



SOFTWARE DEVELOPMENT I

4th lecture



This lecture

- SWEBOK overview.
- Software system construction. Key goals and challenges.
- Business needs analysis.
- Software system modification and maintenance (introduction)





IF... You are „just“ hardcore coder

- If you do not want to be this bridge... You MUST understand how coding „tasks“ are created.





Software Engineering 2

Lab. 1 – 1st iteration, requirements (cont.)

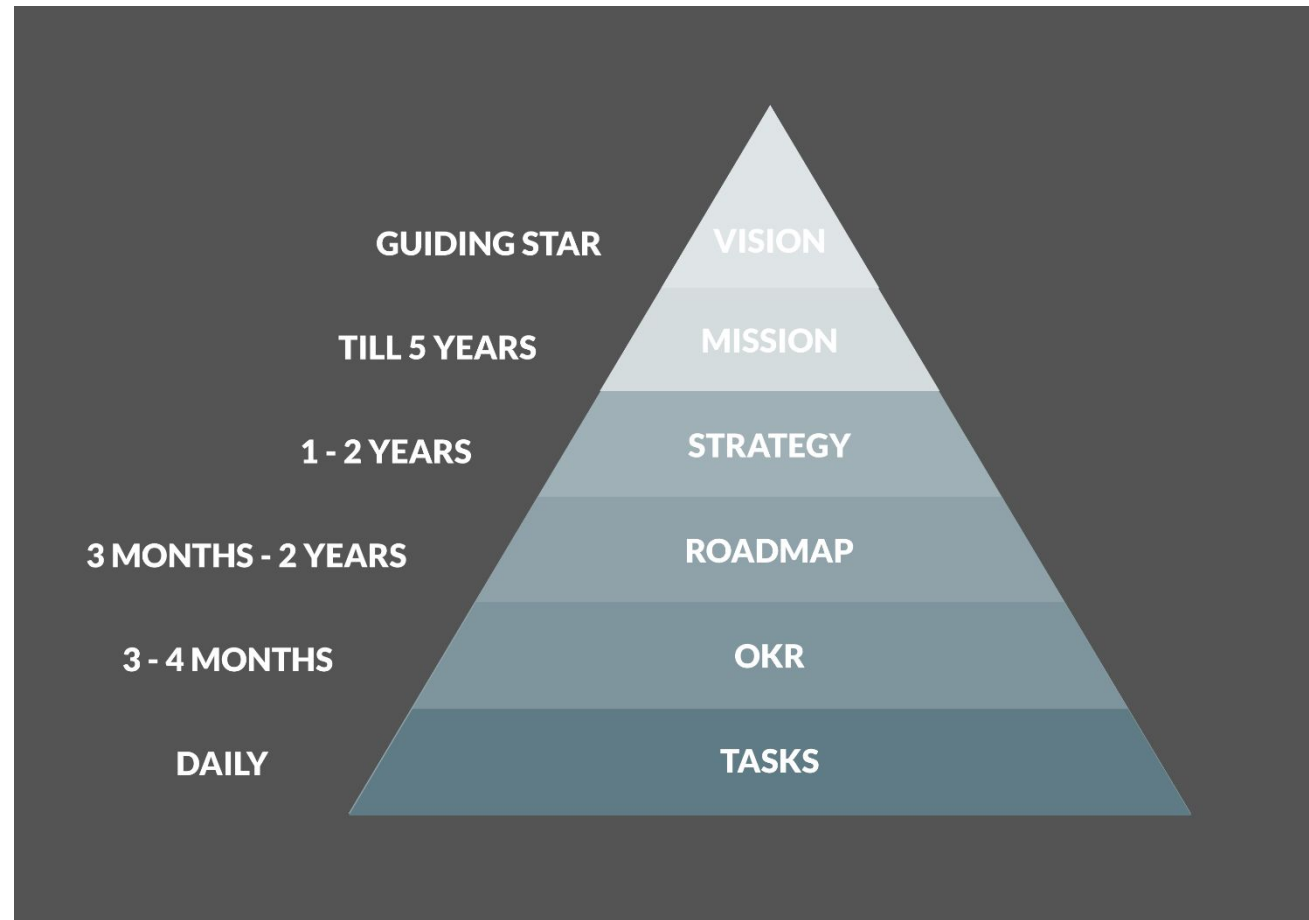
Suggested document structure:

1. Context. Describe the analyzed business, its boundaries, context, customers, partners, etc.
2. Stakeholders and their Needs. Identified and documented stakeholder needs, traceable.
3. Business Model.
 - 3.1 Conceptual Model.
 - 3.2 Behaviour Model.
4. Requirements.
 - 4.1 Functional Requirements. Large part of the business level requirements can be provided in the business model at the relevant context.
 - 4.2 Non-functional Requirements.
5. Validation. Traceability matrices, etc.

What does „theory“ say?



What does „theory“ say?





Vision statements

Vision (sometimes – mission) is the reason why the product exists.

- **Disney:** *To make people happy.*
- **Ikea:** *To create a better everyday life for the many people.*
- **Google:** *to provide access to the world's information in one click.*
- **Microsoft:** *to help people and businesses throughout the world realize their full potential.*



SWEBOK

- SWEBOK - Software
Engineering Body of
Knowledge.




