CONTRIBUTOR TO THE CONCEPT OF STRUCTURED PROGRAMMING, TOP-DOWN DESIGN PROGRAMMING, AND INCREMENTAL DEVELOPMENT. MILLS STATED THAT *"SOFTWARE DEVELOPMENT SHOULD BE DONE INCREMENTALLY IN STAGES WITH CONTINUOUS USER PARTICIPATION AND REPLANNING WITH DESIGN TO COST PROGRAMMING WITHIN EACH STAGE".*

TOM GILB – PROMOTED THE "EVO" ITERATIVE METHOD IN 1976. GILB FOCUS WAS TO BREAK COMPLEX SYSTEMS DOWN INTO SMALL STEPS THAT HAD A CLEAR MEASURE OF SUCCESS. AN ADVANTAGE OF THIS APPROACH WAS THAT IF A STEP FAILED, IT GAVE YOU AN OPPORTUNITY TO INCORPORATE FEEDBACK, ADAPT, AND CONTINUE WITH THE SOFTWARE DEVELOPMENT.

FREDERICK BROOKS – RECOMMEND AN IID APPROACH TO SOFTWARE DEVELOPMENT OVER USE OF THE WATERFALL METHOD IN 1987 STATING THAT UP-FRONT REQUIREMENT SPECIFICATIONS WERE TO BLAME FOR THE HIGH PERCENTAGE OF PROGRAM FAILURES. BROOKS ALSO PUBLISHED THE BOOK TITLED – *THE MYTHICAL MAN MONTH* IN 1985 THAT STATED, *"ADDING MANPOWER TO A LATE SOFTWARE PROJECT MAKES IT LATER".*

BARRY BOEHM – PUBLISHED THE BOEHM'S SPIRAL MODEL THAT PROMOTED ITERATIVE DEVELOPMENT IN 1985.

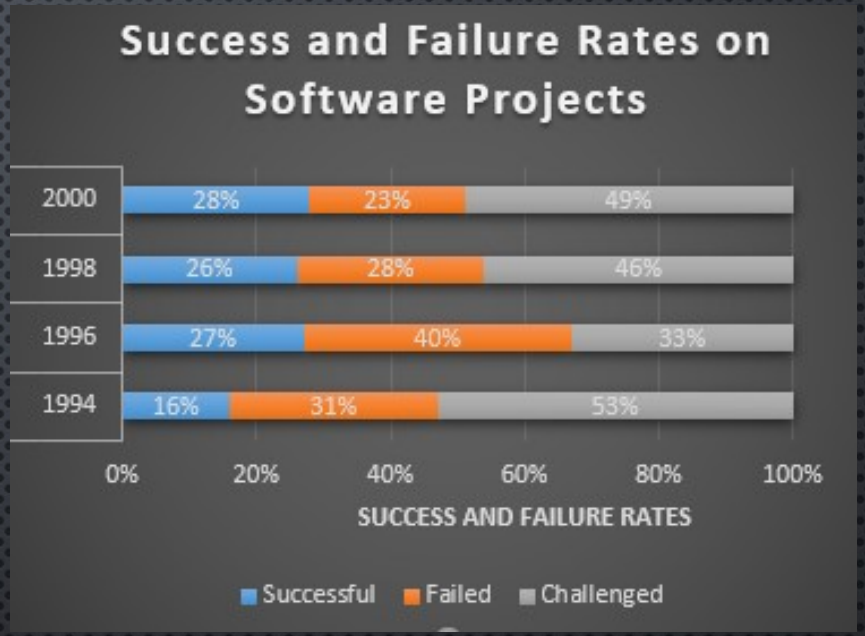
JAMES MARTIN – PROMOTED TIMEBOXED ITERATIVE DEVELOPMENT WITH CUSTOMER PARTICIPATION IN THE 1980S. MARTIN BELIEVED THAT RAPID APPLICATION DEVELOPMENT (RAD) WAS A METHOD TO UNDERSTAND LARGE COMPLEX SYSTEMS. THIS METHOD FOCUSED ON CREATING A PRODUCTION-GRADE PROTOTYPE, LEARNING FROM IT, AND EVOLVING IT UNTIL IT PRODUCED A PRODUCT THAT THE END USER WANTED.

TOM DEMARCO – IDENTIFIED IID METHODOLOGY AS A RISK MITIGATION TECHNIQUE IN 2003.

5. A BUSINESS CASE FOR ITERATIVE DEVELOPMENT

A BUSINESS CASE FOR ITERATIVE DEVELOPMENT CAN BE MADE BASED ON SEVERAL FACTORS INCLUDING:

- PRODUCTIVITY (INCREASES)
- QUALITY (PROCESS AND PRODUCT)
- LESS FAILURES, LESS COST/SCHEDULE IMPACT
- CUSTOMER SATISFACTION (END-PRODUCT MEETS EXPECTATIONS)



BASED ON THE GRAPH TO THE LEFT, IF A COMPANY AVERAGED 10 PROJECTS A YEAR AND EACH PROJECT COST $1M. THEN IN THE 2000 A COMPANY WOULD STAND TO LOOSE 23% OR $2.3M FROM FAILED PROJECTS AND HAVE COST OVERRUNS IN 49% OF THE OTHER PROJECTS.

