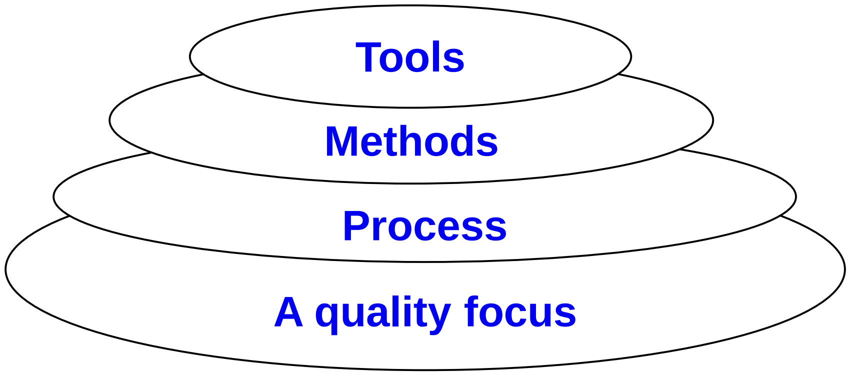
Chapter 2
The Process

Reference text books:

- Software Engineering
  A Practitioner's Approach
  Roger S. Pressman
  Fifth Edition, 2001
- Software Engineering, Ian Sommerville,
  6th Edition, 2000
- Software Project Management, Bob Hughes
  and Mike Cotterell, 2001
- Software System Development: A Gentle Introduction,
  Carol Britton & Jill Doake, Third Edition, 2003

2.1. A Layered Technology

a) Process, Methods, and Tools



- The foundation for SE is the *process* layer. SE process is the glue that holds the technology layers together and enables rational and timely development of computer SW.
- SE *method* provide the technical how-to's for building SW (Communication, Requirements, Design, Code, Testing, Deployment, support)
- SE tools provide automated or semi-automated support for the process and the methods

2.1. A Layered Technology

b) <u>A Generic View of SE</u>

- The work associated with SE can be categorized three generic phases:
  - The *definition phase* focus on ***what***
    - What information is to be processed
    - What function and performance are desired
    - What system behavior can be expected
    - What interfaces are to be established
    - What design constraints exist
    - What validation criteria are required to define a successful system

    * Three major tasks will occur in some form:
      - System or information engineering (Chapter 10 of **[2]**)
      - SW project planning (Chapter 3, 5, 6, and 7 of [**2**])
      - Requirements analysis (Chapter 11, 12, and 21 of **[2]**)

2.1. A Layered Technology

b) <u>A Generic View of SE</u>

- The *development phase* focus on ***how***
  - How data are to be structured
  - How function is to be implemented with a SW architecture
  - How procedural details are to be implemented
  - How interfaces are to be characterized
  - How the design will be translated into programming language
  - How testing will be perform

  * Three specific technical tasks should always occur:
    - ▪ SW design (Chapter 13-16, and 22 of **[2]**)
    - ▪ Code generation and SW testing (Chapter 17, 18, and 23 of **[2]**)

2.1. A Layered Technology

b) <u>A Generic View of SE</u>

- The *support(maintenance) phase* focus on ***Change*** associated with error correction, adaptation, enhancement, prevention. Four types of change are encountered during the support phase:
  - Correction $\rightarrow$ *Corrective maintenance*
  - Adaptation $\rightarrow$ *Adaptive maintenance*
  - Enhancement $\rightarrow$ *Perfective maintenance*
  - Prevention $\rightarrow$ Preventive maintenance

2.2. Software Life-Cycle

- SW Life-Cycle គឺជាដំណាក់កាលគិតចាប់ពីពេលដែល SW ត្រូវបានបង្កើតឡើងរហូតដល់ពេលគេលែងប្រើ (ពីពេលចាប់កកើត ឆ្លើយតបទៅនឹងតម្រូវការ Operate, maintenance រហូតដល់ពេលបោះបង់ចោលលែងប្រើ)។
- SW Life-Cycle ត្រូវបានចែកជា Phase សំខាន់ៗគឺ Analysis, Design, Coding, Testing, maintenance។ ការសម្តែងជា phase ទាំងឡាយមានភាពខុសគ្នាទៅតាមមនុស្សម្នាក់ៗ។

2.2. Software Life-Cycle

- **Primary functions of a software process model**
  - Determine the order of the stages involved in software development and evolution.

2.2. Software Life-Cycle

- **Why are software process models important?**

  They provide guidance on the order in which a software development project should carry out its major tasks

