Lecture 5

Software Engineering
Guide Process Development

Thanit Keatkaew

Computer Engineering , RMUTL

หัวข้อเรียน

1. กระบวนการผลิตซอฟต์แวร์ก่อนหน้าประสบปัญหาปัญหาอะไร
2. Software Engineering คืออะไร และ องค์ประกอบ
3. System Engineering คืออะไร
4. ทบทวน Software Process
5. The conceptual layout of the Guide to the Software Engineering Book of Knowledge (SWEBOK)

1.กระบวนการผลิตซอฟต์แวร์ก่อนหน้าประสบปัญหาปัญหาอะไร

- ✓ การส่งมอบโปรแกรมหรือซอฟต์แวร์ที่ล้าช้ากว่ากำหนดไว้ตอนตนโครงการ
- ✓ เมื่อพัฒนาซอฟต์แวร์เสร็จ ผลลัพธ์ที่ได้ไม่เป็นไปตามที่ตกลงกันไว้ระหว่างผู้จ้างและผู้รับจ้างพัฒนาซอฟต์แวร์
- ✓ มีการใช้งบประมาณเกินหว่าที่กำหนดไว้
- ✓ ระบบรักษาความปลอดภัยของซอฟต์แวร์ ยังไม่มีประสิทธิภาพเพียงพอ
- ✓ ซอฟต์แวร์บางชนิดอาจสร้างความเสียหายให้กับชีวิตและทรัพย์สิน

That why we need ??

Software Engineering

Software Process

2.Software Engineering คืออะไร และ องค์ประกอบ

เป็นศาสตร์เกี่ยวกับวิศวกรรมด้านซอฟต์แวร์ มีเนื้อหาเกี่ยวข้องกับการใช้กระบวนการทางวิศวกรรมในการดูแลการผลิต

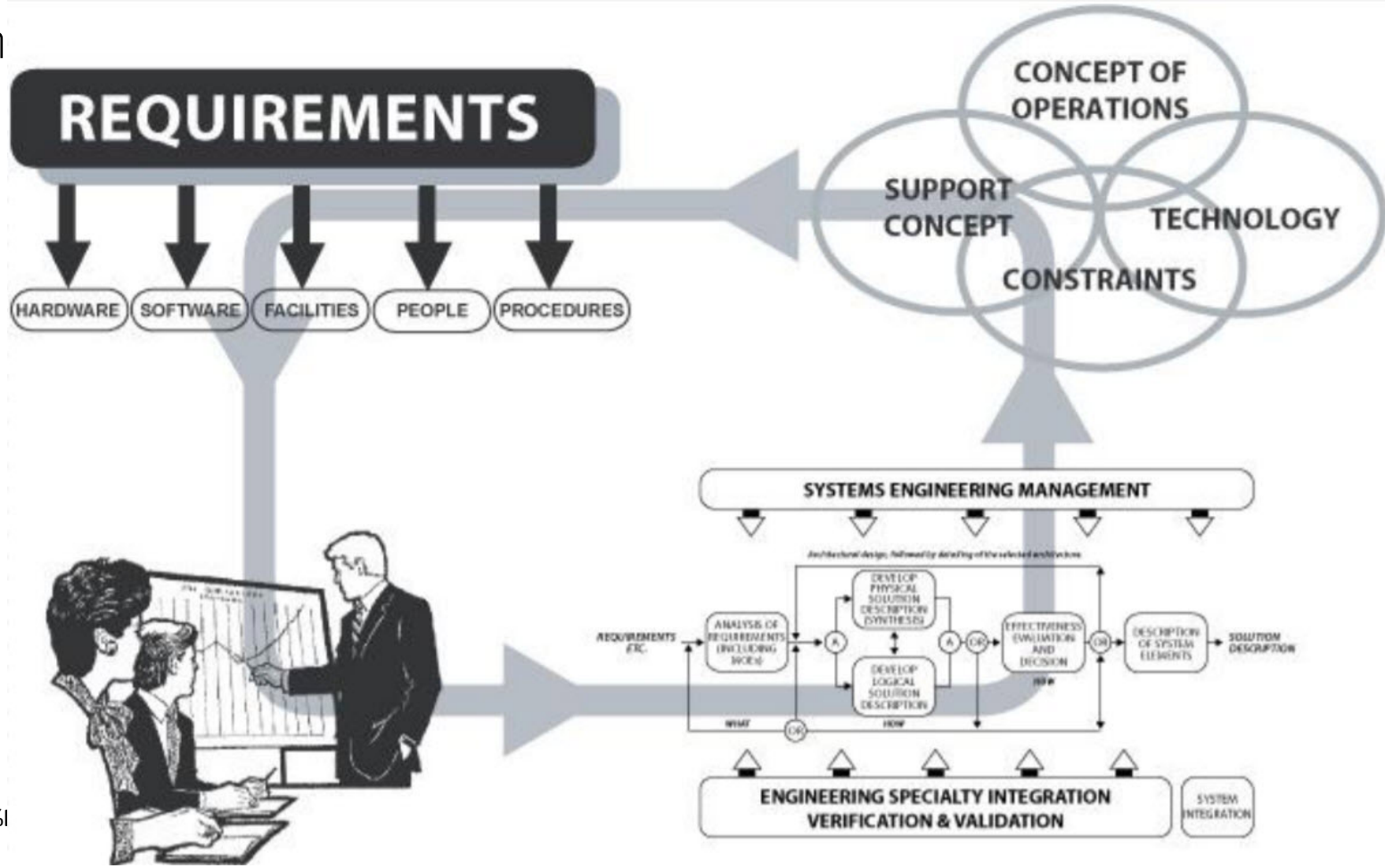
- การเริ่มเก็บความต้องการ
- การตั้งเป้าหมายของระบบ
- การออกแบบ
- กระบวนการพัฒนา
- การตรวจสอบ
- การประเมินผล
- การติดตามโครงการ
- การประเมินต้นทุน
- การรักษาความปลอดภัย
- ไปจนถึงการคิดราคาซอฟต์แวร์เป็นต้น

นิยามของวิศวกรรมซอฟต์แวร์ ในแง่มุมเวลานำไปใช้งาน

- เป็นการศึกษาแขนงหนึ่งในเรื่องของหลักการและกรรมวิธี สำหรับการพัฒนาและบำรุงรักษาระบบซอฟต์แวร์
- วิศวกรรมซอฟต์แวร์เป็นการประยุกต์ความรู้ ทางวิทยาศาสตร์ในการออกแบบและสร้างโปรแกรมคอมพิวเตอร์ และการทำเอกสารที่เกี่ยวข้องเพื่อพัฒนาดำเนินการบำรุงรักษา โปรแกรมเหล่านั้น
- วิศวกรรมซอฟต์แวร์ จะเกี่ยวกับการสร้างตามหลักการทางวิศวกรรมและวิธีการเพื่อให้ได้ซอฟต์แวร์ในลักษณะที่ประหยัดที่สุด หมายถึงเสียค่าใช้จ่ายน้อยที่สุด และสามารถทำงานได้จริง
- วิศวกรรมซอฟต์แวร์ เกี่ยวข้องกับวิธีการการพัฒนาซอฟต์แวร์ โดยมุ่งเน้นไปถึง วิธีการสร้างโดยรวมเทคนิค วิธีการหลากหลายวิธีการเข้าไปในการพัฒนาซอฟต์แวร์ ใหม่ขึ้นมา

