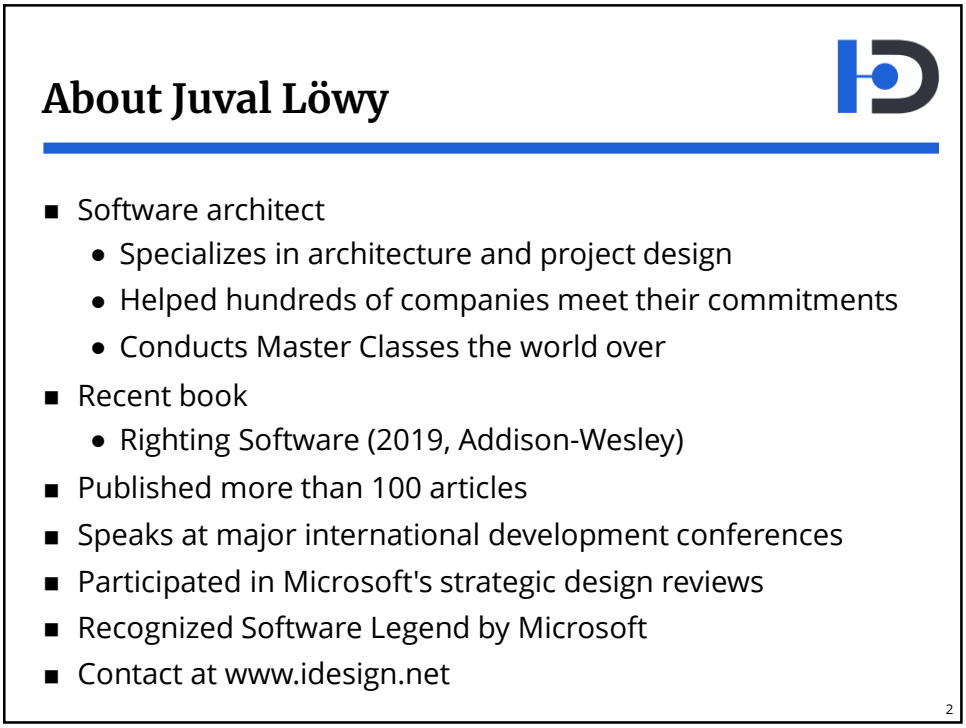


Software Development Crisis

- The software industry is in a deep crisis
  - Multi-dimensional crisis
- Cost
  - Initial development
  - Overall ownership
  - Unacceptable cost of changes leads to "clean slate"
  - `$ > ;`
- Schedule
  - Time to market
  - Time to feature
  - Slips

3

Software Development Crisis

- Requirements
  - Solving the wrong problems
  - System obsolete by release time due to requirements change
- Staffing
  - Unmaintainable by new staff

4

Software Development Crisis

- Quality
  - Up to unusable systems
  - Customer satisfaction

5

Righting the Industry

- Requires fixing all dimensions of the crisis
  - Systems and projects
- Structured engineered methodology
  - The Method

6

Agenda

- System Design
  - 9:00-12:00
- Lunch
  - 12:00-1:00 PM
- Project Design
  - 1:00-4:00 PM

7

System Design

Juval Löwy
www.idesign.net

Avoid Functional Decomposition

- Avoid functional decomposition
  - "Flow-chart" decomposition
  - Basing services on order of logical steps in use cases
  - Functional and time decomposition
- Leads to duplicating behaviors across services
- Extreme edges of the cost/size curve
  - Explosion of services
  - Bloating of services
  - Intricate relationships inside and between

9

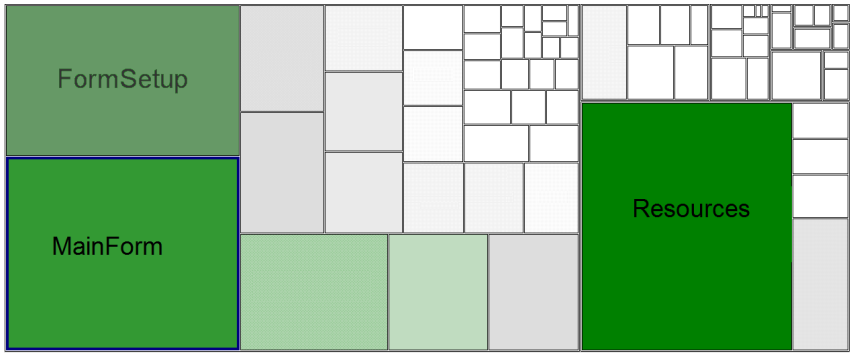
Avoid Functional Decomposition

- Pollutes clients with business logic
  - Implementing uses cases in higher level terms
- Prevents reuse
  - Couples services to order and current use cases
- Prevents single point of entry

10

Avoid Functional Decomposition

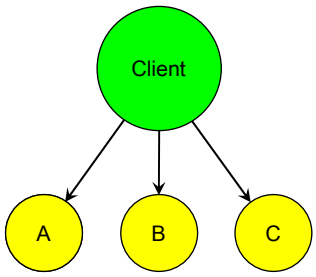
- Makes services too big or too small
  - Often side by side



11

Avoid Functional Decomposition

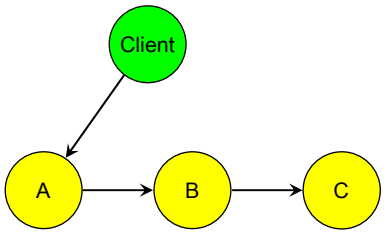
- Clients stitch services
  - Too small services
  - Bloated client



12

Avoid Functional Decomposition

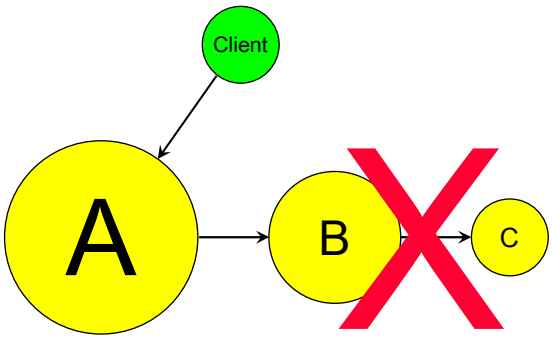
- Too big and bloated
  - Chaining services



13

Avoid Functional Decomposition

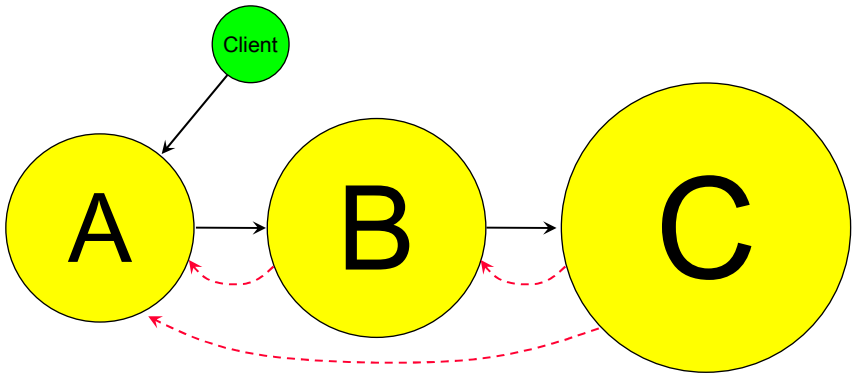
- Too big and bloated
  - Contain details on downstream calls



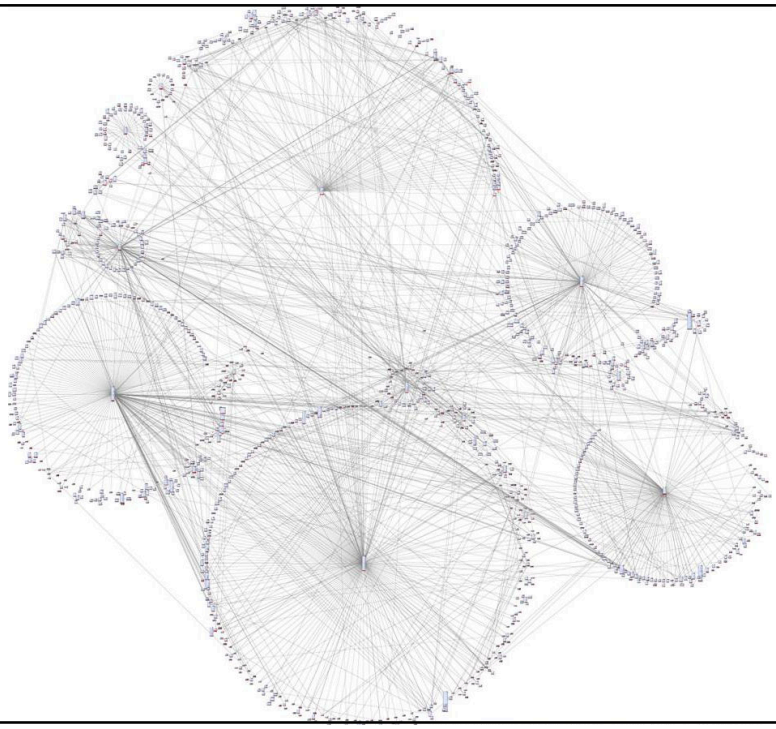
14

Avoid Functional Decomposition

- Too big and bloated
  - Contain details on downstream calls and compensation



15



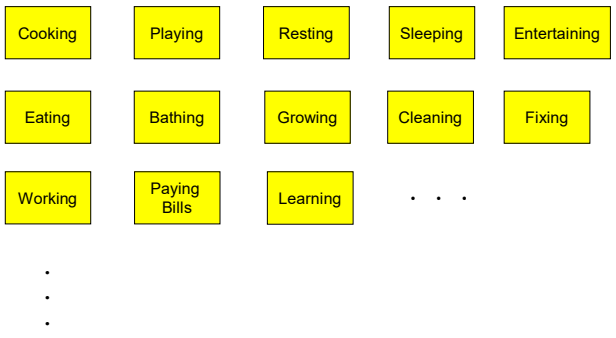
Avoid Functional Decomposition

- Functional decomposition means design adds no value
- Consider performing anti-design effort
- Think about building a house functionally

17

Avoid Functional Decomposition

- Functional decomposition of a house



18

Avoid Functional Decomposition

- How many sub services are under **Cooking**?
  - Breakfast
  - Lunch
  - Dinner
  - Guests
  - Snacks
  - Healthy
  - …
- How would you implement just cooking?

19

Avoid Functional Decomposition

- Domain decomposition of a house

20

Avoid Functional Decomposition

- How would you implement just the kitchen?
- Pumping and bloating
  - Operations grouped based on business objects they require
- Functionality is often spread across multiple areas
  - Composed of smaller functionalities
  - Change involves multiple places
- Brown-field systems often require side-by-side deployment of new system
  - Reconciliation challenge often greater than actual system's
- Regression testing is next to impossible

21

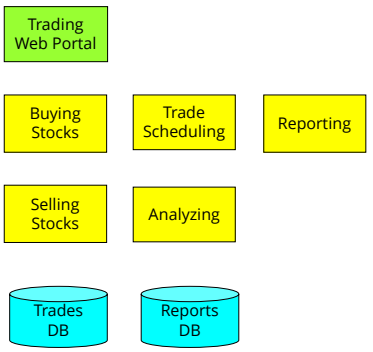
Avoid Functional Decomposition

- Example: stocks trading system
- Requirements
  - Enable traders to buy and sell stocks
  - Schedule trades
  - Issue reports
  - Analyze trades
  - Client is browser
    - Connected sessions
  - After trade/report sends email confirming request or results
  - Data is stored in database

22

Avoid Functional Decomposition

- Naïve functional decomposition



23

Avoid Functional Decomposition

- Client orchestrates Selling Stocks, Buying Stocks, Scheduling Trades, Reporting
  - What will it take to change client to a mobile device?
  - Impact on components
  - Chances of client reuse slim
- Changing email to SMS impacts all components
- Switching to cloud storage impact data access code of all
- Async interactions requires rewrite of all components
- Changing trade items requires new components or massive re-write of old ones
- Changing feed impacts all
- Globalization impacts all

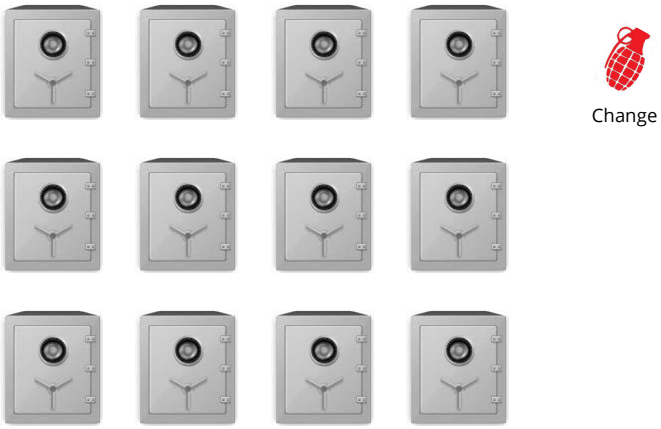
24

Volatility-Based Decomposition

- Decompose based on volatility
  - Identify areas of potential change
    - Can be functional but not domain functional
  - Encapsulate in services
- Implement behavior as interaction between services and subsystems

25

Volatility-Based Decomposition



26

Volatility-Based Decomposition

- Universal principle of good design
- Encapsulate change to insulate
  - Do not resonate with change
- Functional decomposition maximizes impact of change
  - Coupled to it

27

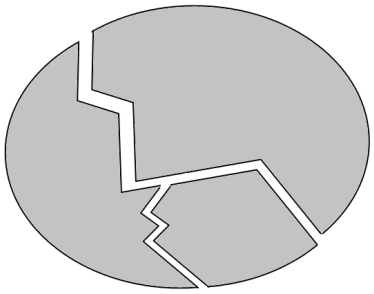
Volatility-Based Decomposition

- Volatility is often not-self evident
  - Takes longer than functional
  - Getting management support
  - Fighting insanity
- Dunning–Kruger effect
- Doctors of old

28

Volatility-Based Decomposition

- Axes of volatility
  - At the same customer over time
  - At the same time across customers



30

Volatility-Based Decomposition

- Axes should be independent
  - Encapsulate from each other as well
  - When not, indicating functional decomposition

31

Volatility-Based Decomposition

- Volatility-encapsulating decomposition of a house
  - At the same customer over time

32

Volatility-Based Decomposition

- Volatility-encapsulating decomposition of a house
  - At the same time across customers

Structure Volatility | Neighbors Volatility | City Volatility

- Axes are independent
  - Neighbors more pronounced on second axis

33

Volatility-Based Decomposition

- Was cooking ever a requirement?
- Functional decomposition handles poorly solutions masquerading as requirements
- Volatility based decomposition is the proper way here as well

34

Volatility-Based Decomposition

- Prepare list of areas of volatility
  - Prior to architecture and decomposition
  - Normal part of requirements gathering and analysis
  - Ask what could change along axes of volatility

35

Volatility-Based Decomposition

- Example: the stocks trading system
- At the same time there are several types of users
  - Who is the user
  - From professional traders to end users reviewing portfolio
- Volatility in users often manifests in volatility in type of client application and technology
  - Web page
  - Mobile device
  - Rich desktop app
- How the users authenticate
  - And authorize

36

Volatility-Based Decomposition

- System notifications is volatile
  - Transport
    - Email
    - Paper mail
    - SMS
  - Publishers
  - Who receives and broadcast nature
- Where data is stored and how to access is volatile
  - Local database
  - Cache
  - Cloud

37

Volatility-Based Decomposition

- Type of connectivity and synchronicity
  - Connected, synchronous lock-step near real-time
  - Queued up trades
  - In or out of order
- Duration of interaction
  - Complete trade in one session
  - Long-running interaction spanning sessions and possibly multiple devices

38

Volatility-Based Decomposition

- The trade item is volatile
  - Over time may want to trade commodities or currencies
- Volatility in trade item implies
  - Volatility in frequency of the market data update
  - Processing workflow of trade itself
  - Analysis of trade
- The locale is volatile with implication on
  - Trading rules
  - UI localization
  - Possible trade items

39

Volatility-Based Decomposition

- Feed is volatile
  - Communication
  - Content
  - Format
  - Frequency
  - External or internal
  - Real or simulated

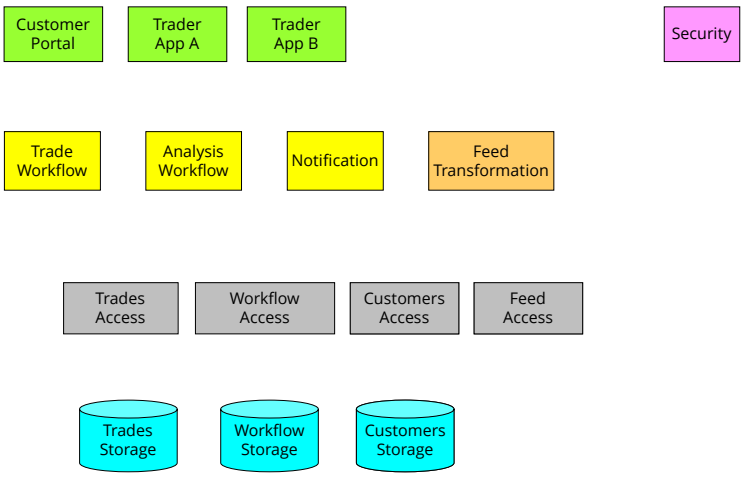
40

Volatility-Based Decomposition

- Observation
  - Not exhaustive list
  - Objective is thinking about what could change and mindset
- Some volatile areas may be out of scope
  - Or relate too much to nature of the business
- Vital to call out the areas of volatility as early as possible
  - Designating a component costs nothing
  - Later on may allocate effort to design and construction
- Once settled on the areas of volatility encapsulate them in components of the architecture

41

Volatility-Based Decomposition



42

Volatility-Based Decomposition

- Transition from list of areas of volatility to services is hardly ever pure 1:1
  - Sometimes a single service encapsulates multiple areas
  - Some areas may map to operational concept
  - May encapsulated in a third party component
- Some mapping is straightforward
- Data storage volatility encapsulated behind **Access Services**
  - Where the storage is
  - What technology is used to access it
  - Referring to the storage as **Storage** and not as Database

43

Volatility-Based Decomposition

- Notification volatility encapsulated in **Notification**
  - How to notify
  - Who is the client
  - Often simple pub/sub will do
- Volatility in trading workflow is encapsulated in **Trade Workflow**
  - What is being traded
  - Specific steps in buying or selling
  - Required customization for local markets
  - Reports
  - Workflow itself even if all else is constant
  - Uses some workflow tool persisting workflow

44

Volatility-Based Decomposition

- Persisting workflow encapsulates volatility
  - Duration of sessions
  - Connectivity type
  - Distributed sessions and multiple devices
- Volatility in analysis is handled exactly the same way as with the trading workflow
- Volatility in feeds is encapsulated in **Feed Transformation**
- **Security** component encapsulates volatility in ways of authenticating and authorizing users
  - Creds storage and access
  - Transfer security

45

Volatility-Based Decomposition

- Anything absent assumed not volatile enough
  - Nothing to encapsulate
- Resist the Siren song of bad habits
  - Tie yourself to the mast of VBD

46

Decomposition And Business

- Avoid encapsulating changes to the nature of the business
  - Hint – very rare indeed
  - Hint – each done poorly
  - The speculating design trap

Decomposition And Business

- Useful posture
  - Design both for you and your competitor
  - Identify nature of the business not to encapsulate
    - Not functionality