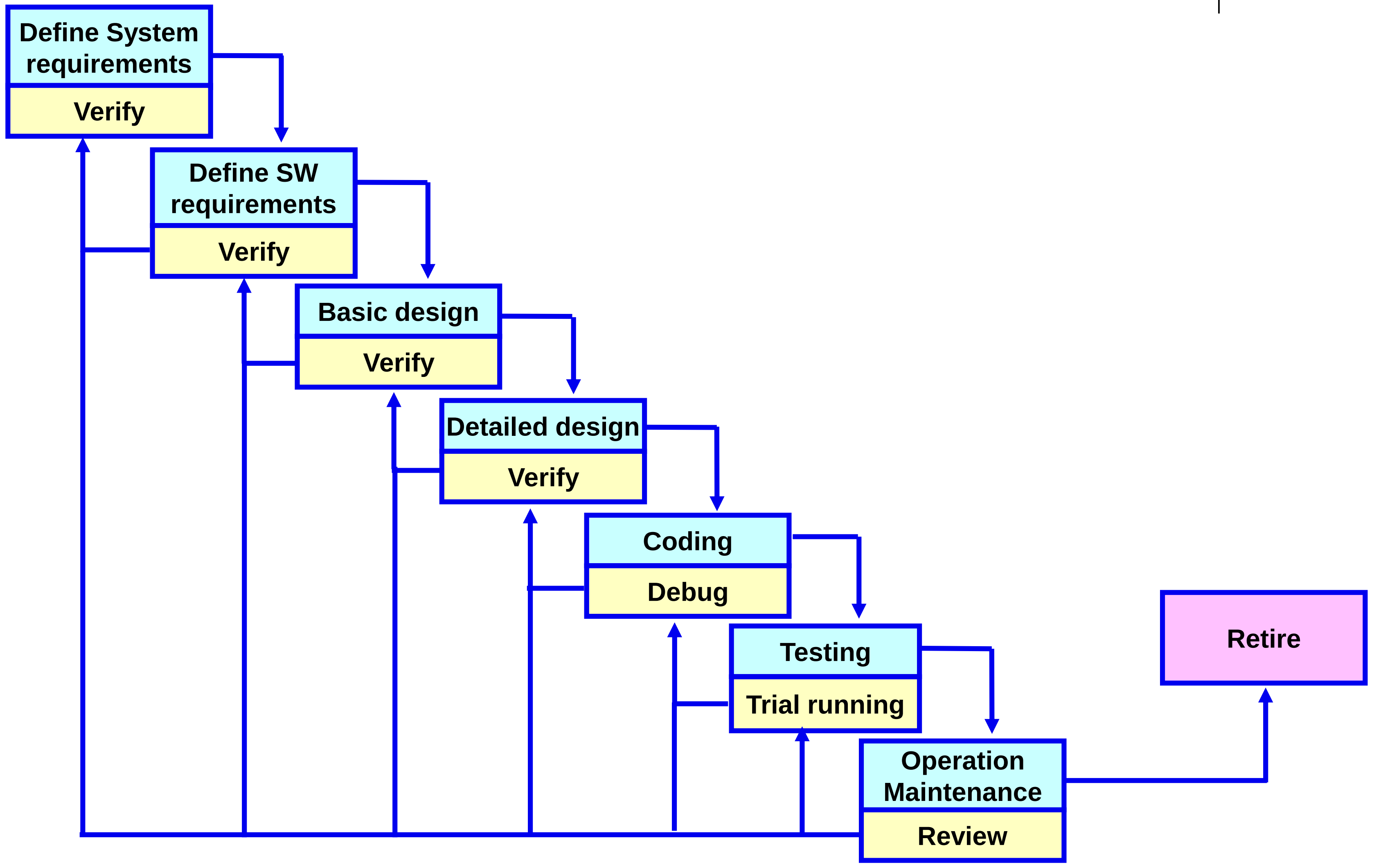
2.2. Software Life-Cycle

1) <u>The Waterfall Model Life Cycle</u>

- ត្រូវបានបង្កើតឡើងក្នុងឆ្នាំ១៩៧០ ដោយលោក Winston W. Royce
- គេប្រើ model នេះនៅពេលគេដឹងច្បាស់លាស់នូវតម្រូវការទាំងឡាយជាមុនសិន។

2.2. Software Life-Cycle

1) The Waterfall Model Life Cycle



2.2. Software Life-Cycle

* **<u>The New Additional of SW Life-Cycle</u>**

(1) Requirements definition and Design phase មានតួនាទីកំណត់ទៅលើគុណភាព SW ដែលប្រើកម្លាំងអស់មួយភាគធំបើប្រៀបទៅនឹងការសរសេរ Code ការធ្វើតេស្ត និងចែកចាយ SW។

(2) ជា Phase ដែលធ្វើឲ្យ Structure របស់ SW កាន់តែមានភាពជាក់ស្តែងទៅតាមរបៀប Top-down។

(3) Design, Coding phase គឺធ្វើតាមរបៀប Top-down រីឯ Testing phase តាមរបៀប Bottom-up។

(4) មុនពេលឈានទៅដល់ Phase បន្តទៀតត្រូវធានាថា Phase ដែលកំពុងអនុវត្តបានធ្វើតេស្តរួចហើយ ដោយពុំមាននៅសល់កំហុសទៀតឡើយ។

(5) ចាំបាច់ត្រូវមានយន្តការត្រួតពិនិត្យគុណភាព ពិនិត្យមើលឡើងវិញរវាង Phaseនីមួយៗ ដើម្បីធានាកុំឲ្យបង្កកំហុសដល់Phaseក្រោយៗទៀត។

(6) ឯកសារនៃ Phaseនីមួយៗមិនត្រឹមតែប្រើសម្រាប់ Phaseក្រោយៗប៉ុណ្ណោះទេ ថែមទាំងសម្រាប់គោលដៅសំខាន់ៗដល់ការត្រួតពិនិត្យ និងធានាគុណភាពនៃ Process នីមួយៗ និងសម្រាប់ SW ខ្លួនឯងទៀតផង។

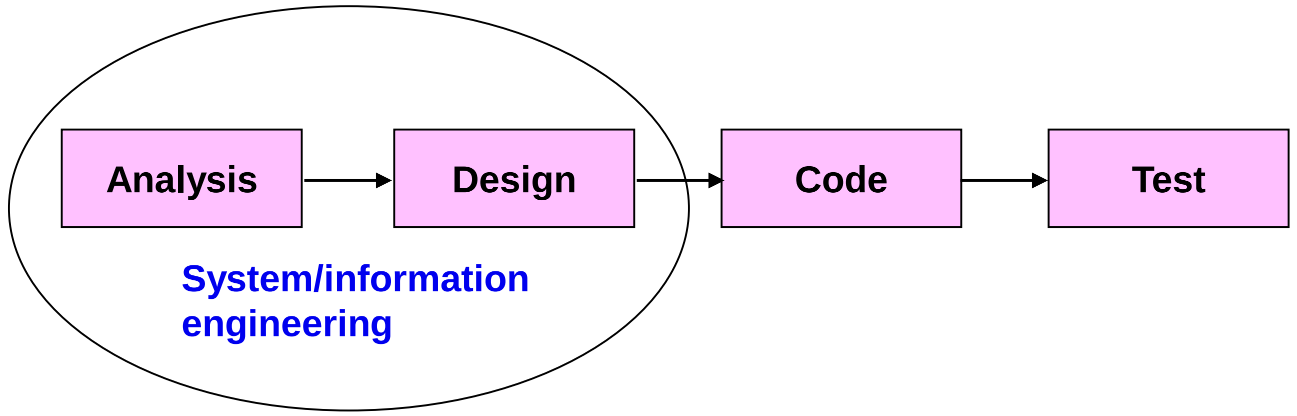
2) Linear Sequential Model

Also known as:

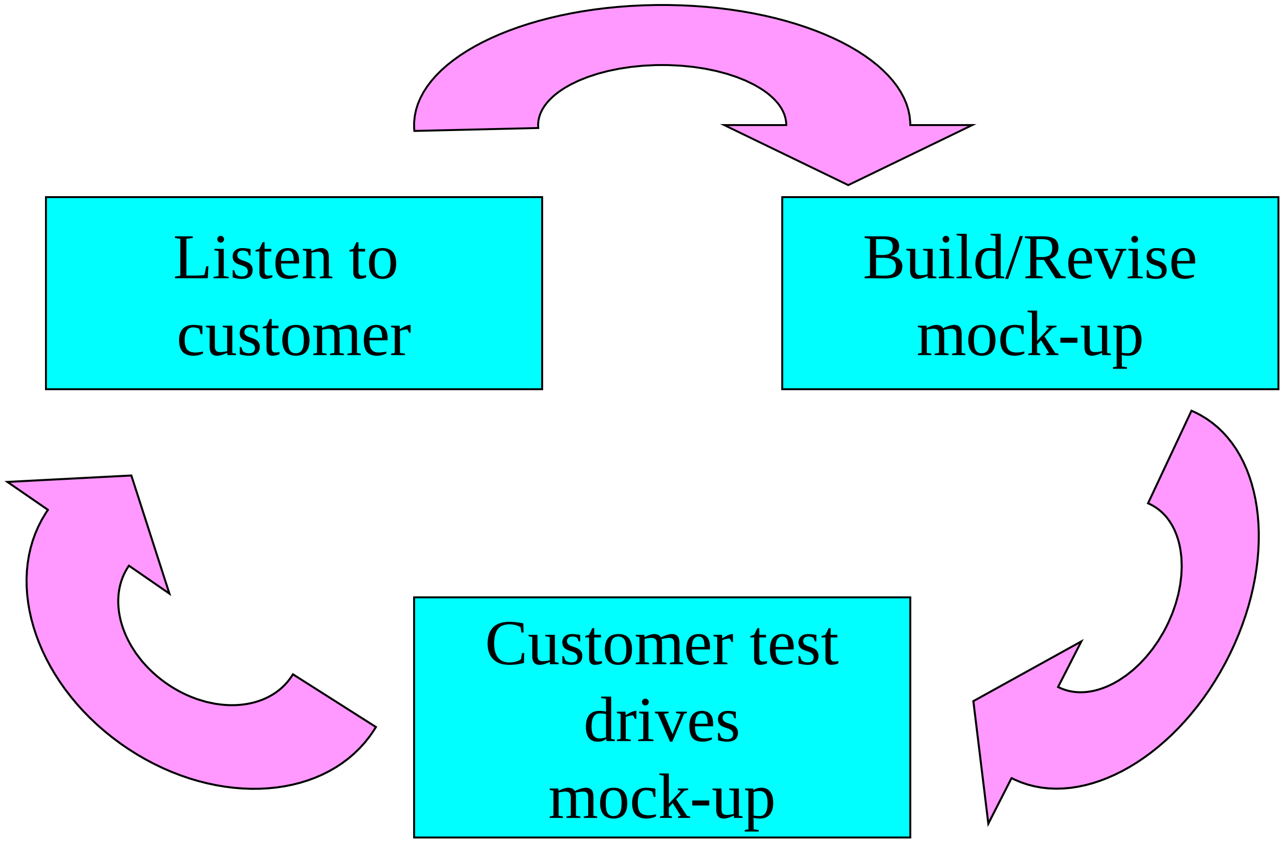
- Classic life cycle model or
- System development life cycle (SDLC) model

* This is a good model to use when requirements are well understood

2) Linear Sequential Model



3) The Prototyping Model (Throw away Model)

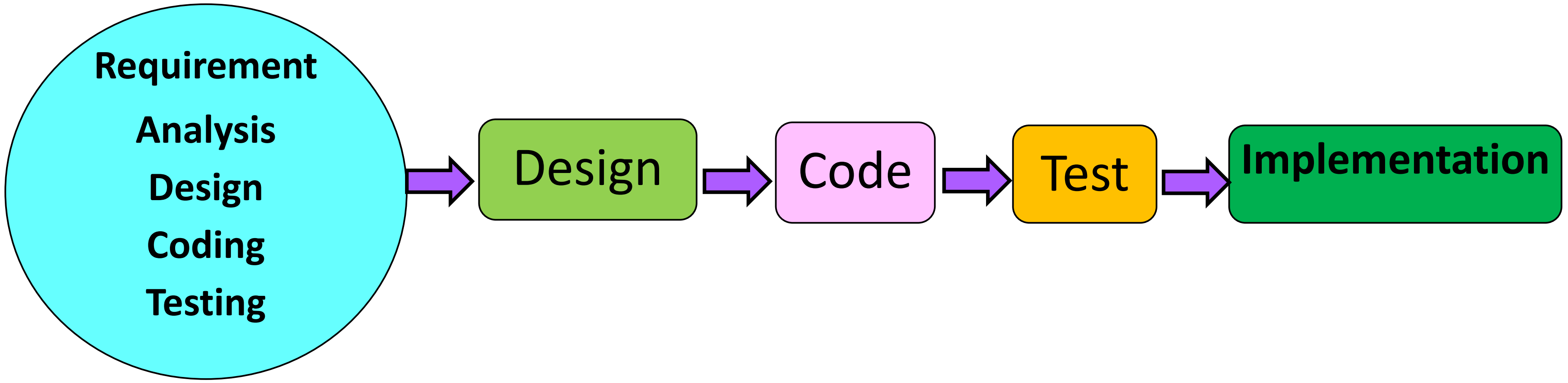


- Objective is to understand the system requirements
- Should start with poorly understood requirements to clarify what is really needed.

