(Sweebok v3 **INTERSECCIÓN** Sweebok V4 **INTERSECCIÓN** ACM-IEEE 2014 **) MÁS** ACM-IEEE 2020

Autores:
Bonilla Pech Russel Adrian
Ortiz Chay Jesus Mateo
Pacab Canul Rodrigo Joaquín
Rodriguez Couoh Orlando Isaías

Análisis de similitudes de Swebok 3, 4 y Currícula de ACM-IEEE 2014

A través de un análisis de los tres documentos, enfocados en el capítulo o sección donde se abordan los temas que corresponden al Diseño del Software, Swebok 3, 4 y la currícula ACM-IEEE del 2014, se halló los siguientes tópicos los cuales están presentes en los tres archivos:

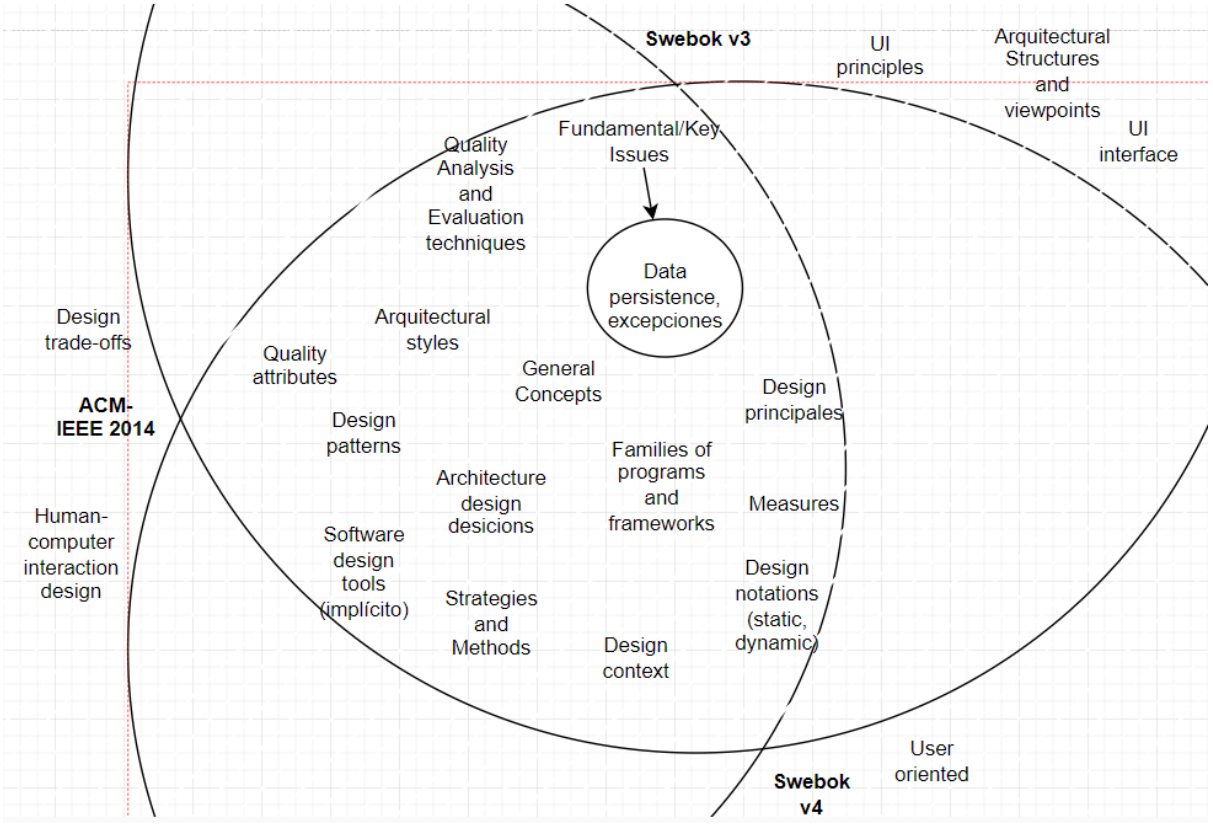


Imagen 1. Diagrama de Venn de los tópicos comunes entre los tres documentos.

La razón por la cual se incluyó el archivo ACM-IEEE 2014, es por el hecho de tener una currícula basada en conocimiento, a comparación de la Computing Curricula 2020, por la misma ACM-IEEE, la cual está basada en competencias como veremos más adelante. A continuación se desglosan los tópicos comunes encontrados justificando las similitudes/diferencias.

General Concepts
Nada diferente en este aspecto, los conceptos generales, pero con otro nombre, como puede ser *Design thinking* o *General Design Concepts*

Design Context (Context of Software design**)**
En los tres documentos, se encontró que hacían referencia a la ubicación del diseño, haciendo mención a los ciclos de vida del software.

Software Design Principles
Se encontró los mismos principios, mismos que están estrechamente relacionados con los conceptos del paradigma de programación orientado a objetos (encapsulamiento, interfaz-implementación, modularización, etc.).

Fundamental/Key Issues in Software Design
Se señalan *Data persistence* y *Exceptions* como las más comunes en los tres documentos.

Architectural Styles
Se encontró contenido similar en los tres documentos.

Design Patterns
En los tres documentos hacen referencia a los tres tipos de patrones de diseño: creacional, estructural, de comportamiento, algunos patrones de diseño son también patrones arquitectónicos.

Architecture Design Decisions
Se encontró contenido similar en ACM-IEEE 2014 y Swebok 3, sin embargo, en Swebok 4, este tiene el nombre de *Design Rationale* que habla de la toma de decisiones importantes.