3.System Engineering คืออะไร

หมายถึงกระบวนการศึกษาและวิเคราะห์ของระบบที่มีความซับซ้อนเพื่อสนับสนุนการทำงานในส่วนของวิศวกรรมซอฟต์แวร์ หรือ กิจกรรมวิศวกรรมระบบ จะถูกดำเนินการไปพร้อม ๆ กับกิจกรรมของวิศวกรรมซอฟต์แวร์ เช่น ในระยะการวางแผนวิศวกรรมซอฟต์แวร์จะต้องวิเคราะห์กระบวนทางธุรกิจของระบบงาน (เป็นกิจกรรมของวิศวกรรมระบบ) ควบคู่ไปกับการวิเคราะห์ความต้องการของลูกค้า



https://www.mindphp.com/%E0%B8%9A%E0%B8%97%E0%B8%84%E0%B8%A7%E0%I

วิศวกรรมระบบ (System Engineering)

วิศวกรรมระบบ ไม่ได้มุ่งเน้นเรื่องของ software อย่างเดียว แต่จะให้ความสำคัญกับส่วนประกอบอื่น ๆ ด้วย ได้แก่

1. กำหนดวัตถุประสงค์ของระบบ
2. กำหนดขอบเขตระบบ
3. แบ่งระบบออกเป็นส่วน ๆ ตามฟังก์ชันการทำงานหรือคุณสมบัติของระบบ
4. พิจารณาความสัมพันธ์ของส่วนประกอบต่าง ๆ ที่เกี่ยวข้องทั้งหมด
5. กำหนดความสัมพันธ์ของปัจจัยนำเข้า ประมวลผล และผลลัพธ์
6. ปัจจัยที่ส่วนเกี่ยวข้องในระบบ ไม่ว่าจะเป็น Hardware , Software , Database หรือแม้แต่ผลิตภัณฑ์ซอฟต์แวร์อื่น ๆ เป็นต้น
7. กำหนดความต้องการในส่วนของการดำเนินการและ Functional ทั้งระบบ
8. สร้างแบบจำลองระบบ เพื่อใช้วิเคราะห์และพัฒนาให้สอดคล้องกับแบบจำลองซอฟต์แวร์ที่สร้างขึ้น
9. นำเสนอและแลกเปลี่ยนข้อคิดเห็นกับผู้ที่เกี่ยวข้องกับระบบ ไม่ว่าจะเป็นผู้ใช้ระบบ เจ้าของระบบ หรือแม้แต่ผู้ที่เกี่ยวข้องกับผลประโยชน์ที่มีต่อระบบ

4.ทบทวน Software Processes

Software is the set of instructions in the form of programs to govern the computer system and to process the hardware components.

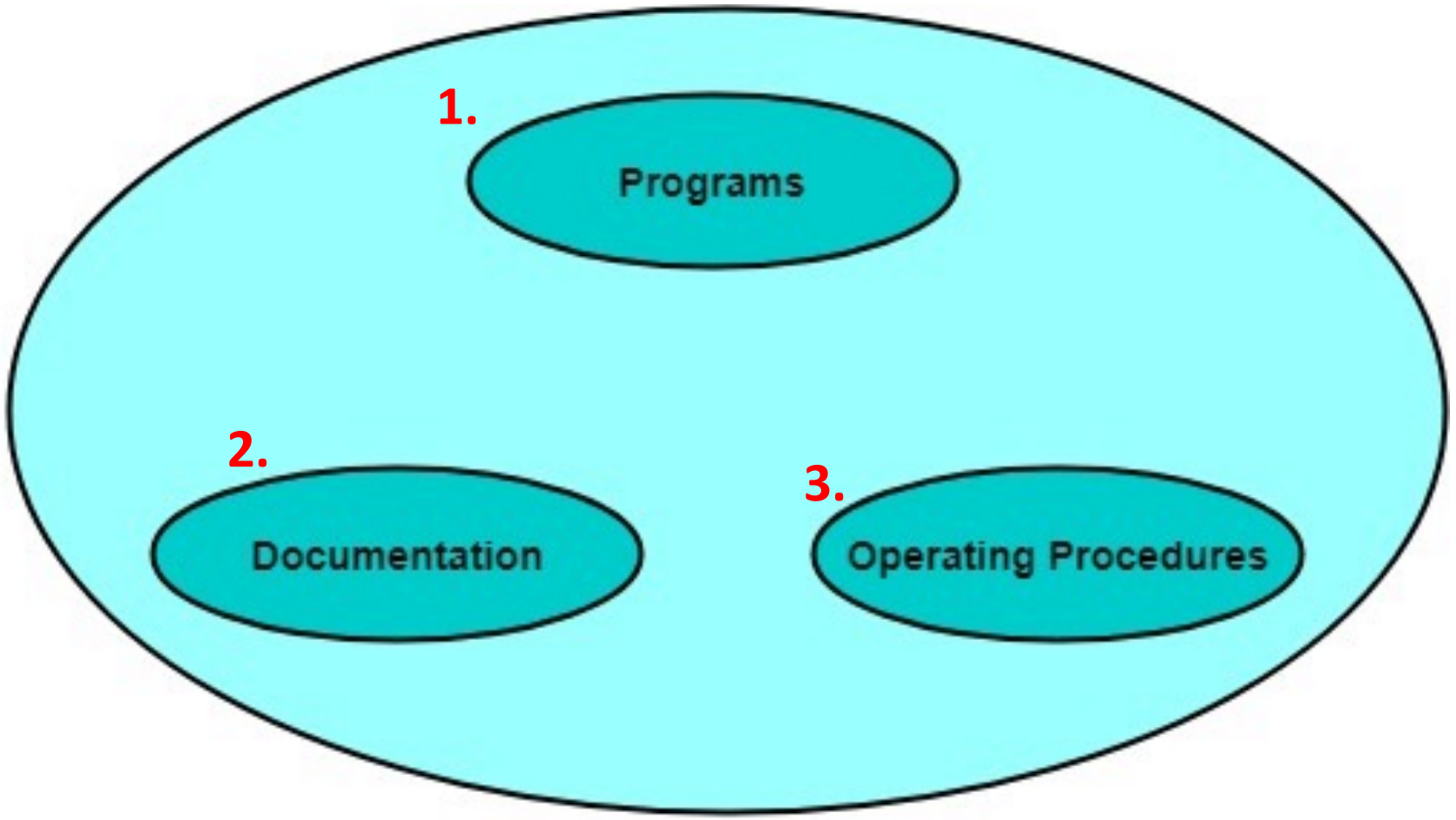
- **To produce a software product** the set of activities is used.
  This set is called **a Software Process**.

- **Software Development :** In this process, designing, programming, documenting, testing, and bug fixing is done.

- **Components of Software :**
  There are three components of the software:

  1.Program, 2. Documentation, and 3. Operating Procedures.