48

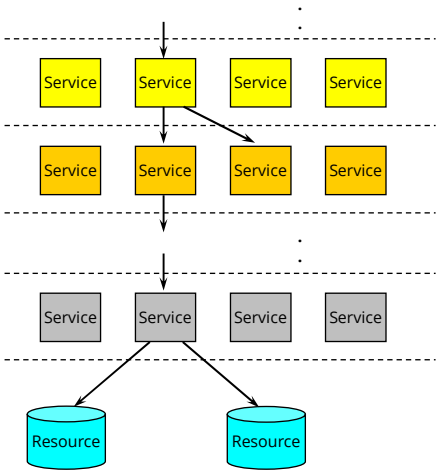
The Method

- Template for common areas to encapsulate
  - A good starting point
  - Encapsulate classic volatile areas
  - Typical runtime behavior and relationships
- Layers encapsulate top-down
- Services inside layers encapsulate sideways

49

Layered Approach

- Systems are typically designed in layers
  - Even simple systems
- Layers layer encapsulation
- All cross-layer entities are services



50

Layered Approach

- Cross-layer calls to services promote and enable
  - Consistency
  - Scalability
  - Fault isolation
  - Security
  - Separation of presentation from logic
  - Availability
  - Throughput
  - Responsiveness
  - Synchronization

51

Structure

- Client
  - AKA Presentation
  - Can be a user or another system
  - Can be variety of client application technologies
    - Encapsulate clients technology volatility
  - Advocate single point of entry to system

52

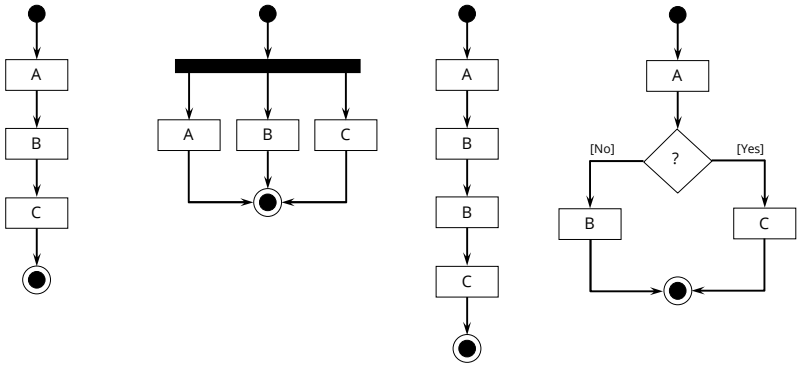
Structure

- Business
  - Managers encapsulate volatility in the sequence in use cases and workflows
    - Each manager is collection of related use cases
  - Engines encapsulates volatility in business rules and activities
  - Manager may use zero or more engines
  - Engines may be shared between managers

53

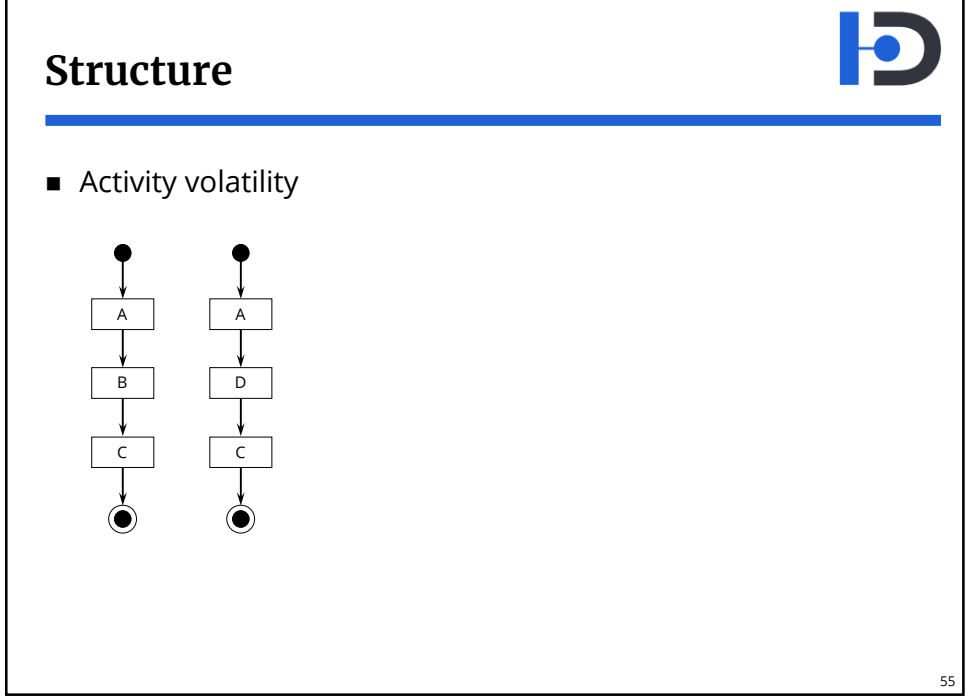
Structure

- Sequence volatility



54

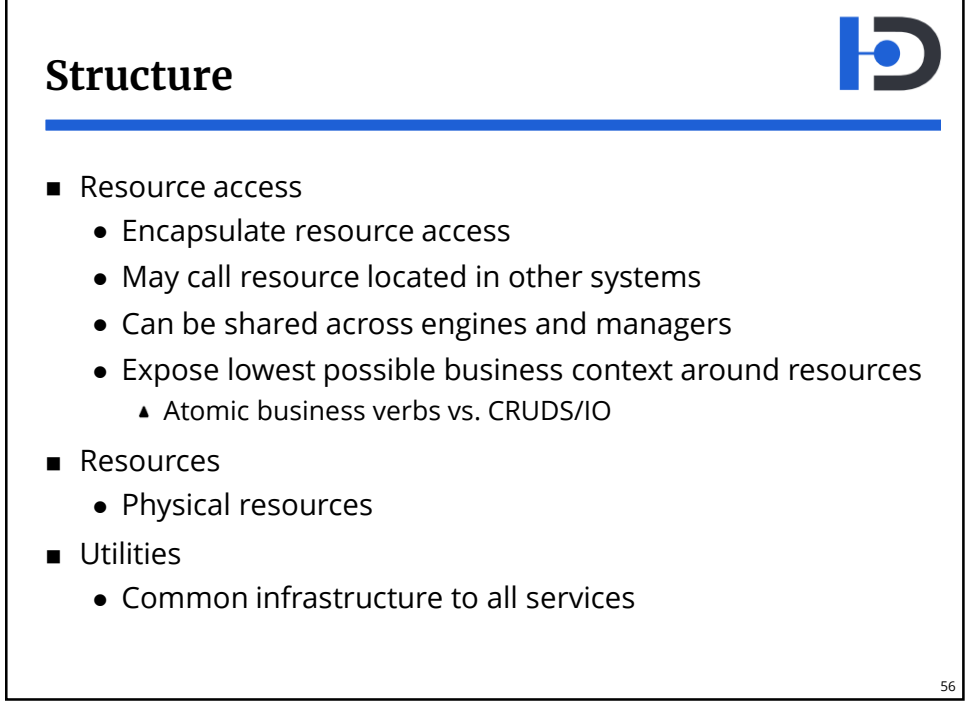
Structure



- Activity volatility

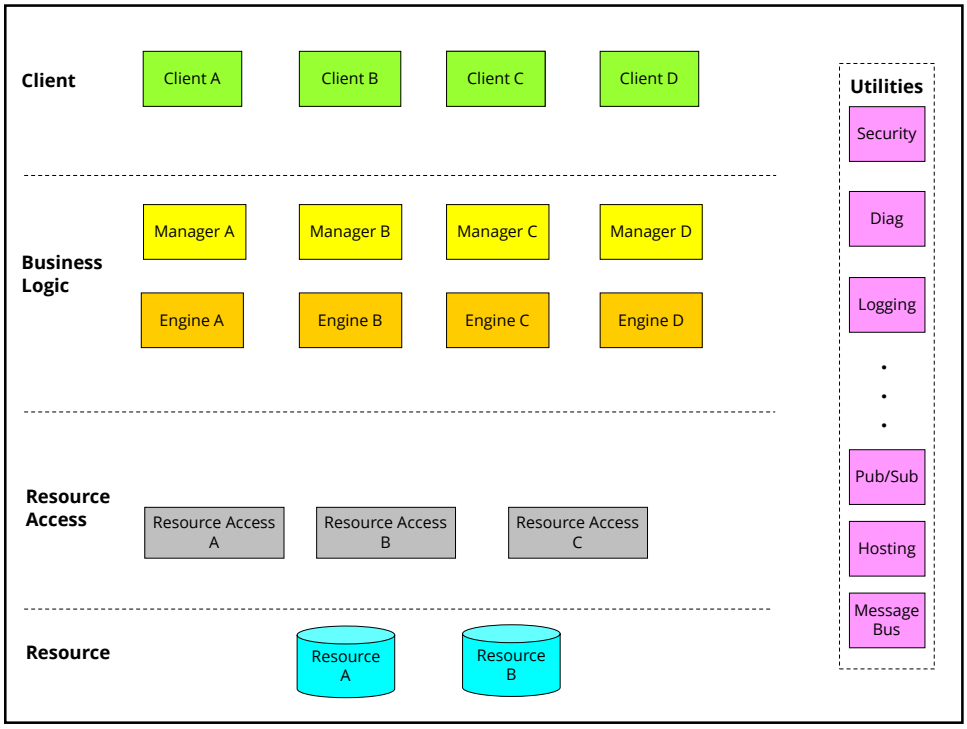
55

Structure



- Resource access
  - Encapsulate resource access
  - May call resource located in other systems
  - Can be shared across engines and managers
  - Expose lowest possible business context around resources
    - Atomic business verbs vs. CRUDS/IO
- Resources
  - Physical resources
- Utilities
  - Common infrastructure to all services

56



Containing Changes

- System must respond to changes in requirements
  - Live system
  - Fighting change kills the system
  - Respond quickly
- Trick is in containing the change
  - Minimize the impact
- Functional decomposition maximizes impact of change
  - Change is spread across multiple places

58

Containing Changes

- Change to use case means change to workflow
  - Manager implementation
  - Not underlying services
- Bulk of effort in system goes into
  - Engines
  - Resource access
  - Resource
  - Clients and UI
  - Utilities and infrastructure
- Essence of agility

59

IDesign

Project Design

Juval Löwy
www.idesign.net

What is Project Design

- Much as you design software system, must **design** the project
  - Accurately calculating duration and cost
  - Devising several good execution options
  - Validating your plan
- All stem from your system design
- Addressing all properly is a hard-core engineering task
  - Requires both architect and project manager to find best overall plan
    - Along with other options for management to choose from
- Project design to project management is what system architecture is to programming

61

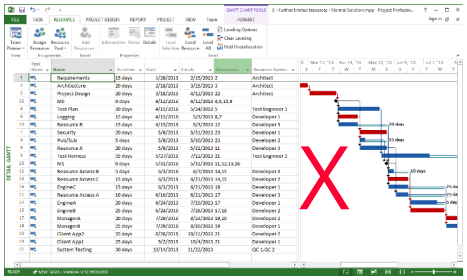
What is Project Design

- Beyond decision making
- Project design is also system assembly instructions
  - Would you buy an IKEA set without assembly instructions?
  - What about your software system?
- Assembling is project management
- Project design produces project plan
  - System design produces the architecture

62

Project Design Overview

- Staffing
- Estimations
- Network
- Schedule
- Cost
- Risk



63

Staffing

- If you want architecture get a competent architect!
- Requirements analysis and architecture are contemplative time consuming activities
  - Single architect is crucial for design integrity
    - Chimera
  - Accountability
  - More firepower does not expedite
  - Single architect usually suffices

Staffing

- In large projects, have a senior architect and junior/apprentice
  - Architects are rare resource
  - Work well together
  - Sounding board
  - Grooming
- Assign a service to individual developer (1:1)
  - Anything else is failure
  - Cannot reliably estimate as well

65

Core Team

- Architect cannot work in isolation
- Need a core team in place
  - Project manager
  - Architect
  - Product manager
- Three logical roles
  - Often map to three individuals
- Core team stays thought the project
  - Other resources are transient

66

Core Team

- Project manager shields team from organization
  - Most organizations create too much noise
  - Act as proxy for team
  - On-going through project
    - Status reports
    - Negotiations
    - Cross-organization constraints
- Architect decompose system into services
  - Project design upfront
  - On-going through project
    - Keeping system on the design

67

Core Team

- Product manager encapsulates customers
  - Proxy to customers
  - Resolves conflicts across customers
  - Negotiates requirements and priorities
  - Communicates expectations
- Core team stays throughout the project
  - Roles morph over time

68

Core Team

- Core team designs project in **fuzzy front end**
  - Requires 15-25% of project duration
    - Depending on constraints
    - From requirements gathering and analysis to system and project design
    - Design is not time consuming
- Reliably answering
  - How long it will take
  - How much it will cost
- Impossible to know without
  - Architecture
  - Project design

69

Educated Decisions

- Pointless to commit or approve project without
  - Cost
  - Schedule
  - Risk analysis
- Pointless to staff project with resources until committed

70

Educated Decisions

- Plan is not single pair of schedule/cost
  - Must provide management with several options trading schedule, cost, and risk
  - Most managers would not opt for cheapest or fastest option
    - Or safest or riskiest
- Whose fault is it when pair is impractical

71

Educated Decisions

- Management commits to plan in Feed Me/Kill Me point
  - Officially, SDP review
  - Even if your "process" does not have it
- Killing beneficial for all if unacceptable
  - CLM for core team
  - Opportunity cost

72

Estimations

- How long will it take?
- Must estimate effort to continue the design
  - Overall
  - For each activity in project
- Overall duration is not sum divided by staffing
  - Dependencies
  - Communication overhead

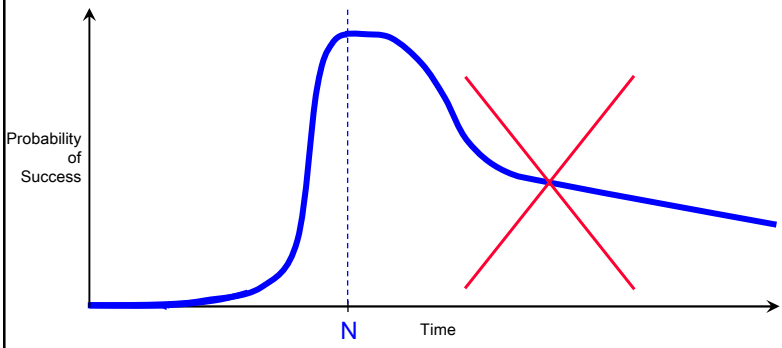
73

Estimations

- Both underestimation and overestimation are deadly
- Padding allows for increased complexity
  - Gold plating
  - Parkinson's Law
- Too aggressive schedule guarantees failure
  - Cutting corners and best practices
- Nominal estimation maximizes probability for success

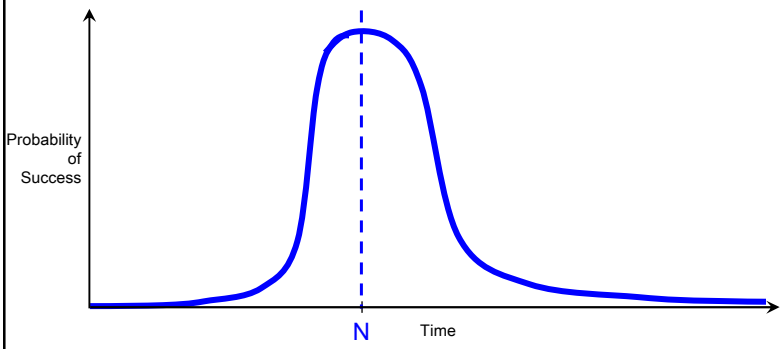
74

Estimations



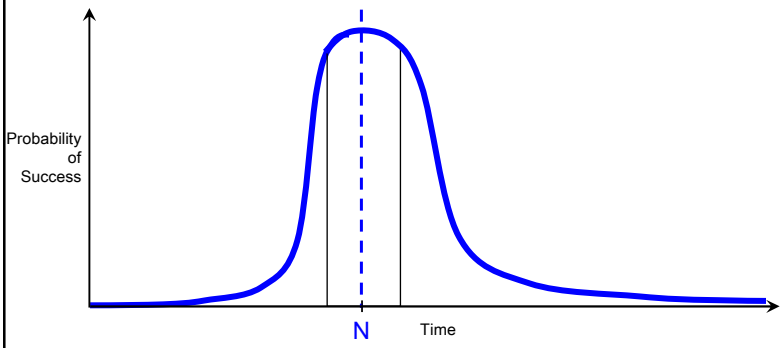
75

Estimations



76

Estimations



77

Estimations

- Good estimations are accurate
  - Not precise
- Do not confuse unknown with uncertain
- Help people overcome fear
  - Even refusal to estimate
    - "Don't know"
  - No entrapment

78

Estimations

- Confronted with uncertainty use orders of magnitudes
  - First to narrow type of units
  - Second to zoom in on value (**2X**)
- Use PERT to handle uncertainty
  - **E** calculated estimation
  - **T** optimistic estimation
  - **M** most likely estimation
  - **P** pessimistic estimation

$$E = \frac{T + 4*M + P}{6}$$

79

Estimations

- Estimation resolution is always quantum of 5 days
  - Simple
  - Aligns with week boundaries
  - Reduces waste in plan
  - Looks great in Microsoft Project
  - Allow errors compensation due to law of big numbers

80

Activity-Based Estimation

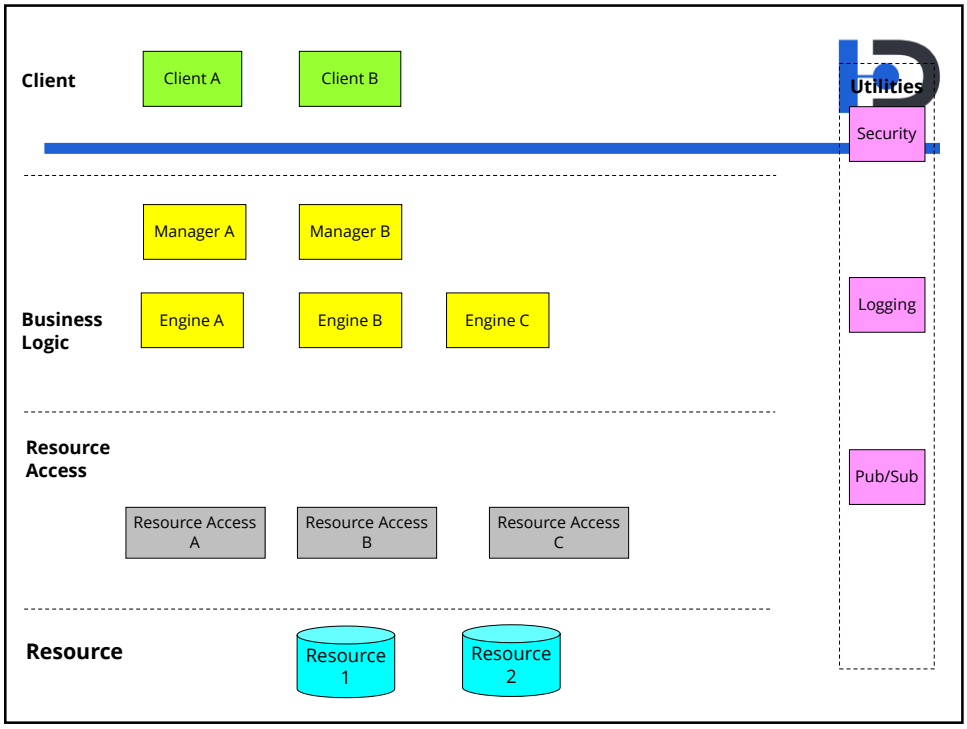
- Done individually per activity
  - Per owner if possible
- Maintain correct estimation dialog
  - Across all levels and roles
- Itemize lifecycle of all services
  - Do not omit:
    - Learning curves
    - Test clients
    - Installation
    - Integration points
    - Peer reviews
    - Documentation