3) The Prototyping Model (Throw away Model)

- You use this model when the client is not clear about the requirements of the proposed system or when there is a conflict in client requirements
- To resolve the conflict, the development team develops a working model so that the requirements of the client become defined

3) The Prototyping Model (Throw away Model)

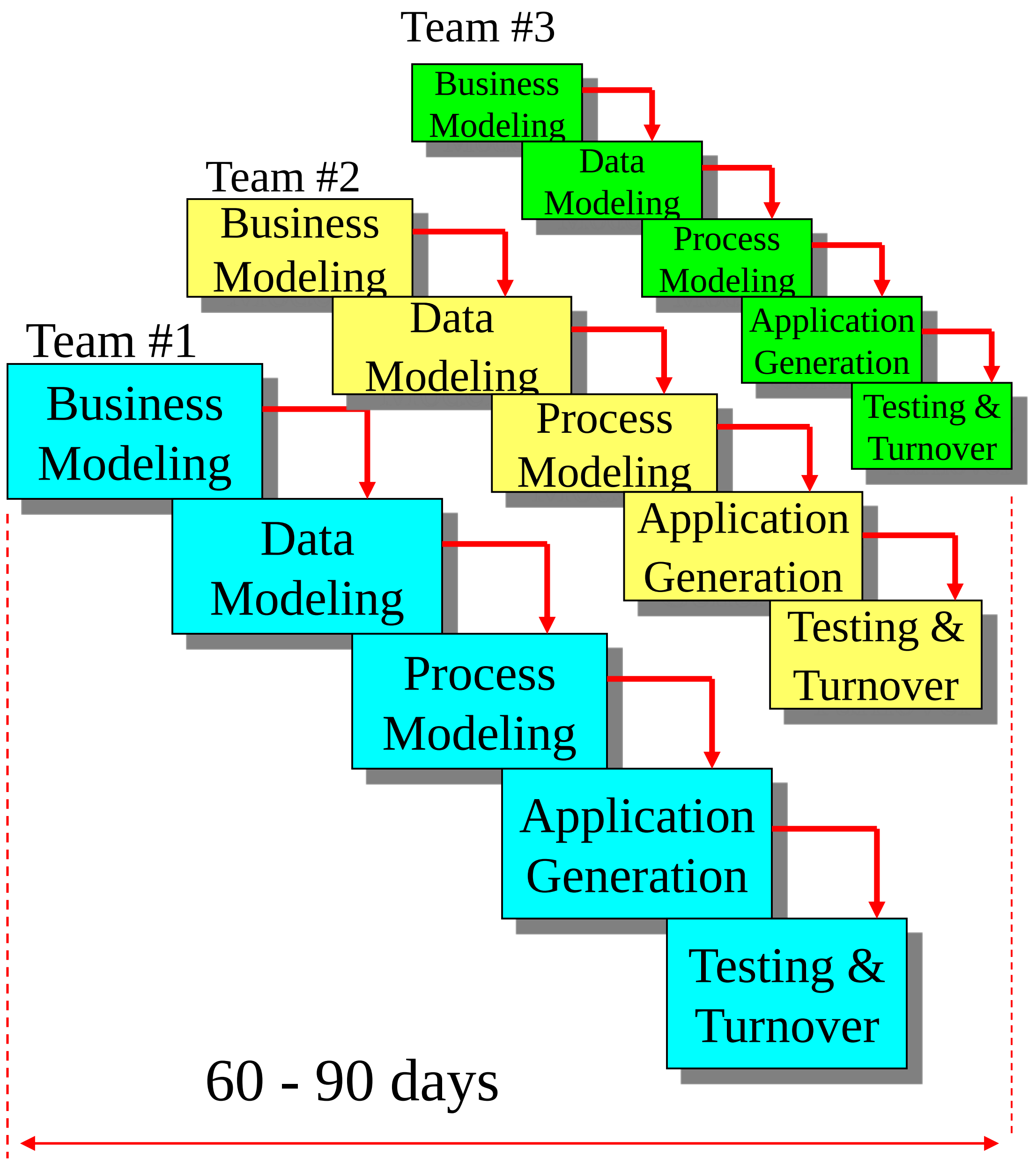


3) The Prototyping Model (Throw away Model)

តើត្រូវធ្វើ Prototyping model នៅពេលណា?

- នៅពេលទើបដឹងនូវគោលបំណងគ្រួសៗនៃ SW មិនទាន់ច្បាស់សេចក្តីលម្អិតនូវអ្វីជា Input ឬ Process យ៉ាងដូចម្តេច ឬមិនទាន់ច្បាស់នូវអ្វីជាតម្រូវការសម្រាប់ output នៅឡើយទេ។
- ប្រើ "The first system" ដើម្បីប្រមូលតម្រូវការពីអ្នកប្រើប្រាស់តាមរយៈ design លឿន។
- Algorithm បច្ចេកទេសប្រើដើម្បីធ្វើ Prototype អាចមិនទាន់លឿន មិនទាន់ល្អ លៃយ៉ាងណាឲ្យតែបានធ្វើជាគម្រូដើម្បីពិភាក្សា ផ្តល់ជាតម្រូវការនៃអ្នកប្រើប្រាស់។

4) The RAD (Rapid Application Development) Model



4) The RAD (Rapid Application Development) Model

- The RAD Model is a high-speed adaptation of the linear sequential model
- Project requirements must be well understood and the project scope tightly constrained
- Developers are often able to use component-based construction techniques to build a fully functional system in a short time period

4) The RAD (Rapid Application Development) Model

RAD Model មានលក្ខណៈ

- ដំណើរការអភិវឌ្ឍន៍ SW តាមបែបបន្ថែម (Incremental software development) គឺកើនឡើងជាជំហានៗ ដែលជុំអភិវឌ្ឍន៍នីមួយៗមានរយៈពេលខ្លី(៦០ ទៅ ៩០ ថ្ងៃ)។
- Component-based construction ដោយលទ្ធភាពប្រើប្រាស់ឡើងវិញ (reusability)។
- រួមមាន Team មួយចំនួន ដែល Team នីមួយៗអនុវត្ត 1 RAD ទៅតាម Phase: Business modeling, Data modeling, Process modeling, Application Generation, Testing and turnover។

4) The RAD (Rapid Application Development) Model

1. Teams should consist of about 6 people, including both developers and full-time users of the system plus anyone else who has a stake in the requirements.

2. Developers chosen for RAD teams should be multi-talented "renaissance" people who are analysts, designers and programmers all rolled into one)

4) The RAD (Rapid Application Development) Model

* Business modeling

Information flow ត្រូវបានបង្កើតជា model ដើម្បីឆ្លើយនូវសំណួរ:

- What information drives the business process?
- What information is generated?
- Who generates it?
- Where does the information go?
- Who processes it?

