



NORTHROP GRUMMAN

DEFINING THE FUTURE



The Challenges of Requirements Decomposition

October 21, 2008

Eliza Siu
Northrop Grumman Corporation

Agenda

1. Introduction – Requirement Composition
2. Challenge 1 – Timing
3. Challenge 2 – High Level of Accuracy
4. Challenge 3 – Non-normal Data
5. Challenge 4 – Simulated Data
6. Recommendations
7. Conclusion
8. Questions and Answers

- What is typically done when requirements are decomposed or flowed down?
 - Split up one function into smaller ones and allocate them to various components, so that when each component performs its function, the entire function will be completed.
 - Performance requirements on timeline/speed/ accuracy, etc. are divided up similarly.
- Requirements must be verifiable or testable
 - Otherwise, one cannot tell if requirements are implemented correctly.
- Requirements need to be sold off

Challenge 1 – Timing (1 of 5)

- Example 1 – “classic” decomposition
 - The system shall complete the task within 10 sec
 - A shall complete its task within 3 sec.
 - B shall complete its task within 4 sec.
 - C shall complete its task within 2 sec.
 - Margin – 1 sec
- At a minimum
 - The start & end points at each component must be measurable
 - Need a well-defined boundary between the 2 components
 - Won’t work if B is an embedded library
 - Architecture & design are important
 - If components are not divided up logically, there will be issues with verifying requirements

Challenge 1 – Timing (2 of 5)

- The classic decomposition works just fine if the 10 seconds is a generous number, and each component's worst case timing is within the allocation.

| Component | Allocation | Typical | Worst case |
|-----------|------------|---------|------------|
| A | 3 | 2 | 2.5 |
| B | 4 | 2 | 3 |
| C | 2 | 1 | 1.5 |
| Total | 9 | 5 | 7 |

Challenge 1 – Timing (3 of 5)

- When the timing is tight

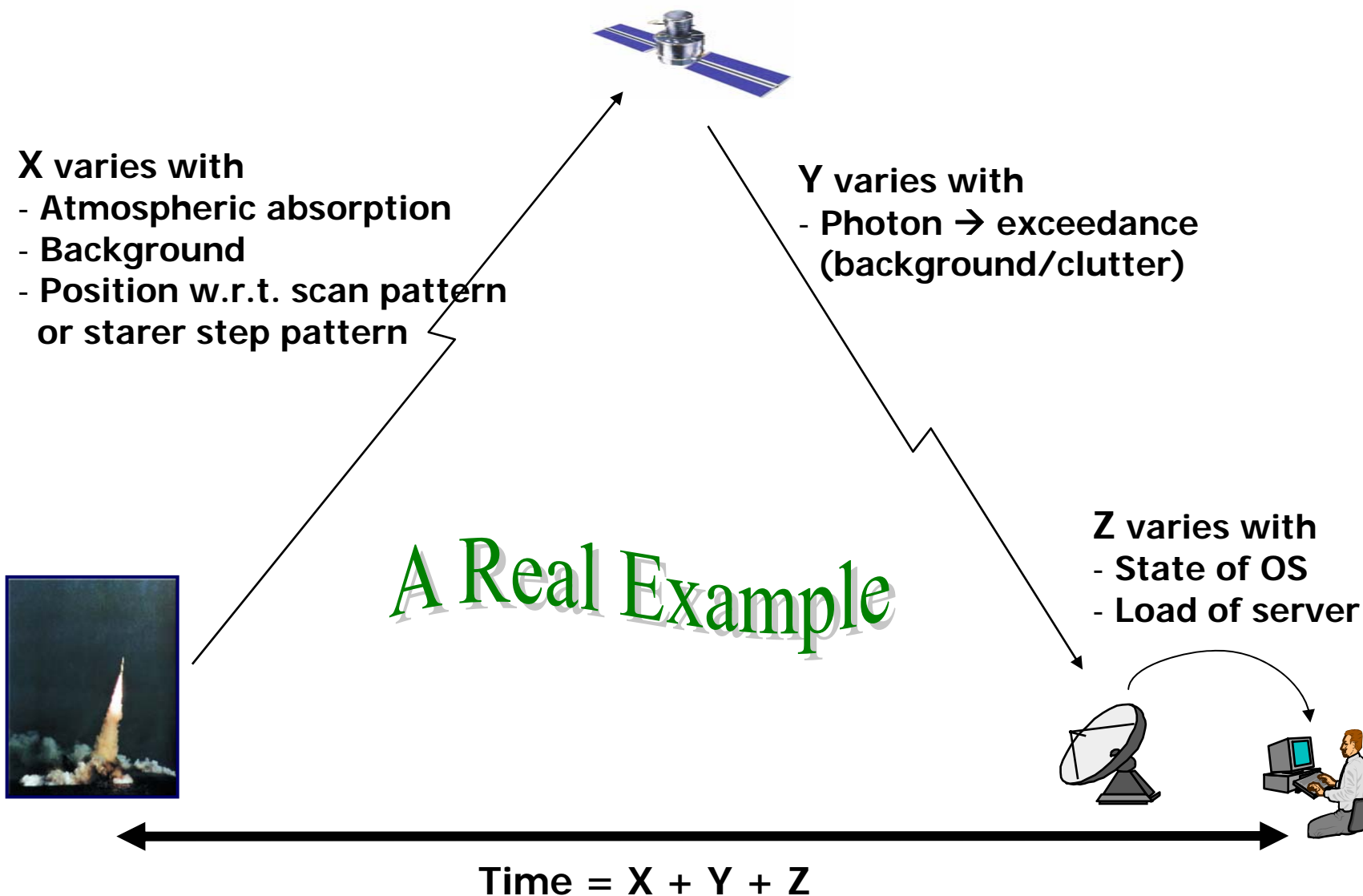
| Component | Allocation | Typical | Worst case |
|-----------|------------|---------|------------|
| A | 3 | 2.5 | 4 |
| B | 4 | 4 | 5 |
| C | 2 | 1.9 | 3 |
| Total | 9 | 8.4 | 12 |