Program vs. Software

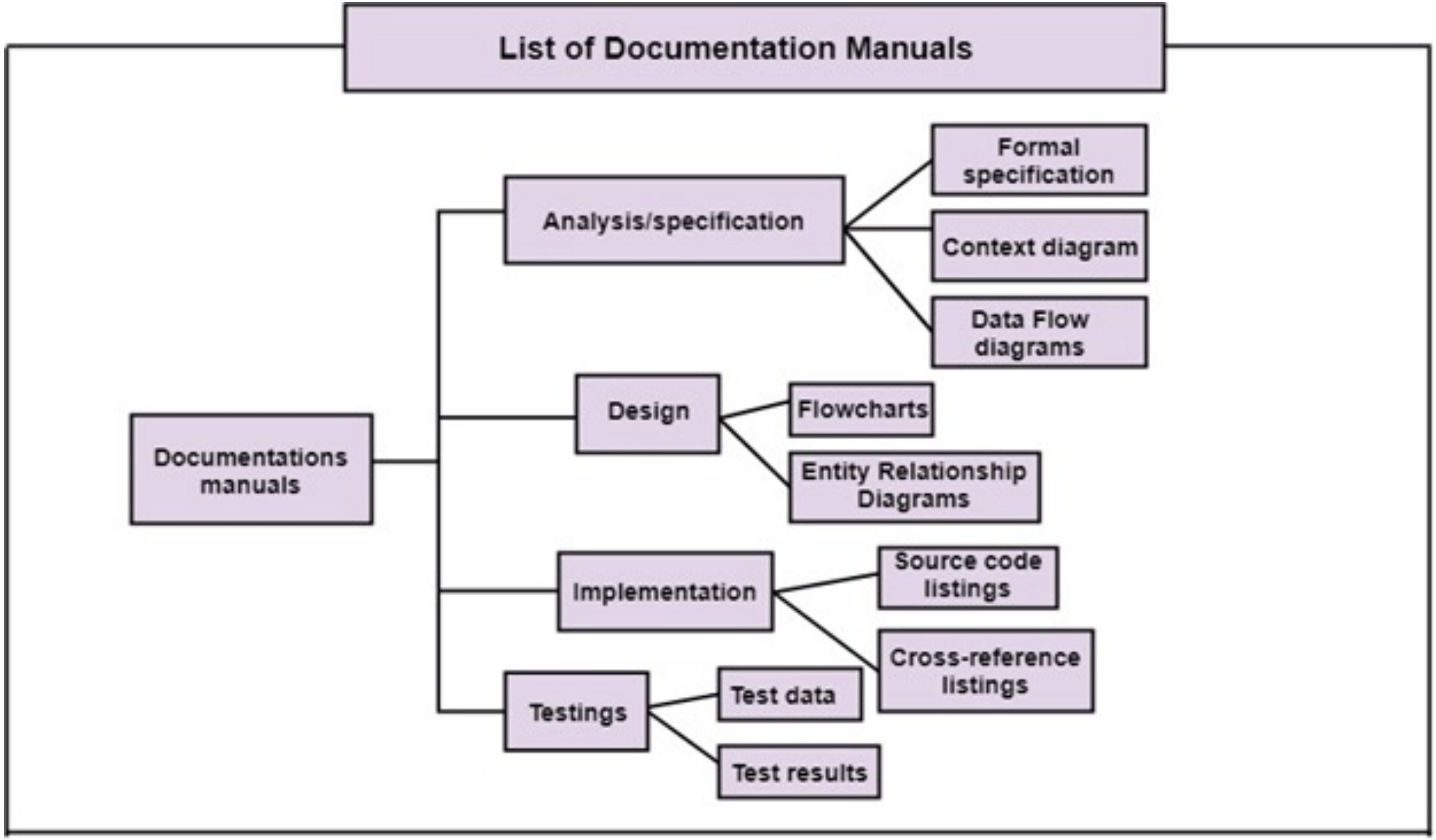


Software= Program + Documentation + Operating Procedures

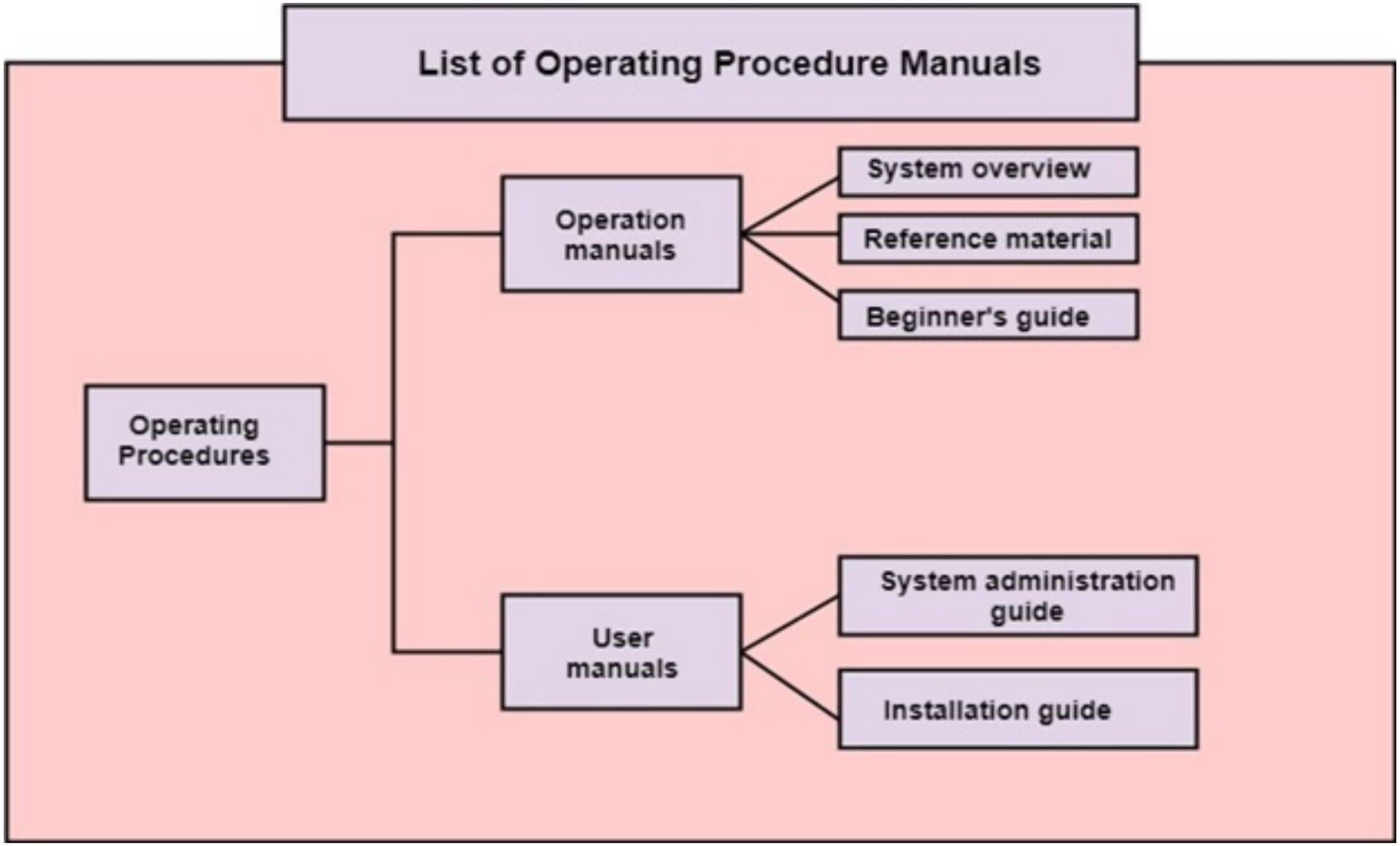
Fig:Components of Software

[Software Processes - javatpoint]

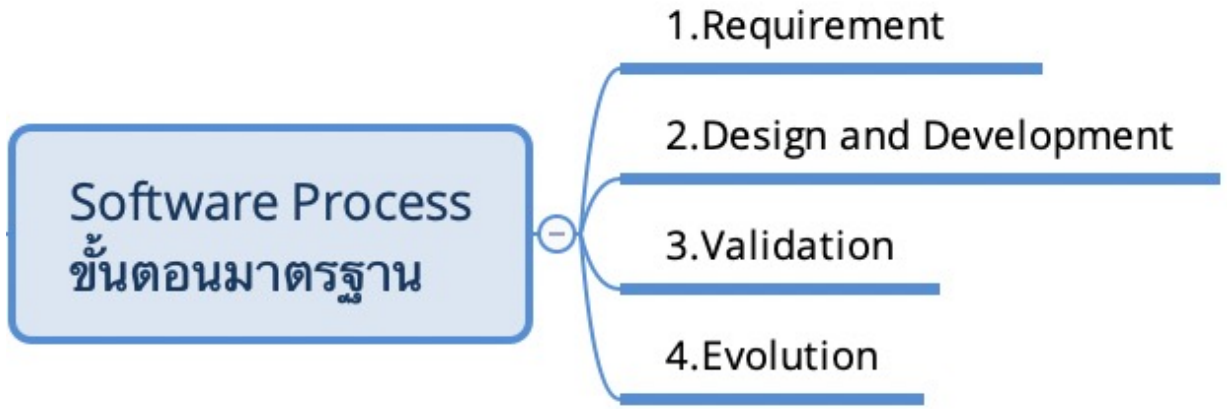
Documentation



Operating Procedures:



4 กิจกรรมหลักของ software processes



1. Software Specifications –

Detailed description of a software system to be developed with its functional and non-functional requirements.