4) The RAD (Rapid Application Development) Model

* Data modeling:

Data objects ចាំបាច់ដើម្បីជាជំនួយដល់កិច្ចការ business ត្រូវបានបង្កើតឡើង។ ទន្ទឹមនឹងនោះ attribute នៃ object នីមួយៗក៏ដូចជាទំនាក់ទំនងរវាង object ទាំងឡាយ ក៏ត្រូវបានកំណត់នៅក្នុងពេលនោះដែរ។

* Process modeling:

The data objects ត្រូវបានបម្លែងទៅជា information flow ចាំបាច់ ហើយអនុវត្តនូវមុខងារ business។ ទន្ទឹមនឹងនេះ Processing descriptions ក៏ត្រូវបានបង្កើតដើម្បីបន្ថែម កែតម្រូវ លុប ស្តារឡើងវិញនៃ data objectនីមួយៗ។

4) The RAD (Rapid Application Development) Model

* Application Generation:

ប្រើបច្ចេកទេសជំនាន់ទី៤ដើម្បីបង្កើត SW ពី Component ដែលមានស្រាប់ ឬបង្កើតចេញជា Component ដែលអាចប្រើប្រាស់ឡើងវិញបាននៅពេលក្រោយទៀត។ ប្រើ Tool ដោយស្វ័យប្រវត្តិ ដើម្បីបង្កើត SW។

- CASE is SW to support SW development and evolution processes
- Activity automation
  - Graphical editors for system model development
  - Data dictionary to manage design entities
  - Graphical UI builder for user interface construction
  - Debuggers to support program fault finding
  - Automated translators to generate new version of a program

4) The RAD (Rapid Application Development) Model

* Testing and Turnover:

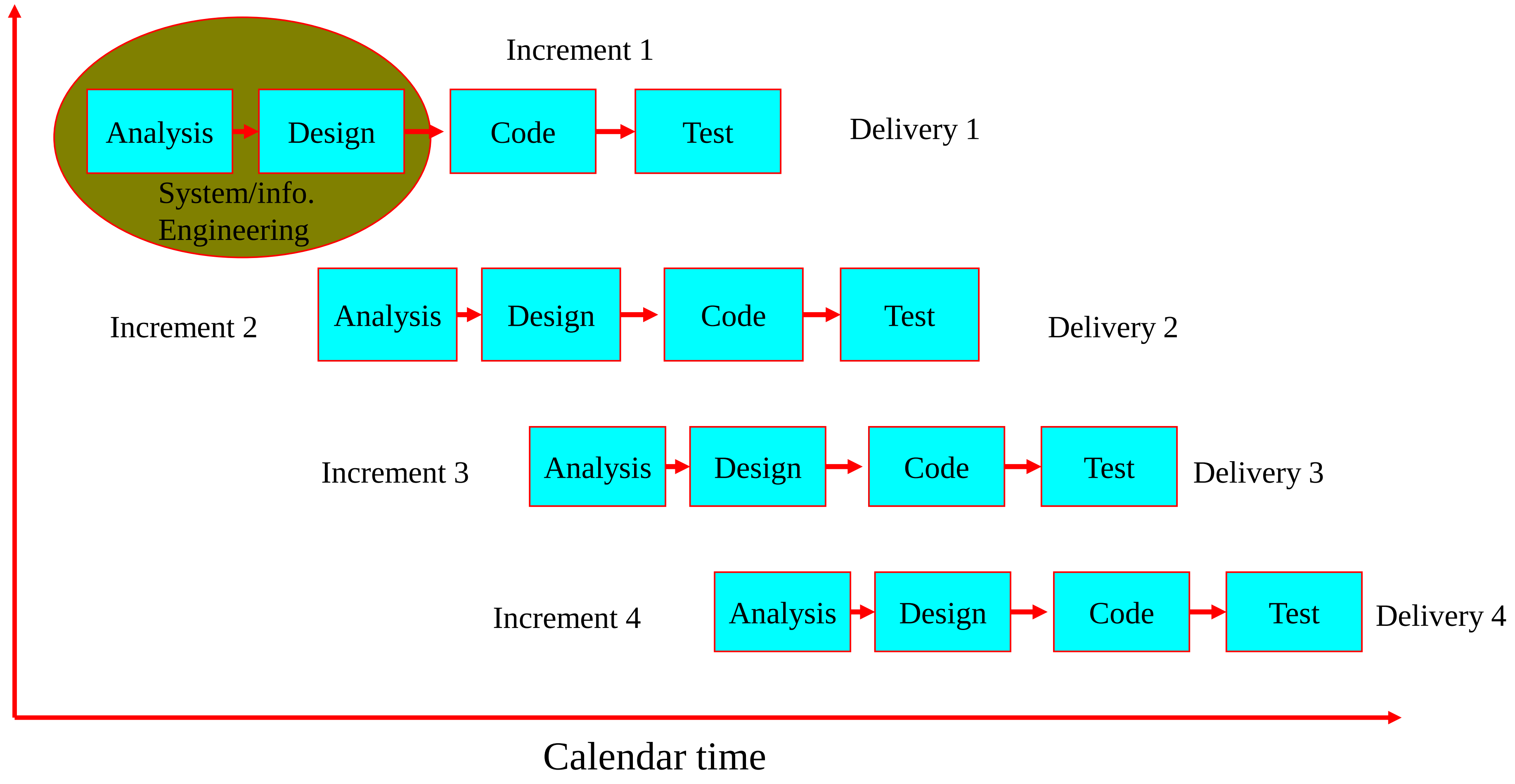
ធ្វើតេស្តសមាសភាពថ្មី និងពិនិត្យមើលគ្រប់ interface (សមាសភាពចាស់ត្រូវបានធ្វើតេស្ត និងប្រើឡើងវិញ)។

* **RAD:** Drawback ?

- ត្រូវការប្រកពធនធានមនុស្សគ្រប់គ្រាន់ដើម្បីបង្កើត Team សម្រាប់មុខងារសំខាន់ៗ។
- តម្រូវឲ្យភាគីទាំងពីរ (Developers and customers) ចុះកិច្ចសន្យាក្នុងរយៈពេលខ្លី ត្រូវតែមាន SW ឲ្យបានគ្រប់គ្រាន់។ ខ្វះការទទួលខុសត្រូវពីភាគីម្ខាង ផាយនឹងធ្វើឲ្យគម្រោងខូចការ។
- RAD ពុំមានភាពប្រសើរសម្រាប់គ្រប់ Applications ឡើយ ពិសេសចំពោះ Application ដែលមិនអាចបំបែកទៅជា Module ឬទាមទារ performance ខ្ពស់។
- បើ Risk ខាងផ្នែកបច្ចេកទេសមានកម្រិតខ្ពស់ គឺមិនត្រូវប្រើ RAD ឡើយ។