Families of Programs and Frameworks
Se encontró contenido similar en ACM-IEEE 2014 y Swebok 3, sin embargo, en Swebok 4 este tiene nombre de *Detailed Design.*

Quality Attributes
Se encontró contenido similar en los tres documentos.

Quality Analysis and Evaluation Techniques
Se encontró contenido similar en los tres documentos.

Measures
Se encontró contenido similar en los tres documentos.

Software Design Notations

Structural Descriptions (Static View)

Behavioral Descriptions (Dynamic View)

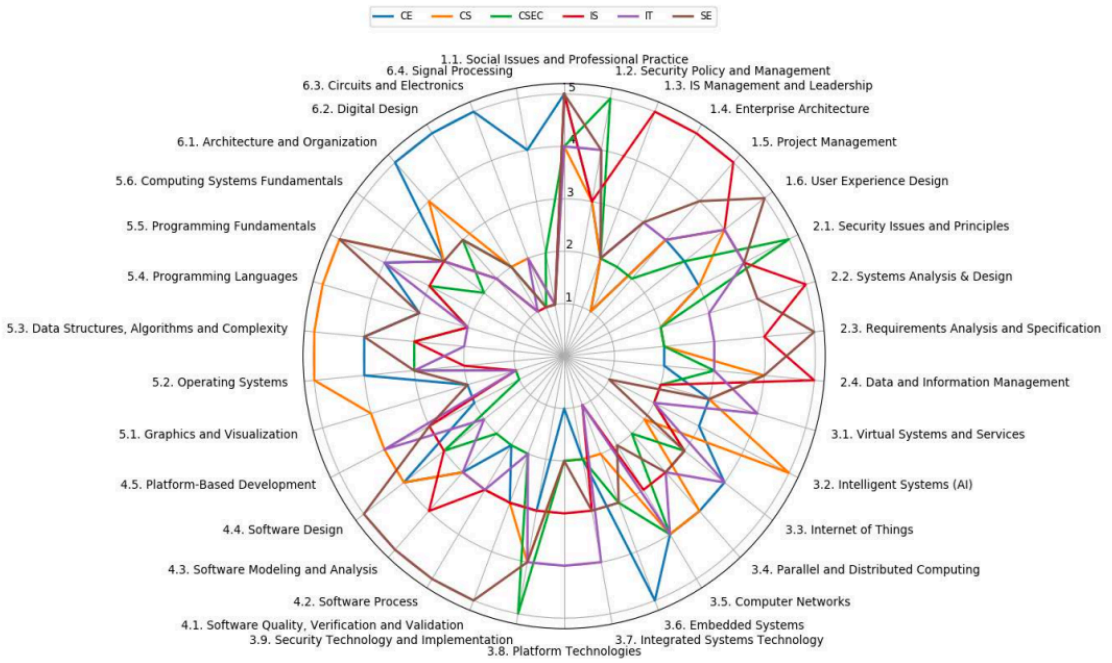
Se encontró contenido similar en Swebok 3 y 4, sin embargo en la currícula de ACM-IEEE 2014, no se específica qué *Notations* son descripciones estáticas o descripciones dinámicas, sin embargo se deduce.

Strategies and Methods
Se encontró contenido similar en los tres documentos. Nótese que en Swebok 4, se encuentra *User Oriented*.

Software Design Tools
A pesar de solamente encontrarse explícitamente en Swebok 3, incluso únicamente mencionando superficialmente, en los tres documentos se hace mención a lo largo de todo el documento sobre el uso de herramientas.

Nota: algunos de los tópicos que se encuentran en la lista, no están precisamente en el mismo apartado, sin embargo se encuentran en otro, esto se debe a una reorganización en el temario de versión a versión de Swebok.

En cuanto al documento **Computing Curricula 2020** podemos observar que los trazos de color café hacen referencia a la Ingeniería de Software (Software engineering) y uno de las áreas de conocimiento más relevantes según la gráfica es el Diseño de Software, de ahí la relevancia de su estudio.



Además en este documento se puede notar que cuando se habla de las competencias que un ingeniero de software debería tener en cuanto al diseño de software a pesar de que se mencionan habilidades técnicas, se hace un énfasis en habilidades no técnicas (soft-skills):

1. Presenta a los tomadores de decisiones empresariales requisitos arquitectónicamente significativos a partir de un documento de especificación de requisitos de software.
2. Evaluar y comparar ventajas y desventajas de posibilidades de diseño alternativas para satisfacer requisitos funcionales y no funcionales y escribir una breve propuesta que resuma las conclusiones clave para un cliente.
3. Producir un diseño de alto nivel de subsistemas específicos que sea presentable para una audiencia no informática considerando patrones arquitectónicos y de diseño.
4. Producir diseños detallados para un cliente para diseños de alto nivel de subsistemas específicos utilizando principios de diseño y aspectos transversales para satisfacer requisitos funcionales y no funcionales.
5. Evaluar las pruebas de software considerando los atributos de calidad en el diseño de subsistemas y módulos para un desarrollador/fabricante.
6. Crear documentos de diseño de software que permitan comunicar de manera efectiva a los clientes el diseño de software, así como a los analistas, implementadores, planificadores de pruebas, etc.

Aquí podemos notar palabras clave que hacen referencia a soft skills tales como comunicación efectiva, análisis, escritura, etc. habilidades muy relevantes para poder aprovechar las habilidades técnicas que nos proporcionará Diseño de Software.