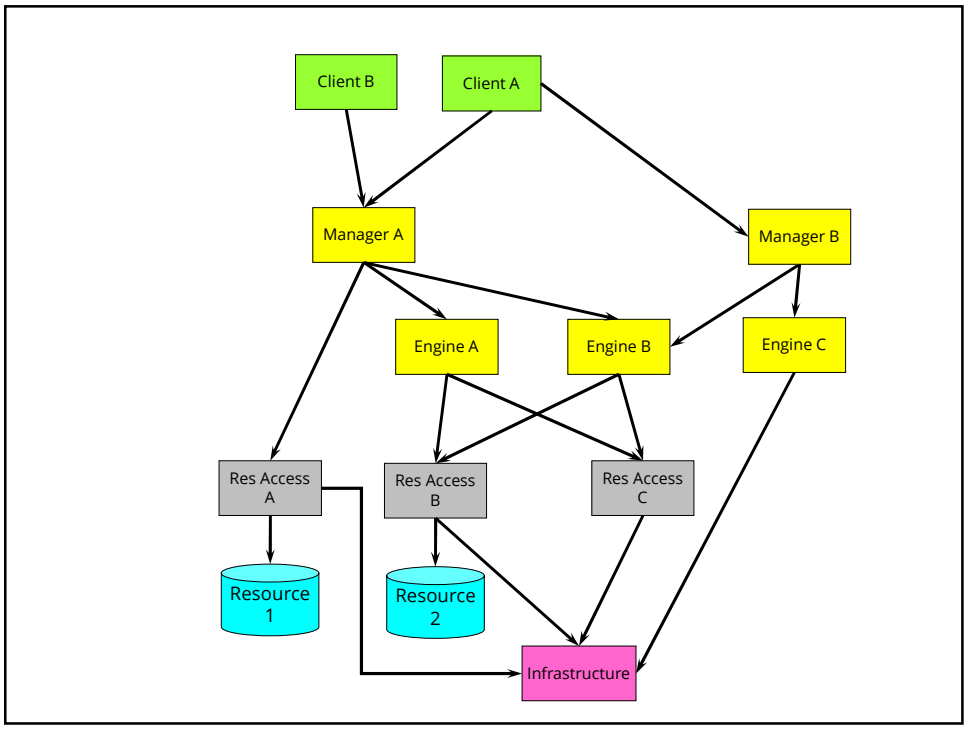
81

Critical Path Analysis

- Driven by architecture
- Pre-requisites
  - Decomposition into services
  - Dependencies tree
    - Driven by use-cases
  - List of non-coding activities
  - Effort estimation per activity
  - Available resources
    - Must repeat analysis per scenario

82

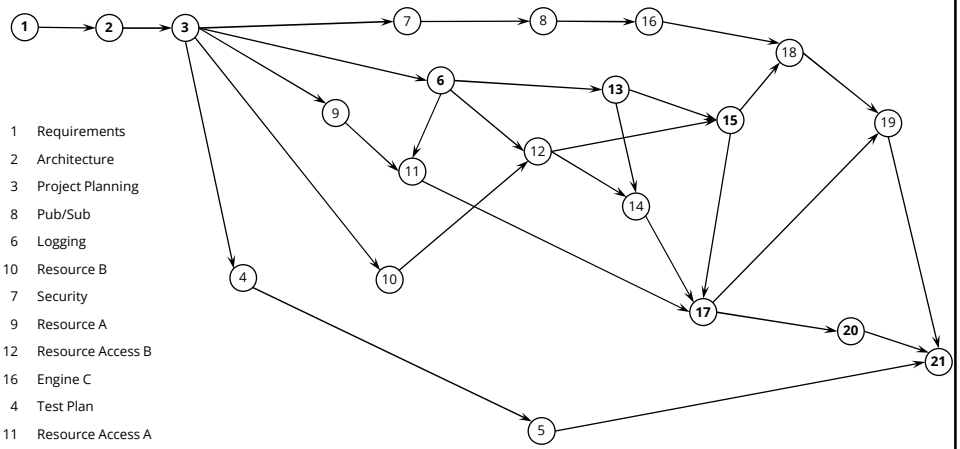




Project Network

- Convert dependencies tree into abstract graph
  - Add non-coding activities

85



1	Requirements
2	Architecture
3	Project Planning
8	Pub/Sub
6	Logging
10	Resource B
7	Security
9	Resource A
12	Resource Access B
16	Engine C
4	Test Plan
11	Resource Access A
13	Resource Access C
14	Engine A
15	Engine B
5	Test Harness
17	Manager A
18	Manager B
19	Client App1
20	Client App2
21	System Testing

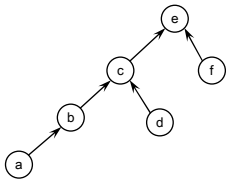
Project Network

- Time to finish each activity
  - Activity effort estimation + time to get to activity
  - Time to get to activity is max of all paths leading to it

```
T(i) = E(i) + Max(T(i-1),T(i-2),…,T(i-j))
```

- Example

```
T(e) = E(e) + Max(T(c),T(f)) = E(e) + Max(T(c),E(f))
T(c) = E(c) + Max(T(b),T(d)) = E(c) + Max(T(b),E(d))
T(b) = E(b) + Max(T(a)) = E(b) + E(a)
```



87

Project Network

- Start at last activity and calculate time to finish per preceding activity
  - Use regression to calculate duration of all paths leading for each activity
- Identify longest path from start to finish
  - Called critical path

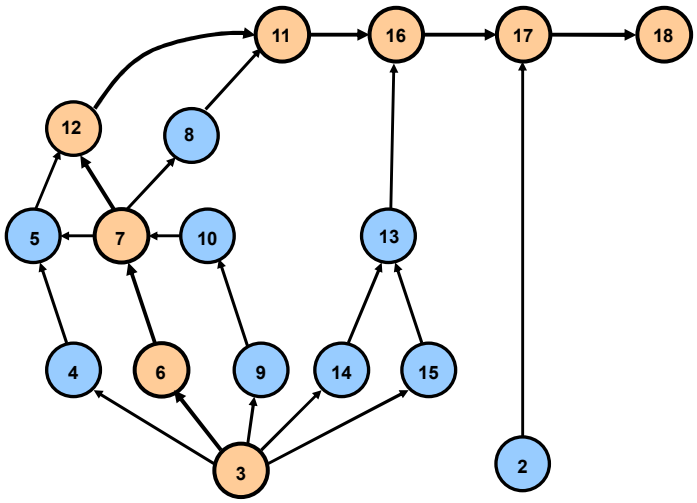
88

Critical Path

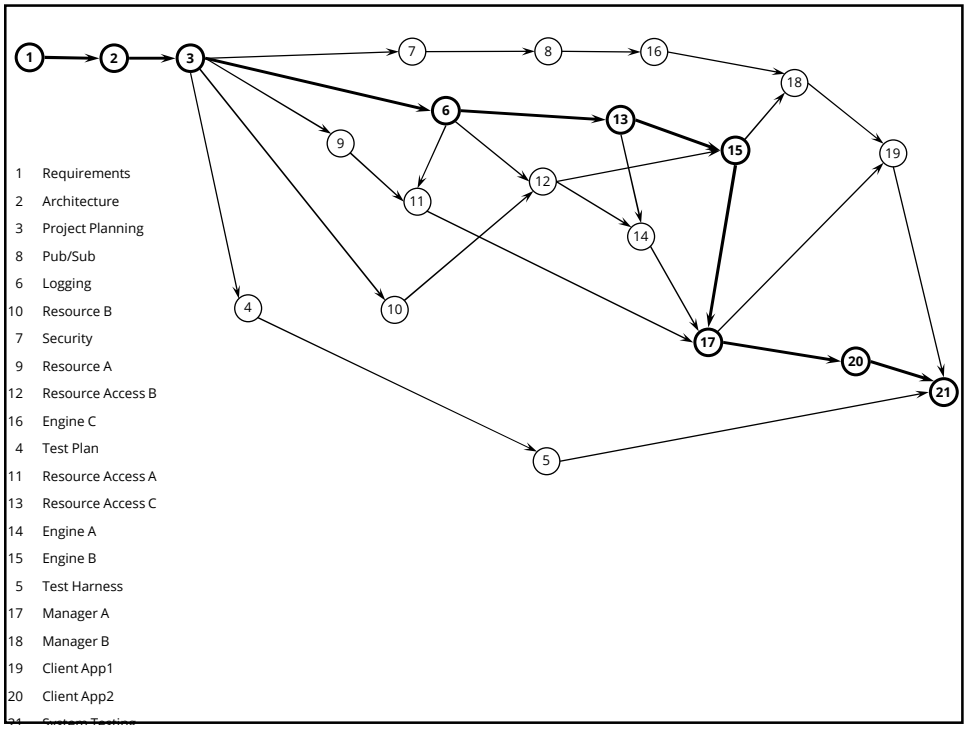
- Longest path from start to finish of project
  - Quickest possible way project can ever be built
  - No project can ever be accelerated beyond its critical path
- Duration of critical path is duration of project
  - Only way to answer "How long will it take?"

89

Critical Path



90

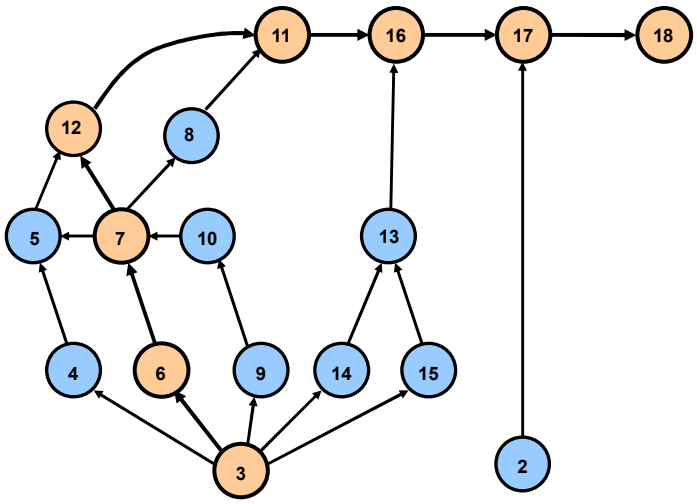


Critical Path and Resources

- Assign resources to critical path first
  - Accelerating non-critical paths does not accelerate project
  - Slowing down critical path absolutely slows down project
- Best resources always go to critical path
  - Where the risk is
- Look for smallest staffing level that allows unimpeded progress along critical path
  - Level will change along path
  - Assume for now unlimited resources

92

Critical Path



93

Float Analysis

- Non-critical activities can float
  - Do not have to start immediately
- Non-critical paths float will vary
- Assign resources to near critical path first
  - Where the risk is
- Can trade float for resources
  - Look for smallest staffing level that allows unimpeded progress on critical path
  - With acceptable risk

94

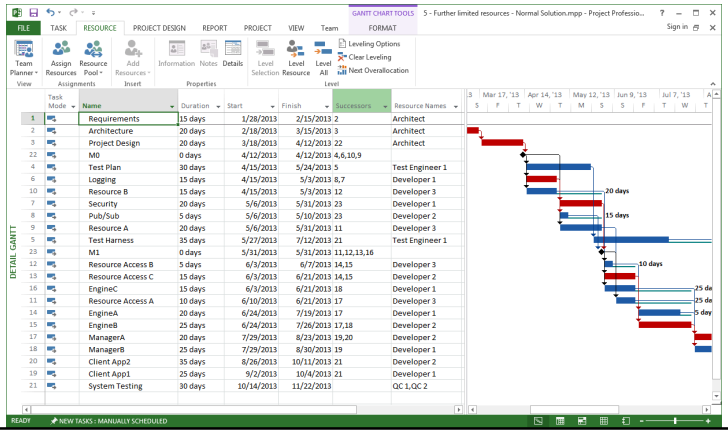
Activities and Resources

- Resource availability may dictate when activities done
  - Beyond logical dependencies
  - Resource dependencies are dependencies
- Assign resources for activities based on
  - Critical path
  - Floats
  - Available resources
  - Constrains
    - Will yield several plans

95

Scheduling Activities

- Use scheduling software to derive actual dates
  - And convert work days to calendar days



96

Scheduling Activities

Owner	Activity	Effort	Planned Completion Date
		0	4/1/2011
BCM1	Build/Configuration Manager	41	5/27/2011
Patrick	Architect - Top Level Design	60	6/23/2011
Patrick	WCF + MVC Training	8	6/27/2011
Diane	Account Data	20	7/25/2011
Patrick	Service Model Pekin	30	8/8/2011
Dianne	Policy Data	11	8/9/2011
Dianne	Support Utilities	35	8/16/2011
Diane	Customer Account Data Access	30	9/5/2011
Pat	Transform Policy Data	28	9/16/2011
Patrick	Pub/Sub Service	23	9/19/2011
Patrick	Security	21	10/18/2011
Dianne	Policy Data Access	36	10/25/2011
Jakob	Customer Manager	57	11/23/2011
Patrick	Service Endpoint Lookup	30	11/29/2011
Dianne	Agency Data	32	12/8/2011
Jakob	Registration Process (UI)	21	12/22/2011
Jakob	Customer Logon Process (UI)	21	12/22/2011
Dianne	Agency Data Access	26	1/13/2012
Jennifer	QA Testing - Stage 1	21	2/20/2012
Jakob	Policy Engine	39	4/6/2012
Dianne	Documents Index Data	47	4/18/2012
Paul	Documents File Store	48	4/19/2012
Jakob	Attach (Customer Mgr)	32	5/22/2012
Jakob	Documents Access	26	5/24/2012
Jakob	Attach Policies (UI)	41	7/18/2012
Jennifer	QA Testing - Stage 2	21	8/16/2012
Jakob	Agency Engine	27	9/24/2012
Dianne	Policy Documents Engine	38	10/9/2012
Pat	Billing Data	67	11/19/2012
Jakob	Policy Manager	48	11/29/2012
Jakob	View attached Policies (UI)	32	1/14/2013
Dianne	Billing Info Access	40	1/14/2013
Jennifer	QA Testing - Stage 3	21	2/12/2013
Dianne	Billing History Engine	41	4/10/2013
Jakob	View Billing Info (Policy Mgr)	43	6/10/2013
Jakob	Activity Manager	29	7/26/2013
Jakob	View Billing Info (UI)	34	7/26/2013
Jennifer	QA Testing - Stage 4	21	8/26/2013
	Total Planned Effort	1247	

97

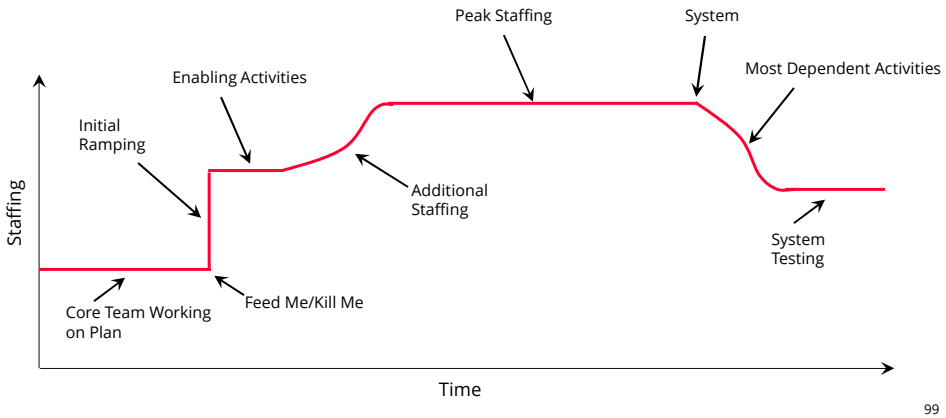
Staffing Distribution

- Only once project approved should assign resources
  - And option chosen
- Not all resources
  - Needed at once
    - Critical path
  - Retired uniformly
- Developers and resources ramp after SDP review
  - Phase in and out
  - Avoid feast/famine cycles
- Each planning option must have own staffing distribution

98

Staffing Distribution

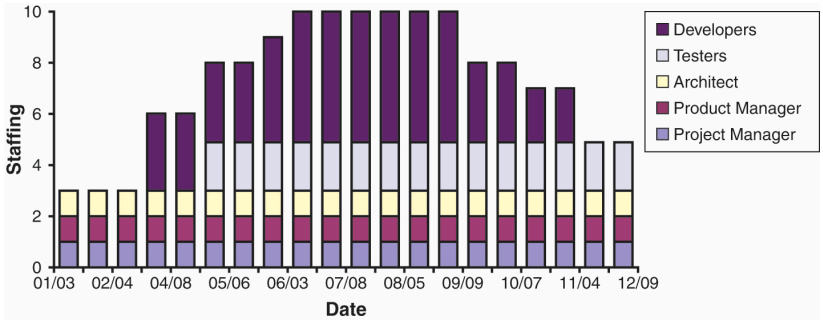
- Resources assigned along this curve



99

Staffing Distribution

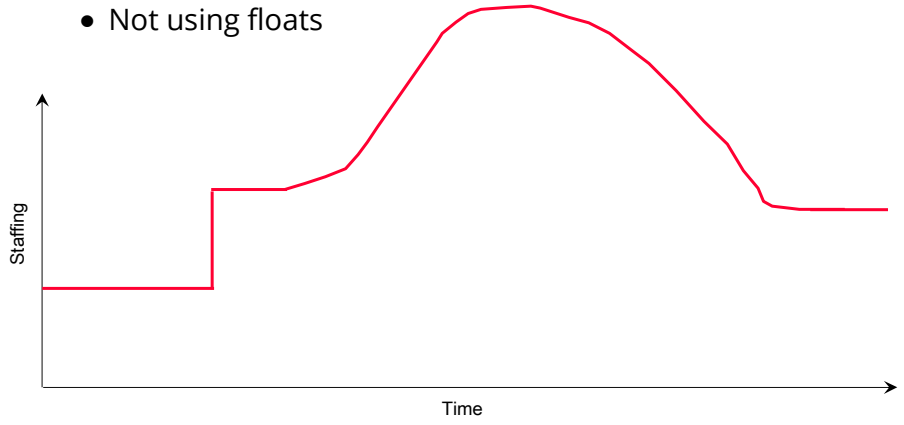
- Sample project
  - Properly staffed



100

Staffing Distribution

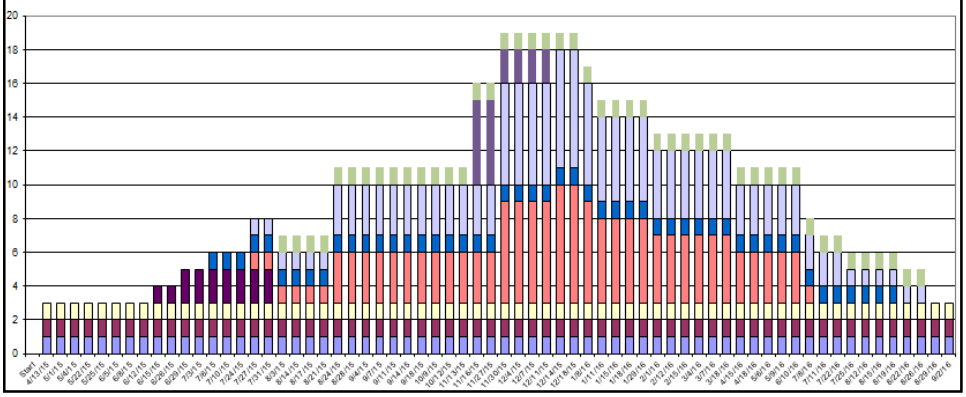
- Avoid peaks
  - Huge waste
  - Not using floats



101

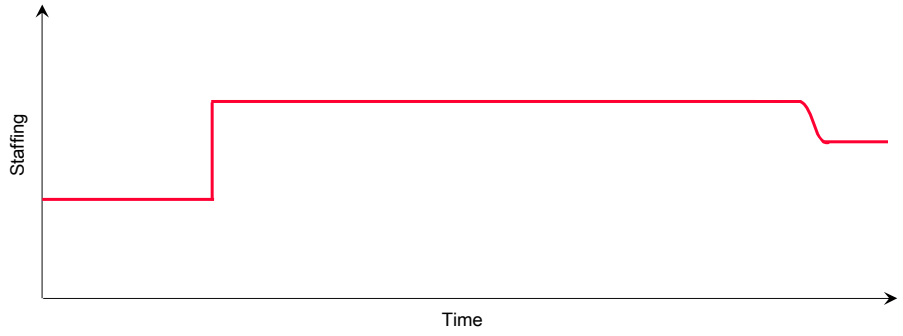
Staffing Distribution

- Avoid peaks
  - Huge waste
  - Not using floats



Staffing Distribution

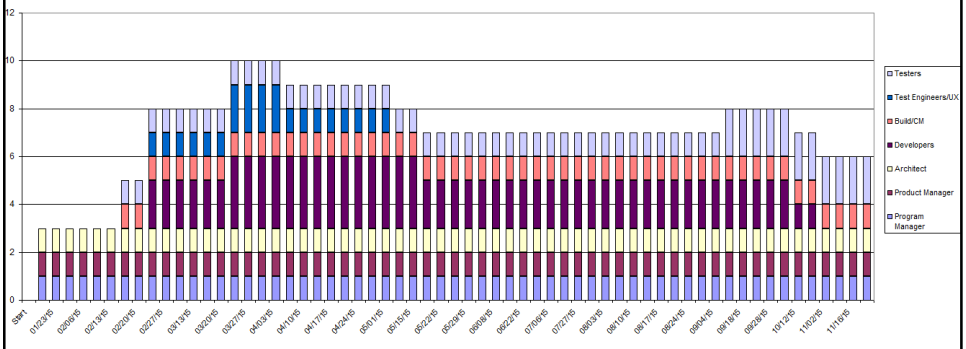
- Avoid flat line
  - Missing hump due to subcritical



103

Staffing Distribution

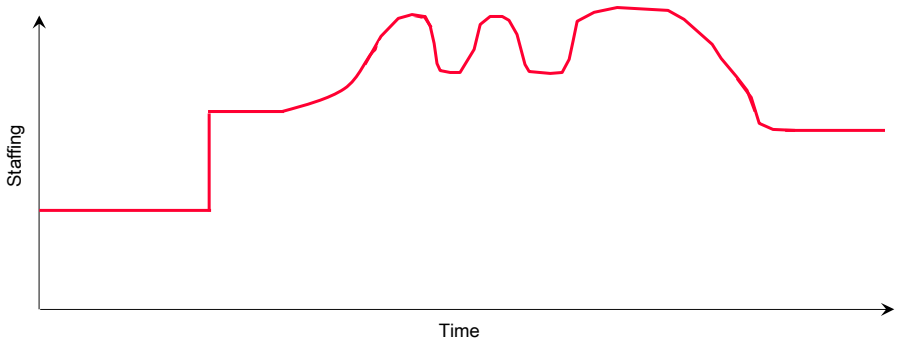
- Deep subcritical staffing sample
  - Unable to ramp up and maintain hump



104

Staffing Distribution

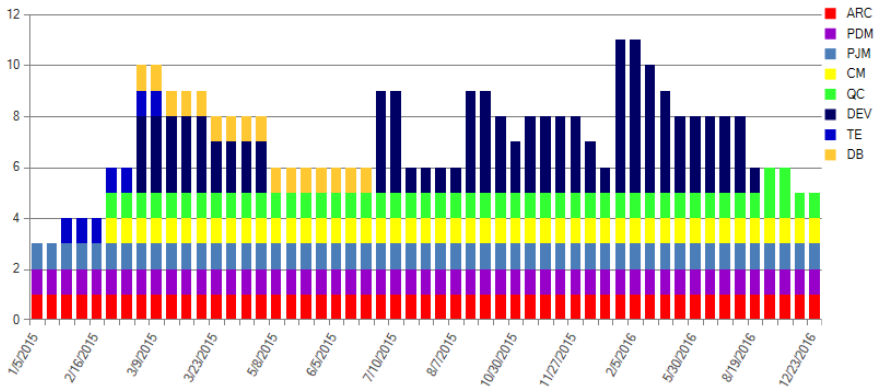
- Avoid erratic staffing
  - Impractical
  - People should not join/leave multiple times same project



105

Staffing Distribution

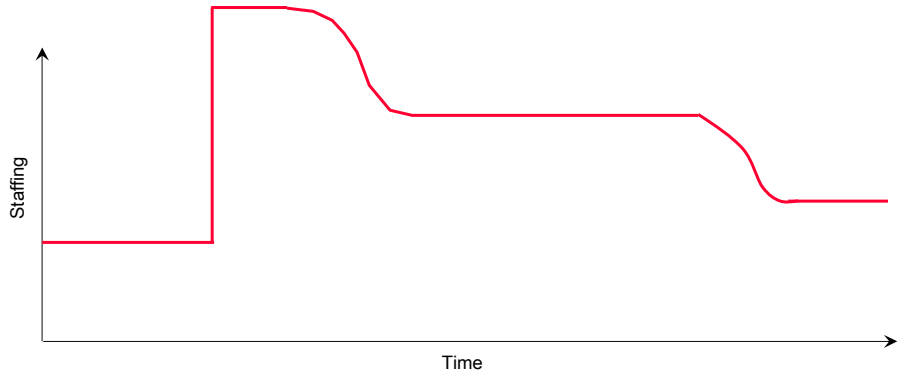
- People joining leaving joining....



106

Staffing Distribution

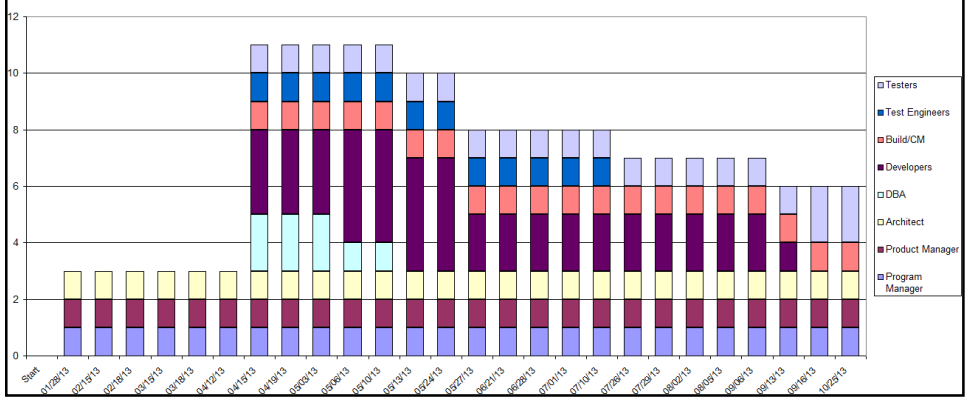
- Avoid risky high ramp
  - No team can ramp that quickly
  - Typically wasteful once hitting critical path



107

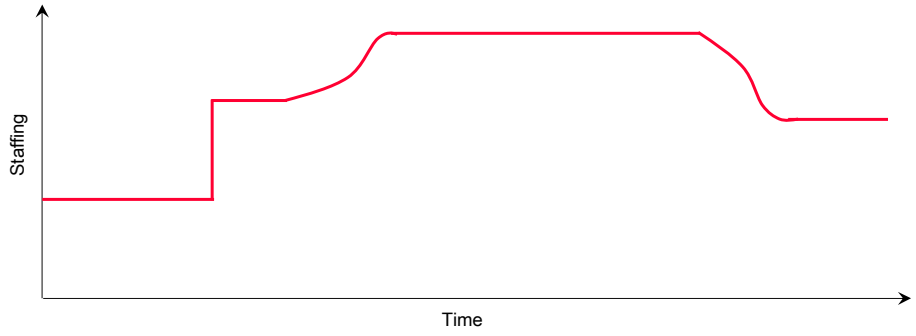
Staffing Distribution

- Risky ramp
  - Then hitting the wall



Project Cost

- Area under staffing curve
  - **Cost = Staffing * Time**
- Only way to answer "How much will it cost?"



109

Project Cost

- Without accurate cost and schedule
  - Perpetually overbid or underbid
  - In-house or external
  - Eventually go out of business
- Would you play poker without looking at the cards?
- Career/team/company survival

110

Roles and Responsibilities

- Architect designs the project
  - Project manager in supporting role
- Project design hinges on
  - Architecture
  - Resources
  - Time

111

Roles and Responsibilities

- Architect has insight and perspective on
  - Architecture
  - Technology
  - Dependencies
  - Design constrains
  - Resource skills
- Project manager has insight and perspective on
  - Resources cost
  - Availability
  - Priorities
  - Feasibility
  - Politics

112

Roles and Responsibilities

- Building the plant is part of the plant design
  - Do not be astronaut architect
- Lack of physical constraints is all the more reason to design
- Project cost and schedule derive from architecture
  - Do not be empty suit project manager
- Managing and executing complex software system requires
  - Intimate familiarity with architecture
  - Close continuous cooperation with dedicated architect

Roles and Responsibilities

- Architects traditionally address
  - Maintainability
  - Reusability
  - Extensibility
  - Feasibility
  - Scalability
  - Throughput
  - Availability
  - Responsiveness
  - Performance
  - Security
- Architects produce design solution that satisfies above requirements and constraints

114

Roles and Responsibilities

- Must add to the list
  - Resources
  - Cost
  - Schedule
  - Risk
- Part of system design as much as the others
- Fundamentally an engineering design task

115

IDesign

Advanced Techniques

Juval Löwy
www.idesign.net

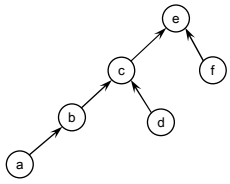
Network Diagrams

- Project is collection of related activities
- Network diagram is model of project
  - And relationship required to complete
  - Much like architecture is model of system

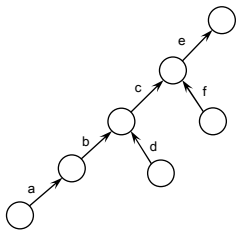
117

Network Diagrams

- Prefer arrow diagram to node diagram



Node Diagram

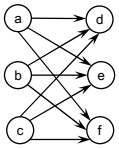


Arrow Diagram

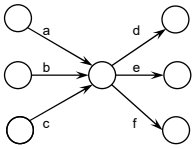
118

Arrow vs. Node Diagrams

- Arrow diagram shines with complex dependencies involving repeated dependencies on activities
  - Avoid line crossing

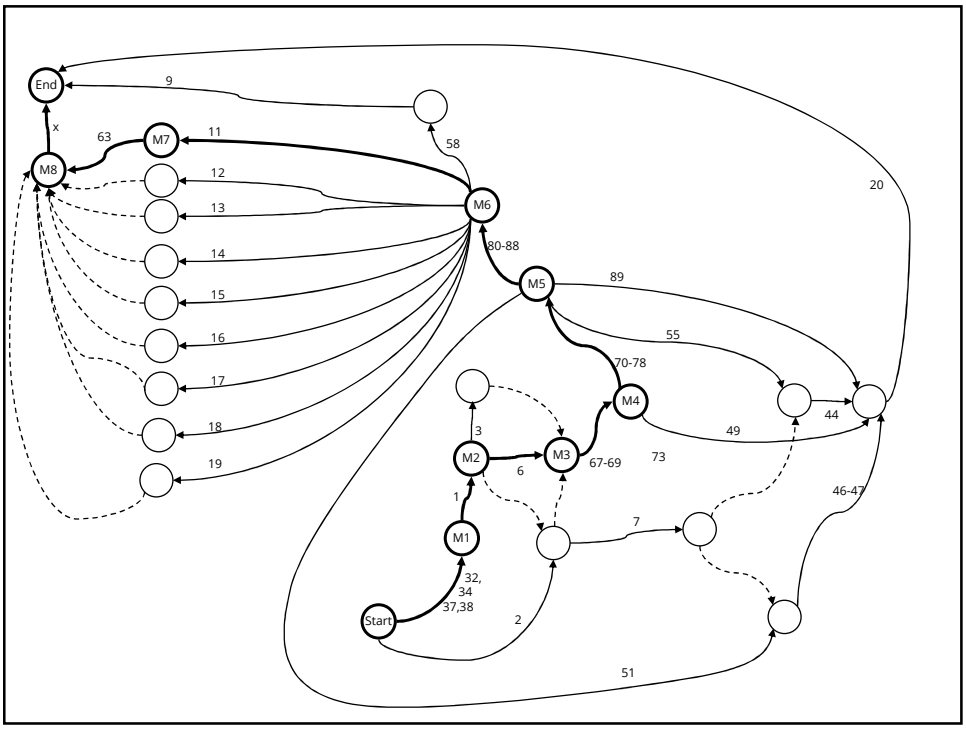


Node Diagram



Arrow Diagram

119



Floats

Juval Löwy
www.idesign.net

Floats

- Critical activities must complete as soon as possible to avoid delay
- Noncritical activities may be delayed without slipping
  - They can "float"

122

Floats Analysis

- Floats are the project safety margins
  - Consume to compensate for unforeseen delays and inflictions in non-critical activities
- Low float projects are at high risk
  - Activities may become critical and delay project

123

Activity Times and Floats

- Microsoft Project calculates floats automatically
  - Called "slacks"

	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Total Slack
1		A	10 days	1/7/2013	1/18/2013		0 days
2		B	10 days	1/7/2013	1/18/2013		6 days
3		C	23 days	1/21/2013	2/20/2013	1	0 days
4		I	13 days	1/21/2013	2/6/2013	1	19 days
5		G	15 days	1/21/2013	2/8/2013	2	8 days
6		H	17 days	1/21/2013	2/12/2013	2	6 days
7		D	12 days	2/21/2013	3/8/2013	3,5	9 days
8		E	14 days	2/21/2013	3/12/2013	6,3,5	7 days
9		F	30 days	2/21/2013	4/3/2013	6,3,5	0 days
10		J	12 days	2/7/2013	2/22/2013	4	19 days
11		L	1 day	2/7/2013	2/7/2013	4	31 days
12		K	9 days	3/13/2013	3/25/2013	7,10,8	7 days
13		M	8 days	2/8/2013	2/19/2013	11	31 days

124

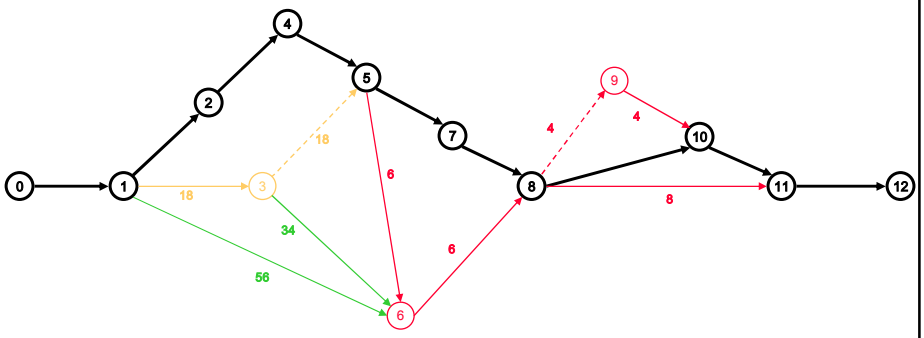
Floats Analysis

- Color code to indicate risk
- Use absolute values
  - Works well in most projects
  - Should customize range
    - 1-10
    - 11-25
    - 26+

125

Floats Analysis

- Visualize risk



126

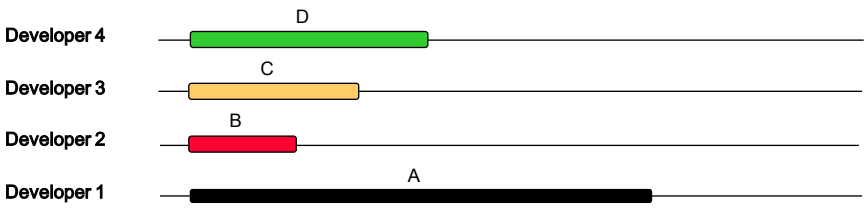
Using Floats

- Floats key project design tool
  - Less for project management
  - Design within adequate resources
  - Design within acceptable risk

127

Using Floats

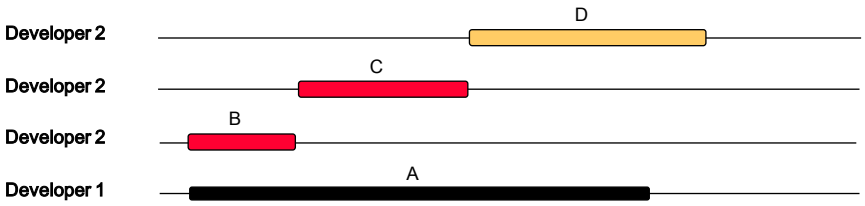
- Can trade float for resources



128

Using Floats

- Can trade float for resources



129

Time-Cost Curve

Juval Löwy
www.idesign.net

Motivation

- There is no way to accelerate code-like-hell
  - You are already coding like hell
- Code-like-hell is not fastest way of building system
  - Effort assigned across critical path
- Code-like-hell is not cheapest way of building system
  - Horrendous waste

131

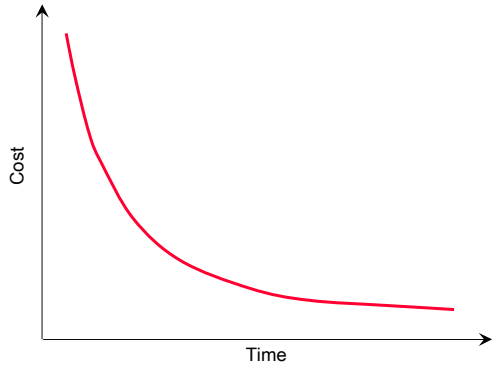
Motivation

- Well-designed projects are fastest way of building system
  - Efficiently assign minimal resources along critical path
- No project can be accelerated beyond its critical path
- How can you accelerate project
  - Compress critical path

132

Time-Cost Curve

- Idealized curve for project or activity



133

Time-Cost Curve

- Additional cost can accelerate project
  - Senior instead of junior developers
  - Test engineers
  - Testers
  - Infrastructure investment
  - Skills and process investment
  - Standards
  - Reviews
  - Parallel work
  - QA
    - Not QC

134

Compression

- Accomplishing same objectives faster
  - Not necessarily the same work
- Only two possible ways
  - Better resources
    - Often will do more work required to finish earlier
  - Concurrent work

135

Project Compression

- Compressing individual activities without changing cross-activities dependencies
- Splitting activities and performing less-depended phases up-stream and parallel to other activities
  - Detailed design
  - Test plan
  - Test harness
  - Simulators for all required services

136

Points On Time-Cost Curve

- Normal solution
  - Assuming unlimited resources
  - Designing project for minimum direct cost
  - Without slowing critical path
- With longer than normal solution cost increases
  - Uneconomical drag-out
  - Probability for success diminishes
  - Complexity increases
  - Gold plating
  - Accumulated overhead

137

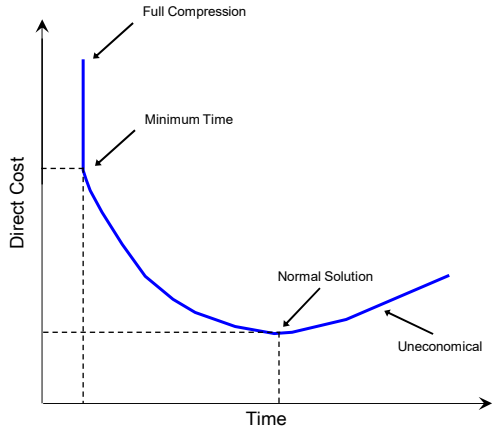
Points On Time-Cost Curve

- Minimum time solution
  - Designing project for shortest duration at minimum cost
  - May redesign project as a whole
  - No point in compressing all activities
- Full compression solution
  - All activities are compressed
  - Duration same as minimum time
  - Costs more than minimum time

138

Time-Cost Curve

- Actual direct cost curve



139

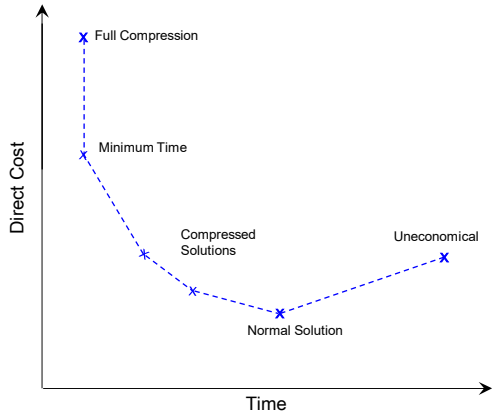
Time-Cost Curve

- Actual curve offers infinite options between minimum time and normal
- Optimal point
  - Somewhere between all-normal and minimum time
- Architect should provide 1-2 practical points in between
  - Reasonable options
  - Likely choice by management
  - Assume enough resources to proceed along critical path
  - Result of some network compression
- Risk is missing element

140

Time-Cost Curve

- Compiled solutions are discrete
  - Finite number of designed solutions



141

Time-Cost Curve

- Do present to management impractical points of
  - Full compression
  - Drag out
- Avoid classic mistakes
  - Schedule below minimum duration
  - Subcritically staffed project pushed right of normal
  - Often unaware of their impracticality

142

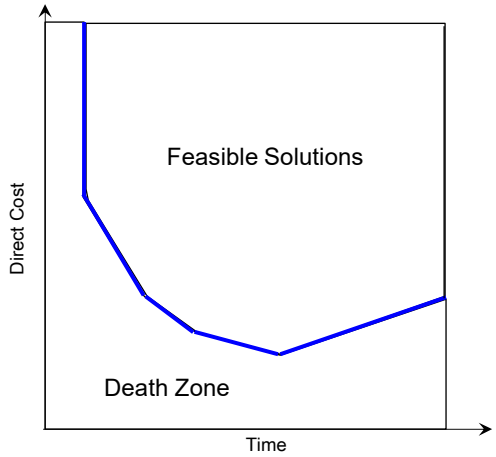
Time-Cost Curve

- Curve shows something paramount: feasibility
  - Points above curve are doable
    - Will cost more for same date
    - Will take longer than should for same cost
  - Points under curve are impossible to commit to
    - With that cost cannot ship on that date
    - With that date cannot ship on that cost
    - Project failed without ever starting

143

Time-Cost Curve

- Points on curve are simply minimum cost for duration



144

Cost Elements

- Cost is product of both direct and indirect elements
- Direct cost when adding direct measurable value
  - Varies over project lifetime
- Direct cost typical elements
  - Developers working on project
  - Testers performing system testing
  - DBA designing database
  - Test engineer designing and building test harness
  - UI/UX designer designing client apps
  - Architect designing system or project

145

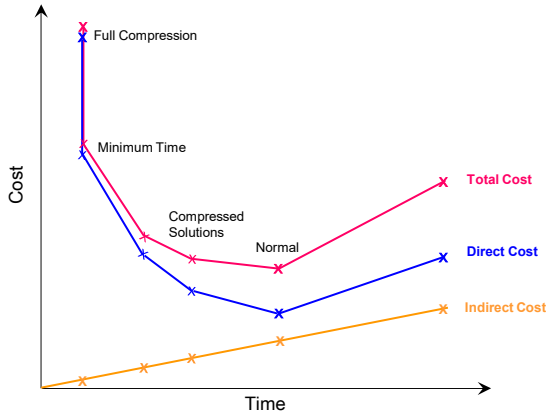
Cost Elements

- Indirect cost is adding on-going indirect immeasurable value
  - Constant with time
- Indirect cost typical elements
  - Core team post SDP review
  - DevOps
  - Developers between assigned tasks
  - Daily build and Build
  - Daily smoke test

146

Time-Cost Curve

- Calculated project cost



147

Time-Cost Curve

- Compressed project will have less indirect cost
- Mitigate somewhat cost of compression

148

Staffing and Cost

- Observation
  - All things being equal, shorter projects are cheaper
  - Even when requiring more expensive resources
- Even without compression
- Classic mistake
  - Trying to reduce cost by throttling resources
    - In quality or quantity
  - Project is longer
  - Costs more

149

IDesign

Quantifying Risk

Juval Löwy
www.idesign.net

Motivation

- For every project there are several design options
  - Some more aggressive than others
- The more aggressive options carry increased risk
  - Probability of success is lower
  - Must take into account when choosing
- Should be able to quantify risk to better evaluate your options
  - Most people recognize the risk axis but ignore it since cannot measure it

151

Objectives

- Measure risk objectively
  - And easily
- Use more than one way of calculating
- Correspond to intuitive evaluation
- Evaluate design options in light of risk as well as cost and schedule
  - Feasibility

152

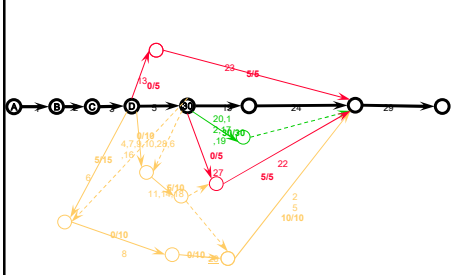
Risk and Floats

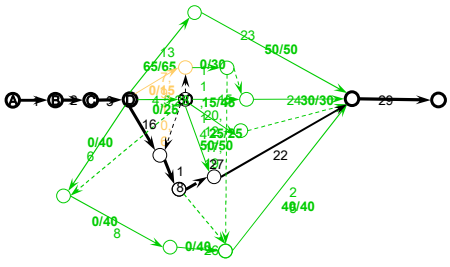
- Floats are objective way of measuring risk
- Changing project design can drastically change its risk

153

Risk and Floats

- Two possible designs for the same project
  - Which would you rather be on?





154

Requirements for Quantifying Risk

- Risk is always relative
  - "Risky"?
    - Compare with what?
  - There is only riskier or safer
- What are units of risk?
- How can you compare options and projects?
- Normalize risk to numerical range
  - 0 to 1 is as good as any
  - 0 minimizing risk
  - 1 maximizing risk
- Different methods should yield comparable results

155

Criticality Risk Index

- Capturing that almost intuitive impression of how much project is critical or at risk
- Classifying activities into risk bins
- Critical activities are riskier
  - Higher likelihood of cost and schedule overrun
- Near critical activities are risky too
  - The more critical the riskier
- Activities with large floats are less risky

156

Criticality Risk Index

- Assign weight for criticality
  - A factor
  - Follow float classification via color coding
- Can shift factors as long as sum is capped and constant

Activity Color	Criticality Factor
Critical (Black)	4
Red	3
Yellow	2
Green	1

157

Criticality Risk Index

- Classify activities based on total floats
  - Color coding
- Same as with floats analysis
- Done for evaluating viability
  - Rather than resource assignment

158

Criticality Risk Index

- Definitions
  - C Number of critical activities
  - R Number of red activities
  - Y Number of yellow activities
  - G Number of green activities
  - N Number of activities
  - W_C Weight of critical activities
  - W_R Weight of red activities
  - W_Y Weight of yellow activities
  - W_G Weight of green activities

159

Criticality Risk Index

- Risk formula

$$Risk = \frac{W_C*C + W_R*R + W_Y*Y + W_G*G}{W_C*N}$$

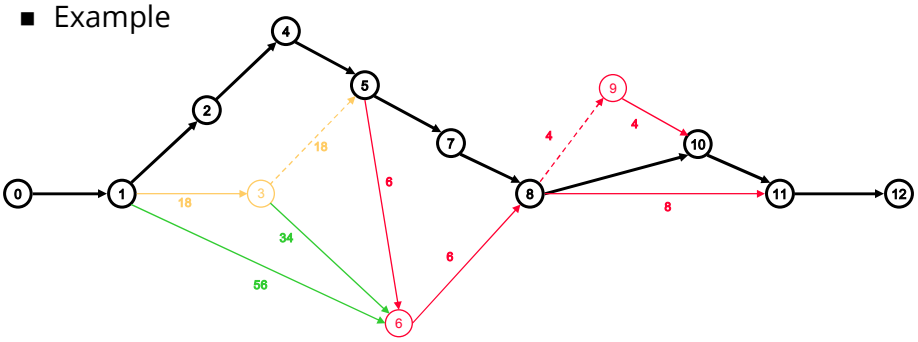
- Or with the specific weights

$$Risk = \frac{4*C + 3*R + 2*Y + 1*G}{4*N}$$

160

Criticality Risk Index

- Example



$$\text{Risk} = \frac{4*9 + 3*4 + 2*1 + 1*2}{4*16} = 0.81$$

161

Criticality Risk Index

- Easy to calculate in Excel

No.	Float	Critical	Red	Yellow	Green
1	0	1	0	0	0
2	0	1	0	0	0
3	56	0	0	0	1
4	0	1	0	0	0
5	18	0	0	1	0
6	34	0	0	0	1
7	0	1	0	0	0
8	6	0	1	0	0
9	0	1	0	0	0
10	6	0	1	0	0
11	0	1	0	0	0
12	0	1	0	0	0
13	8	0	1	0	0
14	4	0	1	0	0
15	0	1	0	0	0
16	0	1	0	0	0
	132	9	4	1	2

Criticality Risk	0.81
Critical factor	4
Red Factor	3
Yellow Factor	2
Green Factor	1
Total	10
Red Limit	10
Yellow Limit	25

162

Criticality Risk Index

- Maximum risk at 1
  - All activities critical
- Minimum risk at W_G/W_C
  - 0.25 in the sample
  - Can risk ever be zero?

163

Activity Risk

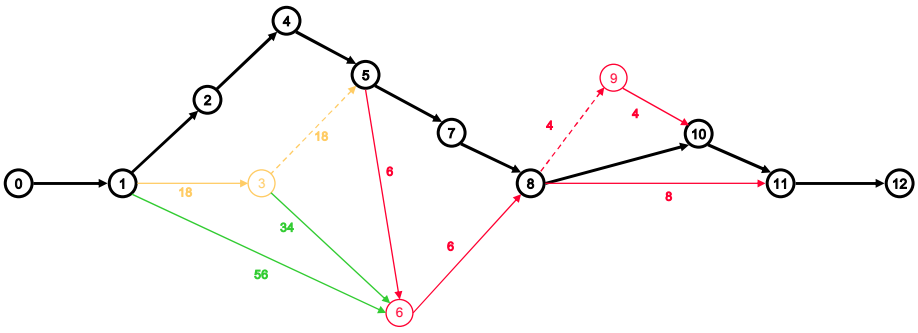
- Calculate each activity's contribution to risk
  - Instead of generic bins
  - More discrete and precise
- Definitions
  - F_i Total float of activity i
  - M Largest total float value across all activities
  - N Number of activities
- Risk formula

$$Risk = 1 - \frac{\sum_{1}^{N} F_i}{N*M}$$

164

Activity Risk

- Example



$$\text{Risk} = 1 - \frac{18+18+34+56+6+6+4+4+8}{56*18} = 0.85$$

165

Activity Risk

- Easy to calculate in Excel
- Maximum risk at 1
  - All activities critical
- Minimum risk at 0
  - No critical path
  - All activities with same maximum float
  - Can risk ever be zero?

166

Criticality Vs. Activity Risk

- For decent size project yield very similar results
- Criticality often requires tweaking
  - Judgment calls
- Criticality reflects intuition better

167

Risk Metrics

- Minimum direct cost is around 0.5
- Acceptable risk is 0.5 to 0.75
  - Symmetry in risk
  - Relates to risk rate of growth
- Normal solution slightly under 0.7
- Risk crossover is around 0.75

168

Risk Decompression

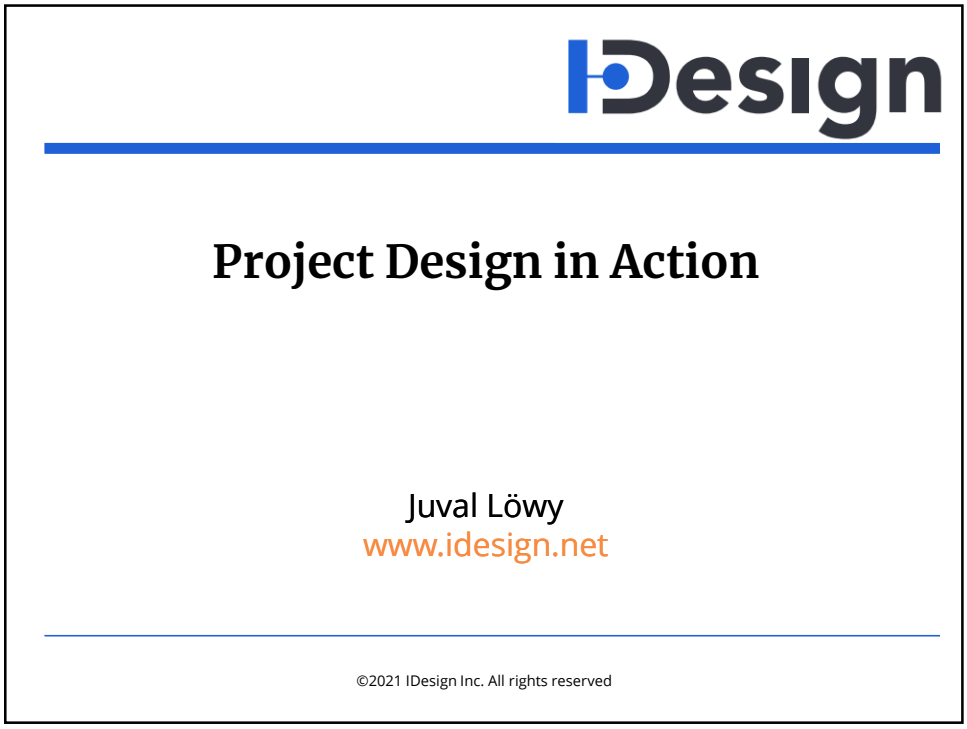
- Deliberately designing for a later date to reduce overall risk
  - Introducing float along critical path
- Objective
  - Increasing probability of meeting commitments
- Motivation
  - Past failures
  - Build trust
  - Too many unknowns
  - Volatile environment

