Revision Report (SDF)

Steering Committee Comments.

1. Instead of “first year” in intro – perhaps use “introductory expertise”
2. Programming style – use some standard style guide (maybe encourage its use through the program)
3. Add something on security / secure coding
4. Add something on Exception handling
5. In Data Structures, add: Bulk/aggregate operations (like map/reduce)
6. In Data Structures we have “write programs that use each of the abstract data types / data structures provided in the language like arrays, ...” – this can be too large. Better to enumerate the fundamental data structures, and avoid the use of “etc.”
7. Shared concepts with SE – list them
8. Disposition: Use one word descriptions like self-directed, inventive, inquisitive,.. (also, last one should be “adapt to different (rather than “new”) tools...”). (Maybe use the illustrative list given in CC20202: Proactive, Self-directed, Passionate, Purpose-driven, Professional, Responsible, Adaptable, Collaborative, Responsive, Meticulous.)
9. Disposition of using online resources – can encourage plagiarism, and there are risks in letting students without experience use un-curated content (they may learn bad practices)

How Incorporated/Responses:

1. For now leaving "first year" as it is widely understood (CS1 and CS2 is a common term used for first year courses, which will cover SDF)
2. This requires program level mandate - the overall SC may consider it
3. We discussed this - it is very hard to add anything about security early, as students still do not know about how programs are executed, the runtime system, etc..
4. Incorporated - a topic and a learning outcome on Exception handling has been added.
5. Bulk data structures - we feel that SDF should focus on basic data structures (some of which are available in some languages). The bulk data structures should be covered in some other KA.
6. Done
7. Done
8. Done (the example terms provided for dispositions have been mostly used).
9. The disposition has been repositioned and reworded: Self-Directed. Seeking out solutions to the issues on their own …

Reviewer 1

Feedback comment: Comment: Basic programming need not include classes. Unless we are using a Java-like language, where the use of classes is necessary to write a working program, class can be avoided. modules/libraries are included later in this KU
**How incorporated:** The topic wording changed suitably: Key modularity constructs such as functions (and methods and classes, if supported in the language) and related concepts like parameter passing, scope, abstraction, data encapsulation, etc.

**Why not incorporated:**

**Date considered:**

Feedback comment: Dealing with compile time and runtime errors (Comment: I think we need not explicitly mention this point here. It belongs to KU-4 SDF/Development Methods, along with the use of general-purpose IDE)

**How incorporated:** In SC, a suggestion was made to include Exception Handling. Combining the SC comment and above, this has been changed to: Dealing with runtime errors in programs (exception handling)

A LO has been added on Exception Handling.

**Why not incorporated:**

**Date considered:**

Feedback comment: Measure the performance of a program (e.g. to assess how performance changes with scale, to compare alternative implementations using alternative data structures, …). (Comment: Emphasis on alternative data structures, as “alternative implementations” could also mean changing the algorithm)

**How incorporated:** Changed to: Measure the performance of a program (e.g. to assess how performance changes with scale, alternative data structures, …).

**Why not incorporated:**
Feedback comment: Basic Code Coverage (to understand whether a sufficient number of tests are added or more tests are required)

- Develop tests to improve coverage of the program code.

How incorporated:

Why not incorporated: We believe coverage should not be included in core - it is not foundational to programming and even some of us in the committee do not cover it in our courses.

Reviewer 2 Comments.

Feedback comment: ... to design and analyze algorithms, ...: This is not emphasized in the topics or examples below, and I recommend shifting it to another KA. There is plenty to be done learning how to use appropriate tools, language-provided structures, and processes.

How incorporated:

(i) The sentiment is changed to: ...they must be able to select and use appropriate data structures and algorithms

(ii) It is listed as a shared concept with the KA - Algorithms (AL)

(iii) In SDF, the knowledge unit "Algorithms and Design" has been renamed as "Algorithms" and the topics and LOs are suitably aligned to this.

Why not incorporated:

Date considered:
Feedback comment: In line with other suggestions in this document, I’d reduce the Algorithms/Design hours – moving it to another KA – and shift the weight to emphasize Development Methods.

How incorporated: We agree. Have cross listed these with the AL KA. Also, in the section "Possible Course Packaging", have clarified that items related to Algorithms may be covered in CS2 and not in CS1

Why not incorporated:

Date considered:

Feedback comment: I feel this should be emphasized in the topics list. Initially, when I read it, I thought it did not reflect the breadth of “software DEVELOPMENT”. With the exception of the final two bullets, it reads as a list of language structures. Example topics: Tools and methods for isolating and identifying the root cause of a bug, Tools and processes for developing and evaluating a test suite, Use of supporting tools such as IDEs and static analyses

How incorporated: SDF has 4 different KUs, and the first one really deals with programming. The "software development" aspects are included in the "Software Development Practices" KU (which is the new name replacing the older "Development Methods"). This KU has tools/IDE, debugging, etc there.

The KU "Fundamental Programming Concepts" has been renamed as "Fundamental Programming Concepts and Practices" to emphasize the actual hands-on programming aspects, which is what the LOs focus on.

Why not incorporated:

Date considered:

Feedback comment: …. uses APIs to get data (e.g., from the web,…..: Neat outcome, but I don’t that it is necessary for the APIs to be used to get data. I’d combine 4/5 to be “Write a program that uses language-provided libraries and frameworks and APIs (where applicable).”

How incorporated: There is no disagreement about having this LO. We feel it is best to leave it as separate and not combine with other LOs.
Why not incorporated:

Date considered:

Feedback comment: .....Trace the flow of control during the execution of a program.

... : I love that this is included, and I would shift it earlier in the list, so it does not appear to be an afterthought. I might also add a suggestion that tracing be used to find faults. For example, “Trace the flow of control during the execution of programs (including both correct and incorrect programs).”

How incorporated: Added

Why not incorporated:

Date considered:

Feedback comment: .....Common algorithms like: Sorting, Searching, Tree traversal, Graph traversal, etc..... Assessing the time/space efficiency of an algorithms through measurements

Suggest reducing weight and moving to another KA. They are not, IMO, aligned with the software development topics, although they do echo and run in parallel with some of the data structure topics.

How incorporated: We feel that some familiarization with algorithms should remain in SDF. As suggested, we have added these LOs as cross cutting concerns with Algorithms (AL), and have also explained that it need not be done in CS1. We have also renamed and repositioned the knowledge unit "Algorithms and Design" which is now just "Algorithms".

Why not incorporated:

Date considered:

Feedback comment: ..... Basic Source Code control.....: Tempted to cut for the first year.
How incorporated: We agree. This has been removed.

Why not incorporated:

Date considered:

Feedback comment: R2 - suggestions on Dispositions.

How incorporated: Most have been accepted. (Then we have added more.)

Why not incorporated:

Date considered: