Attachment 2. Transition complexity calculation

To calculate complexity, the Functional Cognitive Complexity method is applied to a portion of the flowchart (Formula 4). Given the atomicity of blocks, the complexity of removing blocks is assumed to be equal to 1. If removing a block leads to additional changes in the system, it is recommended to either restructure activities so that block removal does not introduce extra complexity or use a more precise complexity evaluation method. According to the proposed technology, Transition Complexity Calculation consists of the following steps:

- Formalizing source and target variation processes using the proposed mathematical model.
- Matching activities.
- Building flowcharts for differing activities.
- Matching sub-activities (components of the flowchart).
- Calculating the complexity of adding/removing sub-activities.
- Calculating the transition complexity of processes using Formula 5.
- Calculating the total transition complexity using Formula 6.

A.3.1 Pure CQRS to Classical CQRS

The query processing is the same across all three variations. We will examine the processes of command handling and projection rebuilding. Let's divide the processes into activities and match them accordingly (Table 9).

Table 9: Command process activities matching

Pure CQRS	Classical CQRS
Create command	Create command
Validate command	Validate command
Route command	Route command
Fetch aggregate pure CQRS	Fetch aggregate classical CQRS
Update aggregate's state	Update aggregate's state
Save aggregate pure CQRS	Save aggregate classical CQRS
Dispatch events	Dispatch events
Route events	Route events
Handle update projection event pure CQRS	Handle update projection event classical CQRS
Notify client	Notify client

Each of the matched activities (*Fetch Aggregate, Save Aggregate, Build New Projection*) is broken down into sub-activities until it reaches the level of atomic operations that either fully match or need to be added/removed.

Fetch an Aggregate

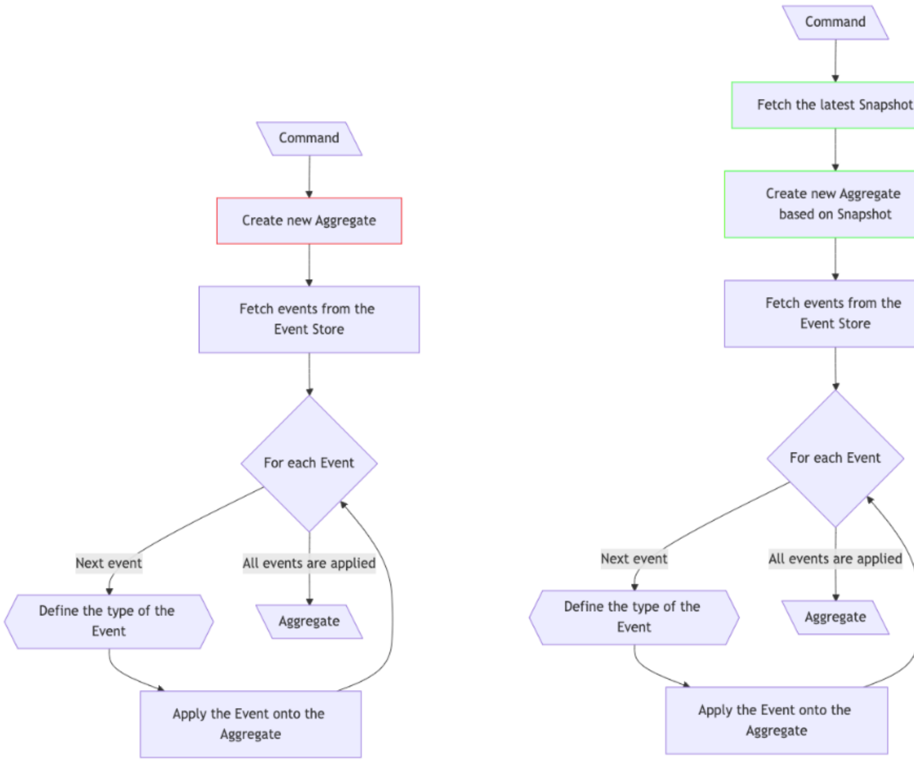

Figure 27: Pure CQRS and Classical CQRS Fetch an Aggregate Flowchart

The flowcharts in Figure 27 illustrate the steps of the Fetch Aggregate algorithm for the Pure CQRS and Classical CQRS variations. The activity of creating a new aggregate without parameters (highlighted in red), does not match any sub-activities in the target approach and will be removed during the transition. Activities marked with green frames are those that need to be implemented. Thus, the transition complexity calculation for the Fetch Aggregate activity, using the Functional Cognitive Complexity method, is as follows:

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

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

$$W_A = 2 + 1 = 3$$

$$S = S_A + S_R = (1 + 1) * 3 + 1 = 7$$

Thus, the transition complexity of an activity (S) is equal to the complexity of adding new sub-activities (S_A) and the complexity of removing sub-activities not present in the target method (S_R). In this example, the complexity of addition is calculated using the Functional Cognitive Complexity method for the relevant part of the flowchart. The complexity of removal is assumed to be equal *1*, considering the atomic nature of the blocks.

Save an Aggregate

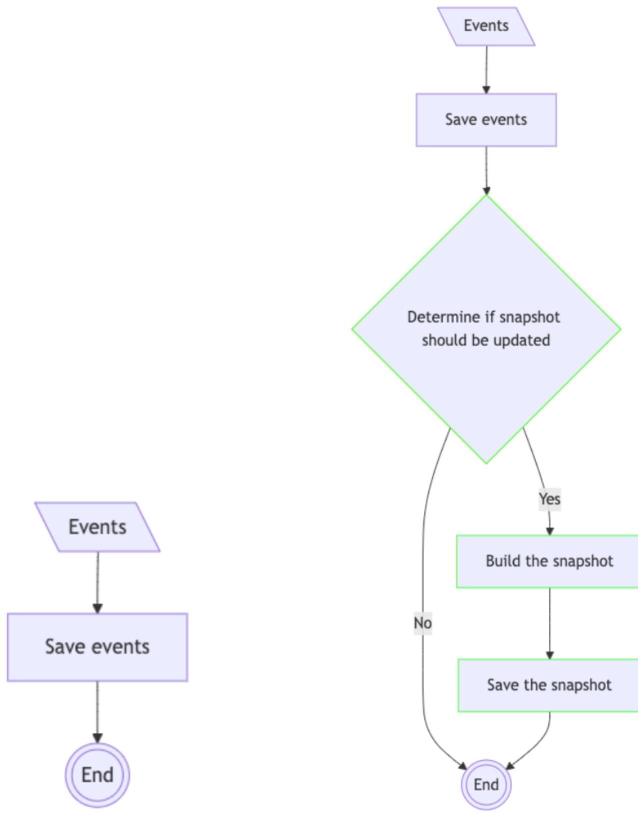

Figure 28: Pure CQRS and Classical CQRS Save an Aggregate Flowchart

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

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

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

$$W_A = 2 + 1 + 2 = 5$$

$$S = S_A = (1 + 0) * 5 = 5$$

Pure CQRS to Classical CQRS transition complexity

Using Formula 5, the transition complexity for each process is calculated, and Formula 6 is used to compute the overall transition complexity for the example IS.

$$\omega_{p_c} = 7 + 5 = 12$$

$$\omega = c_{p_c} * \omega_{p_c} = c_{p_c} * 12$$

An additional complexity factor is the introduction of a new resource (a snapshot database). This complexity is not accounted for in the current calculations.

A.3.2 Pure CQRS to mCQRS

Table 10: Command process activities matching

Pure CQRS	mCQRS
Create command	Create command
Validate command	Validate command
Route command	Route command
Fetch aggregate pure CQRS	Fetch aggregate mCQRS
Update aggregate’s state	Update aggregate’s state
	Apply events onto aggregate
Save aggregate pure CQRS	Save aggregate mCQRS
Dispatch events	Dispatch events
Route events	Route events
Handle update projection event pure CQRS	Handle update projection event mCQRS
Notify client	Notify client