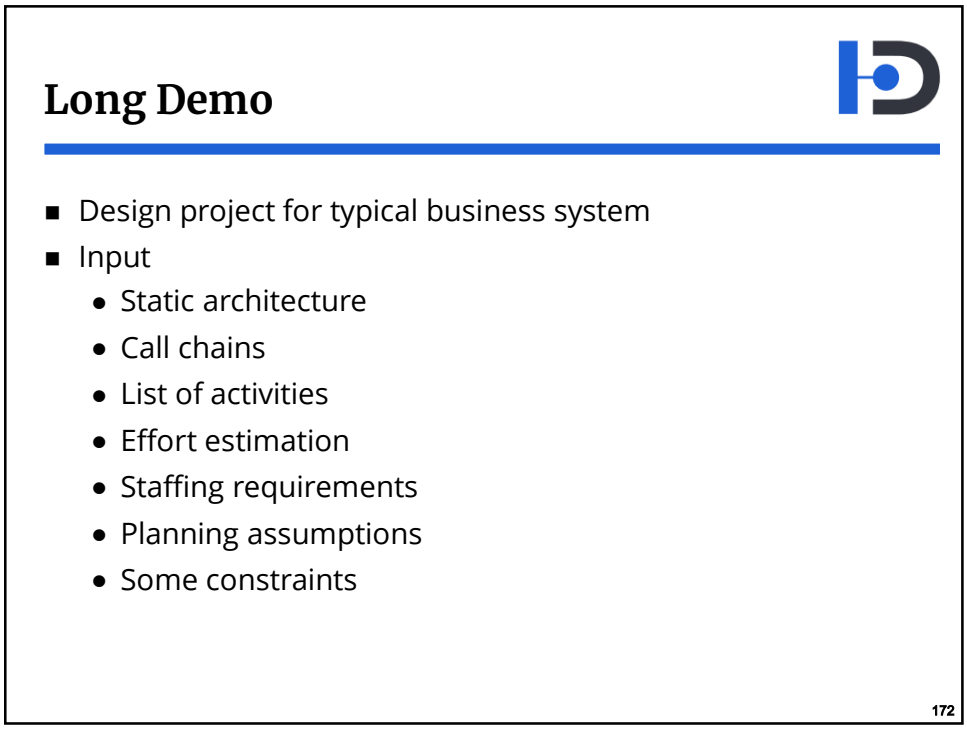
169

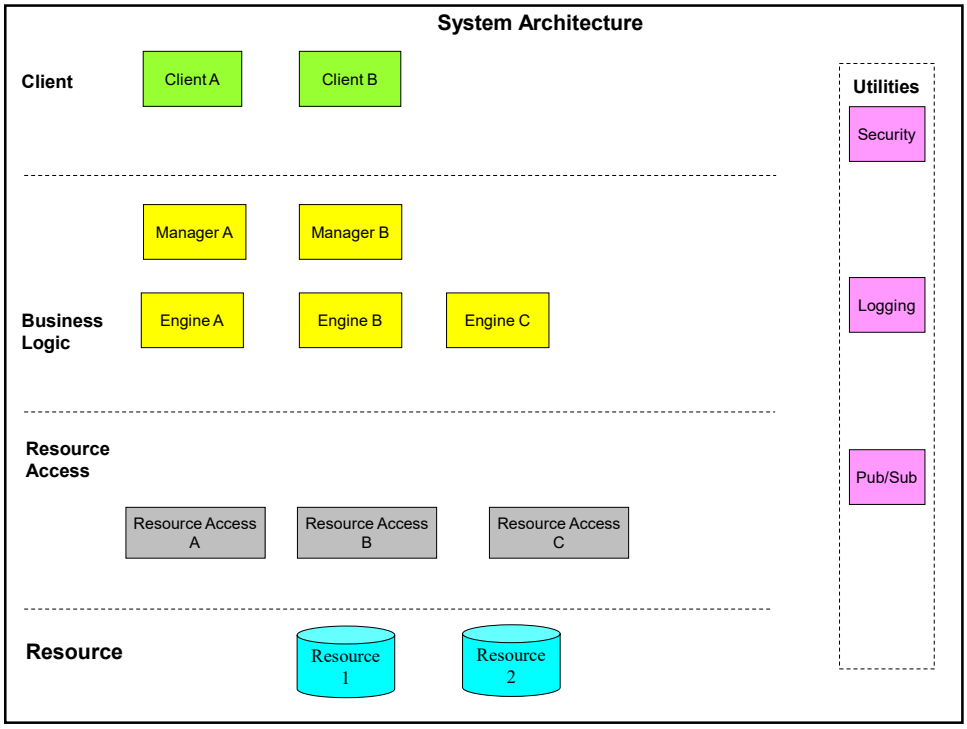
Risk Decompression

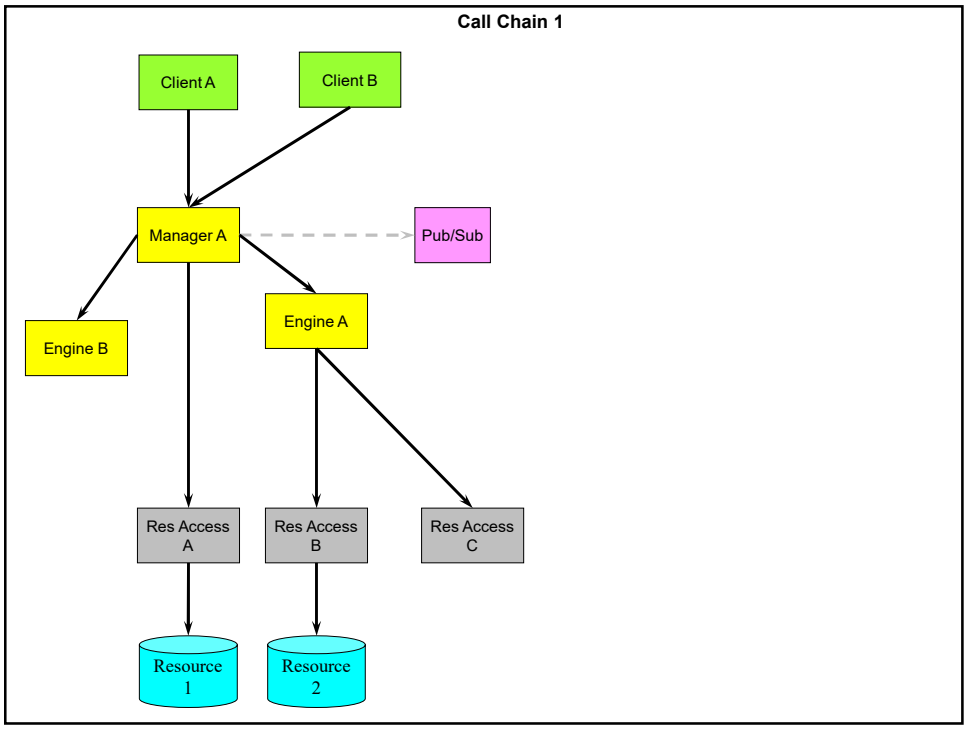
- Decompress by moving completion date of down-stream activities
  - While keeping same resources
- With Microsoft Project
  - Use "Start no earlier than" constraint on activity to move
- When moving multiple activities
  - Decompress earlier activities first
  - Move later activities more

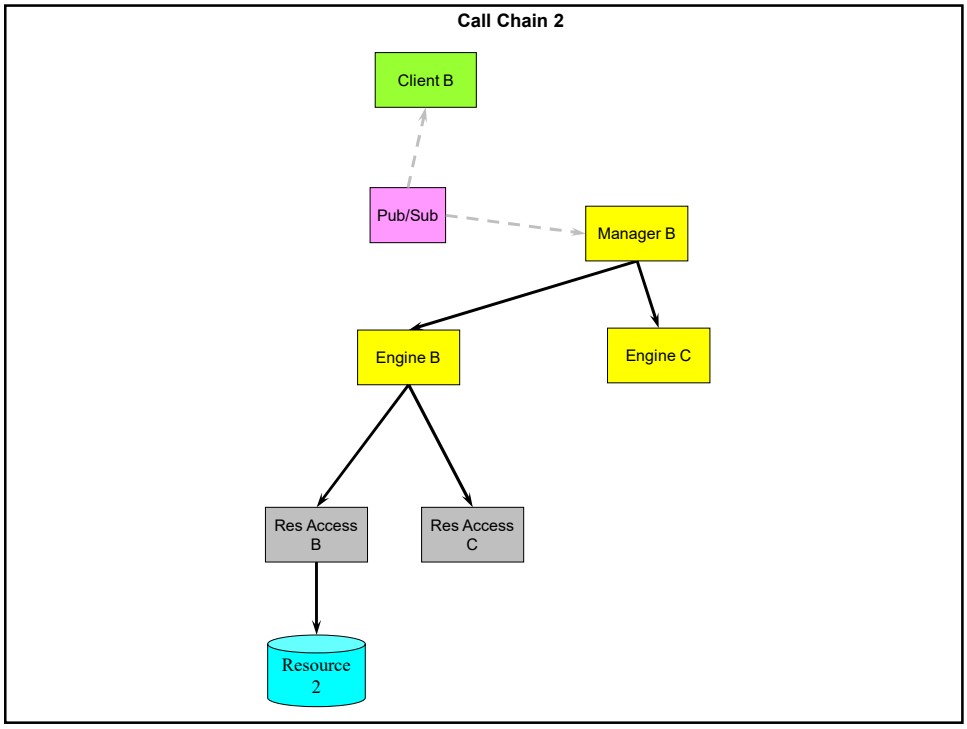
170







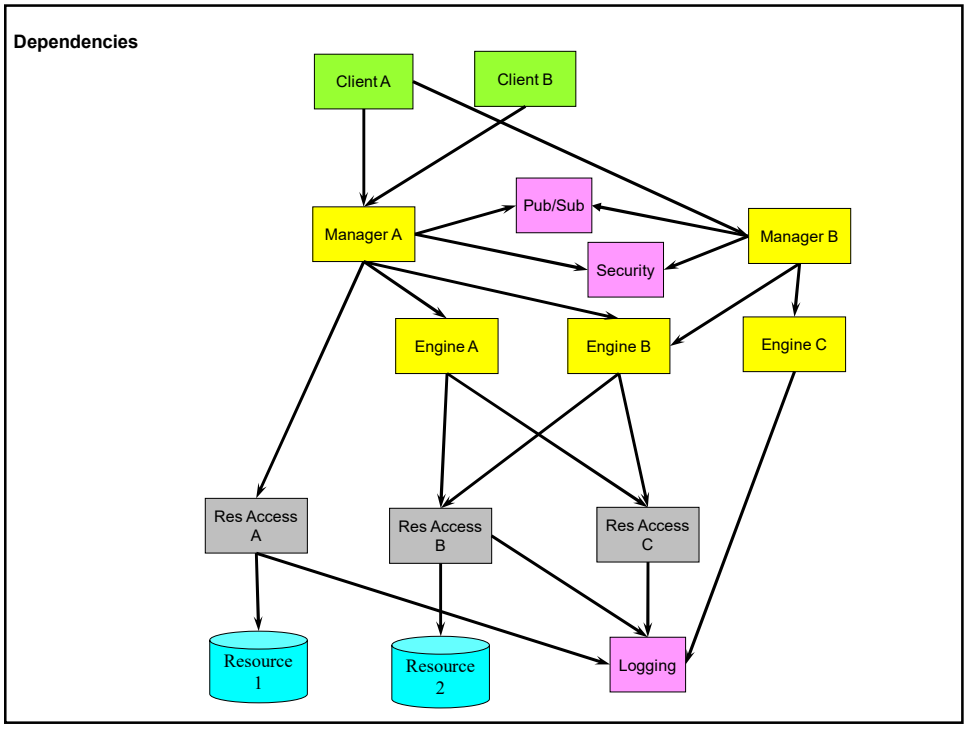




Additional Information

- All entities depend on logbook
- Clients and managers depend on security

176



Adding Activities

- Non-code non-structural activities
  - Requirements
  - Architecture
    - Vertical slice
    - Demo service
  - Project planning
  - Test plan
  - Training
  - Test harness
  - System testing
  - Documentation

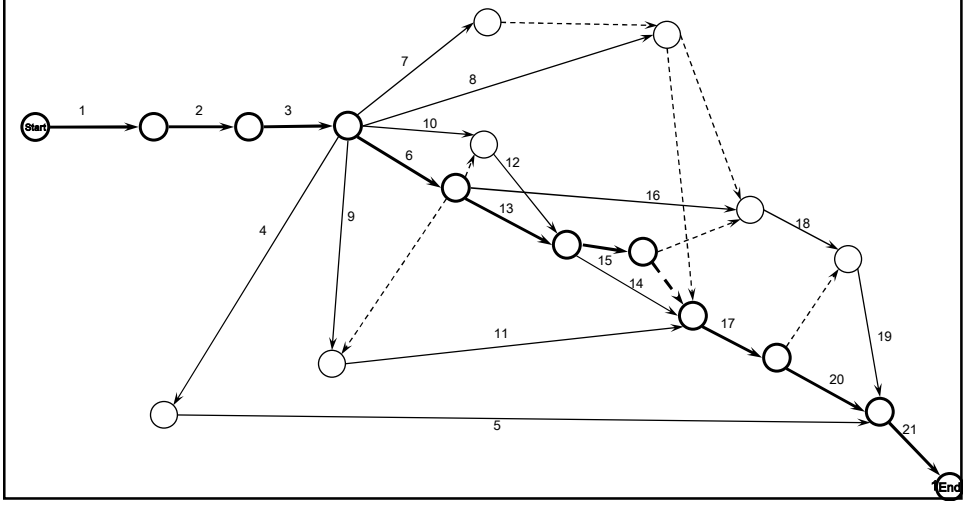
178

Activities

ID	Activity	Duration	Depends on
1	Requirements	15	
2	Architecture	20	1
3	Project Planning	20	2
4	Test Plan	30	3
5	Test Harness	35	4
6	Logging	15	3
7	Security	20	3
8	Pub/Sub	5	3
9	Resource A	20	3
10	Resource B	15	3
11	Resource Access A	10	6,9
12	Resource Access B	5	6,10
13	Resource Access C	15	6
14	EngineA	20	12,13
15	EngineB	25	12,13
16	EngineC	15	6
17	ManagerA	20	7,8,11,14,15
18	ManagerB	25	7,8,15,16
19	Client App1	25	17,18
20	Client App2	35	17
21	System Testing	30	5,19,20

179

Network Diagram



Staffing Requirements

- 1 project manager throughout project
- 1 product manager throughout project
- 1 architect throughout project
- 1 developer per service
- 1 DB architect/specialist for each resource
- 1 tester throughout project
- 1 additional tester for system testing
- 1 test engineer for test plan and test harness

181

Staffing Requirements

- Staffing distribution plan

Role	Planning	Infrastructure	Services	Testing
Architect	X	X	X	X
Project Manager	X	X	X	X
Product Manager	X	X	X	X
Developers		X	X	
Testers			X	X
DevOps		X	X	X

- Can add specialists
  - UX, test engineer, DBA, etc.

182

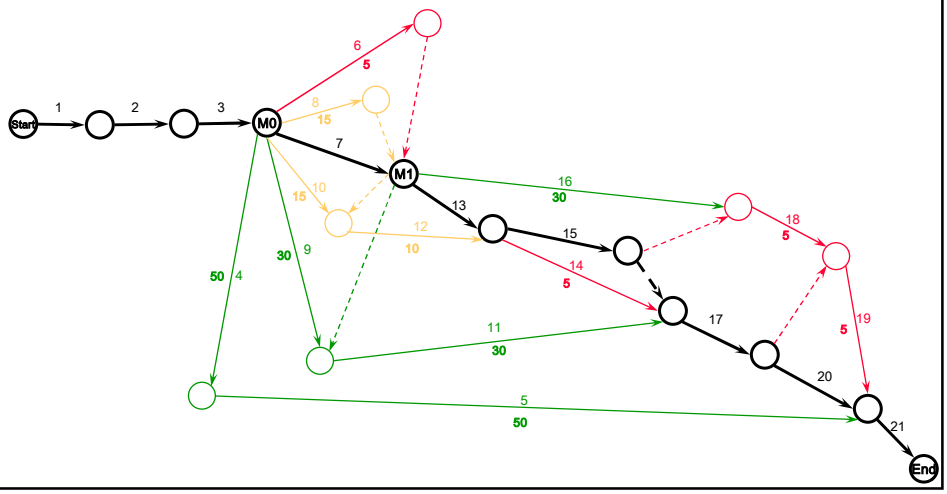
Staffing Requirements

- Assume every resource and rare skill set is available
  - Solution architect
  - DB architect
  - UI designer
  - Test engineer
- Yields first take on project duration and cost

183

Floats Analysis

- Non-critical paths and risk



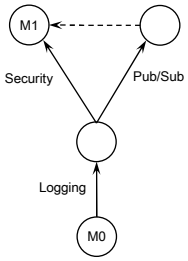
Network and Resource Problems

- There are no infinitely available recourses
- Rare skills are rare
- Undesirable unitizing resources for short periods of time
  - Rump up time
  - Subcontractors can help
- Avoid mass integration points

185

Limited Resources

- Assuming somewhat limited resources or ramp time



186

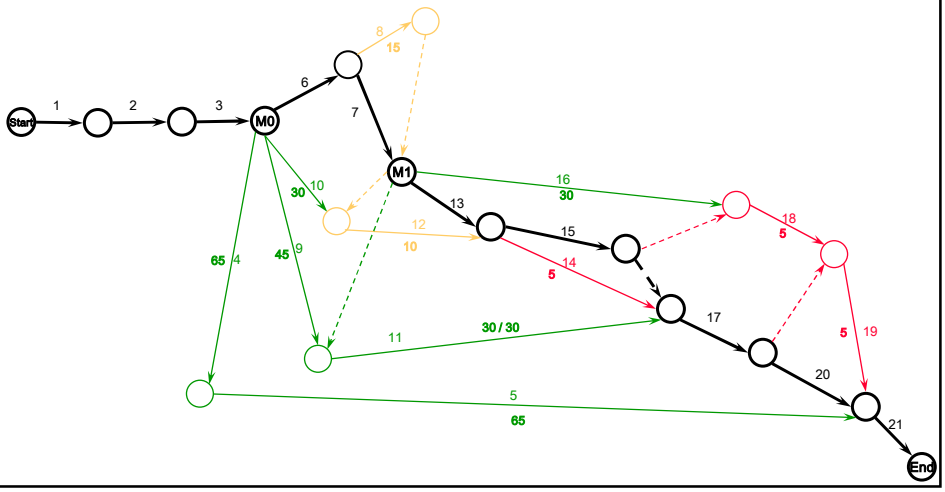
Limited Resources

- Limited to 3 developers
- Result with no change in
  - Critical path
  - Cost
  - Schedule
    - Consumes float

187

Floats Analysis

- Non-critical paths and risk

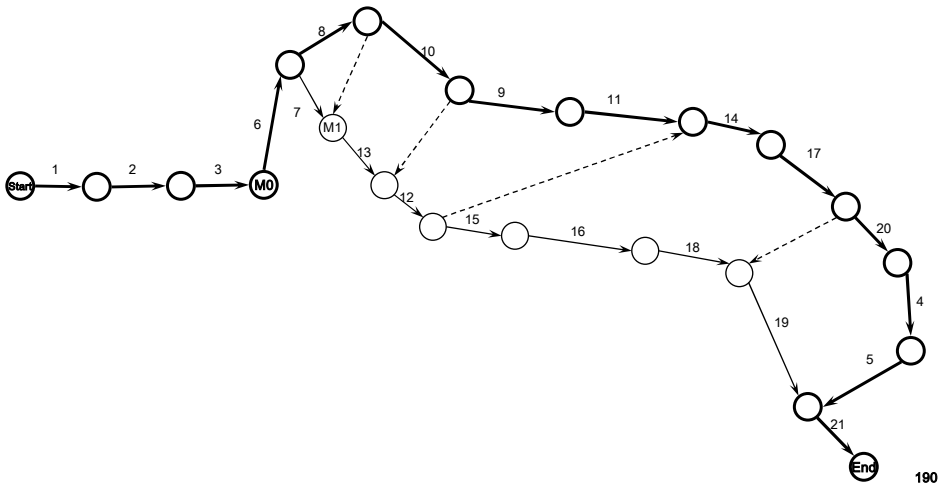


Subcritical Resources

- Only 2 developers
- Project is critically under-staffed
- Limiting factor is not duration of critical path
  - It is availability of resources
- Old critical path does not apply
  - By the time activities on the critical path are complete supporting non-critical activities are not ready
- Redraw network to accommodate resource dependencies
  - Lots of flexibility in order of activities
  - MS-Project with auto-scheduling helps define new network

189

Subcritical Resources



190

Subcritical Resources

- Cost is increased by at least 25%
  - Indirect costs mount too
- Schedule is increased by 35%
- There is no saving with subcritical resources
- Risk index at 0.9
  - Up from 0.73
- Subcritical projects are of high risk

191

Compression

- Could better resources accelerate this project?
- Assume 0.7/1.8 duration/cost is possible with top developer
- Ideally only apply to critical path
- Identify developer that spends the most time on critical path
  - Number of activities
  - Number of days in total
- Could foresee and assign beforehand all critical path to same developer
  - Not always feasible

192

Compression

- Clearly it is better to have Developer 2 be top resource

Resource	Non-Critical Activities	Non-Critical Duration	Critical Activities	Critical Duration
Developer 1	4	85	2	35
Developer 2	1	5	4	95
Developer 1,2	5	90	6	130

193

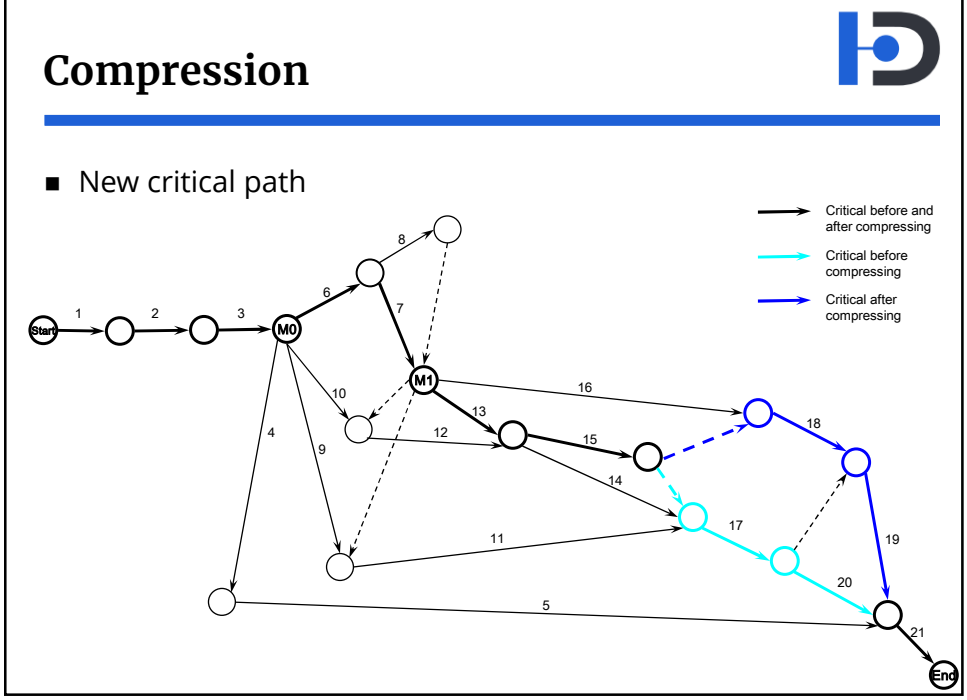
Compression

- Activities with single top developer

ID	Activity	Duration
0	**Start**	0
1	Requirements	15
2	Architecture	20
3	Project Planning	20
4	Test Plan	30
5	Test Harness	35
6	Logging	15
7	Security	20
8	Pub/Sub	5
9	Resource A	20
10	Resource B	15
11	Resource Access A	10
12	Resource Access B	5
13	Resource Access C	10
14	EngineA	20
15	EngineB	20
16	EngineC	15
17	ManagerA	15
18	ManagerB	25
19	Client App1	25
20	Client App2	25
21	System Testing	30

