

Attachment 1. Activities complexity calculation

To calculate complexity, the Functional Cognitive Complexity method is used (Formula 7). This method is applied to flowcharts to assess implementation complexity and to specific parts of the flowchart (marked in green) to evaluate modification complexity. According to the cognitive functional complexity algorithm [50], the calculation consists of several stages as follows:

- Create flowcharts for the algorithm.
- Calculate the cognitive functional complexity for the implementation of the activity.
- Calculate the cognitive functional complexity for the modification of the activity.

A.1.1 Classical CQRS Command process activities

Create a Command.

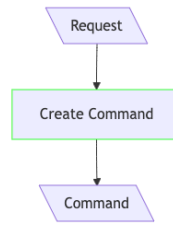


Figure 8: Classical CQRS Create a Command Flowchart

Implementation complexity:

$$BCS_1(sequence): \quad W_1 = 1$$

$$W_I = 1$$

$$S_I = (1 + 1) * 1 = 2$$

Modification complexity:

$$BCS_1(sequence): \quad W_1 = 1$$

$$W_M = 1$$

$$S_M = (1 + 1) * 1 = 2$$

Validate a Command

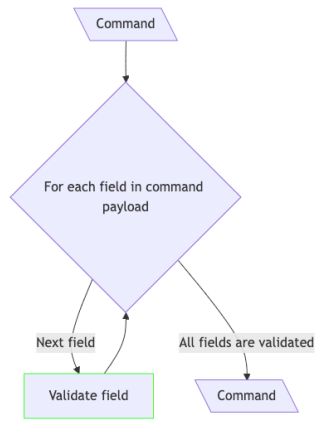


Figure 9: Classical CQRS Validate a Command Flowchart

Implementation complexity:

$$BCS_1(\text{iteration}): \quad W_1 = 3$$

$$BCS_2(\text{sequence}): \quad W_2 = 1$$

$$W_I = 3 + 1 = 4$$

$$S_I = (1 + 1) * 4 = 8$$

Modification complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$W_M = 1$$

$$S_M = (1 + 1) * 1 = 2$$

Route a Command

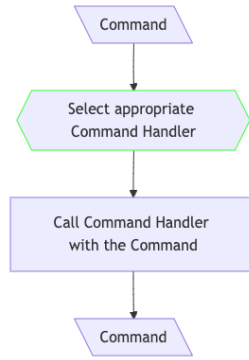


Figure 10: Classical CQRS Route a Command Flowchart

Implementation complexity:

$$BCS_1(\text{branch}): \quad W_1 = 2$$

$$BCS_2(\text{function call}): \quad W_2 = 2$$

$$W_I = 2 + 2 = 4$$

$$S_I = (1 + 1) * 4 = 8$$

Modification complexity:

$$BCS_1(\text{branch}): \quad W_1 = 2$$

$$W_M = 2$$

$$S_M = (1 + 1) * 2 = 4$$

Fetch an Aggregate

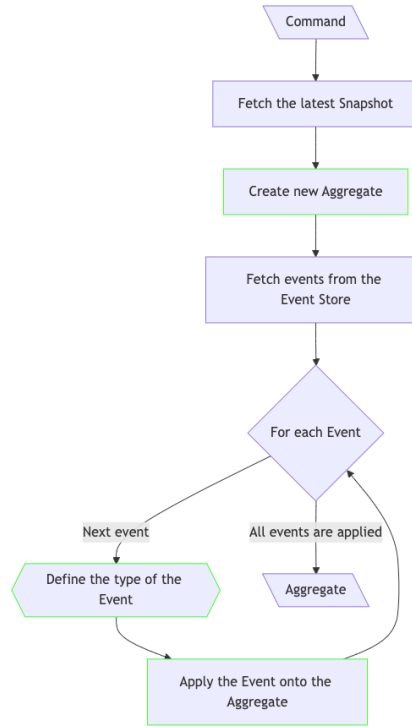


Figure 11: Classical CQRS Fetch an Aggregate Flowchart

Implementation complexity:

$$BCS_1(\text{function call}): \quad W_1 = 2$$

$$BCS_2(\text{sequence}): \quad W_2 = 1$$

$$BCS_3(\text{function call}): \quad W_3 = 2$$

$$BCS_4(\text{iteration}): \quad W_4 = 3$$

$$BCS_5(\text{branch}): \quad W_5 = 2$$

$$BCS_6(\text{sequence}): \quad W_6 = 1$$

$$W_I = 2 + 1 + 2 + 3 + 2 + 1 = 11$$

$$S_I = (1 + 1) * 11 = 22$$

Modification complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$BCS_2(\text{branch}): \quad W_2 = 2$$

$$BCS_3(\text{sequence}): \quad W_3 = 1$$

$$W_M = 1 + 2 + 1 = 4$$

$$S_M = (1 + 1) * 4 = 8$$

Update an Aggregate's state

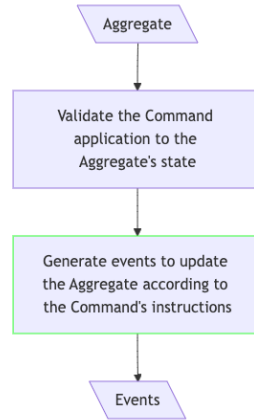


Figure 12: Classical CQRS Update an Aggregate's state Flowchart

Implementation complexity:

$$BCS_1(\text{function call}): \quad W_1 = 2$$

$$BCS_2(\text{function call}): \quad W_2 = 2$$

$$W_I = 2 + 2 = 4$$

$$S_I = (1 + 1) * 4 = 8$$

Modification complexity:

$$BCS_1(\text{function call}): \quad W_1 = 2$$

$$W_M = 2$$

$$S_M = (1 + 1) * 2 = 4$$

Save an Aggregate

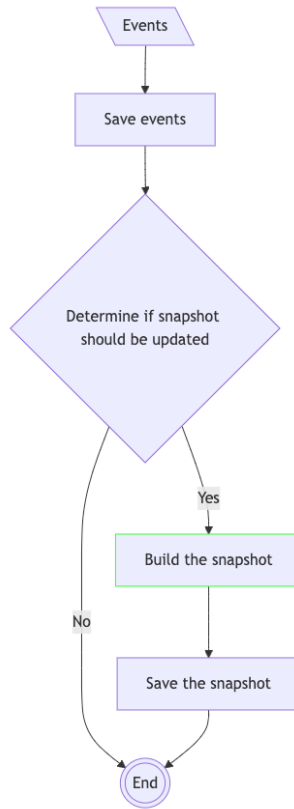


Figure 13: Classical CQRS Save an Aggregate Flowchart

Implementation complexity:

$$BCS_1(\text{function call}): \quad W_1 = 2$$

$$BCS_2(\text{branch}): \quad W_2 = 2$$

$$BCS_3(\text{sequence}): \quad W_3 = 1$$

$$BCS_4(\text{function call}): \quad W_4 = 2$$

$$W_I = 2 + 2 + 1 + 2 = 7$$

$$S_I = (1 + 0) * 7 = 7$$

Modification complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$W_M = 1$$

$$S_M = (1 + 0) * 1 = 1$$

Dispatch events

Each time we add a new process to the dispatcher, we do not implement a loop; instead, a new event condition is simply added to the existing list. Therefore, in complexity calculation, the *iteration* block is considered a *branch* in this case.

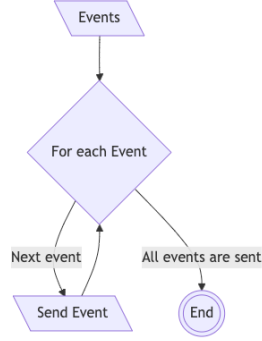


Figure 14: Classical CQRS Dispatch events Flowchart

Implementation complexity:

$$BCS_1(\text{branch}): \quad W_1 = 2$$

$$BCS_2(\text{sequence}): \quad W_2 = 1$$

$$W_I = 2 + 1 = 3$$

$$S_I = (1 + 1) * 3 = 6$$

Modification complexity:

$$W_M = 0$$

$$S_M = (1 + 1) * 0 = 0$$

Route an Event

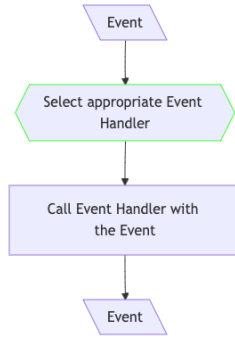


Figure 15: Classical CQRS Route an Event Flowchart

Implementation complexity:

$$BCS_1(\text{branch}): \quad W_1 = 2$$

$$BCS_2(\text{function call}): \quad W_2 = 2$$

$$W_I = 2 + 2 = 4$$

$$S_I = (1 + 1) * 4 = 8$$

Modification complexity:

$$BCS_1(\text{branch}): \quad W_1 = 2$$

$$W_M = 1$$

$$S_M = (1 + 1) * 2 = 4$$

Handle an Event (Update projection)

