

[PCA-06-scaled.jpg \(2560×1051\) \(perfectial.com\)](#)

## Lecture 5: Scalable PCA for Dimensionality Reduction

[COM6012: Scalable ML](#) by [Haiping Lu](#)

YouTube Playlist: <https://www.youtube.com/c/HaipingLu/>

# Week 5 Contents / Objectives

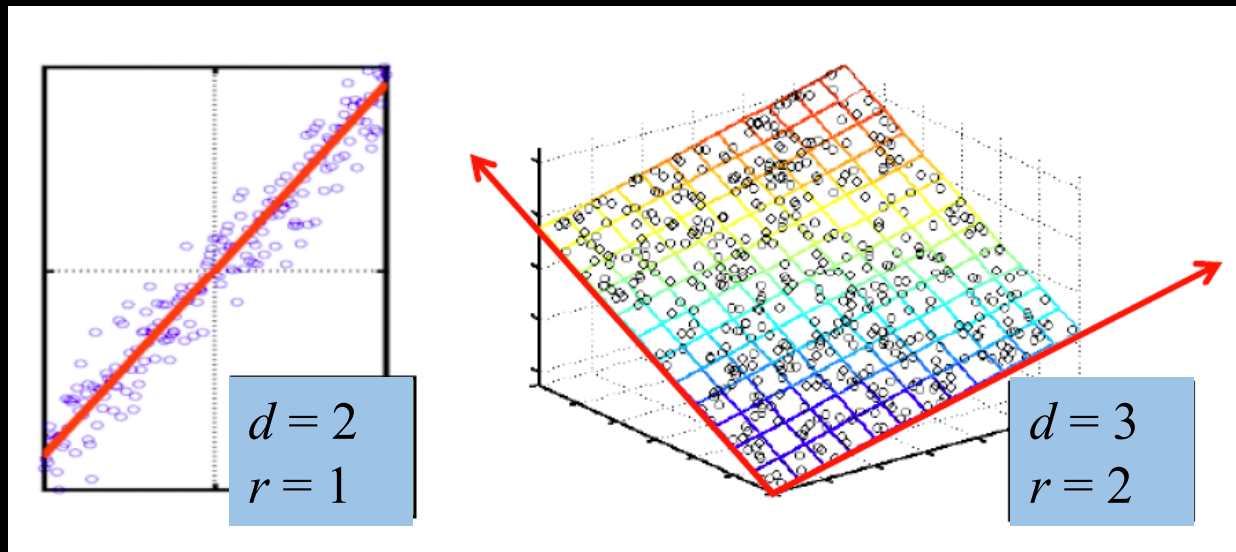
- Scalability Problem of PCA
- PCA via SVD
- Scalable PCA in Spark
- Software Development Life Cycle

# Week 5 Contents / Objectives

- Scalability Problem of PCA
- PCA via SVD
- Scalable PCA in Spark
- Software Development Life Cycle

# Motivation of Dimensionality Reduction

- Raw data: complex and high-dimensional
- **Assumption:** they lie on a low-dimensional subspace
  - Axes of this subspace  $\rightarrow$  representation of the data
  - Simpler, more compact, showing interesting patterns



# Utilities of Dimensionality Reduction

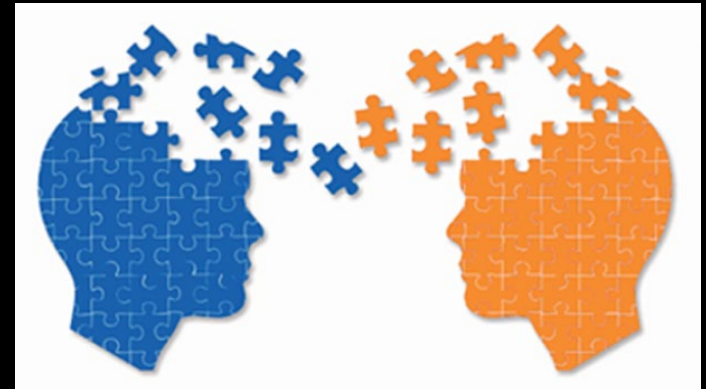
- Discover hidden correlations/topics
- Remove redundant/noisy features
- Interpretation and visualisation
- Easier storage and processing of the data