5) Evolutionary Software Process Model

a) Incremental Model



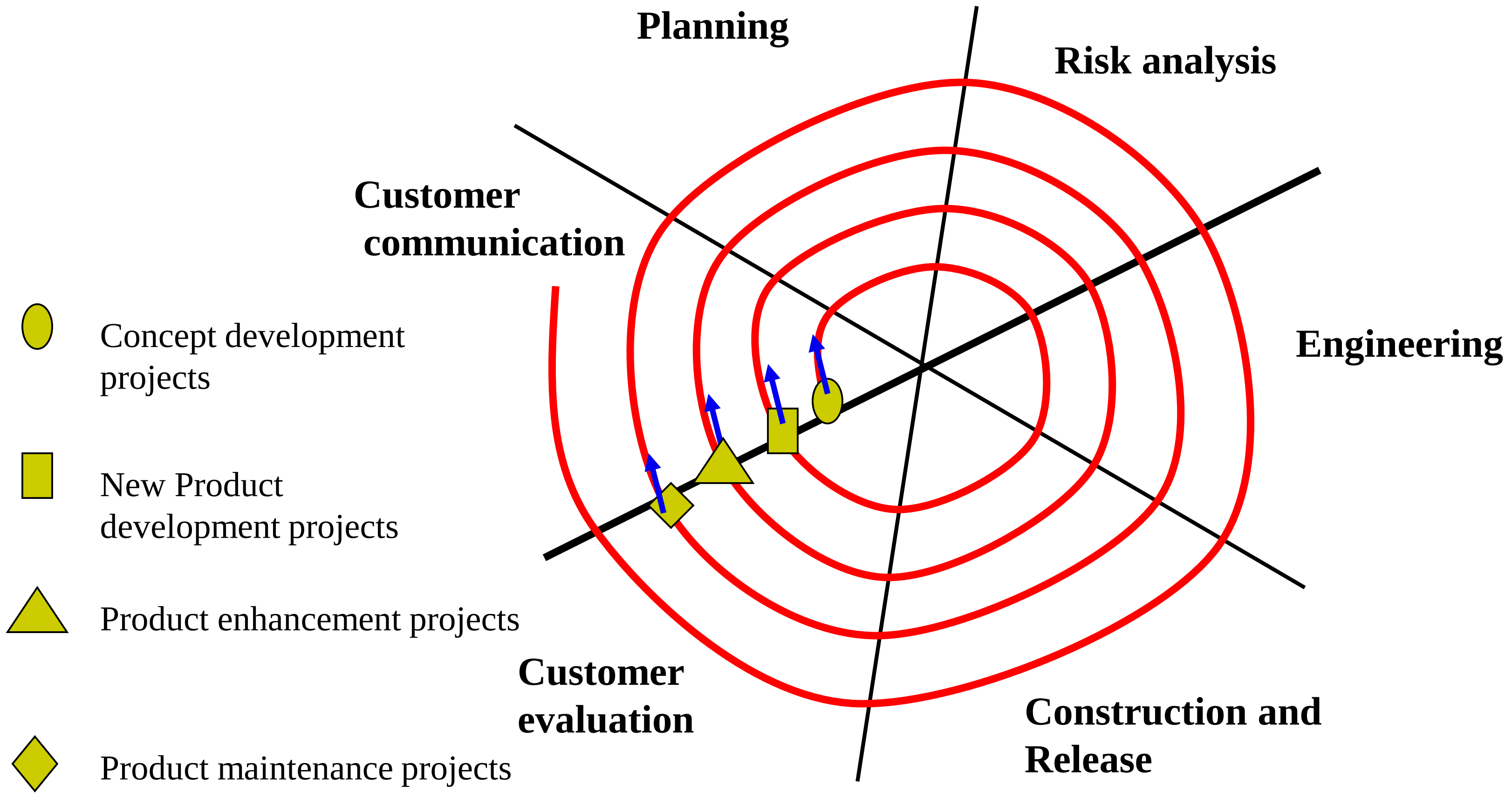
5) Evolutionary Software Process Model

b) The Spiral Model(1988)

- The spiral model may be applicable to projects where: **the projects requirements are very difficult (for large, expensive and complicated projects) and the new technologies are used**

5) Evolutionary Software Process Model

b) The Spiral Model



Planning
Risk analysis
Customer communication
Engineering
Concept development projects
New Product development projects
Product enhancement projects
Customer evaluation
Construction and Release
Product maintenance projects

5) Evolutionary Software Process Model

b) The Spiral Model

- Customer communication: គឺជាដំណាក់កាលដែល Developer និង Customer ធ្វើទំនាក់ទំនងជាមួយគ្នា ដើម្បីស្វែងយល់តម្រូវការ និងគំនិតផ្សេងៗ។
- Planning :កំណត់ឲ្យបានច្បាស់លាស់នូវប្រេកពធនធាន រយៈពេល ថវិកា និងព័ត៌មានផ្សេងៗទៀត។
- Risk analysis: វិភាគមើលទាំង Technical risk និង management risk។
- Engineering : Build one or more representations of the application
- Construction and release: Construct, test, install, and provide user support (Documentation and training).
- Customer evaluation: ទទួលយកប្រតិកម្មពីអតិថិជនត្រឡប់មកវិញនូវ SW representations ក្នុងដំណាក់កាល Engineering និង Installation។

5) Evolutionary Software Process Model

b) The Spiral Model

- **Advantages**
  - High amount of risk analysis
  - Good for large and mission-critical projects.
  - Software is produced early in the software life cycle.

5) Evolutionary Software Process Model

b) The Spiral Model

- **Disadvantages**
  - Can be a costly model to use.
  - Risk analysis requires highly specific expertise.
  - Project's success is highly dependent on the risk analysis phase.
  - Doesn't work well for smaller projects.

5) Evolutionary Software Process Model

c) The Winwin Spiral Model(1994)

- *The WinWin spiral model*, which extends the spiral software development model by adding Theory W activities to the front of each cycle.
- *WinWin*, a groupware tool that makes it easier for distributed stakeholders to negotiate mutually satisfactory (win-win) system specifications.
- The study showed that the WinWin spiral model is a good match for multimedia applications and is likely to be useful for other applications with similar characteristics.

5) Evolutionary Software Process Model

c) <u>The Winwin Spiral Model(1994)</u>

- ដើម្បីសម្រុះសម្រួលឬចរចាររវាង Developer និង Customer ដែលភាគីទាំងពីរ "ឈ្នះ" ដូចគ្នា។
- អតិថិជនទទួលបាន System or product ឆ្លើយតបទៅនឹងតម្រូវការជាមូលដ្ឋាន។
- Developer ទទួលបានថវិកាសមរម្យ តាមរយៈពេលកំណត់សមហេតុផល។
- **សកម្មភាពសំខាន់ៗក្នុងការកំណត់ឲ្យបានច្បាស់នៃsystem:**
  - Identification of the system or subsystem's key "stakeholders"
  - កំណត់លក្ខខ័ណ្ឌឈ្នះ "win condition" នៃ stakeholder
  - សម្រុះសម្រួលលក្ខខ័ណ្ឌឈ្នះនៃភាគីពាក់ព័ន្ធ ដើម្បីឲ្យពួកគេទទួលបាន win-win condition មួយ

5) Evolutionary Software Process Model

c) The Winwin Spiral Model(1994)

* **Typical Cycle of the WinWin Spiral Model**

- Identify the system or subsystem's key stakeholders (1).
- Identify the stakeholders' win conditions for the system or subsystem (2).
- Negotiate win-win reconciliations of the stakeholders' win conditions (3a).
- Elaborate the system or subsystem's product and process objectives, constraints, and alternatives (3b).
- Evaluate the alternatives with respect to the objectives and constraints. Identify and resolve major sources of product and process risk (4).
- Elaborate the definition of the product and process (5).
- Plan the next cycle, and update the life-cycle plan, including partition of the system into subsystems to be addressed in parallel cycles. This can include a plan to terminate the project if it is too risky or infeasible. Secure the management's commitment to proceed as planned (6, 7).

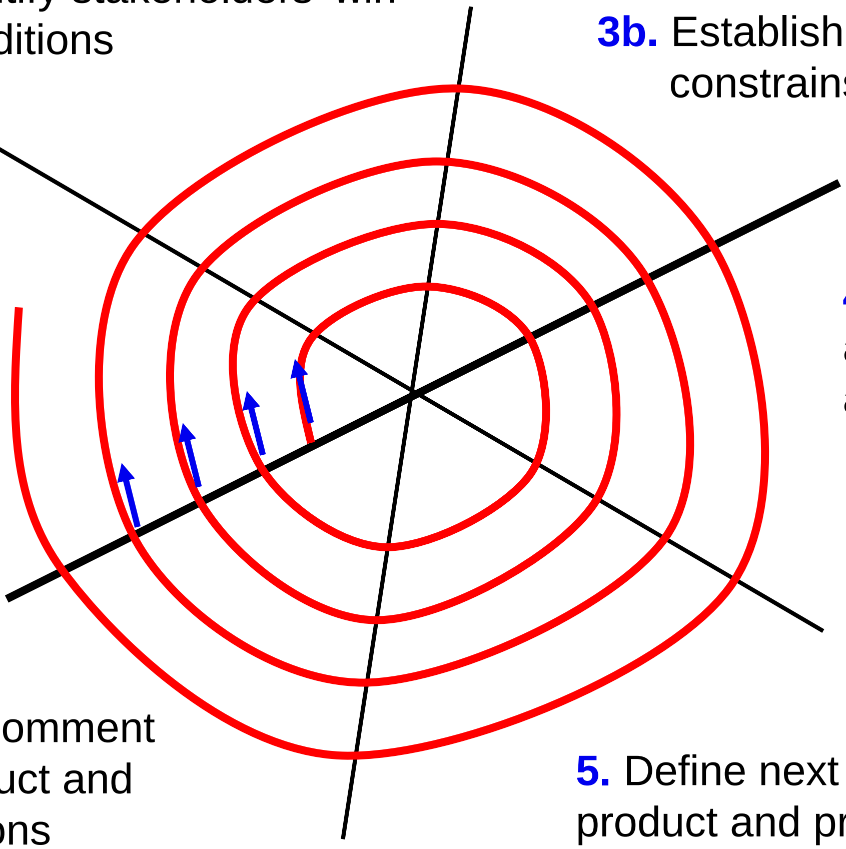
5) Evolutionary Software Process Model

c) The Winwin Spiral Model(1994)

2. Identify stakeholders' win conditions

3a. Reconcile win conditions
3b. Establish next-level objectives, constrains and alternatives