2. Software Development –

Designing, programming, documenting, testing, and bug fixing is done.

3. Software Validation –

Evaluation software product is done to ensure that the software meets the business requirements as well as the end users needs.

4. Software Evolution –

It is a process of developing software initially, then timely updating it for various reasons.

แต่ละกิจกรรมหลักจะประกอบไปด้วย

- **Products**: The outcomes of the an activity.
- **Roles**: The responsibilities of the people involved in the process.
- **Pre and post conditions**: The conditions that must be true before and after an activity.

Software Process Models. - ต้นแบบของการทำ S/W ตาม Software Process

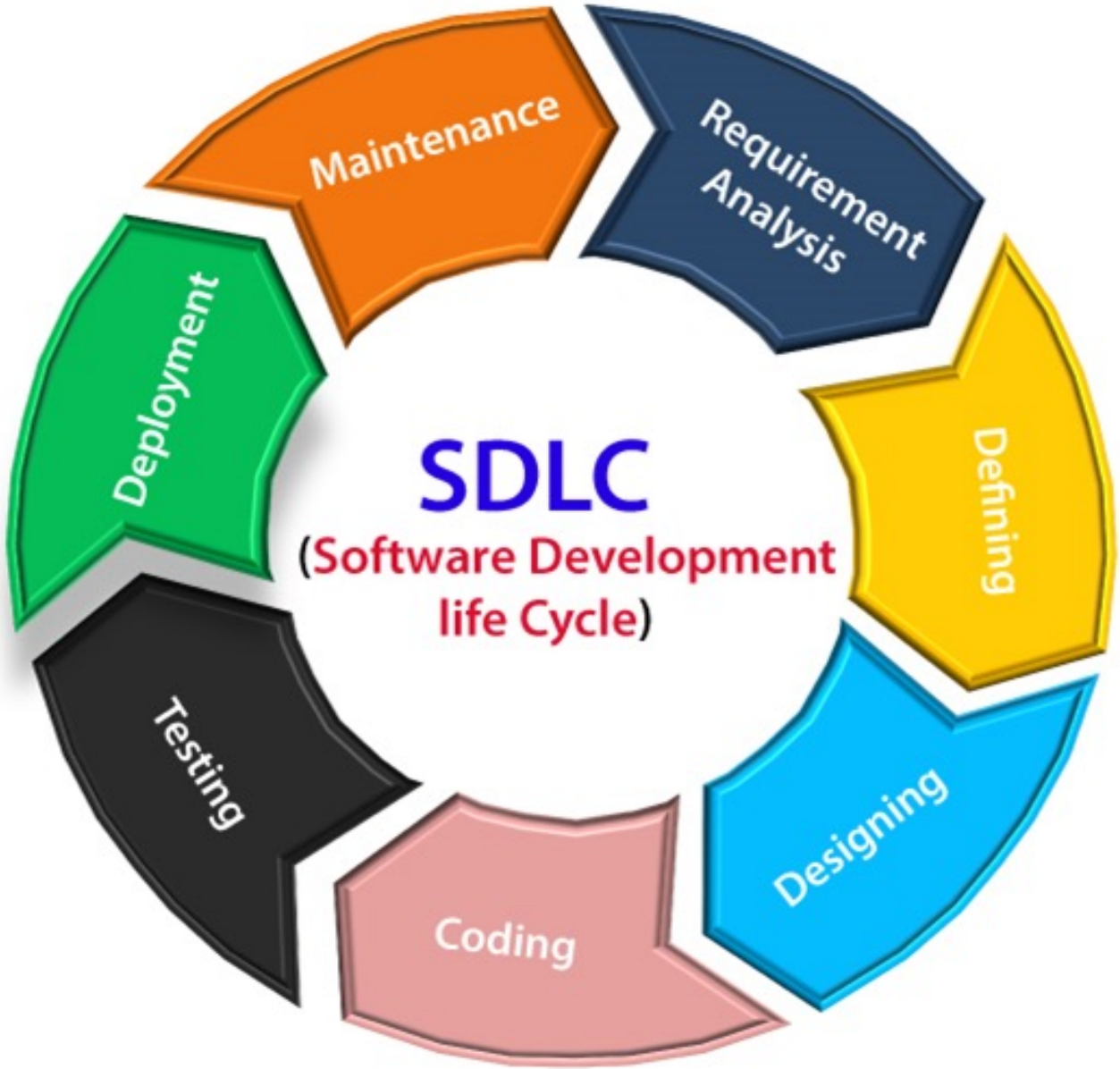
Some methodologies are sometimes known as

***Software Development Life Cycle** (SDLC) methodologies*

though this term could also be used more generally to refer to any methodology.

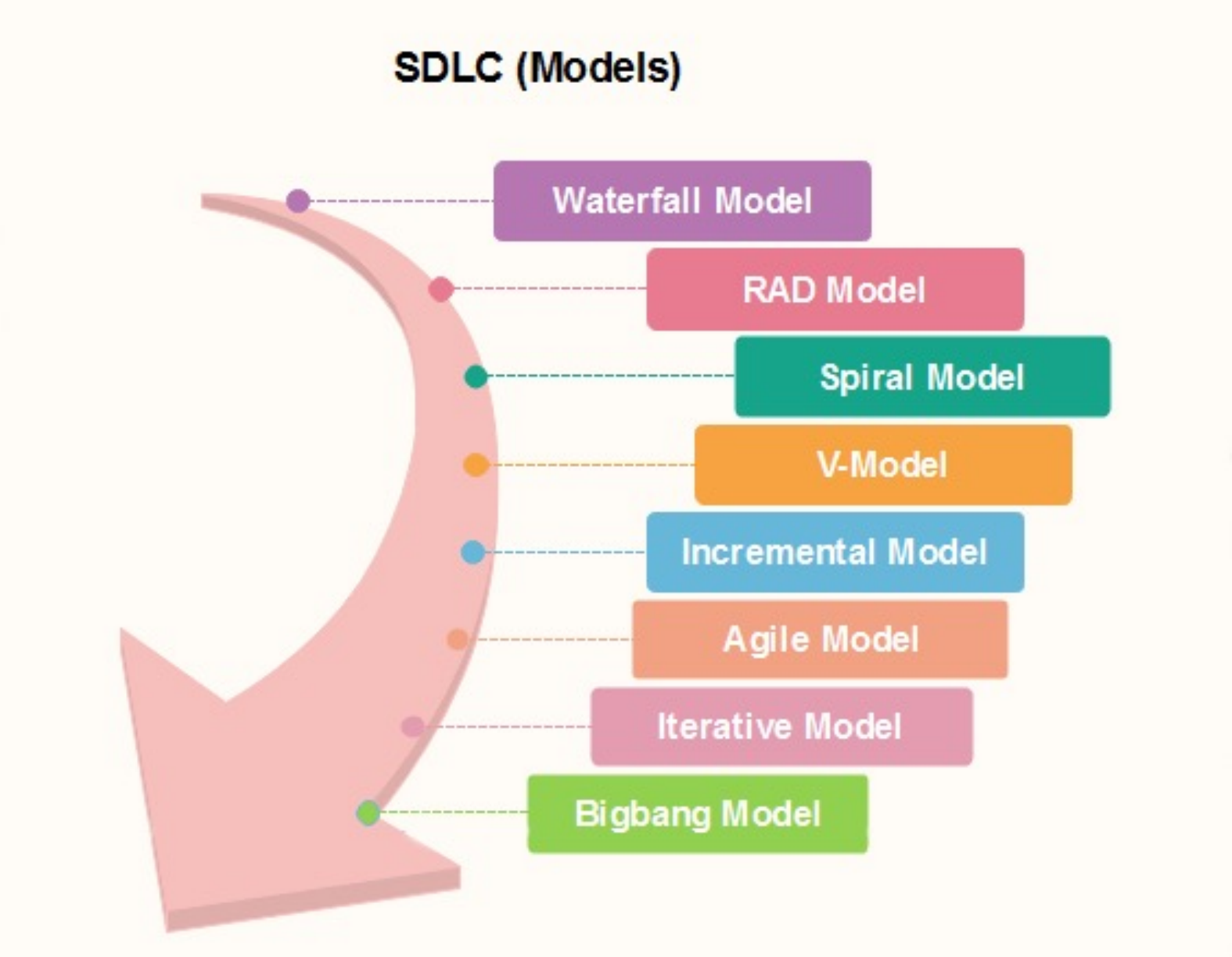
Software Development Life Cycle (SDLC)