[owners-icebergs-blog-image-300x300.jpg \(resettogrow.com\)](#)



[1\\*KvKlx9OnlxdoTfNxWKAY\\_g.jpeg \(480x320\) \(medium.com\)](#)



[Interpreting and Translation Blog: Image \(wordpress.com\)](#)

# PCA $\rightarrow$ Variance Maximisation

- Input:  $n$   $d$ -dimensional data points
- PCA algorithm
  - $X_0 \leftarrow n \times d$  data matrix, data point  $\rightarrow$  row vector  $x_i$
  - $X$ : **subtract mean**  $\bar{x}$  from each row vector  $x_i$  in  $X_0$
  - $\Sigma \leftarrow X^T X$ : Gramian/scatter matrix for  $X$
  - Find eigenvectors and eigenvalues of  $\Sigma$
  - $U (d \times r) \leftarrow$  the top  $r$  eigenvectors (PCs)
- PCA features for  $y$ :  $U^T y$  (dimension:  $d \rightarrow r$ )
  - Zero correlations, ordered by variance

# Scalability Problem of PCA

- Input dimensionality  $\rightarrow$  scatter matrix
  - Images:  $100 \times 100 \rightarrow 10^4$ ;  $1000 \times 1000 \rightarrow 10^6$
  - Scatter matrix  $\Sigma$  is of size  $d^2$ 
    - $d = 10^4 \rightarrow \Sigma$  size  $10^8$
    - $d = 10^6 \rightarrow \Sigma$  size  $= 10^{12}$
- Alternative: Singular Value Decomposition (SVD)
  - Efficient algorithms available
  - Often need just top  $r$  eigenvectors

# Week 5 Contents / Objectives

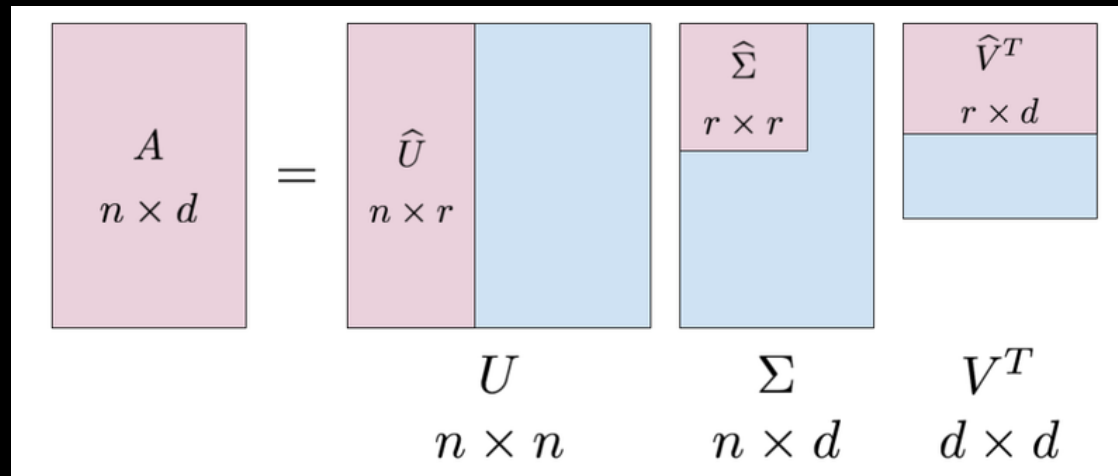
- Scalability Problem of PCA
- PCA via SVD
- Scalable PCA in Spark
- Software Development Life Cycle



# Singular Value Decomposition (SVD)

$$A_{[n \times d]} = U_{[n \times r]} \Sigma_{[r \times r]} (V_{[d \times r]})^T$$

