# Software Engineering

**Lecture 3(b):**
Software engineering method
and process models

# Outline

- Programming vs. software development
- SE, computing, and engineering
- Characteristics of software
- What is Software Engineering (SE)
- Software development process model
  - life cycle: waterfall, spiral
- Case study: KEngine
  - keyword search engine

# References

- Liskov & Guttag (2001):
  - Chapter 11
- Sommerville (2011):
  - Chapters 2, 3
- Boehm (1988)
- LeBlanc et al. (2006): SE curriculum guide 2004

# Programming vs. Software Development

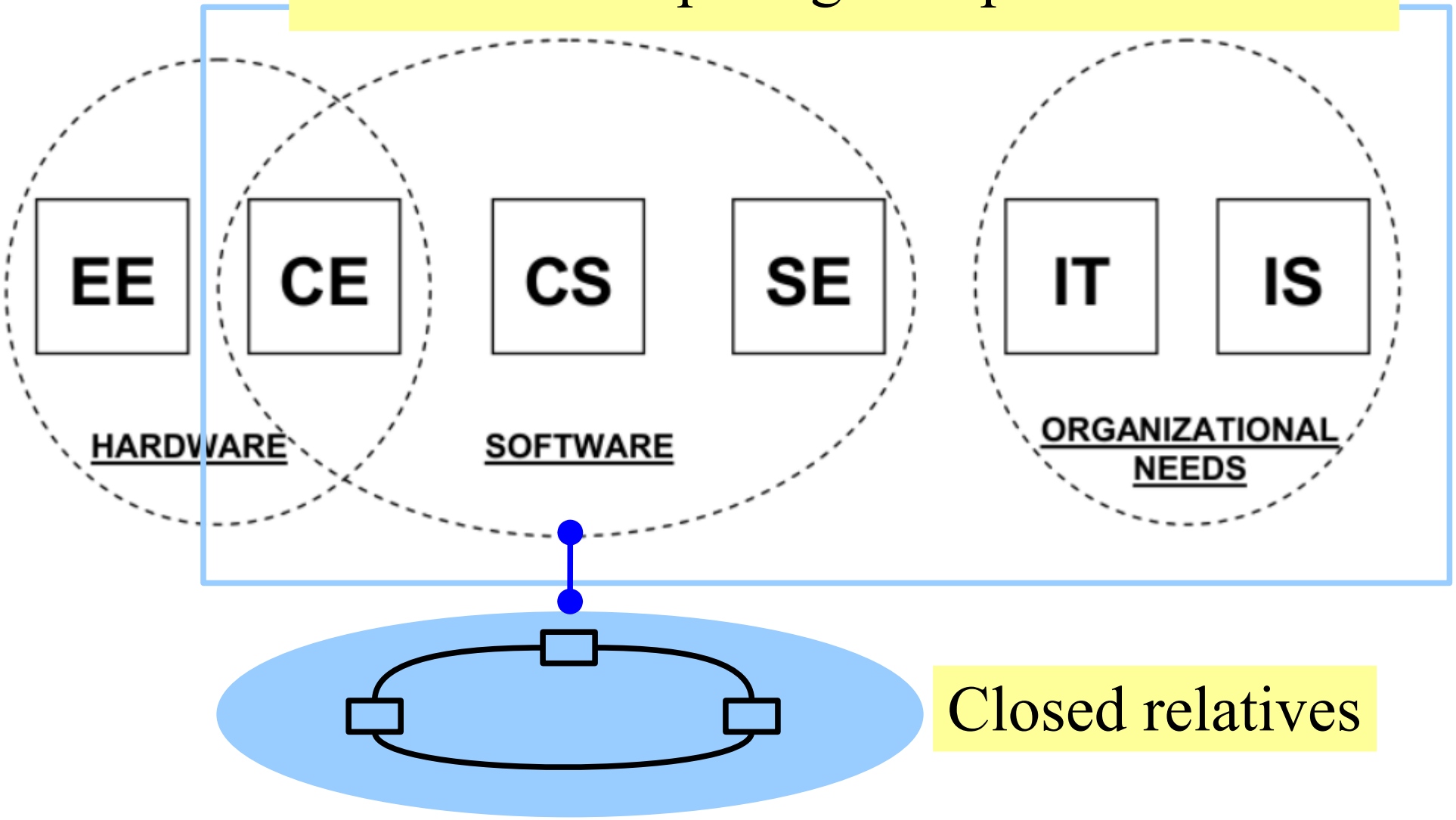
- **Programming** (coding):
  - a **specific task** in software development, which involves **writing** specific computer instructions to solve **a given problem** in one or more programming languages (C#, Java, etc.)
- **Software development**:
  - a **broader creative** process of constructing a software (a 'bigger' program) for a particular purpose (which typically constitutes solving **a set of problems**).
  - consists of several phases: programming is only one step in one of these phases

# Programming vs. Software Development (2)

- "*Computer programmers* ***write code*** to create software programs. They  turn the program designs created by software developers and engineers into instructions that a computer can follow." U.S. Bureau of Labor Statistics
- "*Software developers* are the creative minds behind computer programs. Some ***develop*** the applications that allow people to do specific tasks on a computer or other device. Others develop the underlying systems that run the devices or control networks." U.S. Bureau of Labor Statistics