194

Compression



- New critical path

Compression

- Total cost unchanged from normal solution
  - Less indirect cost
  - Compensating for direct cost of compression
- Duration reduced by 4%
  - New critical path limit

196

Compression

- Additional compression with Developer 1
  - On top of Developer 2

Resource	Non-Critical Activities	Non-Critical Duration	Critical Activities	Critical Duration
Developer 1	2	35	4	85
Developer 2	3	45	2	30
Developer 1,2	5	80	6	115

197

Compression

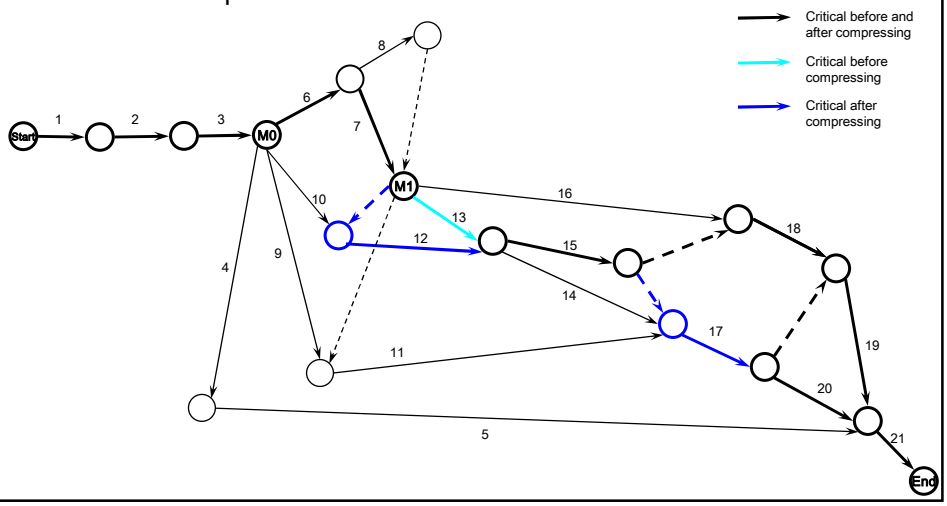
- Activities with two top developers

ID	Activity	Duration	Depends on
0	**Start**	0	
1	Requirements	15	
2	Architecture	20	1
3	Project Planning	20	2
4	Test Plan	30	22
5	Test Harness	35	4
6	Logging	10	22
7	Security	15	6
8	Pub/Sub	5	6
9	Resource A	20	22
10	Resource B	15	22
11	Resource Access A	10	23,9
12	Resource Access B	5	23,10
13	Resource Access C	10	23
14	EngineA	15	12,13
15	EngineB	20	12,13
16	EngineC	10	23
17	ManagerA	15	11,14,15
18	ManagerB	20	15,16
19	Client App1	20	17,18
20	Client App2	25	17
21	System Testing	30	5,19,20

198

Compression

- New critical path



Compression

- Cost reduced by additional 3%
- Schedule reduced by additional 11%

200

Accelerating Schedule

- Adding top resources does little for acceleration
- Real acceleration enabled by working in parallel
  - Remove dependencies
    - Carefully
    - Additional investment up front
- Create new compressed critical path
- Parallel work typically increases
  - Risk due to lower float
  - Required competence
  - Peak team size
  - Noise

201

Accelerating Schedule

- Removing upfront all dependencies between coding activities practically impossible
  - Pure waterfall
  - Diminishing return when all paths are near critical

202

Infra+Clients Design in Front End

- Developer 1 and 2 work in parallel to front-end on
  - **Infra**structure
  - Clients apps
    - Requirements
    - Test plan
    - UI design

203

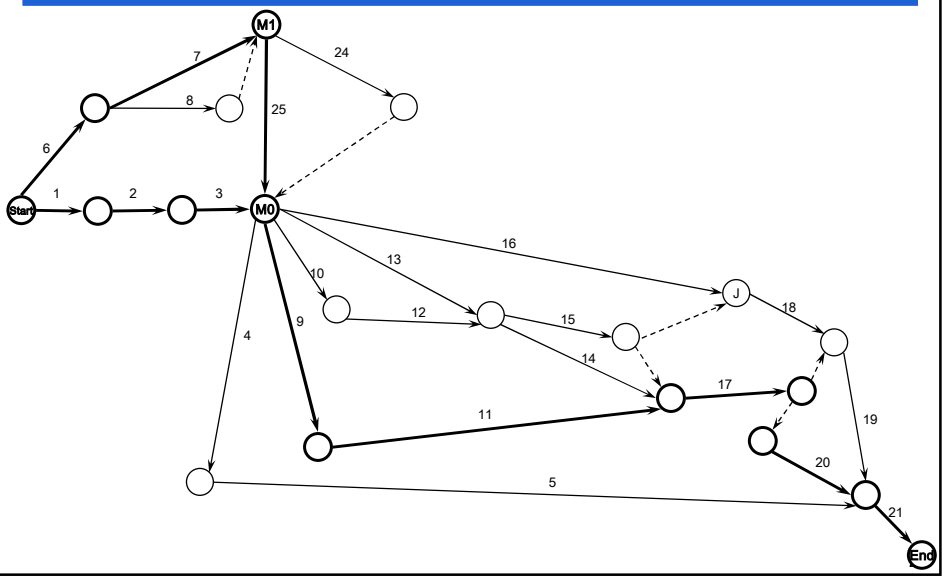
Infra+Clients Design in Front End

- Activities

ID	Activity	Duration	Depends on
0	**Start**	0	
1	Requirements	15	
2	Architecture	20	1
3	Project Planning	20	2
4	Test Plan	30	22
5	Test Harness	35	4
6	Logging	10	
7	Security	15	6
8	Pub/Sub	5	6
9	Resource A	20	22
10	Resource B	15	22
11	Resource Access A	10	9
12	Resource Access B	5	10
13	Resource Access C	10	22
14	EngineA	15	12,13
15	EngineB	20	12,13
16	EngineC	10	22
17	ManagerA	15	14,15,11
18	ManagerB	20	15,16
19	Client App1	15	17,18
20	Client App2	20	17
21	System Testing	30	5,19,20
22	M0	0	3,24,25
23	M1	0	7,8
24	Client App1 Design	10	23
25	Client App2 Design	15	23

204

Infra+Clients Design in Front End

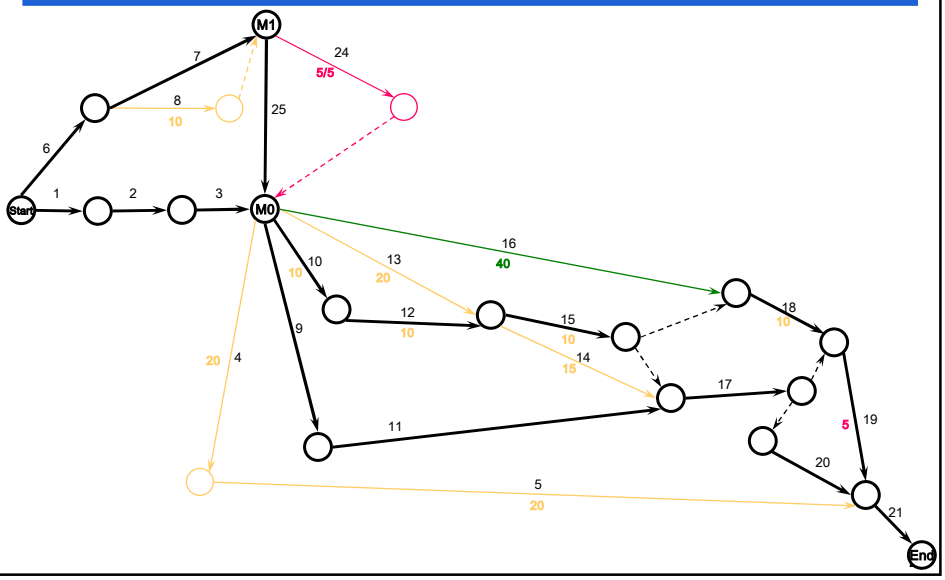


Infra+Clients Design in Front End

- Cost increased by 6%
- Duration reduced by 8%

206

Floats Analysis



Simulators

- Develop simulators for services
- Split clients into development against simulators and integration

208

Simulators

- Two additional developers required
  - Managers simulators
  - Clients implementation
- Also required for client integration at the end
  - Two developers for each integration

209

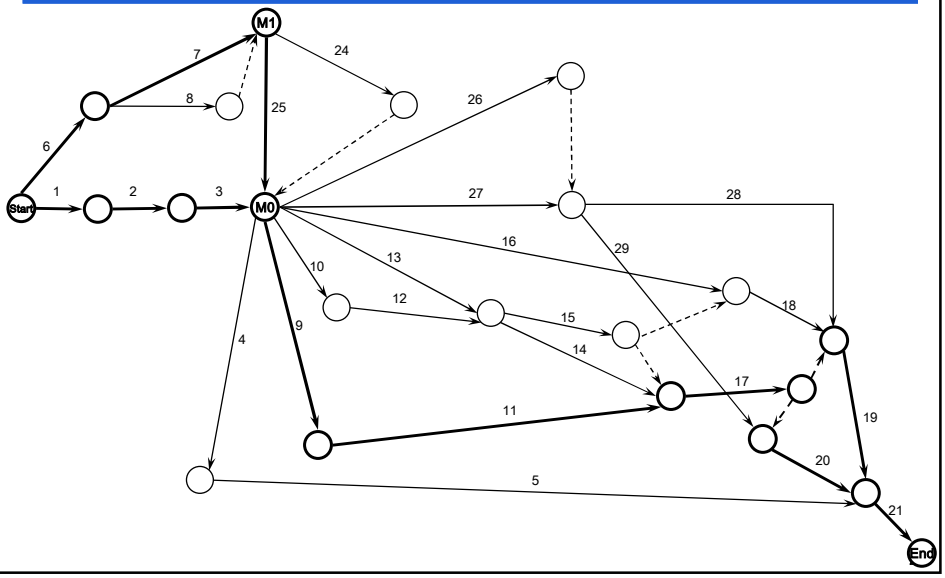
Simulators

- Activities

ID	Activity	Duration	Depends on
0	**Start**	0	
1	Requirements	15	
2	Architecture	20	1
3	Project Planning	20	2
4	Test Plan	30	22
5	Test Harness	35	4
6	Logging	10	
7	Security	15	6
8	Pub/Sub	5	6
9	Resource A	20	22
10	Resource B	15	22
11	Resource Access A	10	9,23
12	Resource Access B	5	10,23
13	Resource Access C	10	22,23
14	EngineA	15	12,13
15	EngineB	20	12,13
16	EngineC	10	22,23
17	ManagerA	15	14,15,11
18	ManagerB	20	15,16
19	Client App1 Integration	5	17,18,28
20	Client App2 Integration	5	17,29
21	System Testing	30	5,19,20
22	M0	0	3,24,25
23	M1	0	7,8
24	Client App1 Design	10	23
25	Client App2 Design	15	23
26	ManagerA Simulator	15	22
27	ManagerB Simulator	20	22
28	Client App1	15	26,27
29	Client App2	20	26

210

Simulators



Simulators

- Cost increased by 1.5%
- Duration reduced by 9%

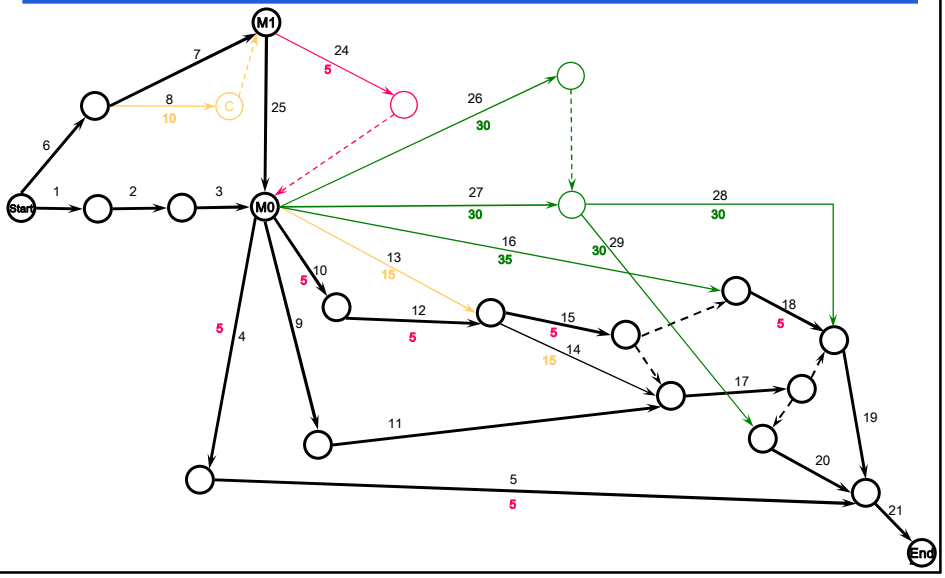
212

Network Compression

- There is a lot of integration pressure at the end of the project
- Fairly coupled and risky at full compression
- Overall compression compared with normal solution
  - Duration decreased by 28%
  - Cost increased by 4%
    - High indirect cost
  - Direct cost increased by 59%

213

Floats Analysis



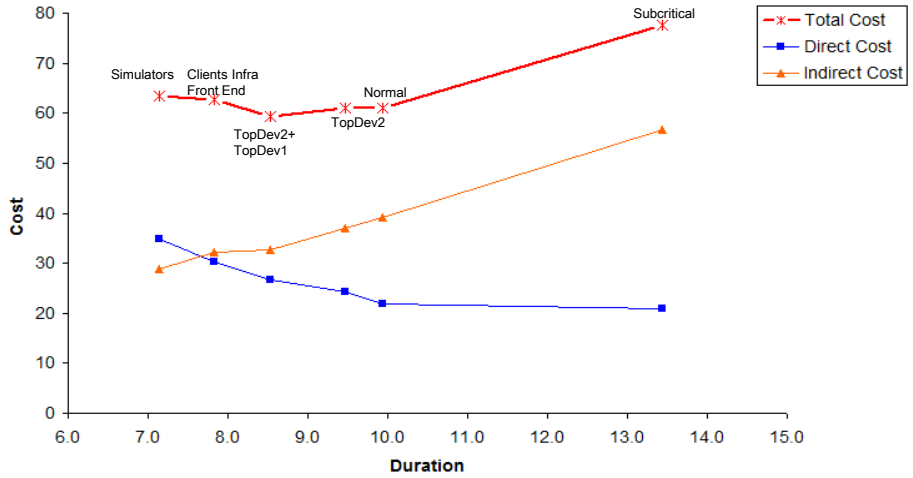
Floats Analysis

- While further compression is theoretically possible (services) in practice this is as far as the project design team should go
  - Low probability of success compressing any further
  - Wasting time designing improbable projects
- Natural limit compressing any project
  - 25-30%

215

Time Cost Curve

- Solutions compiled so far



216

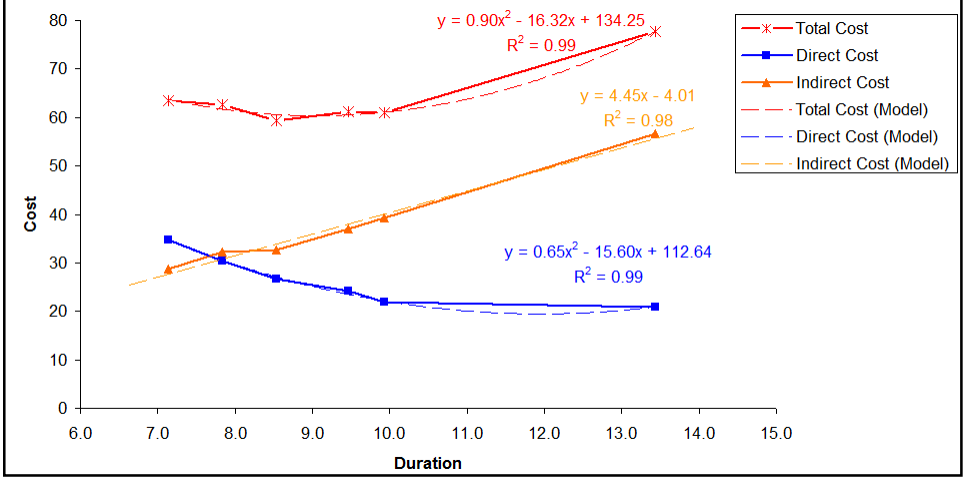
Time Cost Curve

- Can find correlation model
  - Indirect cost as straight line
  - Direct cost as polynomial of second degree
  - Total cost as polynomial of second degree

217

Time Cost Curve

- R^2 better than 95%



Time Cost Curve

- Total cost can be correlated but better just sum of other two
  - Indirect is true straight line
  - Exact sum of direct and indirect model
- Formulas

$$\text{Indirect Cost} = 4.45T - 4.01$$

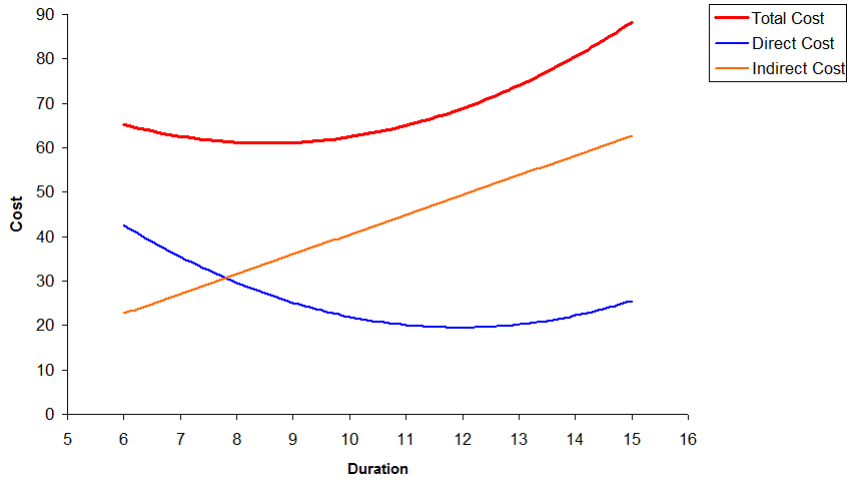
$$\text{Direct Cost} = 0.65T^2 - 15.6T + 112.64$$

$$\text{Total Cost} = 0.65T^2 - 11.15T + 108.63 = \text{Direct} + \text{Indirect Cost}$$