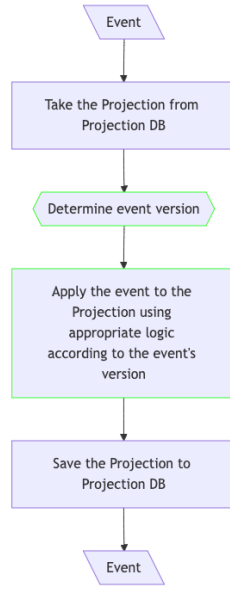


Figure 16: Classical CQRS Handle an Event (Update projection) Flowchart

Implementation complexity:

$$BCS_1(\text{function call}): \quad W_1 = 2$$

$$BCS_2(\text{branch}): \quad W_2 = 2$$

$$BCS_3(\text{sequence}): \quad W_3 = 1$$

$$BCS_4(\text{function call}): \quad W_4 = 2$$

$$W_I = 2 + 2 + 1 + 2 = 7$$

$$S_I = (1 + 1) * 7 = 14$$

Modification complexity:

$$BCS_1(\text{branch}): \quad W_1 = 2$$

$$BCS_2(\text{sequence}): \quad W_2 = 1$$

$$W_M = 2 + 1 = 3$$

$$S_M = (1 + 1) * 3 = 6$$

Notify clients

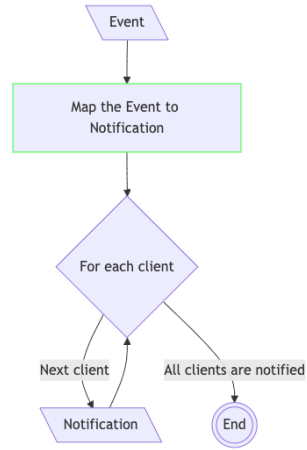


Figure 17: Classical CQRS Notify clients Flowchart

Implementation complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$BCS_2(\text{iteration}): \quad W_2 = 3$$

$$BCS_3(\text{function call}): \quad W_3 = 2$$

$$W_I = 1 + 3 + 2 = 6$$

$$S_I = (1 + 1) * 6 = 12$$

Modification complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$W_M = 1$$

$$S_M = (1 + 1) * 1 = 2$$

A.1.2 mCQRS Command process activities (which differ from the classic CQRS)

Fetch an Aggregate

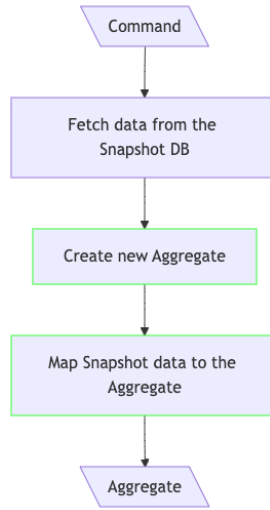


Figure 18: mCQRS Fetch an Aggregate Flowchart

Implementation complexity:

$$BCS_1(\text{function call}): \quad W_1 = 2$$

$$BCS_2(\text{sequence}): \quad W_2 = 1$$

$$BCS_3(\text{sequence}): \quad W_3 = 1$$

$$W_I = 2 + 1 + 1 = 4$$

$$S_I = (1 + 1) * 4 = 8$$

Modification complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$BCS_2(\text{sequence}): \quad W_2 = 1$$

$$W_M = 1 + 1 = 2$$

$$S_M = (1 + 1) * 2 = 4$$

Apply events onto an Aggregate

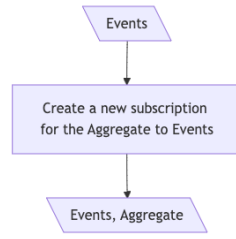


Figure 19: mCQRS Apply events onto an Aggregate Flowchart

Implementation complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$W_I = 1$$

$$S_I = (1 + 2) * 1 = 3$$

Modification complexity:

$$W_M = 0$$

$$S_M = (1 + 2) * 0 = 0$$

Save an Aggregate

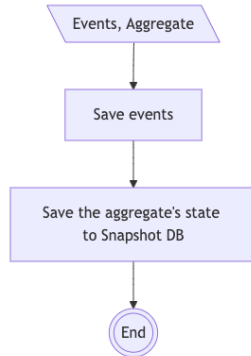


Figure 20: mCQRS Save an Aggregate Flowchart

Implementation complexity:

$$BCS_1(\text{function call}): \quad W_1 = 2$$

$$BCS_2(\text{function call}): \quad W_2 = 2$$

$$W_I = 2 + 2 = 4$$

$$S_I = (1 + 1) * 4 = 8$$

Modification complexity:

$$W_M = 0$$

$$S_M = (1 + 1) * 0 = 0$$

Handle an Event (Update projection)

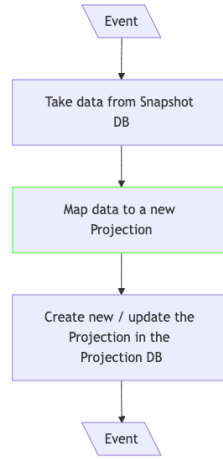


Figure 21: mCQRS Handle an Event (Update projection) Flowchart

Implementation complexity:

$$BCS_1(\text{function call}): \quad W_1 = 2$$

$$BCS_2(\text{sequence}): \quad W_2 = 1$$

$$BCS_3(\text{function call}): \quad W_3 = 2$$

$$W_I = 2 + 1 + 2 = 5$$

$$S_I = (1 + 1) * 5 = 10$$

Modification complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$W_M = 1$$

$$S_M = (1 + 1) * 1 = 2$$

A.1.3 Query process activities

Create a Query

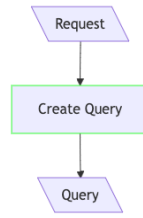


Figure 22: Create a Query Flowchart

Implementation complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$W_I = 1$$

$$S_I = (1 + 1) * 1 = 2$$

Modification complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$W_M = 1$$

$$S_M = (1 + 1) * 1 = 2$$

Validate a Query

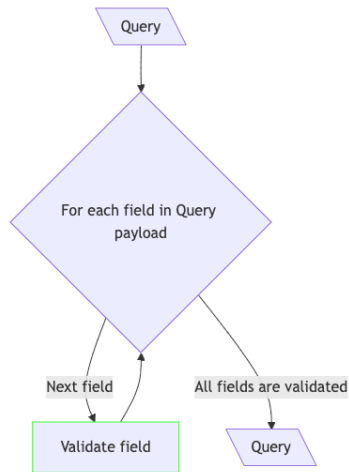


Figure 23: Validate a Query Flowchart

Implementation complexity:

$$BCS_1(\text{iteration}): \quad W_1 = 3$$

$$BCS_2(\text{sequence}): \quad W_2 = 1$$

$$W_I = 3 + 1 = 4$$

$$S_I = (1 + 1) * 4 = 8$$

Modification complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$W_M = 1$$

$$S_M = (1 + 1) * 1 = 2$$

Fetch a Projection

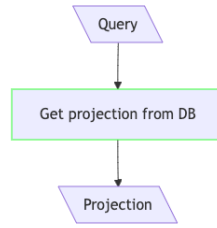


Figure 24: Fetch a Projection Flowchart

Implementation complexity:

$$BCS_1(\text{function call}): \quad W_1 = 2$$

$$W_I = 2$$

$$S_I = (1 + 1) * 2 = 4$$

Modification complexity:

$$BCS_1(\text{function call}): \quad W_1 = 2$$

$$W_M = 2$$

$$S_M = (1 + 1) * 2 = 4$$

Map a Projection to a DTO

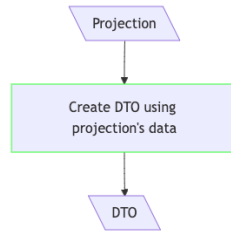


Figure 25: Map a Projection to a DTO Flowchart

Implementation complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$W_I = 1$$

$$S_I = (1 + 1) * 1 = 2$$

Modification complexity:

$$BCS_1(\text{sequence}): \quad W_1 = 1$$

$$W_M = 1$$

$$S_M = (1 + 1) * 1 = 2$$

Return a DTO

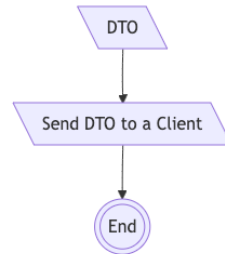


Figure 26: Return a DTO Flowchart

Implementation complexity:

$$BCS_1(\text{function call}): \quad W_1 = 2$$

$$W_I = 2$$

$$S_I = (1 + 0) * 2 = 2$$

Modification complexity:

$$W_M = 0$$

$$S_M = (1 + 0) * 0 = 0$$