# SE in computing (1)

Computing disciplines




Closed relatives

*Source: ACM SE curriculum (2004)*

# SE in computing (2)

- Computer Engineer (CE): software in hardware devices
- Software Engineering (SE): software satisfies robust real-world requirements
- Computer Science (CS): software is the currency for expressing computing ideas

# SE and CS

- SE's foundation is CS:
  - CS: foundational concepts, techniques and tools
  - SE: development process
- CS:
  - build "clever" software, devise new ways (e.g. new algorithm, language, etc.)
- SE:
  - builds "complex" (large) software,

    in a disciplined manner (i.e. the process)
  - focus on quality

# SE and engineering: similarities

- **decision**-based
- **measure** things
- use a disciplined **process**
- operate effectively as part of a **team**
- multiple **roles**
- use **tools** systematically
- reuse **designs** and design artifacts
- **advance** principles, standards, and best practices

# SE and engineering: differences

- SE's **foundations** are primarily in CS not natural sciences
- **discrete** rather than continuous mathematics
- **abstract/logical** entities instead of concrete/physical artifacts
- **no "manufacturing"** phase as such
- "maintenance" means **evolution** (not conventional wear and tear)

# Characteristics of software

- Non-physical
- Interact with other real-world systems:
  - social systems
  - physical systems

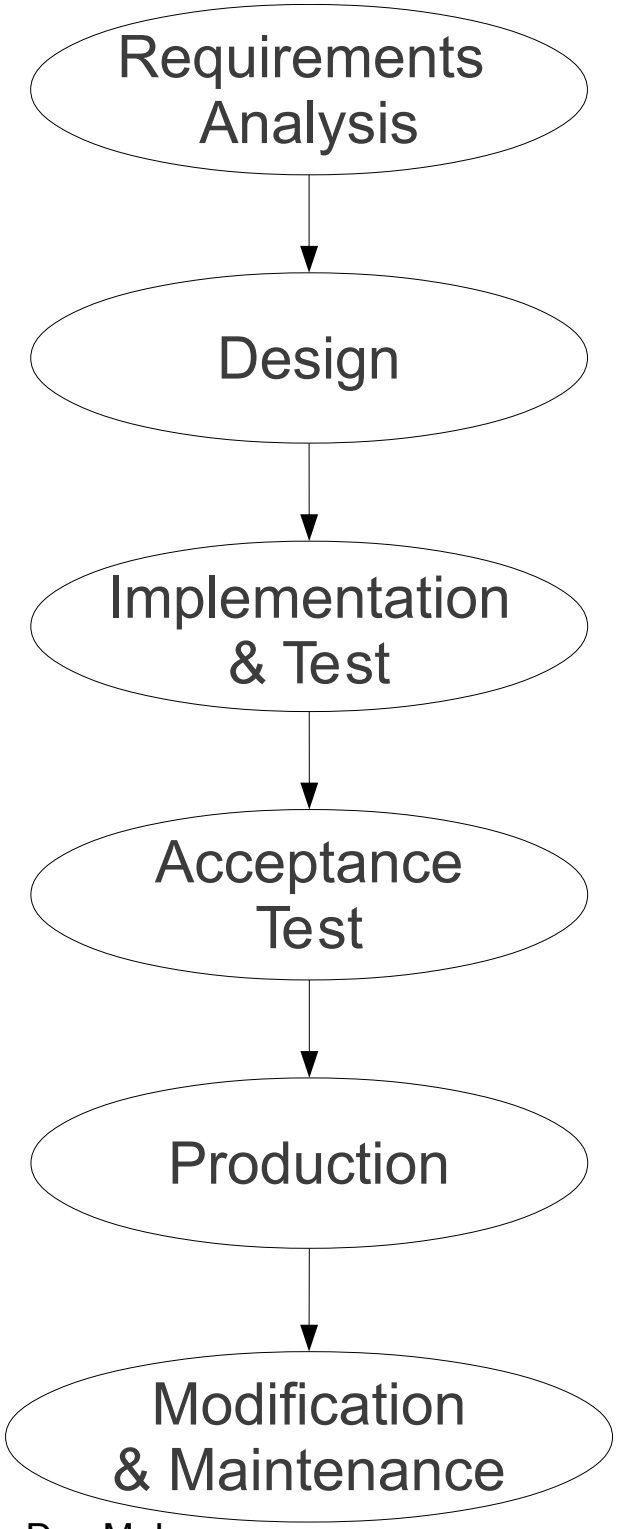
# What is SE?

- An engineering discipline for software development
- Objectives:
  - quality
  - suitable for purpose
- Use a software process

# Software development life cycle

- A sw. development life cycle consists of 6 phases:
  - Requirements analysis
  - Design
  - Implementation & testing
  - Acceptance test
  - Production
  - Maintenance
- Different **process models** for executing the phases:
  - which model to use depends on the nature of software