Challenge 1 – Timing (4 of 5)

- ➔ In the case when the time allowed is very tight, requiring each component to individually satisfy the timing allocation may be a problem.
 - By measuring the time for the whole system, there is a much better chance of passing the requirement.

- ➔ May need customer agreement to bypass the tests for individual components and test the system.

Challenge 1 – Timing (5 of 5)



- Example 2A
 - a) The system shall do something 99% of the time.
 - b) The system shall do something 99.99% of the time.

- Mathematical distinction
 - a) To be able to distinguish 99% from 98%, at least 100 test cases must be run
 - b) To be able to distinguish 99.99% from 99.98%, at least 10,000 test cases must be run

- In order to obtain results that are statistically meaningful, larger samples are needed, so the no. of test cases needed will be even bigger
- Cost for running a test can be expensive
- Therefore, the consequence of higher accuracy must be understood

- Example 2B

How does the higher accuracy creep in at decomposition?

- The system shall do something 0.99 of the time
 - The accuracy is divided up as follows:
 - A shall do its task 0.9995 of the time
 - B shall do its task 0.9905 of the time
 - So that $0.9995 \times 0.9905 > 0.99$

- Therefore, the consequence & trade-off of decomposing requirements this way must be understood
 - Is the higher accuracy for each component needed?
 - With the much larger number of test runs for each component, is the system gaining a higher accuracy as a result?