SDLC Cycle represents the process of developing software.

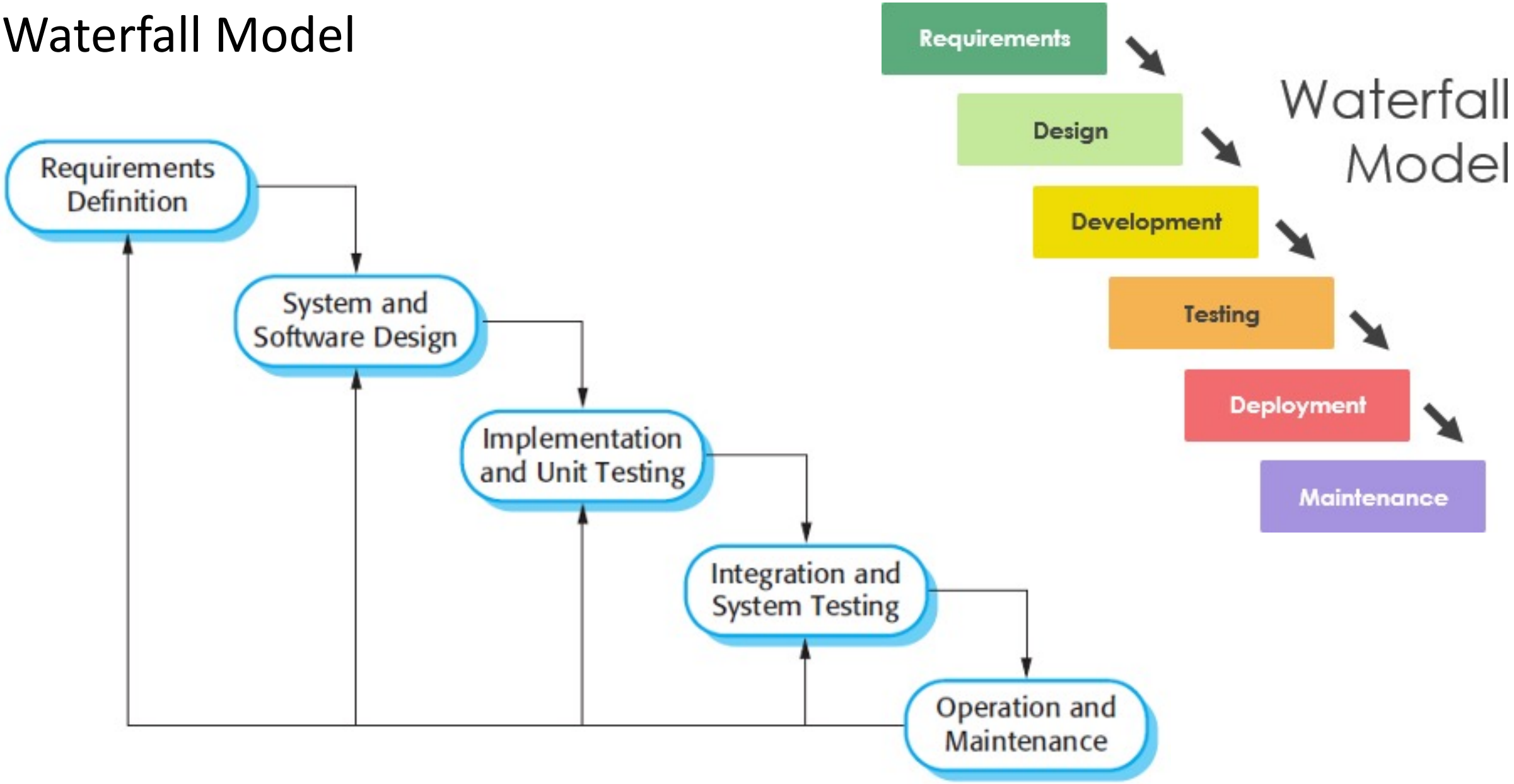


[SDLC Models - javatpoint]

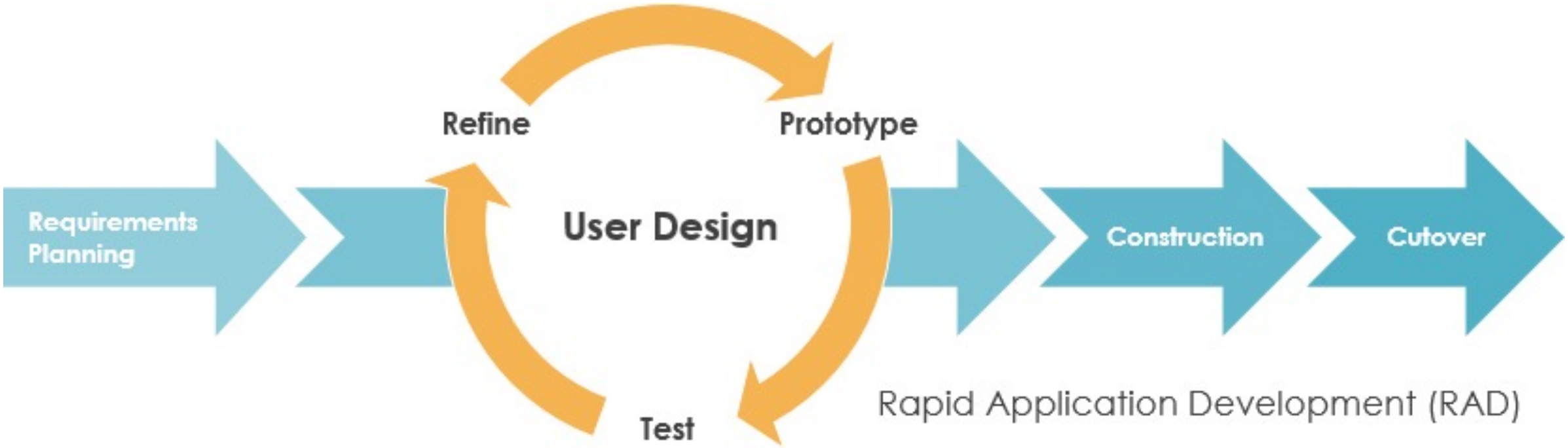
Software Process Models. - ต้นแบบของการทำ S/W ตาม Software Process หรือ SDLC Models