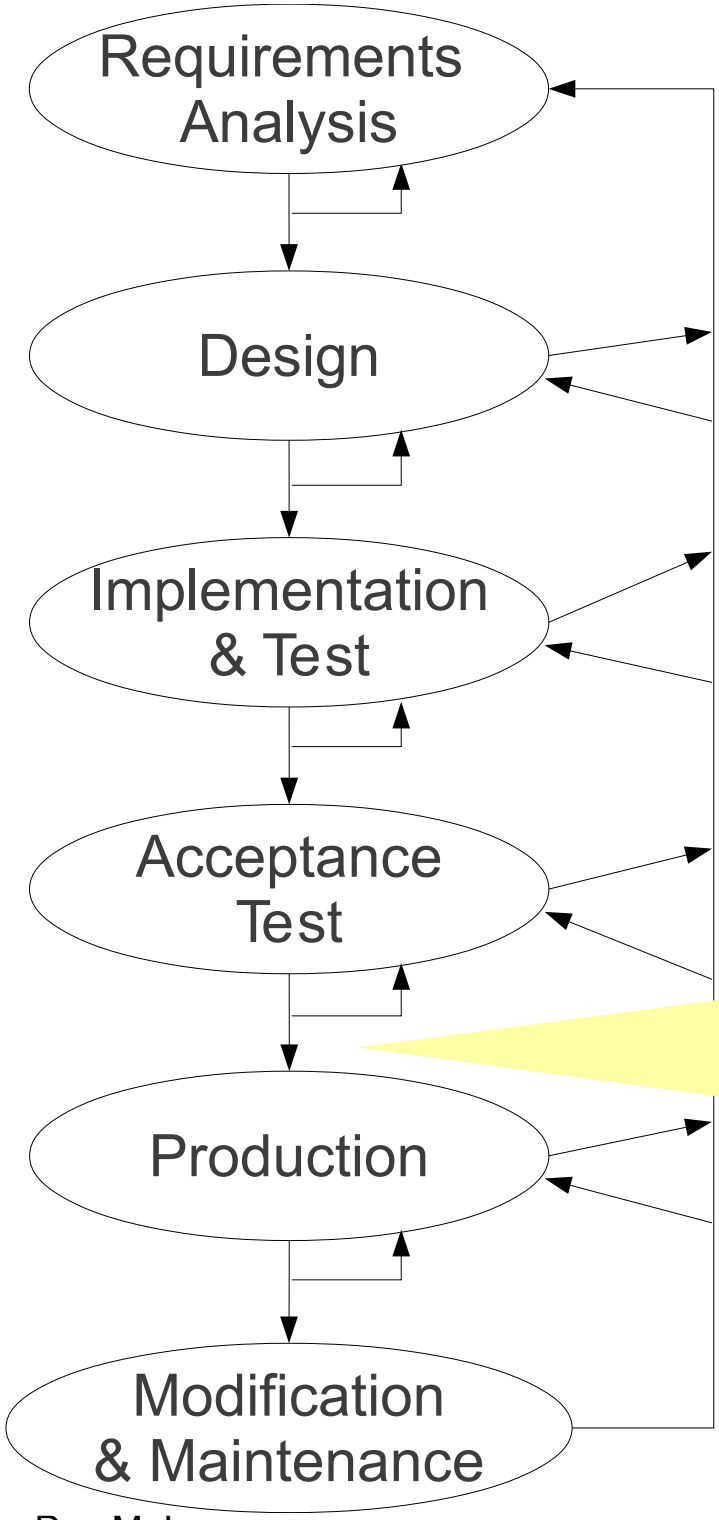
# Waterfall process model




phases are performed
in **sequence**
(errors are detected very late on)

(Liskov & Guttag, 2001)