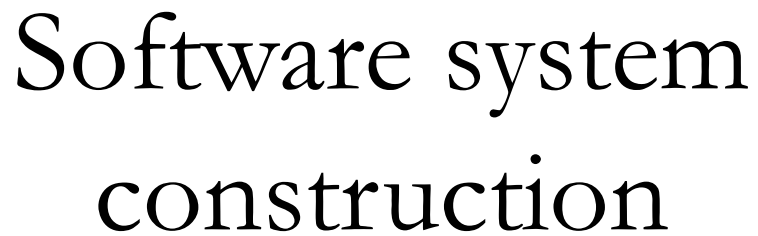
*Guide to the Software
Engineering Body of Knowledge*

Editors

Pierre Bourque
Richard E. (Dick) Fairley



IEEE  computer society



- detailed creation of working software through a combination of coding, verification, unit testing, integration testing, and debugging.
- it is strongly linked to Software Design and Software Testing

```

    ...;e.s.r||(i||!i)},"margin"(s,r,sale,n
    "padding"(s,parseFloat(Pt(e,"border"st i"width"
    null)r.e.style.t:(Ut.test(r))return r;i s(v.supp
    elay");t.remove();if(n=="none"n=="")Pt(i.body.app
    etype.html><html><body>"),Ht.close();t Ht.body.appendCh
    object">t ">">"}",i,n,r));else if( n v.type(t) == "obj
    nght;if(v.isFunction(n))for(;u a;u--)r o u,s \+ .tes
    l(c u);f--u a f(n,r,i),typeof_u=="string"&(c
    in.n)n r t((s r?e i(i{}))r n r);i v.extend
    ),i=t&(i e.mimeType|n.getResponseHeader("content-ty
    t(o),r o)}function.On(e,t){var n,r,i,s,o=e.dataTypes.s
    "i||a["+w.+i];if(!n)for(r.in.a){s=r.split(".");if(s
    or","error:n?l:"No conversion from."+u+".to."+i}}u=i}re
    t({})}function.$n(){return.setTimeout(function(){qn=t,l
    ,u=v.Deferred().always(function(){delete a.elem}),a=f
    ith(e,[f]),!1}},f=u.promise({elem:e,props:v.extend({}
    (e,f.opts,t,n,f.opts.specialEasing[t]|f.opts.easing)
    ps;qn(l,f.opts.specialEasing);for(;i<o;i++){r=Xn[i].c
    ess(f.opts.progress).done(f.opts.done,f.opts.complete
    &&(i=s[1],s=e[n]=s[0]),n!=r&&(e[r]=s,delete e[n]),o=
    ,g=e.nodeType&&Gt(e);n.queue||(l=v._queueHooks(e,"fx"
    y.fire())))),e.nodeType===1&&("height"in.t||"width"i
    block":p.zoom=1)),n.overflow&&(p.overflow="hidden",v.
    "toggle";if(s==(g?"hide":"show"))continue;m.push(r)}
    oveData(e,"fxshow",!0);for(t.in.d)v.style(e,t,d[t]))
    prototype.init(e,t,n,r,i)}function.Zn(e,t){var n,r={h
    ow:!1}var n,r,i=e.document,s=e.location,o=e.navigator
    lice,c=Array.prototype.indexOf,h=Object.prototype.toS
    0]+([\s\uFEFF\xA0]+$/g,w=/^(?:[#<]*(<[\w\W]+>)[^>]*
    )/g,C=/^-ms-/ ,k=-([\da-z])/gi,L=function(e,t){return
    letachEvent("onreadystatechange",A),v.ready()),O={};
    typeof e=="string"){e.charAt(0)=="<"&&e.charAt(e.le
    isPlainObject(n)&&this.attr.call(e,n,!0),v.merge(thi
    ctor(n).find(e)}return.v.isFunction(e)?r.ready(e):(
    (this)},get:function(e){return.e==null?this.toArray(
    "find"?r.selector=this.selector+(this.selector?" ":"
    ){return.e+=e,e===-1?this.slice(e):this.slice(e,e-1
    },1,call(arguments).join(",")},map:function(e){retu
    v.fn,v.extend-v.fn.extend-function(){var e,n,r,i,x,
    p(a:f;a--)}if((e=arguments[a])!=null)for(n in e){
    re v,v.extend(noConflict:function(){return e