Waterfall Model

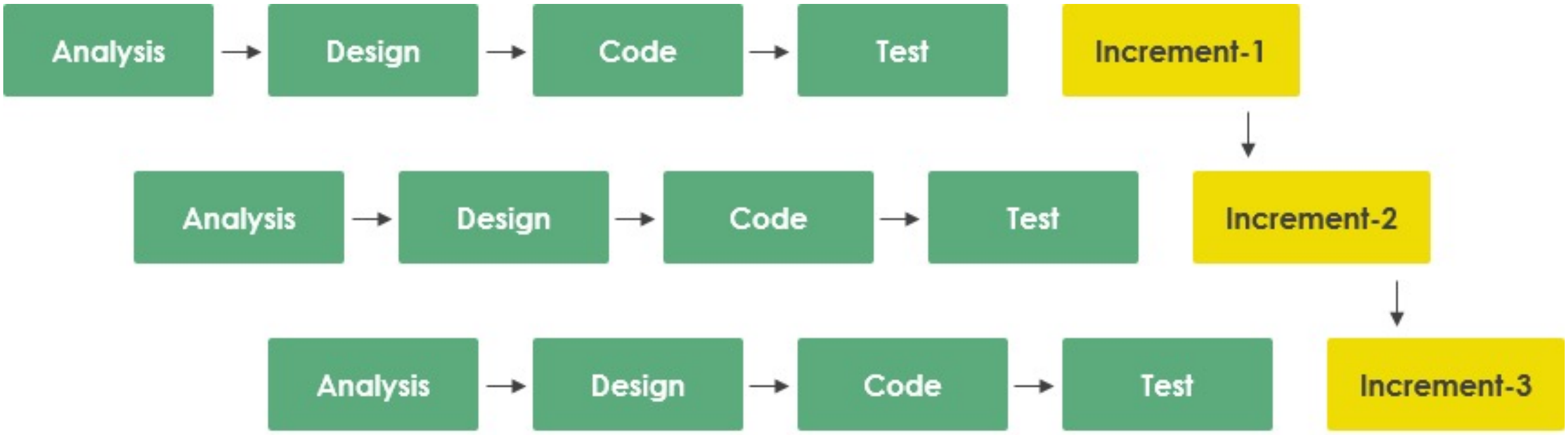


RAD model



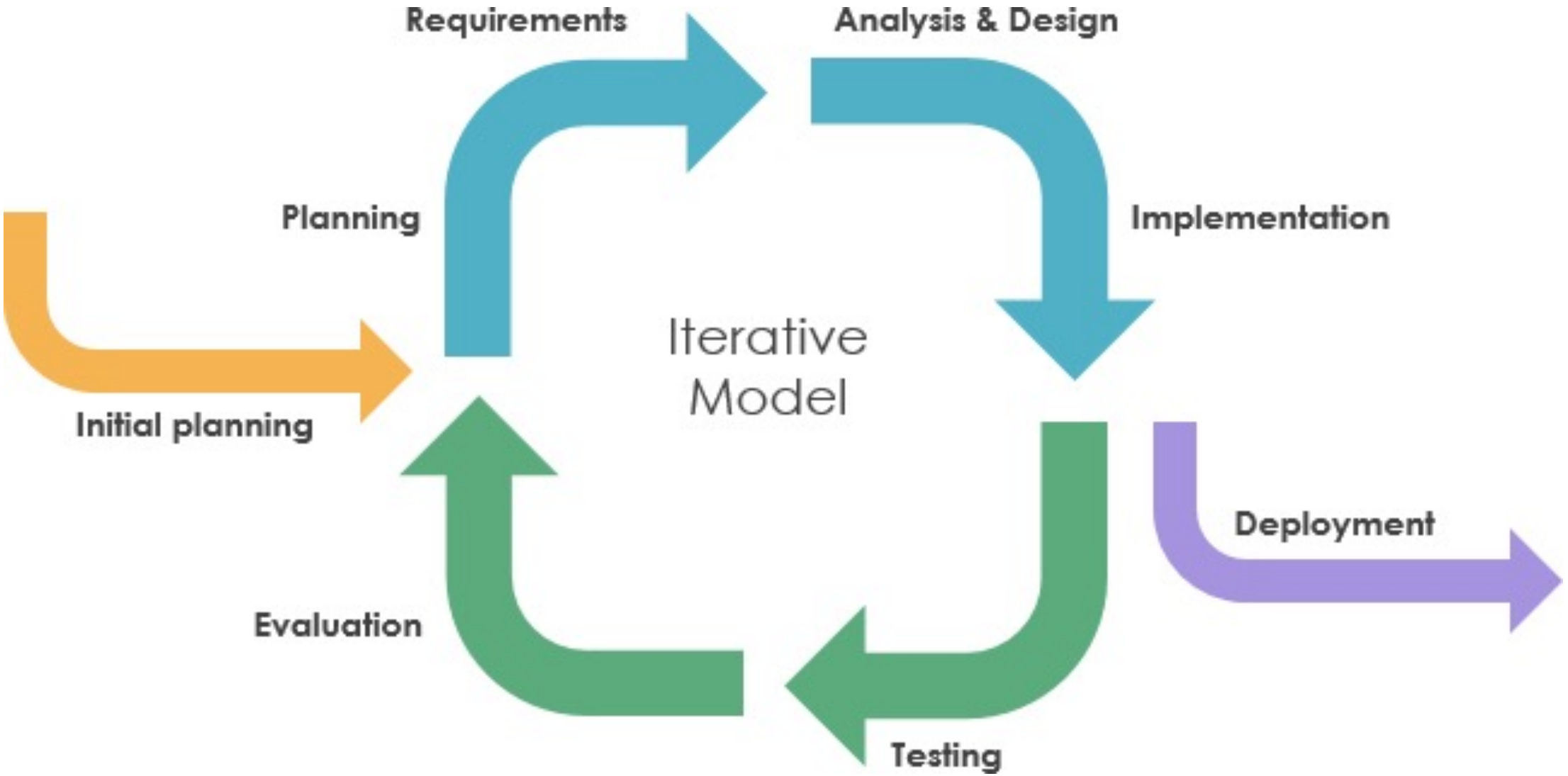
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover

Incremental model

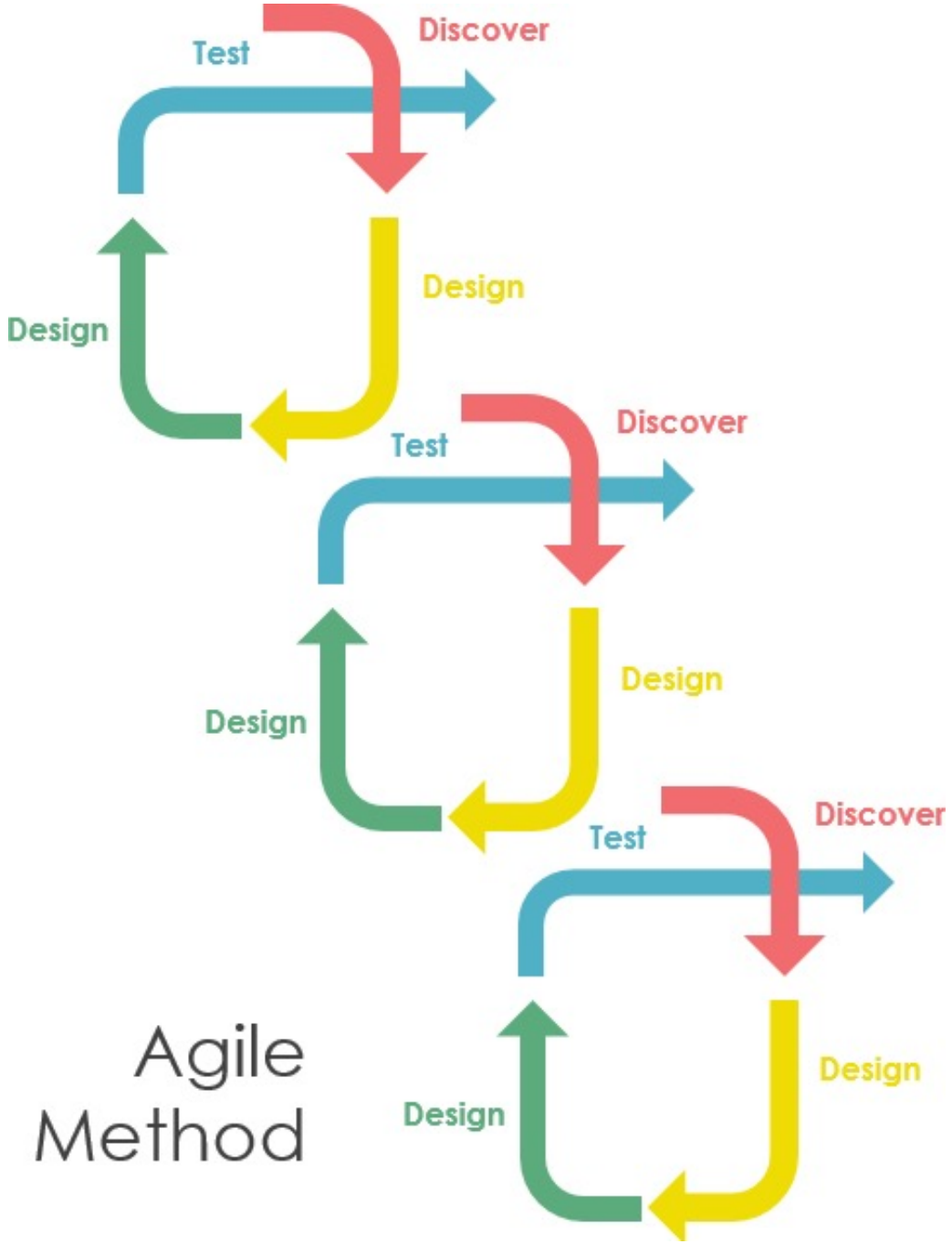


Incremental Model

Iterative Model



Agile model


Test
Discover
Design
Design
Test
Discover
Design
Design
Test
Discover
Design
Design
Agile
Method

5. The conceptual layout of the Guide to the Software Engineering Book of Knowledge (SWEBOK)

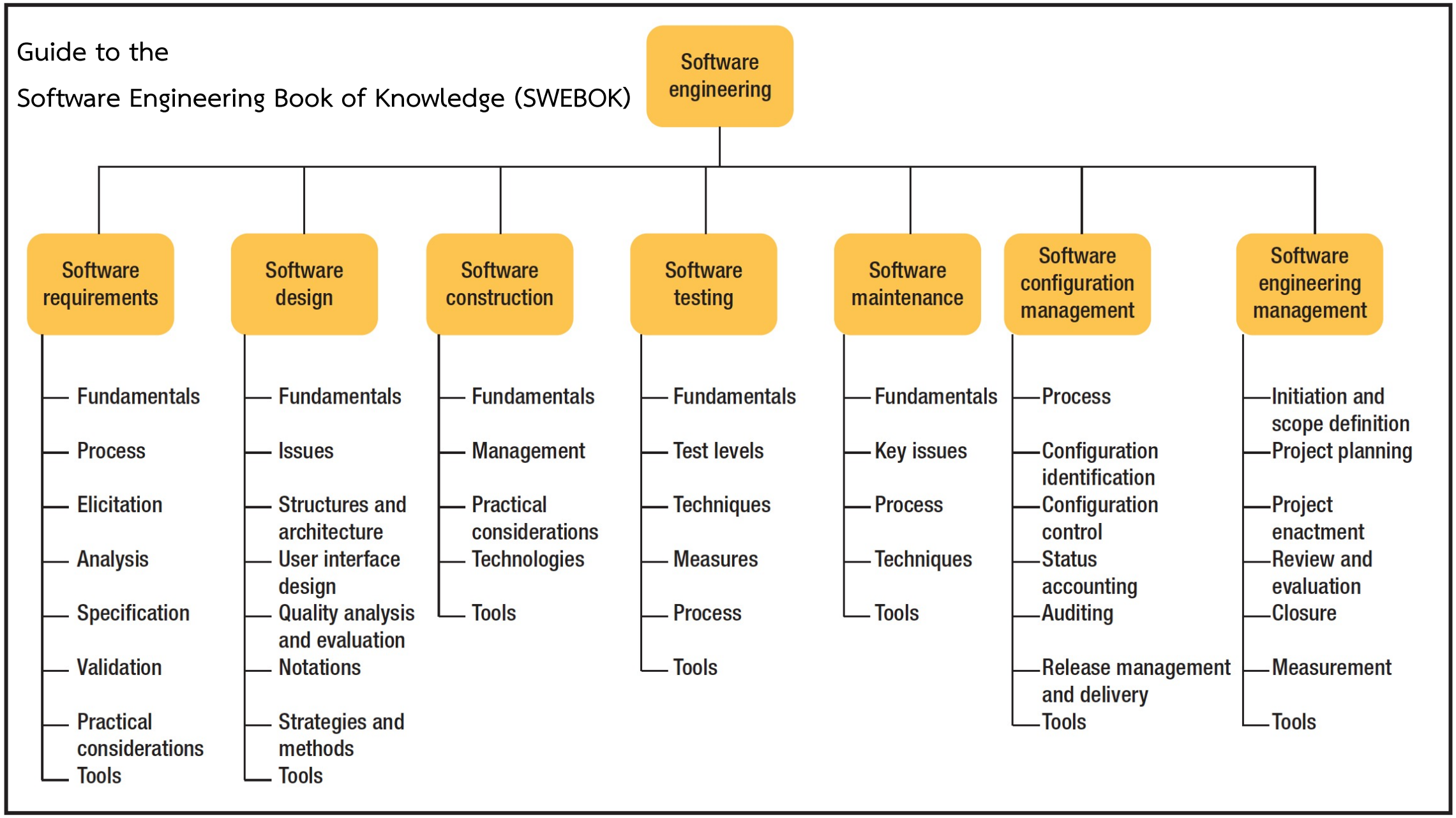


FIGURE 1. Part of the conceptual layout of the *Guide to the Software Engineering Body of Knowledge* (*SWEBOK Guide*). The boxes show major topics, with subtopics listed in the descending structures.

https://twitter.com/ieeesoftware/status/695502544324403200

Principles that Guide Process

Principle #1. Be agile

Principle #2. Focus on quality at every step

Principle #3. Be ready to adapt

Principle #4. Build an effective team

Principle #5. Establish mechanisms for communication and coordination

Principle #6. Manage change

Principle #7. Assess risk

Principle #8. Create work products that provide value for others

Modeling Principles

In SE work, two classes of models can be created

1.Requirements models (also called analysis models)