# Spiral model



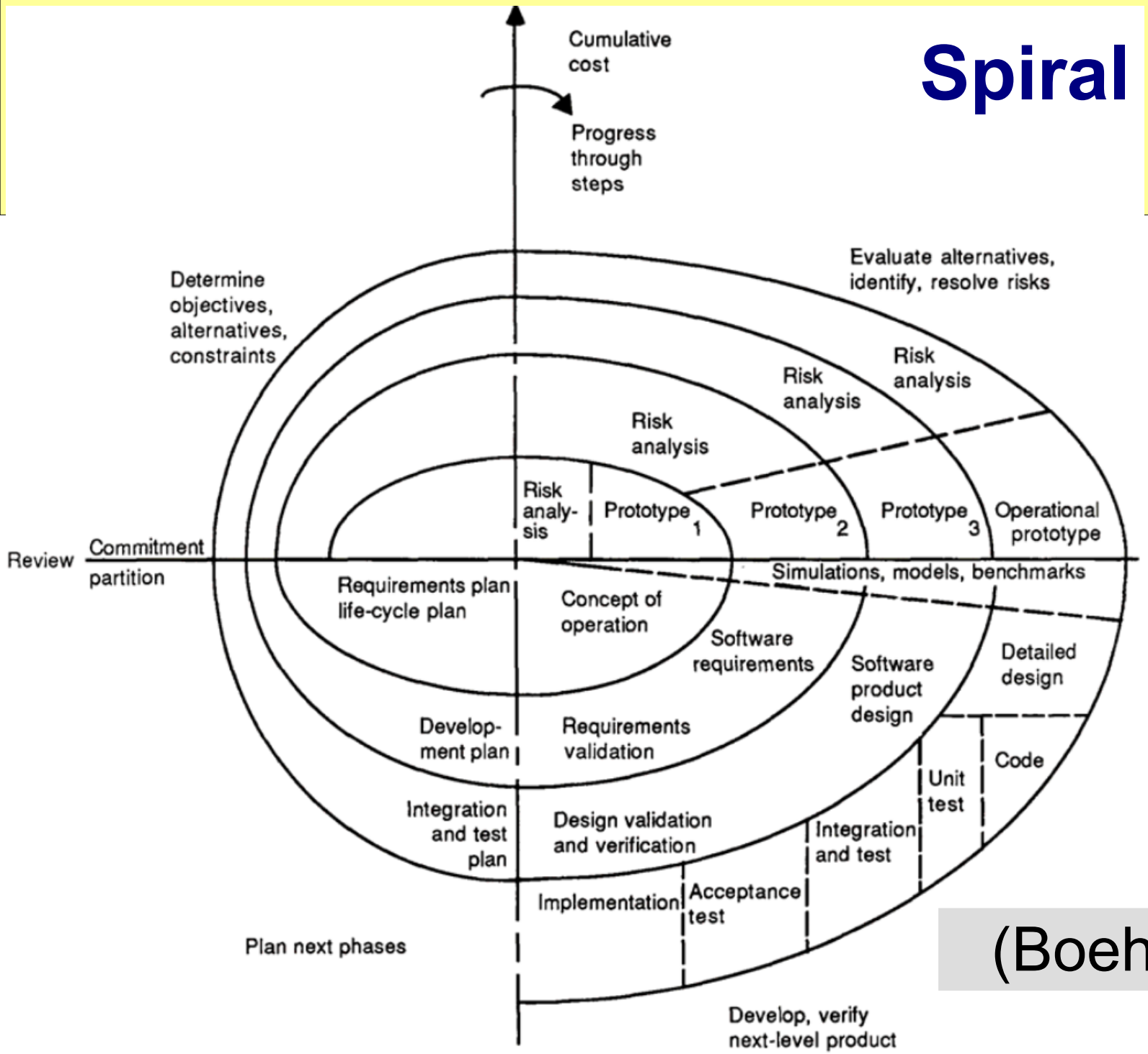

feedback loops allow errors to be detected at each phase

phases are performed in **iterations** with **feedback loops**

2-way interactions between phases

(Liskov & Guttag, 2001)

# Spiral model (2)



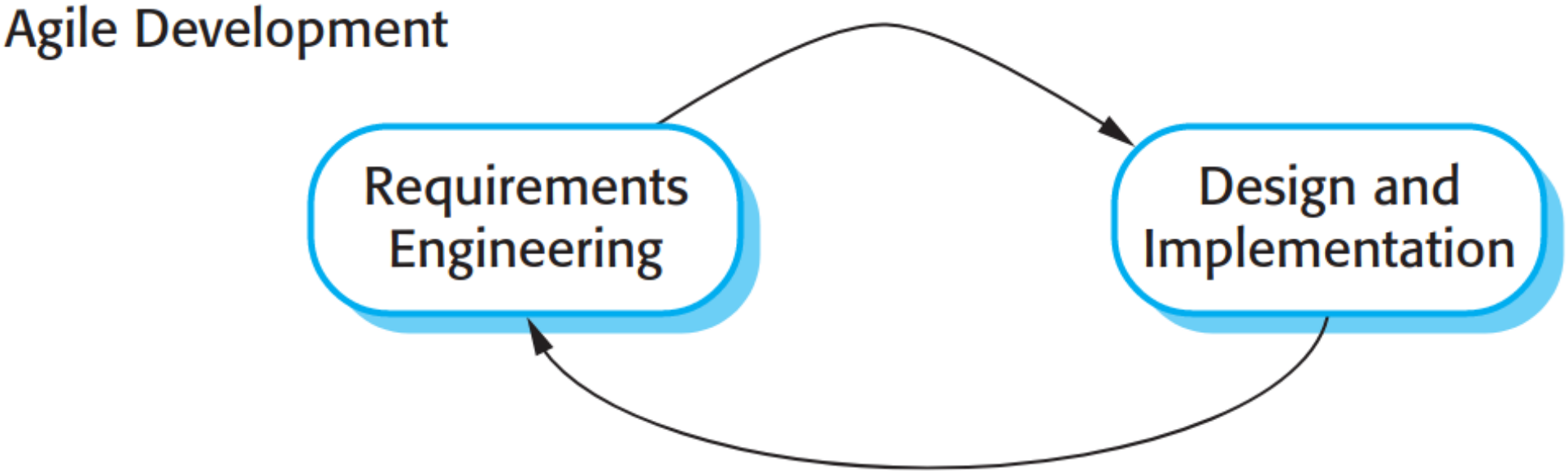

(Boehm, 1988)

# Incremental model

- Develop software through several **increments** (versions)
- Activities are **interleaved** rather than separate, with rapid feedback across activities
- Cheaper and **easier to make changes** while software is being developed
- Early increments include the most important or most urgently required functionality