- $A$ :  $n \times d$  matrix
- $r$ : the rank of the matrix
- $U$ :  $n \times r$  matrix, column orthonormal,  $U^T U = I$
- $\Sigma$ :  $r \times r$  diagonal matrix, strength of each factor
- $V$ :  $d \times r$  matrix, column orthonormal,  $V^T V = I$



# SVD $\leftrightarrow$ Eigen-decomposition

- SVD gives
  - $X = U \Sigma V^T$
- Eigen-decomposition gives
  - $B = X^T X = W \Lambda W^T$
- $U, V, W$ : **orthonormal**  $\rightarrow U^T U = I, V^T V = I, W^T W = I$
- $\Sigma, \Lambda$ : diagonal
- Relationship:
  - $XX^T = U \Sigma V^T (U \Sigma V^T)^T = U \Sigma V^T (V \Sigma^T U^T) = U \Sigma^2 U^T$
  - $X^T X = V \Sigma^T U^T (U \Sigma V^T) = V \Sigma \Sigma^T V^T = V \Sigma^2 V^T$
  - $B = X^T X = W \Lambda W^T = V \Sigma^2 V^T$

# PCA via SVD

- $X_0 \leftarrow n \times d$  data matrix, data point  $\rightarrow$  row vector  $x_i$
- $X$ : subtract mean  $\bar{x}$  from each row vector  $x_i$  in  $X_0$
- $U \Sigma V^T \leftarrow$  SVD of  $X$
- The top  $r$  **right singular vectors**  $V$  of  $X \rightarrow$  the PCs
- The singular values in  $\Sigma =$  the square roots of the eigenvalues of  $X^T X$

# Example on a Document x Term

Term Document	data	information	retrieval	brain	lung
CS-TR1	1	1	1	0	0
CS-TR2	2	2	2	0	0
CS-TR3	1	1	1	0	0
CS-TR4	5	5	5	0	0
MED-TR1	0	0	0	2	2
MED-TR1	0	0	0	3	3
MED-TR1	0	0	0	1	1

- $d = 5$  but  $r=2 \rightarrow$  two bases  $[1\ 1\ 1\ 0\ 0]$  &  $[0\ 0\ 0\ 1\ 1]$
- $U$ : document-to-concept similarity matrix
- $V$ : term-to-concept similarity matrix
- $\Sigma$ : its diagonal elements  $\rightarrow$  strength of each concept

# Interpretation

Term Document	data	information	retrieval	brain	lung
CS-TR1	1	1	1	0	0
CS-TR2	2	2	2	0	0
CS-TR3	1	1	1	0	0
CS-TR4	5	5	5	0	0
MED-TR1	0	0	0	2	2
MED-TR1	0	0	0	3	3
MED-TR1	0	0	0	1	1

retrieval

inf. brain lung

data

CS

MD

doc-to-concept similarity matrix

CS-concept

MD-concept

strength of CS-concept

CS-concept

term-to-concept similarity matrix

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

# SVD – Dimensionality Reduction

- To reduce the dimensionality further (3 zero singular values have already been removed)
  - Best rank-1 approximation  $\rightarrow$

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The image shows the SVD decomposition of a 7x5 matrix. The original matrix is on the left. It is equal to the product of three matrices. The first matrix is a 7x2 matrix of singular vectors, with the first column crossed out by a large yellow 'X'. The second matrix is a 2x2 matrix of singular values, with the second value (5.29) crossed out by a large yellow 'X'. The third matrix is a 2x5 matrix of right singular vectors, with the second row crossed out by a large yellow 'X'.