```



Minimizing complexity

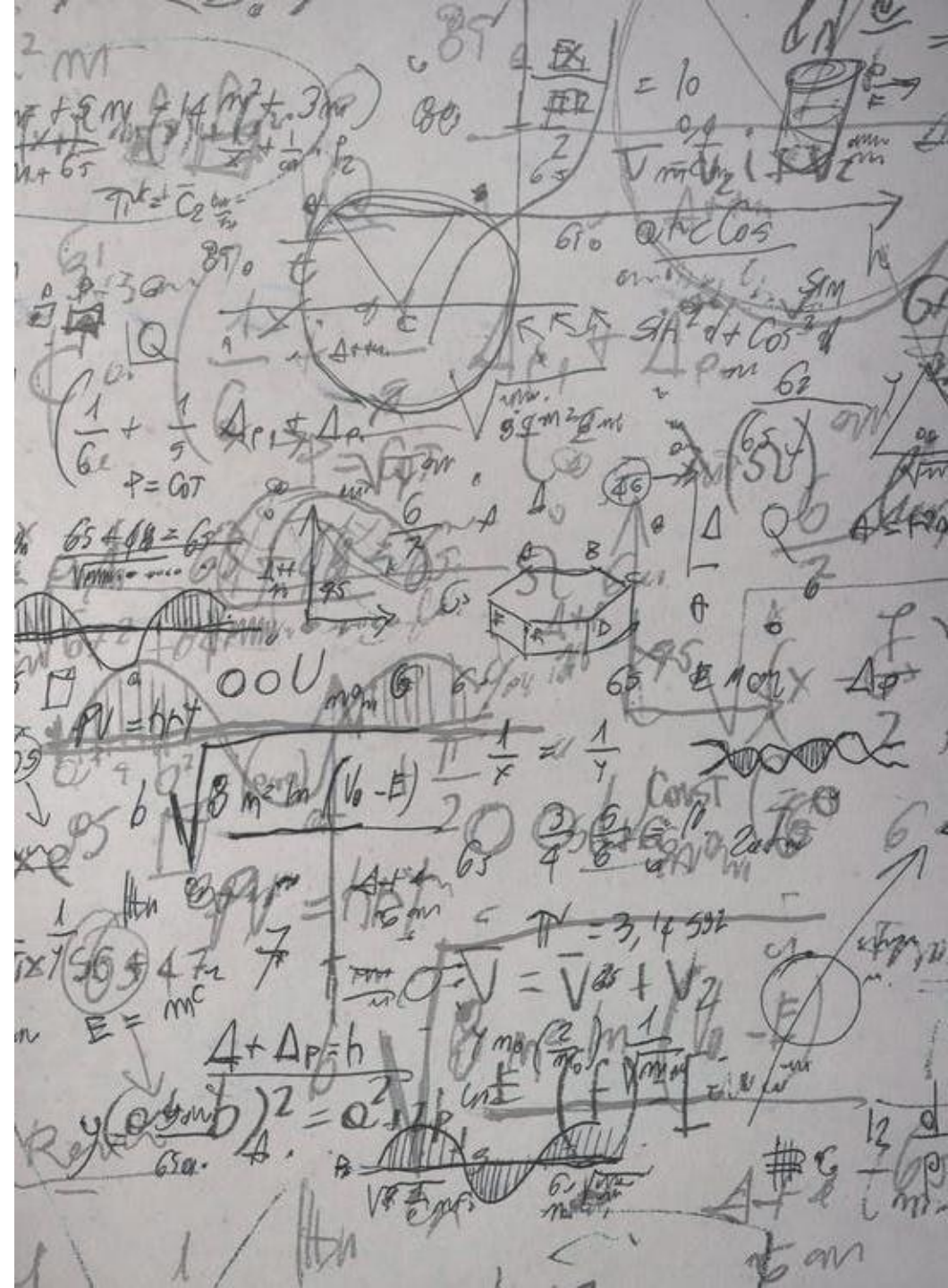
- Most **people are limited** in their ability to hold complex structures and information in their working memories, especially over long periods of time.
- The need to **reduce complexity** applies to essentially every aspect of software construction and is particularly critical to testing of software constructions.
- Basically, reduced complexity is achieved through emphasizing code creation that **is simple and readable rather than clever**



Minimizing complexity

Achieved by:

- making use of standards
- modular design
- specific coding techniques
- construction-focused quality techniques





Constructing for verification

- Building software in such a way that faults can be readily found by the software engineers writing the software as well as by the testers and users during independent testing and operational activities.
- Good examples:
 - following coding standards to support code reviews and unit testing,
 - organizing code to support automated testing,
 - restricting the use of complex or hard-to-understand language structures
 - etc.



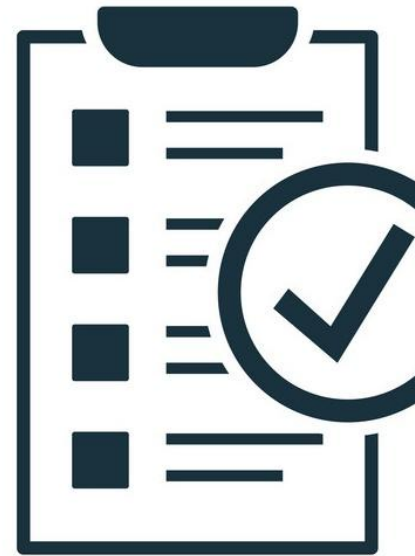
Reuse

- Reuse refers to using existing assets in solving different problems.
- In software construction, typical assets that are reused include libraries, modules, components, source code, etc.
- Systematic reuse can enable significant software productivity, quality, and cost improvements.
- Two types:
 - construction **for reuse**: means to create reusable software assets
 - construction **with reuse**: means to reuse software assets in the construction of a new solution



Standards in construction

- Helps achieve a project's objectives for efficiency, quality, and cost.
- Examples:
 - communication methods (for example, standards for document formats and contents)
 - programming languages
 - coding standards (naming conventions, layout, and indentation)
 - platforms (interface standards for operating system calls)
- Can be external and internal



Anticipating change

- Most software **will change** over time!
- **Anticipation** of change drives many aspects of software construction.
- Anticipating change helps software engineers build **extensible software**, which means they can enhance a software product without disrupting the underlying structure.
- Basically: *you have to write your code **expecting multiple changes** on it!*



Measurement(s)