# Agile model

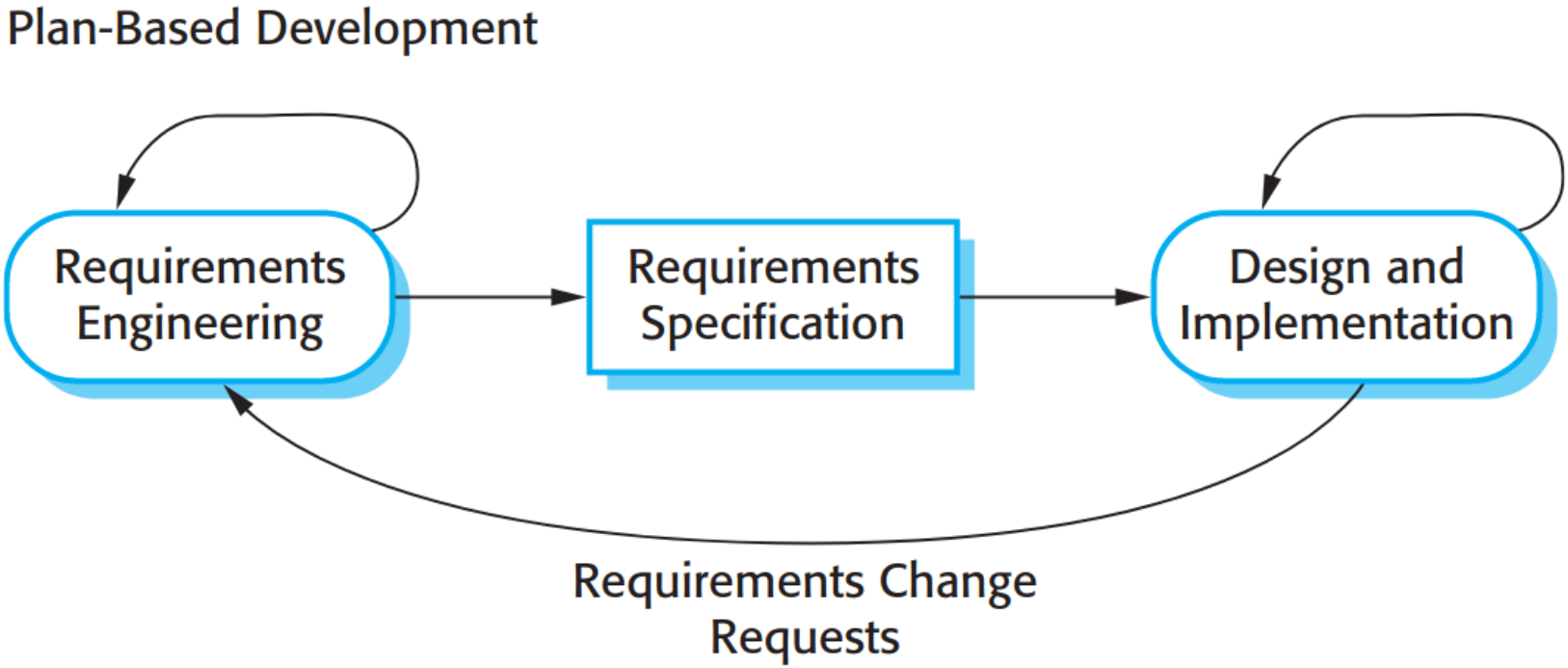
- A popular form of incremental development
- Iteration occurs ***across*** activities
- Simplified communication between activities




(Sommerville, 2011)

# Plan-driven models (e.g. waterfall)

- Iteration ***within*** activities
- Formal outputs between activities




(Sommerville, 2011)