# Week 5 Contents / Objectives

- Scalability Problem of PCA
- PCA via SVD
- Scalable PCA in Spark
- Software Development Life Cycle

# Three PCA APIs in Spark MLlib

- DataFrame-based API – [PCA](#) ([source code](#), [Scala doc](#))
  - `pyspark.ml.feature.PCA(k=None, inputCol=None, outputCol=None)`
- RDD-based API – [RowMatrix](#) ([source code](#), [Scala doc](#))
  - `computePrincipalComponents(k)`
  - **Scalable**: `computeSVD(k, computeU=False, rCond=1e-09)`

```
465 @Since("1.6.0")
466 def computePrincipalComponentsAndExplainedVariance(k: Int): (Matrix, Vector) = {
467     val n = numCols().toInt
468     require(k > 0 && k <= n, s"k = $k out of range (0, n = $n]")
469
470     if (n > 65535) {
471         val svd = computeSVD(k)
472         val s = svd.s.toArray.map(eigValue => eigValue * eigValue / (n - 1))
473         val eigenSum = s.sum
474         val explainedVariance = s.map(_ / eigenSum)
```



# SVD in Spark MLlib (RDD)

- $U: m \times k ; \Sigma : k \times k ; V: n \times k$
- Assumption:  $n$  (dimensionality)  $< m$  (# samples)
- Different methods based on computational cost
  - If  $n$  is small ( $n < 100$ ) or  $k$  is large compared with  $n$  ( $k > n/2$ ), compute  $A^T A$  first and then compute its top eigenvalues and eigenvectors **locally** on the driver
  - Otherwise, compute  $(A^T A)_v$  in a **distributive** way and send it to ARPACK to compute  $(A^T A)$ 's top eigenvalues/eigenvectors on the driver node

# Selection of SVD Computation

```
334     if (n < 100 || (k > n / 2 && n <= 15000)) {
335         // If n is small or k is large compared with n, we better compute the Gramian matrix first
336         // and then compute its eigenvalues locally, instead of making multiple passes.
337         if (k < n / 3) {
338             SVDMode.LocalARPACK
339         } else {
340             SVDMode.LocalLAPACK
341         }
342     } else {
343         // If k is small compared with n, we use ARPACK with distributed multiplication.
344         SVDMode.DistARPACK
345     }
346     case "local-svd" => SVDMode.LocalLAPACK
347     case "local-eigs" => SVDMode.LocalARPACK
348     case "dist-eigs" => SVDMode.DistARPACK
349     case _ => throw new IllegalArgumentException(s"Do not support mode $mode.")
```

# Week 5 Contents / Objectives

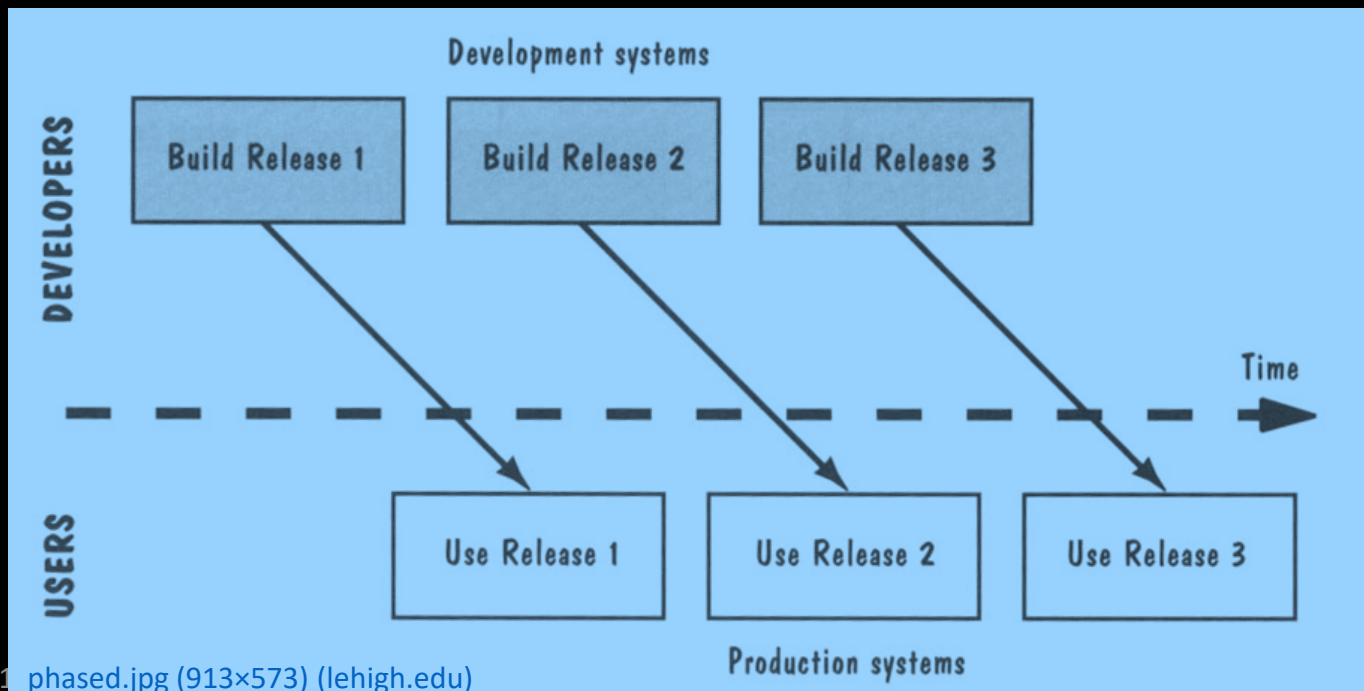
- Scalability Problem of PCA
- PCA via SVD
- Scalable PCA in Spark
- **Software Development Life Cycle**

# Software and Changes

- A virtue of software: relatively easy to change
  - Otherwise, it might as well be hardware
- Planning for change
  - Good *comments* describe meaning of code to facilitate and reduce the cost of software maintenance
  - *Modularity* help manage change because modules help to isolate and localize change

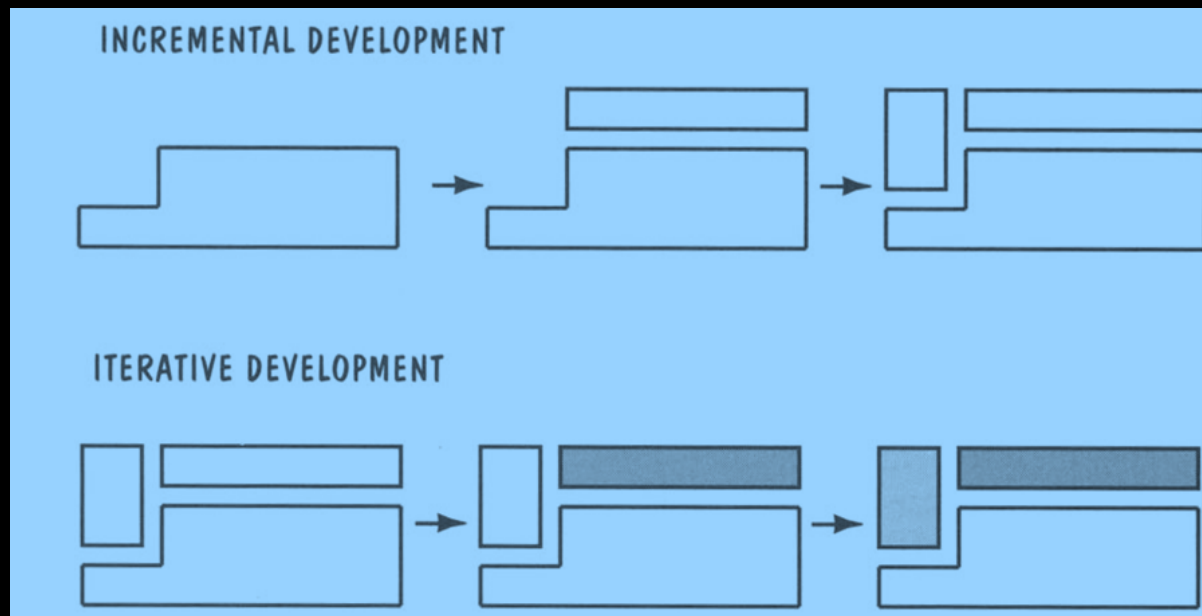
# Phased Development

- Reduce cycle time, deliver in pieces, let users have some functionality while developing the rest
- Two or more systems in parallel
  - The **operational/production** system in use by customers
  - The **development** system to replace the current release