1. Identify next-level stakeholders

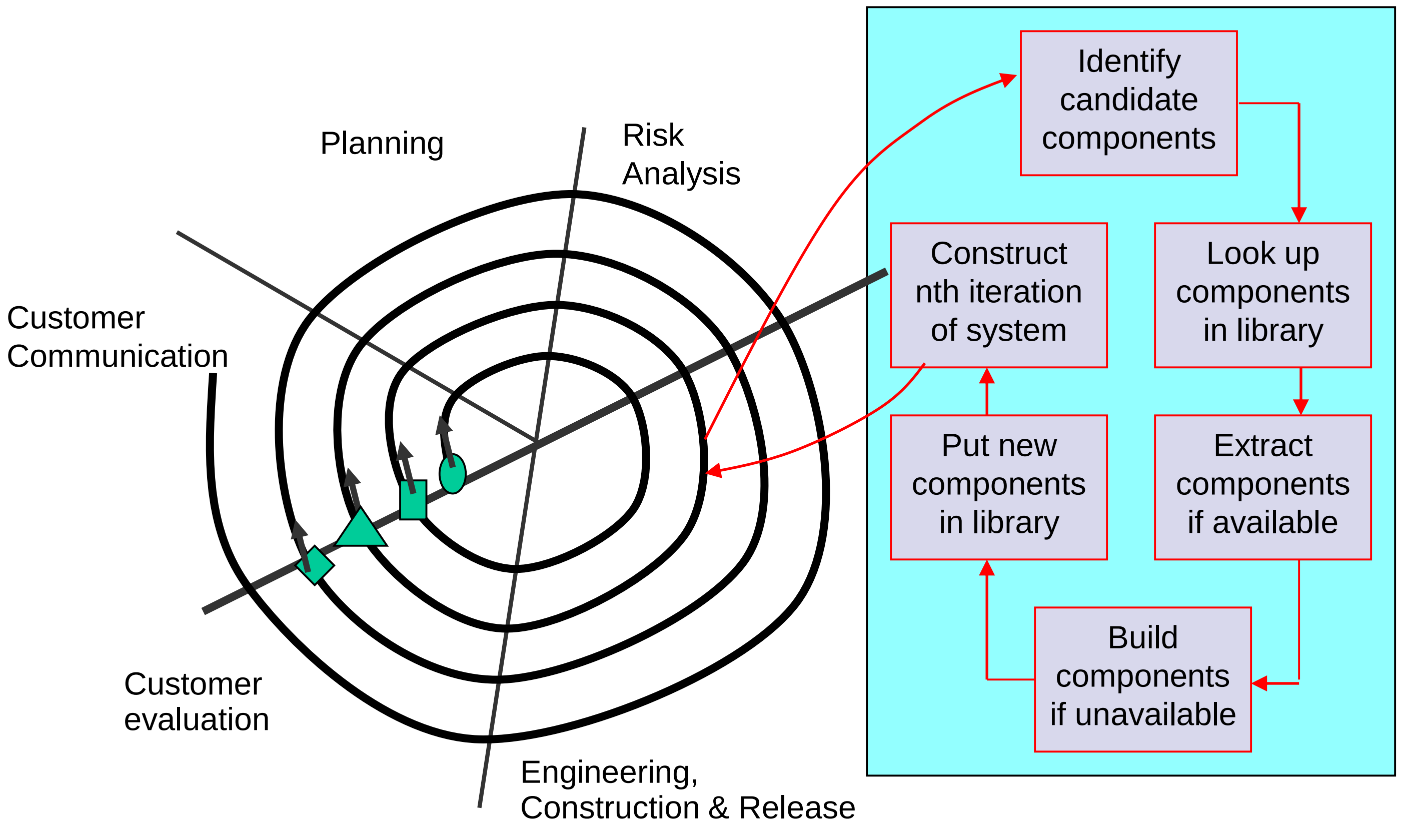


4. Evaluate process and product alternatives and resolve risks

7. Review and comment
6. Validate product and process definitions

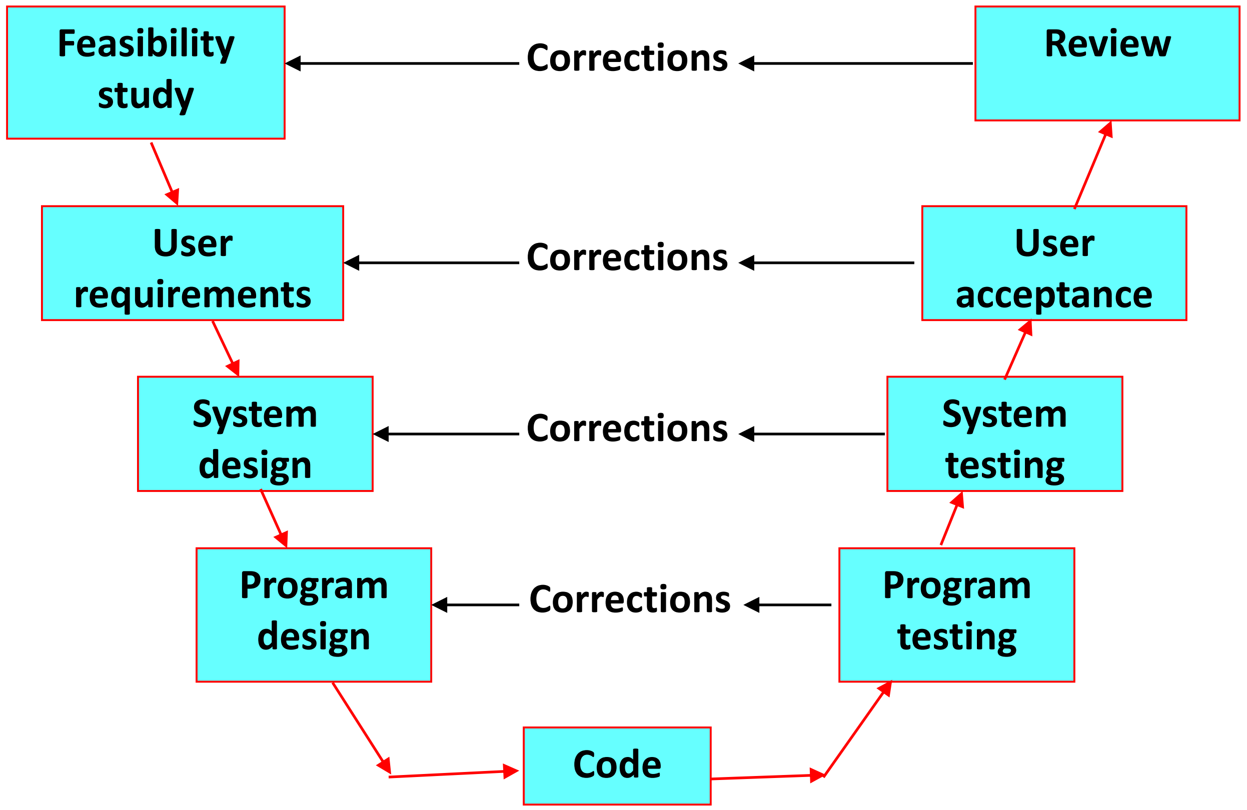
5. Define next level of product and process, including partitions

6) The Component-Based Development Model


Planning
Risk Analysis
Customer Communication
Customer evaluation
Engineering, Construction & Release
Identify candidate components
Look up components in library
Extract components if available
Build components if unavailable
Put new components in library
Construct nth iteration of system

7) The V-process model

- Software requirements clearly defined and known
- Software development technologies and tools is well known



2.3. How to Select the Right SDLC

- **STEP 1:** Learn the about SDLC Models
- **STEP 2:** Assess the needs of Stakeholders
- **STEP 3:** Define the criteria
  - Is the SDLC suitable for the size of our team and their skills?
  - Is the SDLC suitable for the selected technology we use for implementing the solution?
  - Is the SDLC suitable for client and stakeholders concerns and priorities?
  - Is the SDLC suitable for the geographical situation (distributed team)?
  - Is the SDLC suitable for the size and complexity of our software ?

2.3. How to Select the Right SDLC

  - Is the SDLC suitable for the type of projects we do?
  - Is the SDLC suitable for our software engineering capability?
  - Is the SDLC suitable for the project risk and quality insurance?
- **STEP 4: Decide**
- **STEP 5: Optimize**