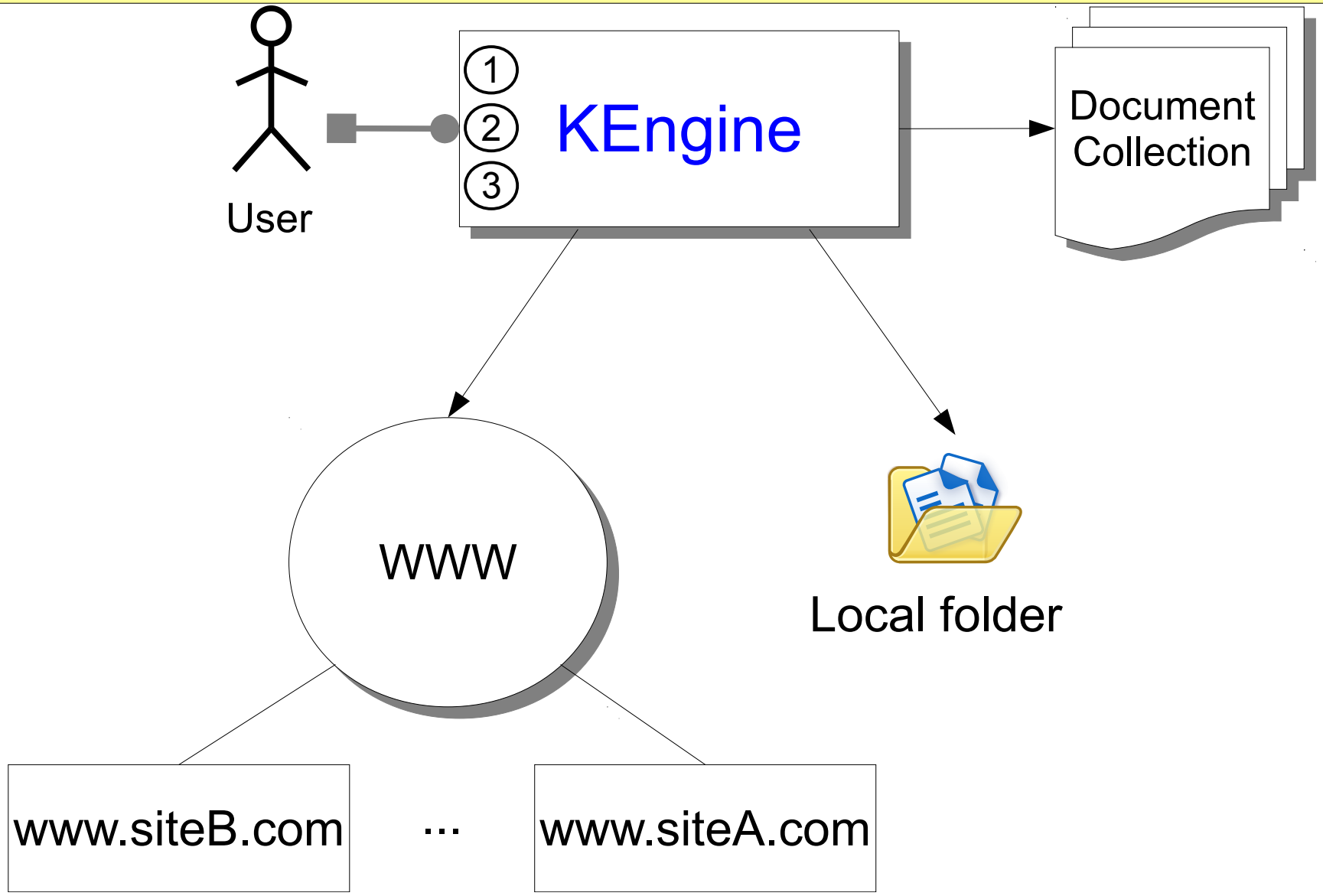
# Agile model characteristics

- Small increments:
  - new software releases are made available to customers every 2-3 weeks
- Involve customers to get rapid feedback on changing requirements
- Minimize documentation:
  - informal communications rather than formal meetings & written documents

# Case study: KEngine

- Liskov's Section 12.4 + some modifications
- A keyword search engine that enables a user to perform three basic tasks:
  1. To ***obtain*** a collection of *documents*
  2. To ***retrieve*** documents by title
  3. To ***search*** for relevant documents using keywords

# KEngine overview



# Document

- A sequence of words
- HTML document:
  - has title and body
- Examples...

# Document (d1)

```
<html>
<head>
  <title> welcome to my page </title>
</head>
<body>
<p>
this is a test page to test the simple
Doc parser
</body>
</html>
```

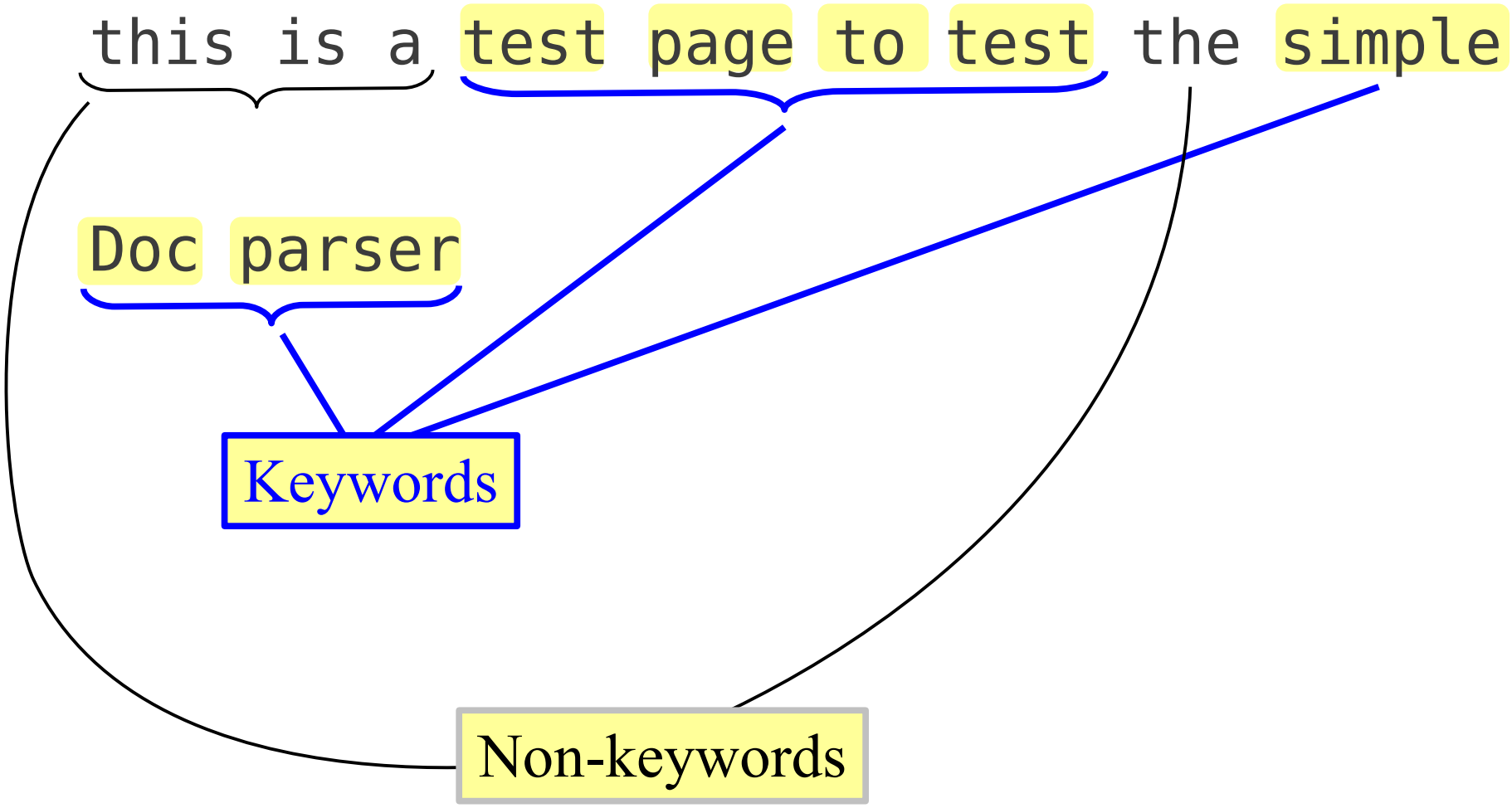
# Document (d2)

```
<html>
<head>
  <title> welcome to my page </title>
</head>
<body>
<p>
another test page
</body>
</html>
```