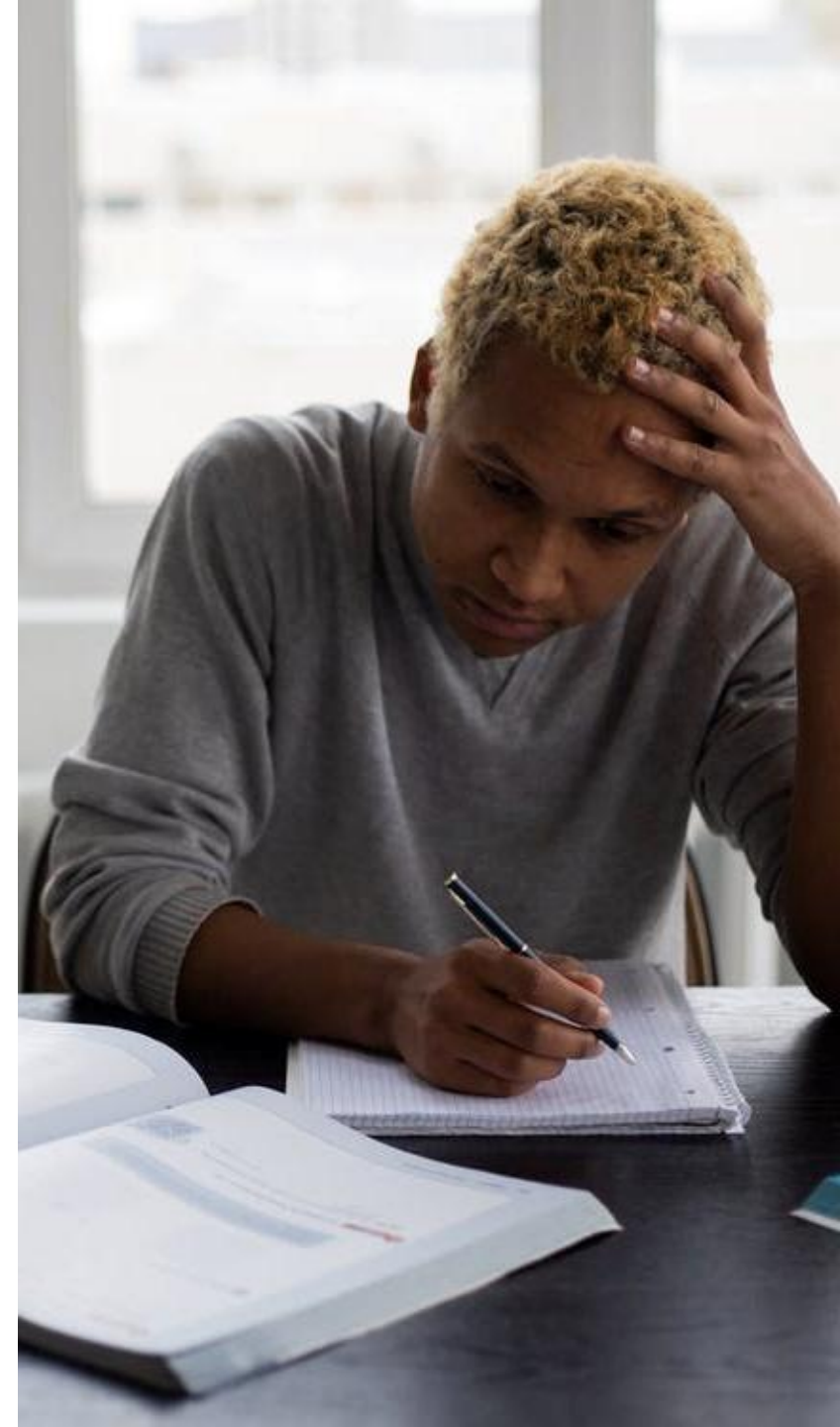
- Can be useful for purposes of managing construction, ensuring quality during construction, and improving the construction process.
- Examples:
 - code developed,
 - code modified,
 - code reused,
 - code destroyed,
 - code complexity,
 - code inspection statistics,
 - fault-fix and
 - fault-find rates,
 - etc.





Coding considerations

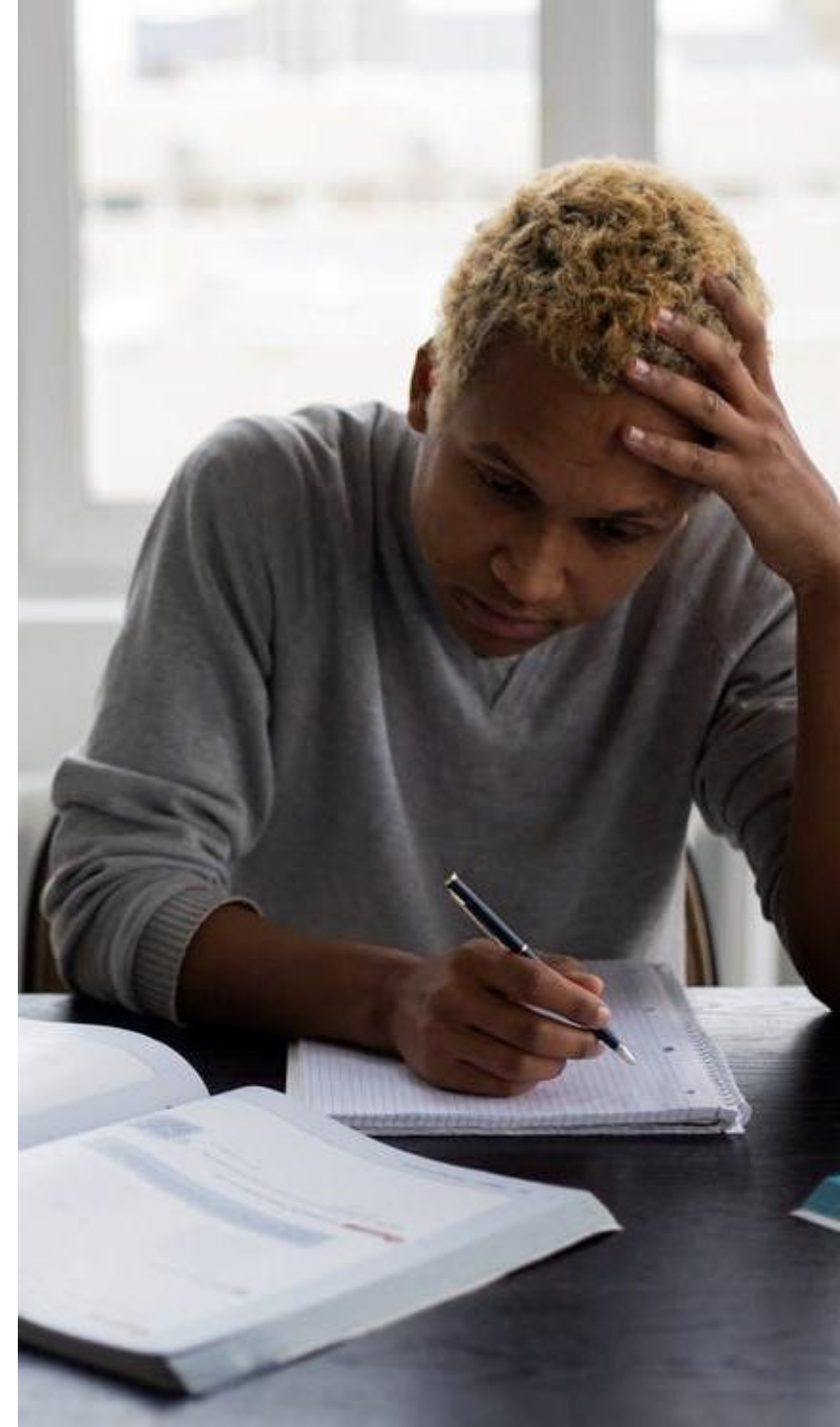
- Techniques for creating **understandable** source code, including **naming conventions** and source code layout
- Use of classes, enumerated types, variables, named constants, and other similar entities
- Use of control structures
- Handling of **error conditions**—both anticipated and exceptional (input of bad data, for example)