219

Time Cost Curve

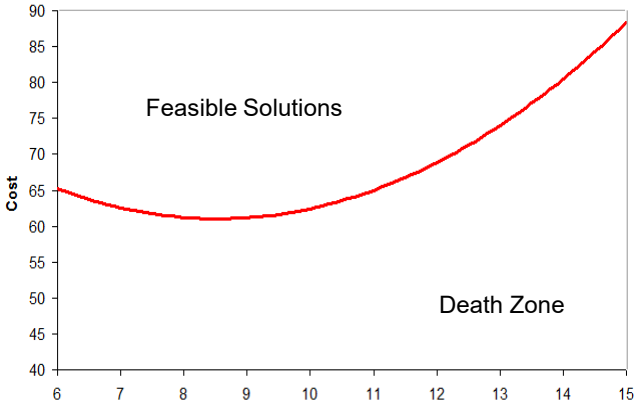
- Pure model



220

Time Cost Curve

- Can now avoid the death zone
  - And answer intelligently to quick questions



221

Time Cost Curve

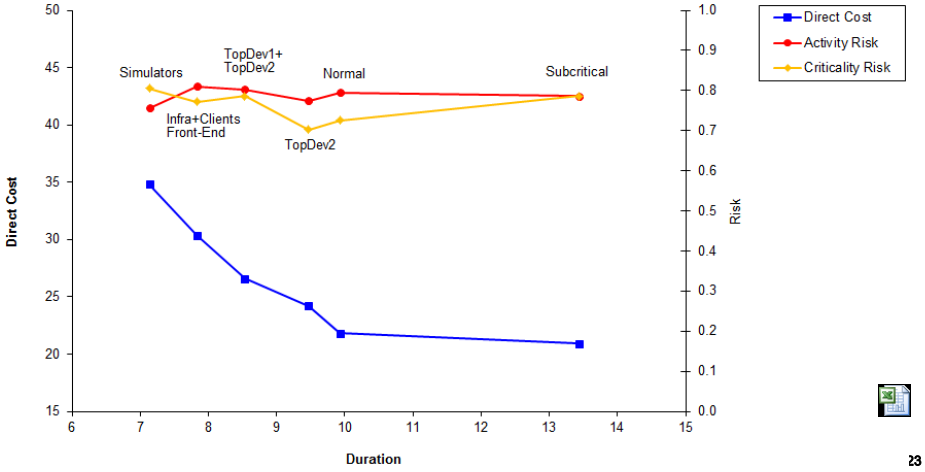
- Example
  - Suppose given 9 months and 4 people
- You will need at least 7 on average

```
Total Cost    = 0.65*9^2 – 11.15*9 + 108.63 = 61.2
Average staff = 61.2/9 = 6.8
```

  - Normal solution has 68% average staffing
    - You will need as many as 10
    - Less will go subcritical
- 9 months and 36 man-months is not even on the chart

Quantifying Risk

- Each compression option also changed project risk



23

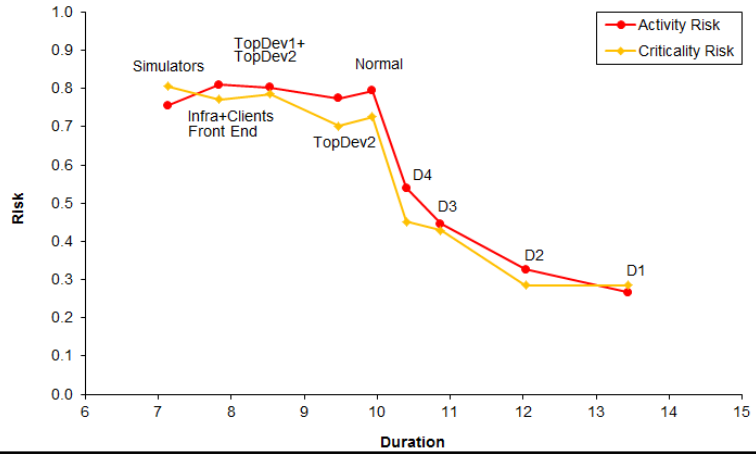
Quantifying Risk

- The good news
  - Criticality and activity risk track closely
- The bad news
  - All design options so far are high risk
    - And similar value too
  - Model based on subcritical point that should be avoided
- Solution
  - Risk decompression
  - Try 2 weeks, 1 month, 2 months, 3.5 months

224

Risk Decompression

- Complete risk curve



225

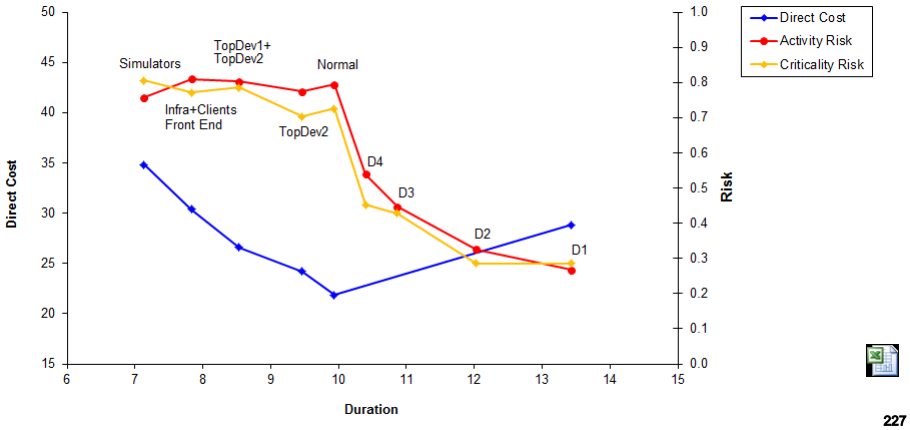
Risk Decompression

- Risk tipping point around D4
  - Decompressing even a little decreases risk substantially
  - D3 if you want safer since D4 is right on the edge

226

Cost and Risk Decompression

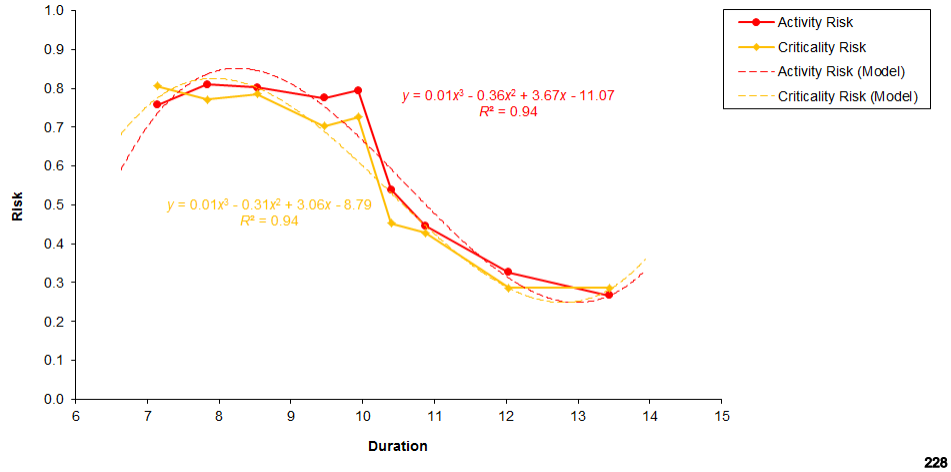
- D1 adds 3.5 months or 7 man-months for 2 developers



227

Modeling Risk

- Can add trend lines



228

Modeling Risk

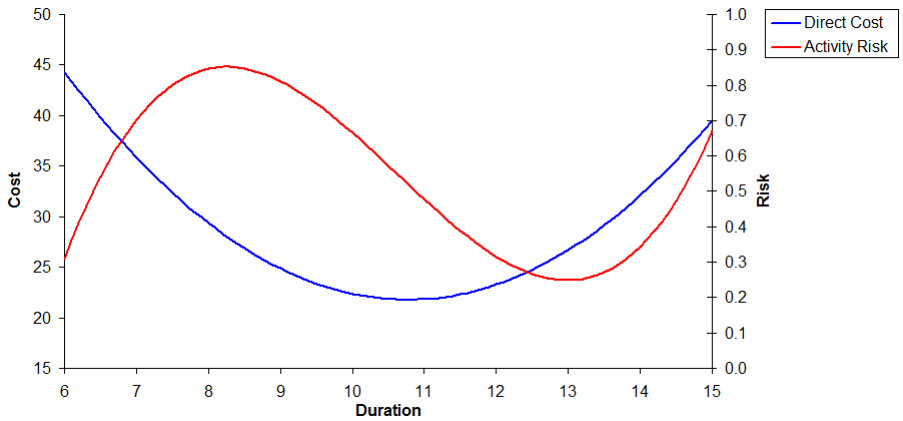
- Use activity risk only for example
  - Higher values so a bit more conservative
- Can produce risk equation for this project
  - How risk changes with planned duration

$$R = 0.01T^3 - 0.36T^2 + 3.67T - 11.07$$

229

Modeling Risk

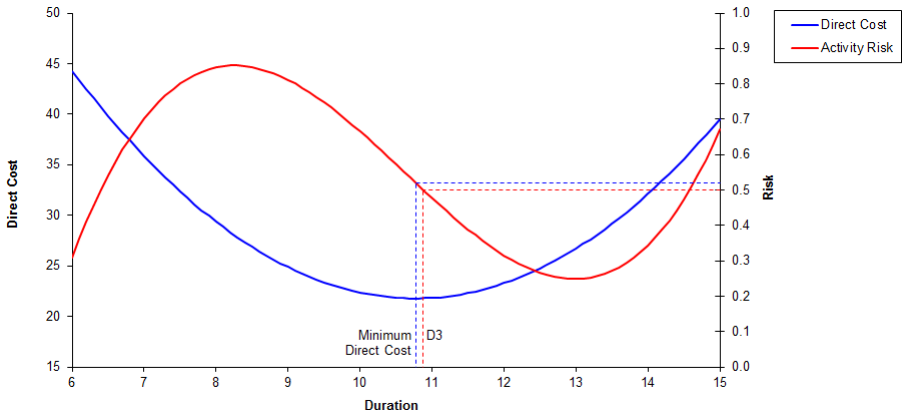
- Plot risk vs. direct cost model



230

Quantifying Risk

- Minimum direct cost and risk



231

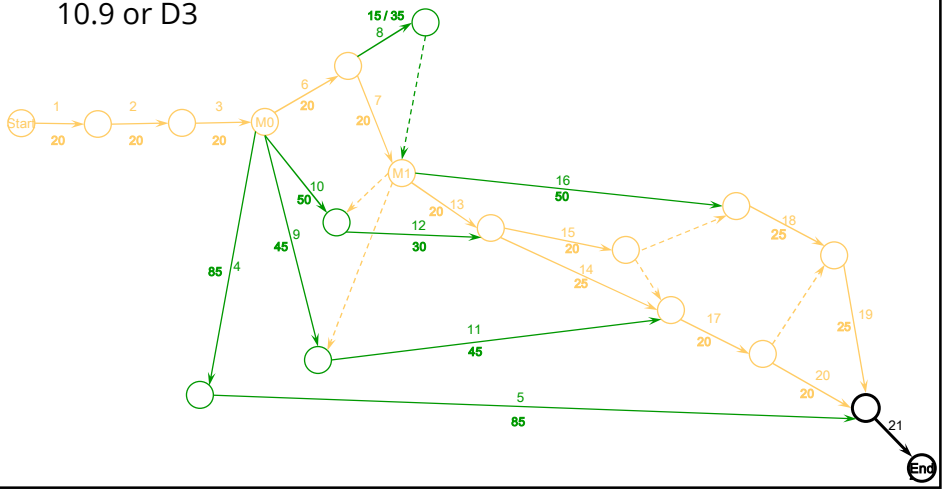
Quantifying Risk

- Minimum cost at 0.52
  - As expected
  - 10.8 months or just before D3 at 10.9 months
  - D3 at exactly 0.50
- Minimum direct cost is also least acceptable risk
  - For critically staffed projects
  - Optimal in cost duration and risk
- Highest probability of delivering on plan
  - Should always strive to design around minimum direct cost
  - Other options are there almost to explain why not to

232

Quantifying Risk

- Optimal decompression is to minimum direct cost point of 10.9 or D3



Planning and Risk

- Anything worth doing requires risk
  - Risk-free projects are not worth doing
- Assume less than 0.3 is too safe
  - Criticality risk cannot even go below 0.25
- This project risk tops at 0.85
  - Will you ever design for maximum risk?
- IDesign's thumb rule is 0.75 as likely upper practical limit
  - Too fragile and slip likely when > 0.75
- Can derive exclusion zones for planning options
  - 0.3 to 0.75 is acceptable

234

Risk Crossover Point

- Initially risk rises faster than cost
  - Risk maximized before minimum duration or all-crash
- Designing for maximum risk is unwise
  - Where is the cut off point?
- Can gage acceptable risk
  - When risk rises slower than cost you should probably stop compressing
  - Called risk crossover point
  - Conservative point
  - IDesign risk model typically at 0.75

235

Risk Crossover Point

- Requires comparing risk and cost rates of change
  - Analytically if model available
  - Numerically in Excel without a model is easy too
  - Must scale risk and cost first

236

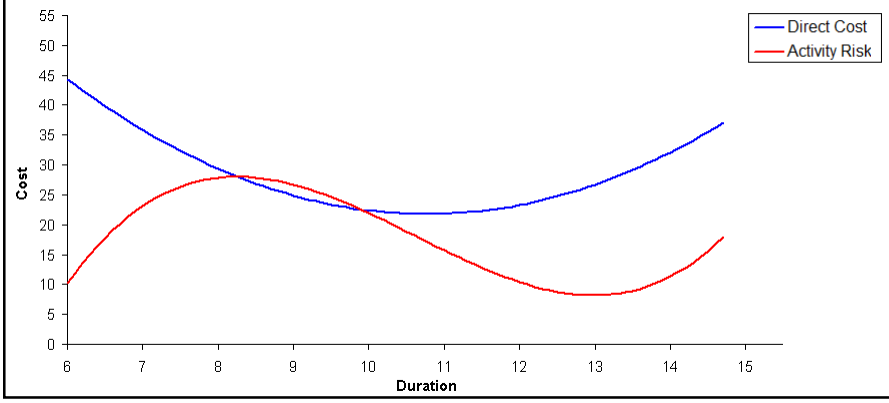
Scaling Risk and Cost

- Risk maximized when first derivative is zero
- Calculus reminder
  - $y = ax^3 + bx^2 + cx + d$
  - $Y' = 3ax^2 + 2bx + c$
- In our case
  - $R = 0.01T^3 - 0.36T^2 + 3.67T - 11.07$
  - $R' = 0.03T^2 - 0.72T + 3.67$

237

Scaling Risk and Cost

- Can plot risk and direct cost on same axis and scale



238

Risk Crossover Point

- Acceptable risk is when all are true
  - Left of minimum risk
  - Right of max risk
  - Risk rises faster than cost in absolute value to scale
    - $F * |R'| > |C'|$

$$C' = 1.98T^2 - 21.32$$

$$R' = 0.03T^2 - 0.72T + 3.67$$

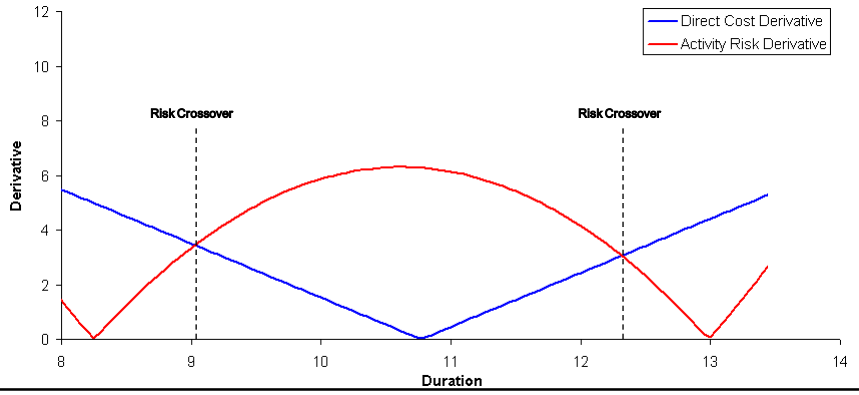
$$32.9*|0.03T^2-0.72T+3.67| > |1.98T - 21.32|$$

$$9.03 < T < 12.31$$

239

Risk Crossover Point

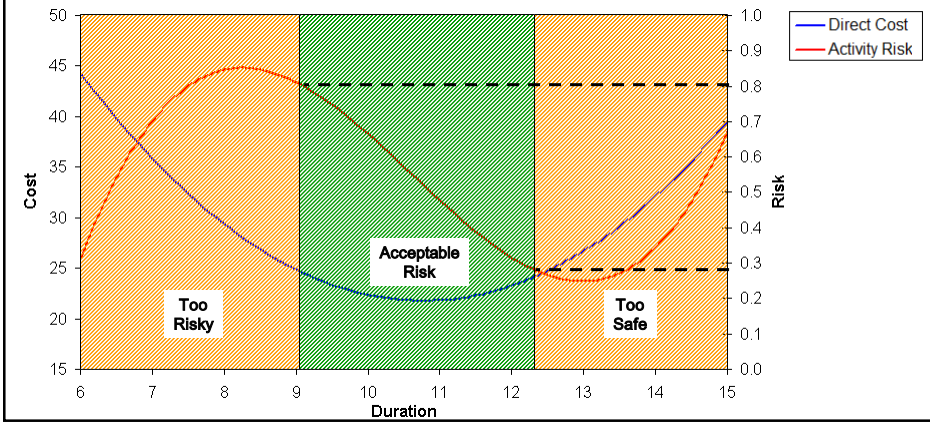
- There are two crossover points



240

Risk Crossover Point

- R(9.03) = 0.81
- R(12.31) = 0.28



241

When to Design a Project

- Several possible answers
- Always
- ROI question
  - Cost and time of designing project vs. saving in cost and time
  - Since it takes few days to a week for basic design it is no-brainer for most projects
    - In a week 17 iterations on a project while training
  - The larger the scope the more should invest in finding exact optimal solution across cost duration and risk
- With aggressive schedules

242

When to Design a Project

- Really integrity and responsibility question
  - If project funded 100% by your money, would you like architect and project manager to invest in project design?
    - A little investment or a lot?
  - If project manager is personally liable for cost of slips beyond commitments would PM invest in project design?
    - A little investment or a lot?
    - But the company always pays for it
    - Why are you callus or cavalier or complacent with their money?
- Getting ahead in life

243

More at DevIntersection

- Actors – The Past and Future of Software Engineering
  - Tuesday 10:30 AM
- Composable Design
  - Tuesday 12:00 PM

244

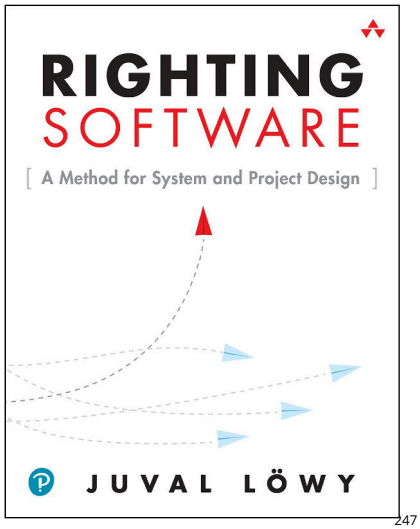

IDesign
Resources
Juval Löwy
www.idesign.net
©2021 IDesign Inc. All rights reserved


Resources
Slides
www.idesign.net/DevInt
246

Righting Software

- Addison-Wesley
  - December 2019

- www.rightingsoftware.org



247

Next IDesign Master Classes

- Architect's Master Class
  - August 2-6th, 2021, California
- Project Design Master Class
  - September 13-17th, 2021, California
- www.idesign.net

248