- Statistical distinction

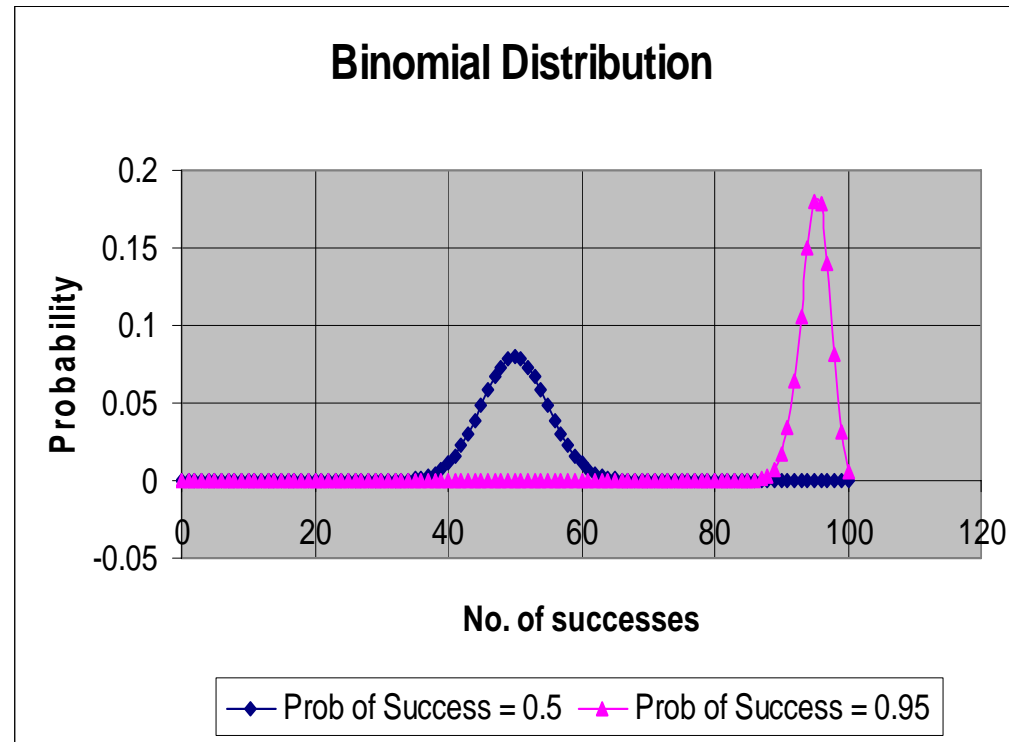
- In order to obtain results that are statistically meaningful, larger samples are needed. Furthermore, the more the data does not follow the normal distribution, the bigger the sample size should be.



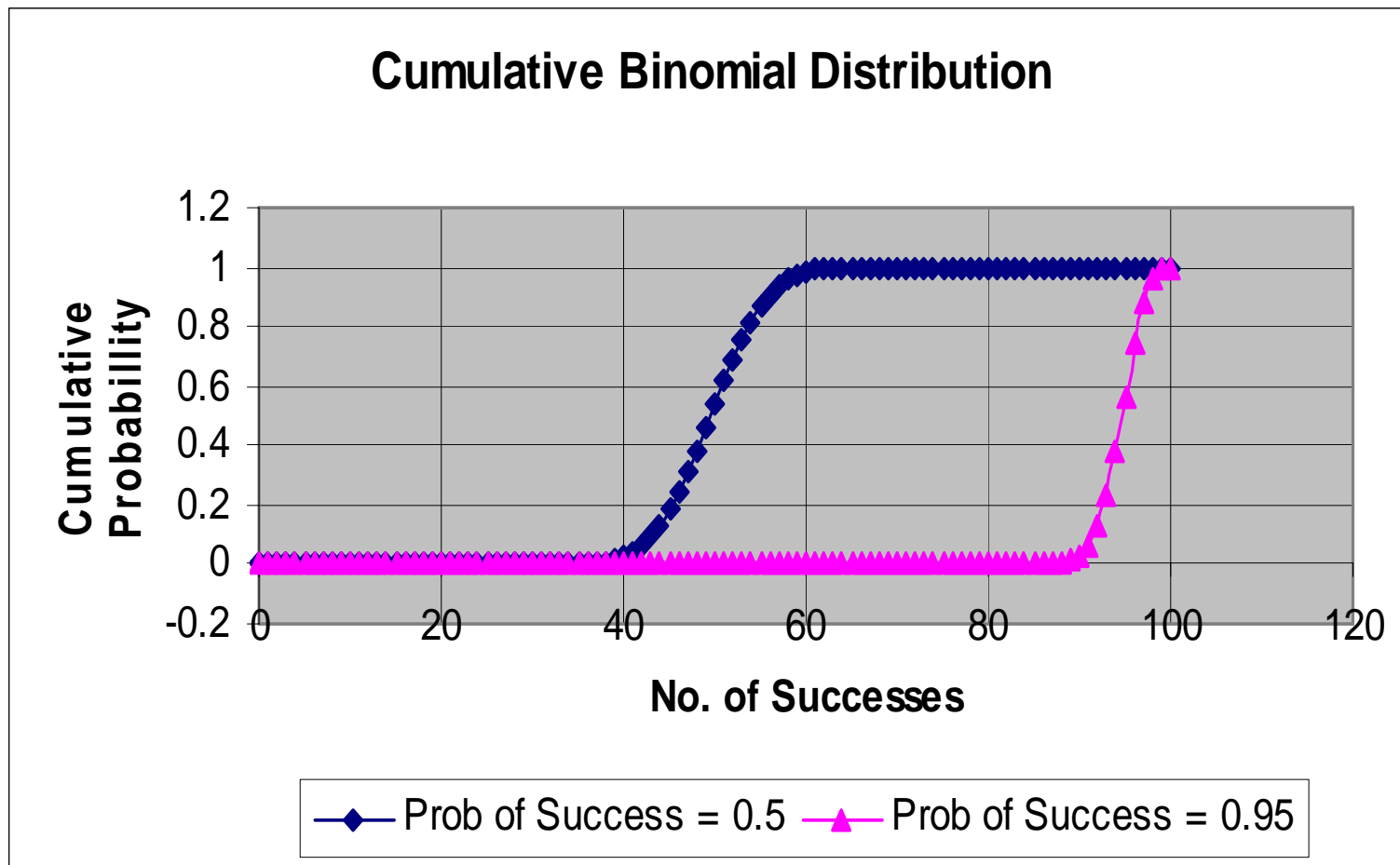
The distribution of the data measured is usually not well understood.

Challenge 3 – Non-normal Data (2 of 5)

- An example of binomial distribution
 - Probability of success of each trial is 0.5 or 0.95
 - Total no. of trial is 100



Challenge 3 – Non-normal Data (3 of 5)




Challenge 3 – Non-normal Data (4 of 5)

- Example 3
 - The system shall do something 1/month
- In both Examples 2 & 3, the probability of success is close to zero or one, i.e. the data is far from a normal distribution.

Assuming a minimum of 10 samples are needed

- 99.99% => need to run test case
(10,000 X 10) = 100,000 times
- 1/month => need to run test for 10 months

 If the requirement is on the order of 1/year, will the program ever have enough time to test the requirement?

Challenge 3 – Non-normal Data (5 of 5)

- If there are not sufficient resources to run the large no. of tests needed
 - If the requirement says ≥ 0.95
 - If values measured are 0.945, 0.955 & 0.951, is it conclusive that the system meets the requirement?

- Several possible solutions to this situation
 - Build a better system
 - Run more test cases
 - Buyer is willing to accept the risk
 - The smaller the no. of samples, the higher the risk

- Limitation of Simulated Data

- In order to verify requirements, simulated data will often be needed. However, the limitations of simulated data (or the models from which the data is created) must be understood

- The physical system that needs to be modeled/simulated is often not well understood to the level of precision required.

E.g.

How well do we think we can simulate the weather data for the next 10 days?

How well can we simulate a coin toss & to what level of fidelity?

- In Example 2, the simulated data for testing 99.99% needs to be many times more accurate than the data for testing 99%. Therefore, the models also need to be much more accurate.

- Program Decision

- The program needs to decide how much resources should be spent on modeling and simulation
 - Is it worth spending a lot of resources to improve the simulated data?
 - An alternative to spending resources on simulation is to conduct verification after the system is fielded. This is also a decision that the program needs to make.

- Never write requirements that are impossible to test or cost a lot to test
- Get involved as early as possible
 - Try to influence upper level requirements as early as possible, so that you won't have bad parent or grandparent requirements
 - Participate in other systems engineering activities, such as architecture development, and look out for potential problems
 - The earlier a problem is discovered, the less expensive it is to fix the problem

- If you are stuck with them, need to work with customer to find a way out, e.g.
 - Help customer to understand the
 - Problem
 - Alternatives
 - Cost vs benefit of each alternative
 - Get customer agreement on testing the system without testing individual components
 - Example 1 & 2B
 - Get customer agreement on a lower level of accuracy
 - Example 2A
 - Convince customer to accept higher risk
 - Example 3
 - Get customer agreement on testing the system after it is fielded
 - Example 4

- Systems/Requirements engineers need to
 - Know how to write/develop/decompose requirements, and also understand the impact of requirements written in various manners
 - Understand how the system works
 - Be proactive and get involved as early as possible
 - Be involved in higher level requirements, architecture, etc.
 - Include the effort in the BOE
 - Work with customers

- Contact Info
 - Eliza.Siu @ ngc.com
 - 626-812-1013

NORTHROP GRUMMAN

DEFINING THE FUTURE