Coding considerations

- Prevention of code-level security breaches (e.g. buffer overflows or array index bounds)
- Resource usage via use of exclusion mechanisms and discipline in accessing serially reusable resources (including threads and database locks)
- Source code organization (into statements, routines, classes, packages, or other structures)
- Code documentation
- Code tuning





Construction quality

- unit testing and integration testing
- test-first (or test-driven) development
- use of assertions and defensive programming
- debugging
- inspections
- technical reviews
- security-oriented reviews
- static analysis





Business needs analysis

- **Business analysis** is
 - set of tasks and techniques
 - used to work as a liaison among stakeholders
 - in order to understand the structure, policies, and operations of an organization,
 - and to recommend solutions that enable the organization to achieve its goals.





Business needs analysis

- **Business analysis** involves
 - understanding how organizations function
 - to accomplish their purposes,
 - and defining the capabilities an organization requires
 - to provide products and services to external stakeholders.





Business analysis

- Business analysis may be performed to understand the current state of an organization or to serve as a basis for the later identification of business needs.
- In most cases, however, business analysis is performed to define and validate solutions that meet business needs, goals, or objectives.



Business Analyst (BA)

- A **business analyst** is any person who performs business analysis activities, no matter what their job title or organizational role may be.
- Business analysis practitioners include not only people with the job title of business analyst, but may also include:
 - business systems analysts, systems analysts, requirements engineers, process analysts, product managers, product owners, enterprise analysts, business architects, management consultants and others.



BABOK: BA techniques

- Acceptance and Evaluation Criteria Definition (9.1)
- Brainstorming (9.3)
- Business Rules Analysis (9.4)
- Data Dictionary and Glossary (9.5)
- Data Flow Diagrams (9.6)
- Data Modeling (9.7)
- Decision Analysis (9.8)
- Document Analysis (9.9)
- Interviews (9.14)
- Metrics and Key Performance Indicators (9.16)
- Non-functional Requirements Analysis (9.17)
- Organization Modeling (9.19)
- Problem Tracking (9.20)
- Process Modeling (9.21)
- Requirements Workshops (9.23)
- Scenarios and Use Cases (9.26)



Requirement analysis

- At its most basic, a **software requirement** is a property that must be exhibited in order to solve some problem in the real world.
- An essential property of all software requirements is that they be **verifiable**. Sometimes that might be difficult.
- Software requirements should be stated as clearly and as unambiguously as possible, and, where appropriate, **quantitatively**.



REQUIREMENTS ARE RIGHT BEFORE CODING.

HOW DO YOU START BEFORE THAT?

BMI • Business model canvas

<div><div>● Key partners</div><div>Who are your most important partners? Which key resources do you acquire from partners? Which key activities do your partners perform?</div></div>	<div><div>● Key activities</div><div>What are the activities you perform every day to create & deliver your value proposition?</div></div> <div><div>● Key resources</div><div>What are the resources you need to create & deliver your value proposition?</div></div>	<div><div>● Value propositions</div><div>What is the value you delivery to your customer? Which of your customer's problems are you helping to solve? What is the customer need that your value proposition addresses? What is your promise to your customers? What are the products and services you create for your customers?</div></div>	<div><div>● Customer relationships</div><div>What relationship does each customer segment expect you to establish and maintain?</div></div> <div><div>● Channels</div><div>How does your value proposition reach your customer? Where can your customer buy or use your products or services?</div></div>	<div><div>● Customer segments</div><div>For whom are you creating value? What are the customer segments that either pay, receive or decide on your value proposition?</div></div>
<div><div>● Cost structure</div><div>What are the important costs you make to create & delivery your value proposition?</div></div>			<div><div>● Revenue streams</div><div>How do customers reward you for the value you provide to them? What are the different revenue models?</div></div>	



MOST Analysis

Mission

A statement declaring what business the organisation is in, and what it is intending to achieve