# Word

- A word provides data
  - excl. HTML tags
- Two types: **keyword** and **non-keyword**
- Non-keyword is an uninteresting word:
  - e.g. commonly found words: an/a/the
- A keyword is an interesting word
- **Keyword frequency**: the number of times a keyword appears in a document

# Words example



# Keyword frequencies

<"test",<d1,2>> <"test",<d2,1>>

- word "test" appears twice in d1, once in d2

<"page",<d1,1>> <"page",<d2,1>>

<"to",<d1,1>> <"simple",<d1,1>>

<"Doc",<d1,1>> <"parser",<d1,1>>

<"another",<d2,1>>

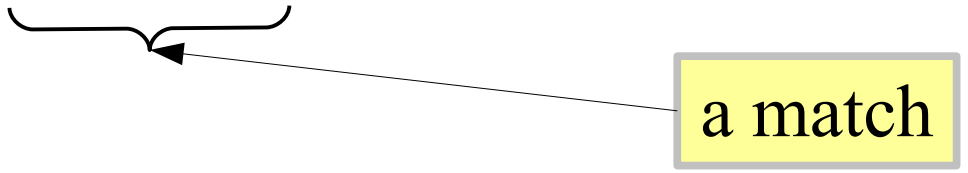
# Query

- A query consists of a set of keywords
- A query result is a set of matches
- Match = <d,f>
  - d: a document containing all query keywords
  - f: sum of the frequencies of the query keywords d
- Matches are sorted in desc. order