# Iterative/Incremental Development

- **Incremental**: partition a system by functionality
  - Early release: small, functional subsystem
  - Later releases: add functionality
- **Iterative**: improve overall system in each release

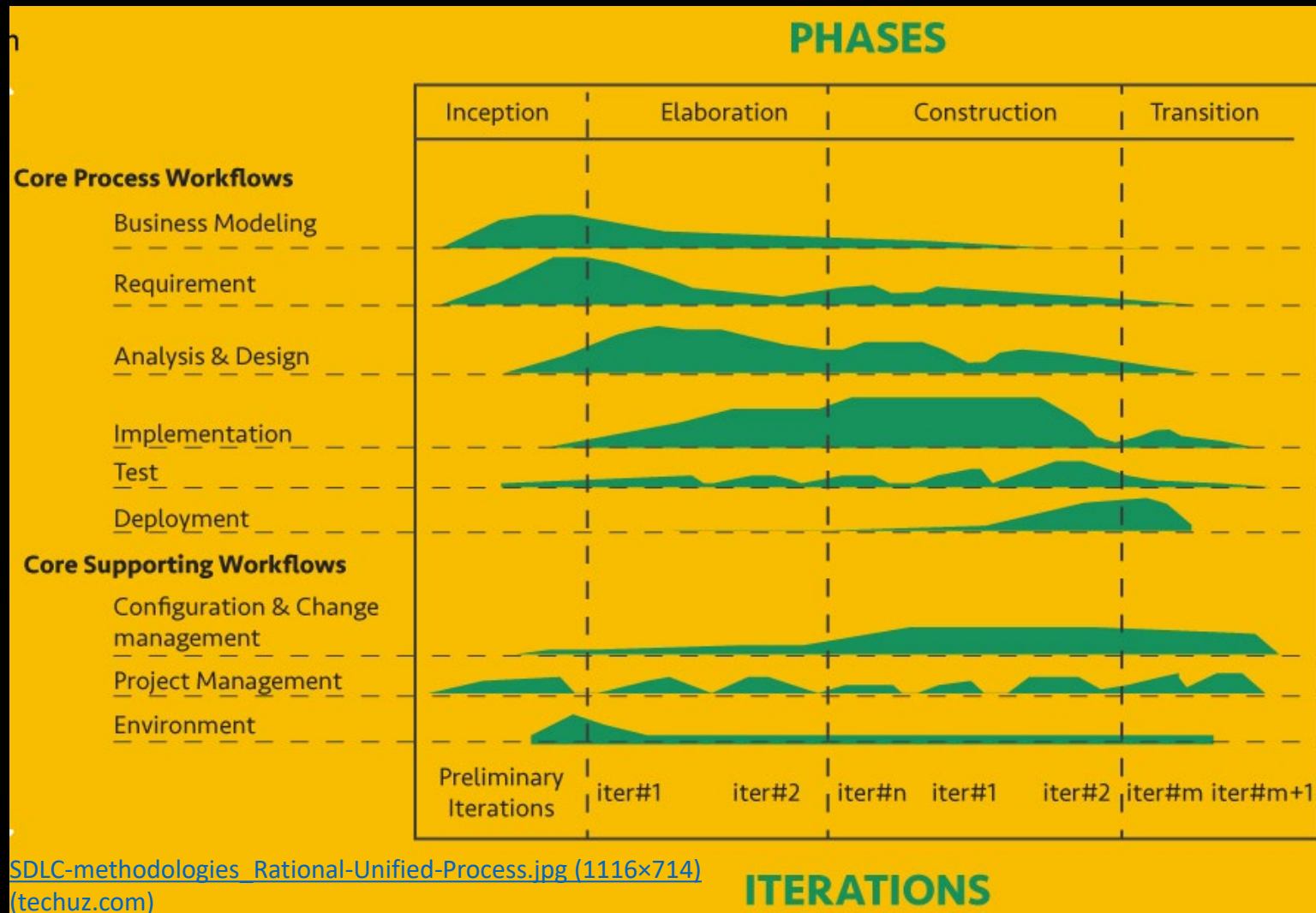


[iterative.jpg \(902×539\) \(lehigh.edu\)](#)

# Lifecycle Phases

- **Inception**: rationale, scope, and vision
- **Elaboration**: “Design/Details”
  - detailed requirements and high-level analysis/design
- **Construction** – “Do it”
  - build software in increments, tested and integrated, each satisfying a subset of the requirements
- **Transition** – “Deploy it”
  - beta testing, performance tuning, and user training
- Phases: NOT the classical requirements/design/coding/implementation processes
- Phases iterate over many cycles

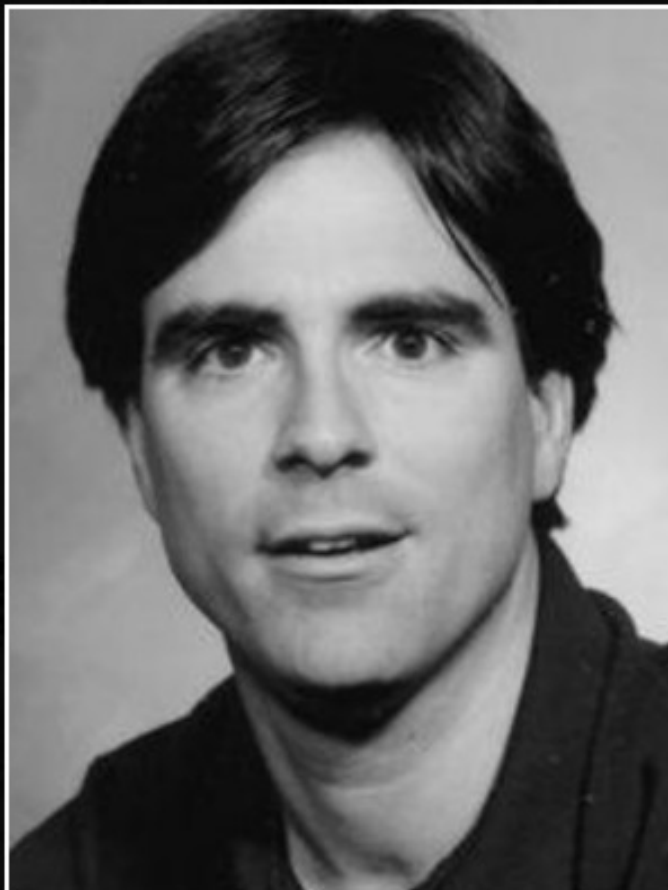
# Phases: Iterative & Incremental





# Work on Your Project

- Get a **small subset** or a reduced version to study, develop, debug, and test
- Break down big/difficult problem into smaller/easier sub-steps (**avoid black-box debugging**)
- Be structured, organised, and logical
- Keep good documentation
- Get help online (e.g. search) and keep the references



Engineering isn't about perfect solutions; it's about doing the best you can with limited resources.

— *Randy Pausch* —

**AZ QUOTES**

[Randy Pausch quote: Engineering isn't about perfect solutions; it's about doing the best... \(azquotes.com\)](http://azquotes.com)

# Acknowledgement & References

- Acknowledgement
  - Some slides are adapted from the [MMDS book](#) slides and the slides on software process life cycles by [Glenn Blank](#)
- References
  - [Chapter 11](#) of the [MMDS book](#)