Objectives

The goals against which the organisation's achievements can be measured

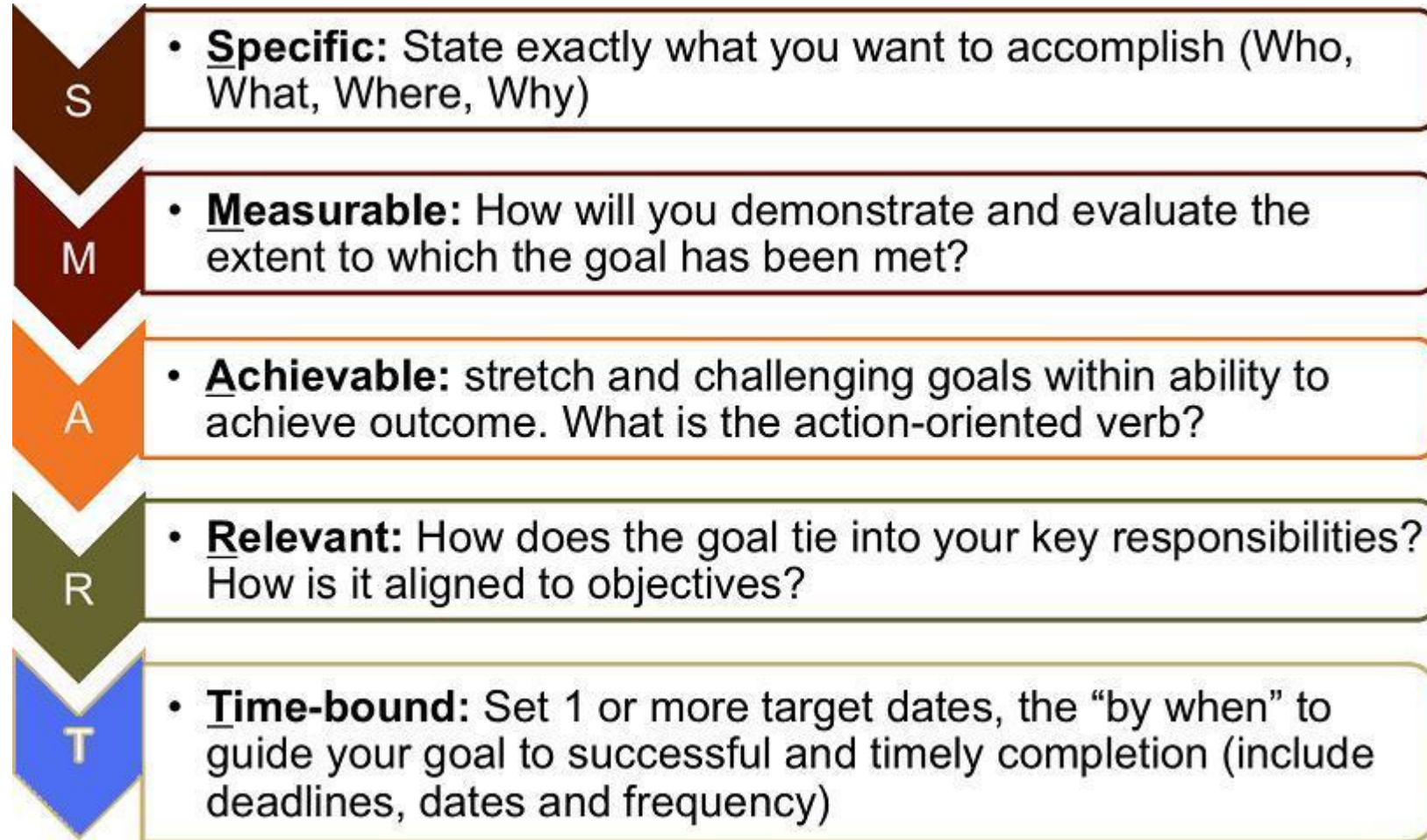
Strategy

The medium to long-term methods for achieving organisational objectives

Tactics

The short term activities with which the strategy is implemented

SMART Goals





Stakeholdership management

- Stakeholder: a member of "groups without whose support the organization would cease to exist"
- Each stakeholder group has different stakes at company or product.

Examples:

- Employees
- Communities
- Shareholders
- Creditors
- Investors
- Government
- Customers
- Owners
- Financiers
- Managers



Stakeholder management

Stakeholders:	Stakeholder's concerns: ^[11]
Government	taxation, VAT , legislation , employment, truthful reporting, legalities, externalities...
Employees	rates of pay, job security , compensation, respect, truthful communication, appreciation, acknowledgement, recognition.
Customers	value, quality, customer care, ethical products.
Suppliers	providers of products and services used in the end product for the customer, equitable business opportunities.
Creditors	credit score, new contracts, liquidity.
Community	jobs, involvement, environmental protection, shares, truthful communication.
Trade unions	quality, worker protection, jobs.
Owner(s)	profitability, longevity, market share, market standing, succession planning, raising capital, growth, social goals.
Investors	return on investment, income.



In Software Engineering (SWEBOK)

- Stakeholders should provide support, information, and feedback at all stages of the software life cycle process.
- For example, during the early stages, it is critical to identify all stakeholders and discover how the product will affect them.
- So that sufficient definition of the stakeholder requirements can be properly and completely captured.
- It is vital to maintain open and productive communication with stakeholders for the duration of the software product's lifetime.



OKR

- John Doerr introduced Google to OKR, has a formula for setting goals:

I will _____ as measured by _____.

- OKR:

I will (Objective) as measured by (this set of Key Results).



OKR example (1)

- **Objective:** Create an Awesome Customer Experience
- **Key Results:**
 - Improve Net Promoter Score from X to Y.
 - Increase Repurchase Rate from X to Y.
 - Maintain Customer Acquisition cost under Y.



OKR example (2)

- **Objective:** Delight our customers
- **Key Results:**
 - Reduce revenue churn (cancellation) from $X\%$ to $Y\%$.
 - Increase Net Promoter Score from X to Y .
 - Improve average weekly visits per active user from X to Y .
 - Increase non-paid (organic) traffic to from X to Y .
 - Improve engagement (users that complete a full profile) from X to Y .



OKR example (3)

- **Objective :** Refactor our old user management module
- **Key Results :**
 - Survey 5 external API users regarding issues with our authentication
 - Discuss the user management code usage with 5 engineers having used it in production
 - Rewrite and launch new version of our user management module
 - Rewrite the API user authentication for new version



OKR example (4)

- **Objective:** Improve our testing procedures
- **Key Results :**
 - Implement test-driven development in 3 new development teams
 - Increase unit test coverage to 75% of code
 - Conduct a security assessment of our codebase using automated tools
 - Make sure satisfaction score of product management to the testing team is at least 7.5
 - Discover at least 100 bugs and open issues in old code not reviewed in 6 months

5 why's

- **Five whys** (or **5 whys**) is an iterative interrogative technique used to explore the cause-and-effect relationships underlying a particular problem.
- The primary goal of the technique is to determine the root cause of a defect or problem by repeating the question "Why?".
- Imagine „Peeling the onion“





SWOT

Strengths

- What do we do well?
- What have our customers or partners told us they like about us?
- In what areas do we outpace our competitors?
- What's unique about our business, products, or services?
- What assets do we own?
(Intellectual property, proprietary technology, capitol)

Weaknesses

- What can we improve?
- What are our customers or partners dissatisfied with?
- Where do we fall behind our competitors?
- Where are we lacking in knowledge or resources?