# Example

Query:{"test"}

→ Result:{<d1,2>,<d2,1>}

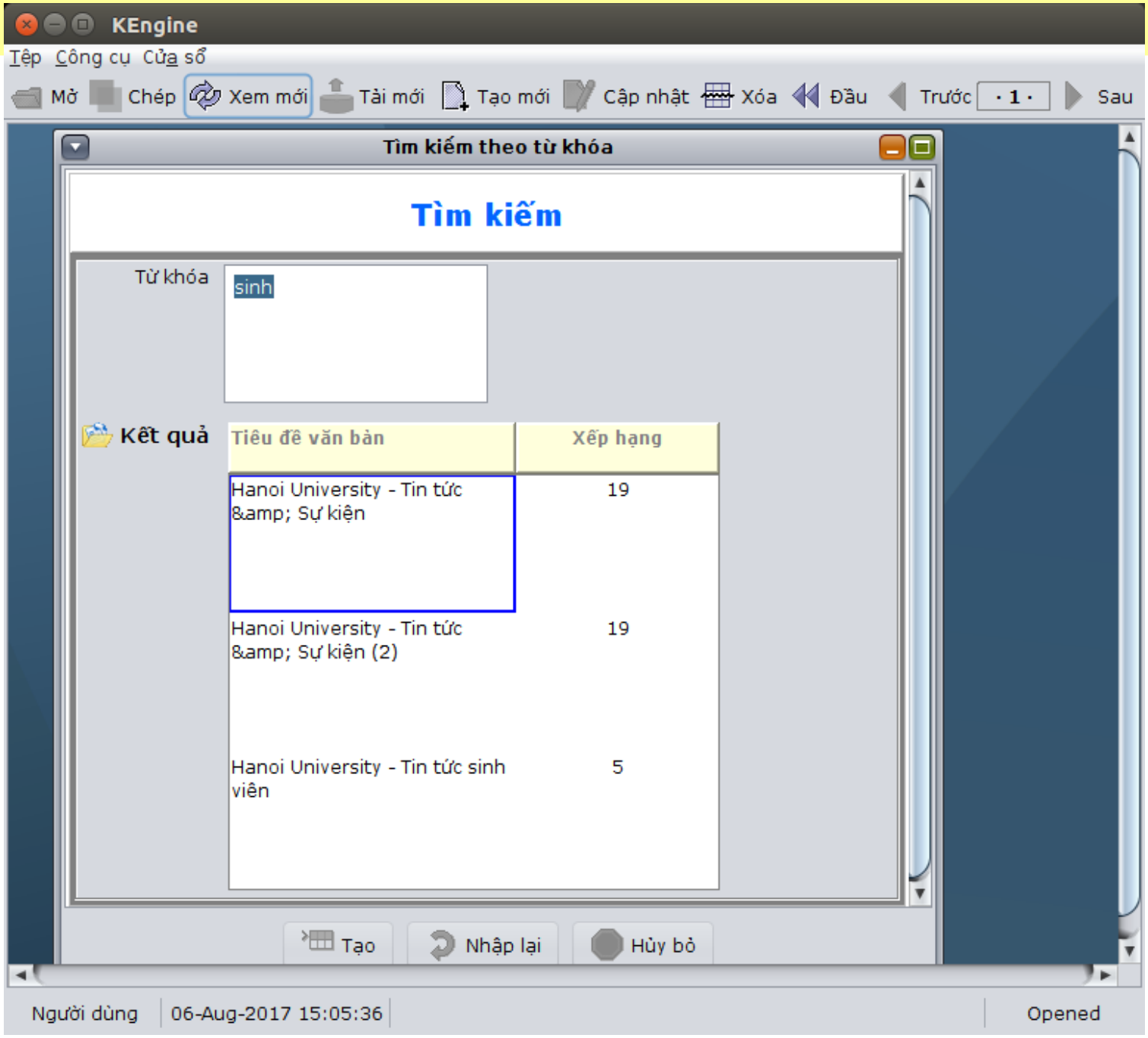



Query:{"test","page"}

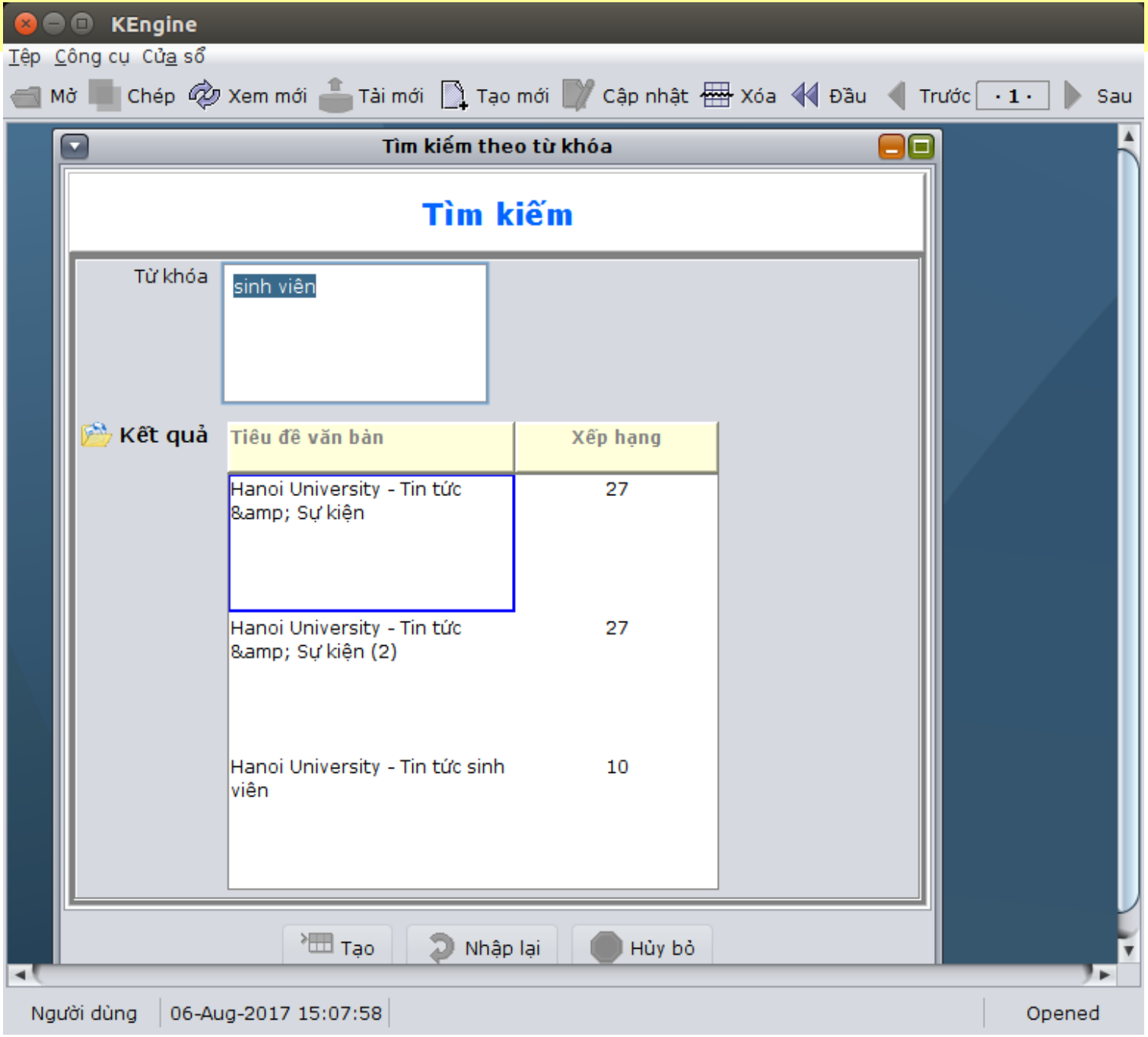
→ Result:{<d1,3>,<d2,2>}

?

# *KEngine Demo*: 1- Enter keyword "sinh"



# 2- Enter keywords "sinh viên"



# Q & A