BY ADOPTING IID METHODOLOGIES BOTH PROJECT FAILURE RATES AND CHALLENGED RATES WOULD BE REDUCED, THEREBY INCREASING THE COMPANY'S PROFITABILITY.

6. THE HISTORICAL ACCIDENT OF WATERFALL VALIDITY

WINSTON ROYCE - PUBLISHED A PAPER IN 1970 TITLED *"MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS (LSS)"* THAT RECOMMEND TO DO THE WATERFALL PROCESS TWICE. TO FIRST HAVE A THROW-AWAY PROTOTYPE EFFORT PRIOR TO IMPLEMENTING THE PROJECT WHEN THERE ARE UNKNOWN FACTORS.

DOD-STD-2167 WAS ADOPTED IN THE 1980S WHICH REQUIRED THE USE OF THE WATERFALL MODEL COMBINED WITH DOCUMENT-DRIVEN REVIEWS. MOST IMPLEMENTORS OF LSS LOST SIGHT OF THE NEED TO PROTOTYPE WHEN UNKNOWN FACTORS ARE INVOLVED.

MANY OTHER STANDARDS WERE BASED ON DOD-STD-2167

WATERFALL WAS SIMPLE: DO REQUIREMENTS, DESIGN, AND IMPLEMENTATION.

WATERFALL GAVE THE ILLUSION OF AN ORDERLY, PREDICTABLE, ACCOUNTABLE, AND MEASURABLE PROCESS WITH SIMPLE DOCUMENT DRIVEN MILESTONES.

UP-FRONT SPECIFICATIONS WERE PROMOTED BY SYSTEM ENGINEERING ORGANIZATIONS.

CMMI INFLUENCED ORGANIZATIONS TO FOLLOW A DOCUMENT DRIVEN DEVELOPMENT WHICH WAS IN LINE WITH A WATER FALL METHODOLOGY.

WordPress.com - Creative Commons

REFERENCES

AGILE & ITERATIVE DEVELOPMENT, A MANAGER'S GUIDE, CRAIG LARMAN, EIGHTH EDITION, ADDISON WESLEY, NEW YORK, NY, COPYRIGHT 2004 BY PEARSON EDUCATION , INC.

SPECIFIC SOURCES THE AUTHOR QUOTED:

[DECK94] – DECK, M. 1994. "CLEANROOM SOFTWARE ENGINEERING: QUALITY IMPROVEMENT AND COST REDUCTION." *PROCEEDINGS, 12TH PACIFIC NORTHWEST SOFTWARE QUALITY CONFERENCE.*

[JARZOMBEK99] – JARZOMBEK, J 1999. *THE 5TH ANNUAL JAWS S3 PROCEEDINGS.*

[JOHNSON02] - JOHNSON, J. 2002. KEYNOTE SPEECH, XP 2002, SARDINIA, ITALY

[JONES00] – JONES, C. 2000. *SOFTWARE ASSESSMENTS, BENCHMARKS, AND BEST PRACTICES*. ADDISION-WESLEY.

[JONES95] –JONES, C. 1995. *PATTERNS OF SOFTWARE FAILURE AND SUCCESS.* INTERNATIONAL THOMPSON PRESS.

[JONES97] - JONES, C. 1997. *APPLIED SOFTWARE MEASUREMENTS*. MCGRAW HILL.

REFERENCES

SPECIFIC SOURCES THE AUTHOR QUOTED:

[MACCORMACK01] – MAC CORMACK, A. 2001. PRODUCT-DEVELOPMENT PRACTICES THAT WORK.” *MIT SLOAN MANAGEMENT REVIEW*. 42(2)

[MANZO02] – MANZO, H, 2002. “ODYSSEY AND OTHER CODE SCIENCE SUCCESS STORIES.” CROSSTALK: THE JOURNAL OF DEFENSE SOFTWARE ENGINEERING, OCT. 2002, USA DOD.

[MARTIN91] – MARTIN, J. 1991. RAPID APPLICATION DEVELOPMENT. MACMILLAIN

[MKCC03]- MAC CORMACK, A. KEMERER, C., CUSUMANO, M., AND CRANDALL, B. 2003. “EXPLORING TRADE-OFFS BETWEEN PRODUCTIVITY & QUALITY IN SELECTION OF SOFTWARE DEVELOPMENT PRACTICES.” WORKING DRAFT SUBMITTED TO IEEE SOFTWARE.

[MV101] – MAC CORMACK. A., VERGANTI, R., AND IANSITI, M. 2001. “DEVELOPING PRODUCTS ON INTERNET TIME: THE ANATOMY OF A FLEXIBLE DEVELOPMENT PROCESS.” MANAGEMENT SCIENCE. JAN 2001.

[STANDISH 98] - JIM JOHNSON, ET. AL 1998. CHAOS: A RECIPE FOR SUCCESS, 1998. PUBLISHED REPORT. THE STANDISH GROUP

[THOMAS01] - THOMAS, M. 2001. “IT PROJECTS SINK OR SWIM” BRITISH COMPUTER SOCIETY REVIEW.