Opportunities

- What emerging trends can we take advantage of?
- Which of our strengths might be valuable to potential partners?
- What adjacent markets might we tap into?
- Are there geographic locations with less competition?

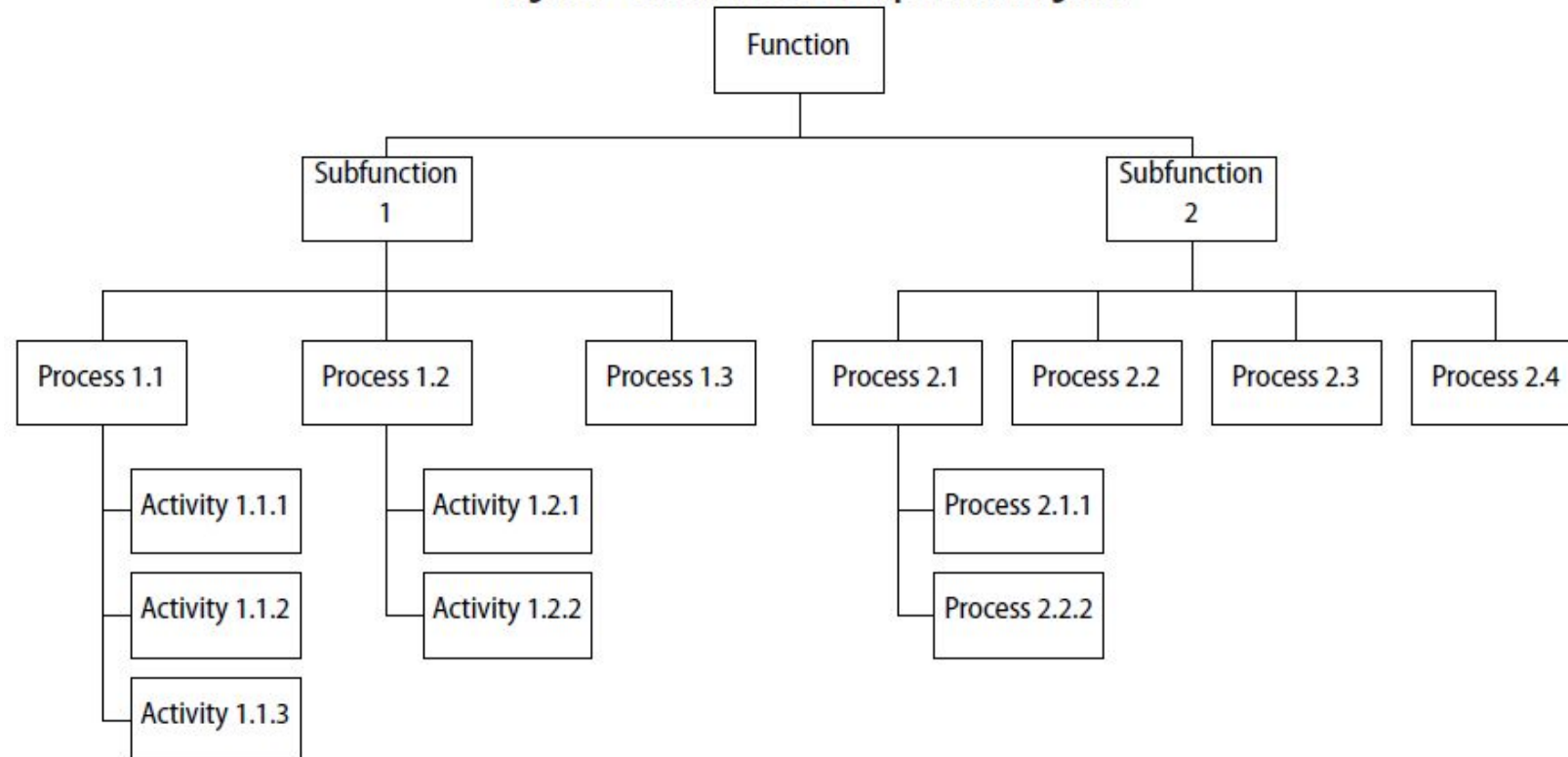
Threats

- What is our competition doing?
- How could our weaknesses leave us vulnerable?
- What market trends are we unprepared for?
- What economic or political issues could impact our business?