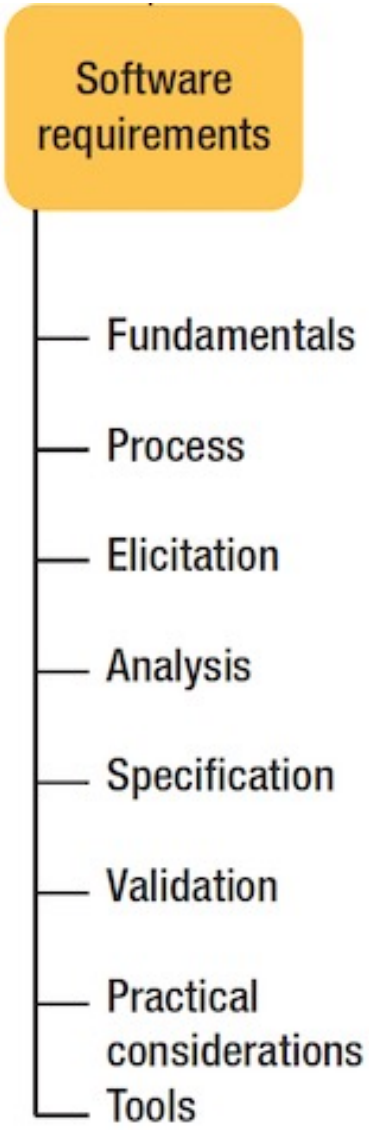
แสดงถึงความต้องการของลูกค้าโดยแสดงภาพใน 3 ลักษณะ

- the information domain
- the functional domain
- the behavioral domain

2.Design models แสดงถึงคุณลักษณะของซอฟต์แวร์ ใน 3 ลักษณะได้แก่

- the architecture
- the user interface
- component-level detail

Requirements Modeling Principles



Principle #1. The information domain of a problem must be represented and understood.

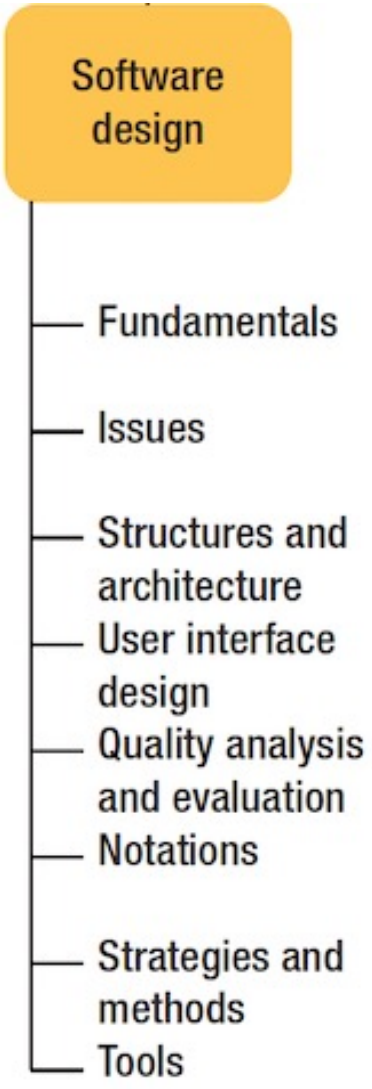
Principle #2. The functions that the software performs must be defined.

Principle #3. The behavior of the software (as a consequence (*ผล พวง ผลพั ธ์*) of external events) must be represented.

Principle #4. The models that depict information, function, and behavior must be partitioned in a manner that uncovers detail in a layered (or hierarchical) fashion.

Principle #5. The analysis task should move from essential information toward implementation detail.

Design Modeling Principles



Principle #1. Design should be traceable to the requirements model.

Principle #2. Always consider the architecture of the system to be built.

Principle #3. Design of data is as important as design of processing functions.

Principle #4. Interfaces (both internal and external) must be designed with care.

Principle #5. User interface design should be tuned to the needs of the end-user. Stress ease of use.

Principle #6. Component-level design should be functionally independent.

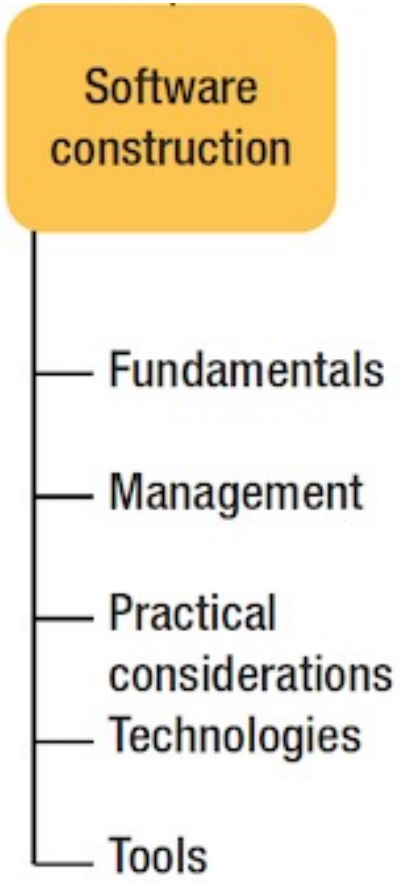
Principle #7. Components should be loosely coupled to each other than the environment.

Principle #8. Design representations (models) should be easily understandable.

Principle #9. The design should be developed iteratively.

Principle #10. Creation of a design model does not preclude using an agile approach.

Construction Principles



Coding principles and concepts

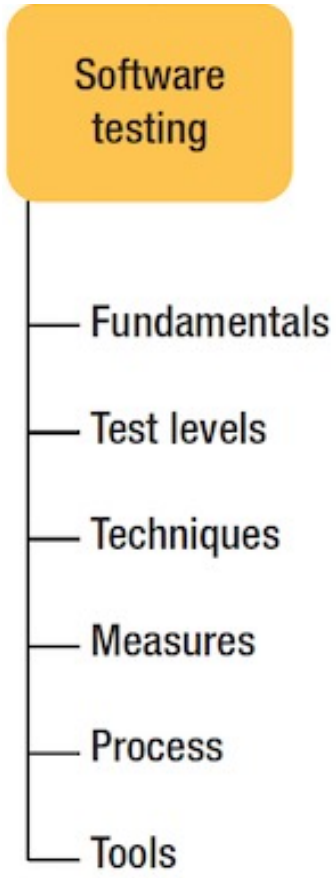
are closely aligned programming style,

programming languages, and programming methods.

Testing principles and concepts

lead to the design of tests that systematically uncover different classes of errors and to do so with a minimum amount of time and effort.

Testing Principles



Principle #1. All tests should be traceable to customer requirements.

Principle #2. Tests should be planned long before testing begins.

Principle #3. The Pareto principle applies to software testing.

Principle #4. Testing should begin “in the small” and progress toward testing “in the large.”

Principle #5. Exhaustive testing is not possible.

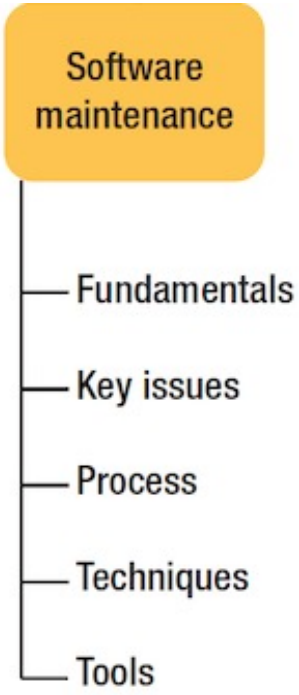
Principle #6. Testing effort for each system module commensurate to expected fault density.

Principle #7. Static testing (i.e. document review) can yield high results.

Principle #8. Track defects and look for patterns in defects uncovered by testing.

Principle #9. Include test cases that demonstrate software is behaving correctly.

Deployment Principles



Principle#1. Customer expectations for the software must be managed.

Principle#2. A complete delivery package should be assembled and tested.

Principle#3. A support regime must be established before the software is delivered.

Principle#4. Appropriate instructional materials must be provided to end-users.

Principle#5. Buggy software should be fixed first, delivered later.

END