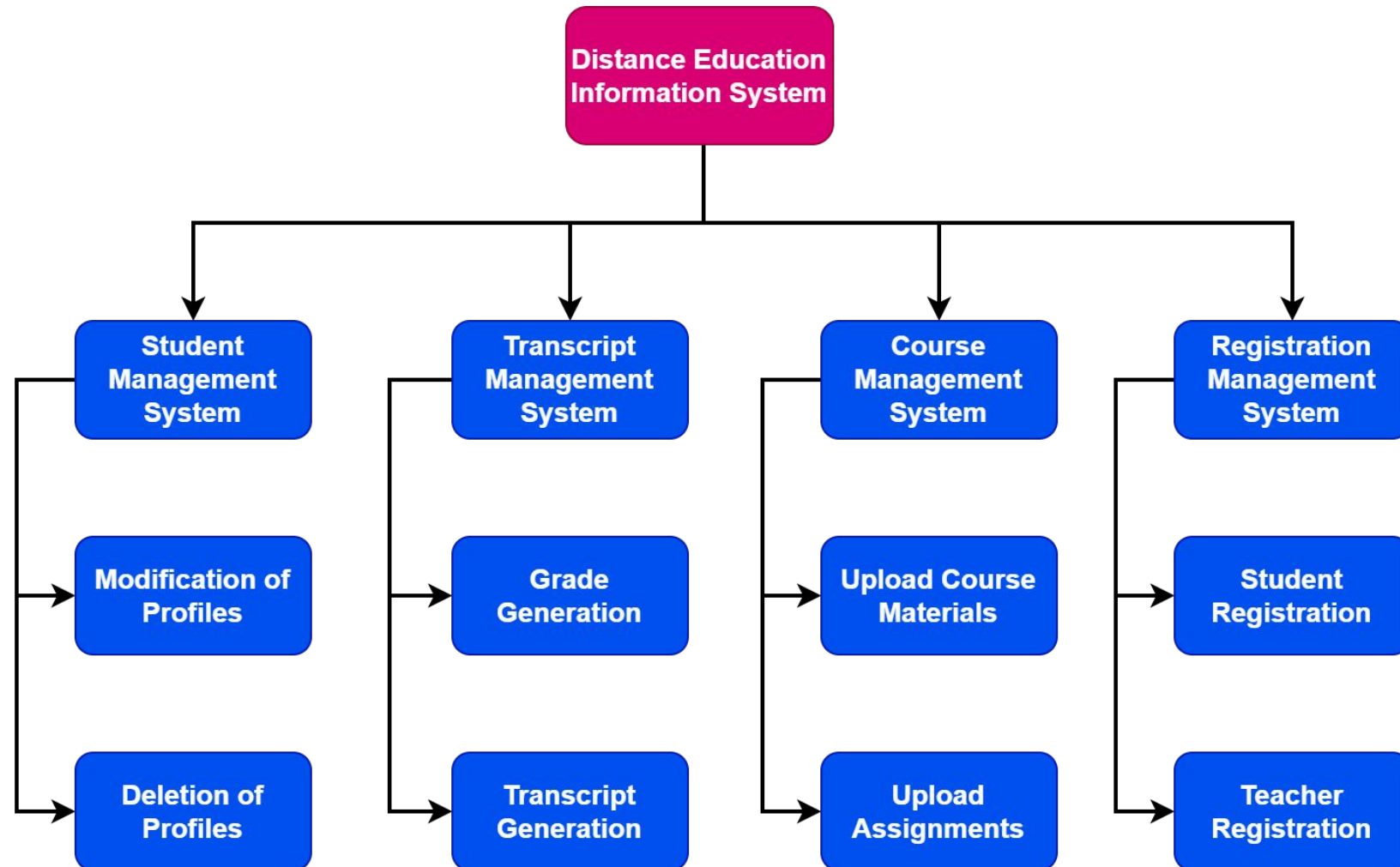
Functional decomposition (1)

- To understand the scope of work and to break the solution scope into smaller work products or deliverables

Figure 9–6: Functional Decomposition Diagram



Functional decomposition (2)





Software system modification and maintenance

- **Software maintenance** in software engineering is the modification of a software product after delivery to correct faults, to improve performance or other attributes.
- A common perception of maintenance is that it is bug-fixing.
- ~80% of maintenance effort is used for non-corrective actions.
- **Code refactoring** is taken to reduce the complexity, ignoring the type of maintenance.



Maintenance is done to...

- correct faults;
- improve the design;
- implement enhancements;
- interface with other software;
- adapt programs so that different hardware, software, system features, and telecommunications facilities can be used;
- migrate legacy software;
- retire software.





Code refactoring

- process of restructuring existing computer code—changing the factoring—without changing its external behavior.
- intended to improve the design, structure, and/or implementation of the software (its non-functional attributes), while preserving its functionality.
- Largely – to improve maintainability and extensibility.



[...] Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. Code for readability.

John F. Woods



Maintainability

The ease with which a product can be maintained in order to:

- correct defects or their cause,
- repair or replace faulty or worn-out components without having to replace still working parts,
- prevent unexpected working conditions,
- maximize a product's useful life,
- maximize efficiency, reliability, and safety,
- meet new requirements,
- make future maintenance easier, or
- cope with a changed environment.



Literature for reading

- *IEEE – SWEBOK Guide V3*
 - *Chapter 4. Software construction.*
 - *Chapter 1. Software requirements.*
- A Guide to the Business Analysis Body of Knowledge® (BABOK® Guide) Version 2.0
- <https://www.businessmodelsinc.com/about-bmi/tools/business-model-canvas/>



Further reading (1)

- <http://www.atlassian.com/blog/jira-align/scaled-agile-okrs/>
- <https://hbr.org/2012/02/the-5-whys.html>
- <https://www.businessmodelsinc.com/about-bmi/tools/business-model-canvas/>
- <https://www.toolshero.com/strategy/most-analysis/>
- <http://inmyownterms.com/writing-terminology-project-goals/>
- [https://en.wikipedia.org/wiki/Stakeholder \(corporate\)](https://en.wikipedia.org/wiki/Stakeholder_(corporate))
- <https://felipecastro.com/en/okr/what-is-okr/>



Further reading (2)

- <https://www.quora.com/What-are-some-examples-of-good-OK-Rs-for-software-engineers-engineering-managers-and-individual-contributors>
- https://en.wikipedia.org/wiki/Software_maintenance
- https://en.wikipedia.org/wiki/Code_refactoring
- <https://en.wikipedia.org/wiki/Maintainability>



Next time: C#-specific and OOP-specific properties

- Exceptional C# properties.
- Delegates, anonymous types, lambda expressions.
- Events.
- Exceptions and their handling.
- Reflection, dynamic typing.