Fetch an Aggregate

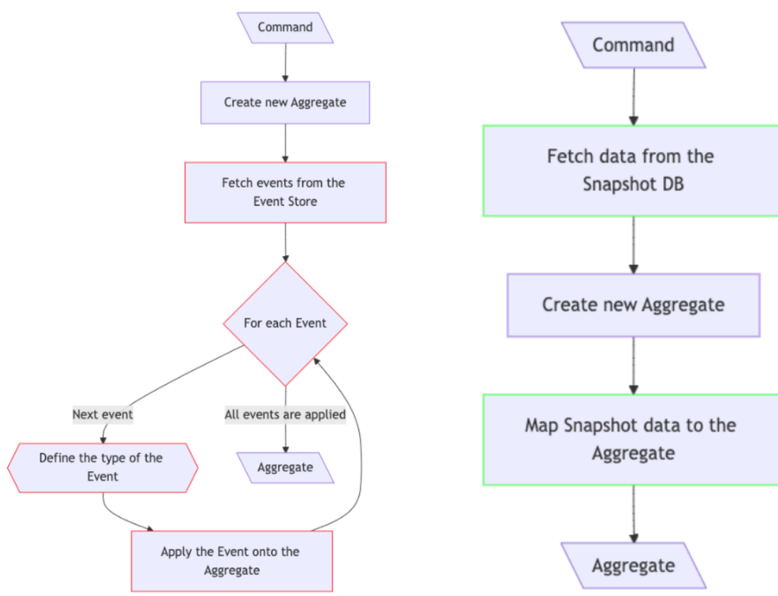

Figure 29: Pure CQRS and mCQRS Fetch an Aggregate Flowchart

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

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

$$W_A = 2 + 1 = 3$$

$$S = S_A + S_R = (1 + 1) * 3 + 4 = 10$$

Apply events onto an Aggregate

There is no Pure CQRS implementation for this activity.

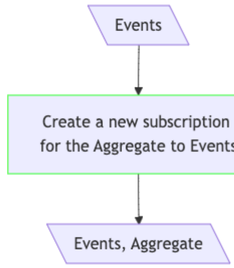

Figure 30: mCQRS Apply events onto an Aggregate Flowchart

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

$$W_A = 1$$

$$S = S_A = (1 + 2) * 1 = 3$$

Save an Aggregate

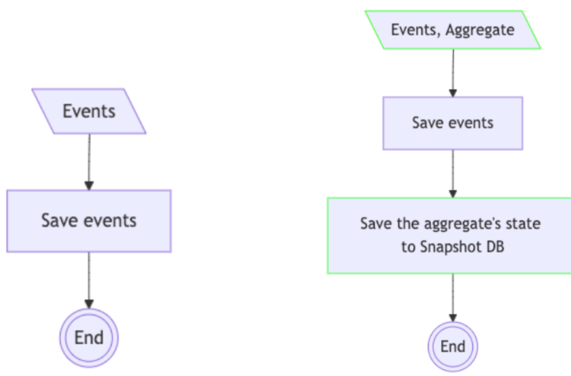

Figure 31: Pure CQRS and mCQRS Save an Aggregate Flowchart

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

$$W_A = 2$$

$$S = S_A = (2 + 0) * 2 = 4$$

Handle an Event (Update projection)

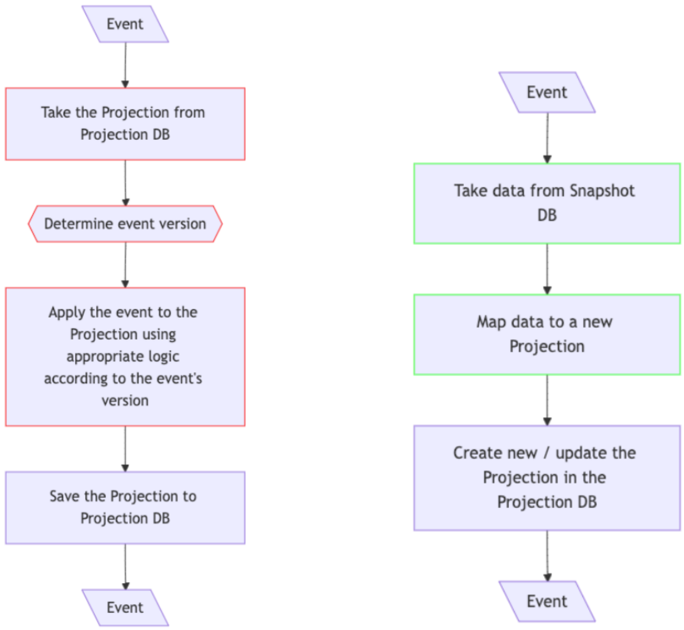

Figure 32: Pure CQRS and mCQRS Handle an Event Flowchart

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

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

$$W_A = 2 + 1 = 3$$

$$S = S_A + S_R = (1 + 1) * 3 + 3 = 9$$

Pure CQRS to mCQRS transition complexity

$$\omega_{p_c} = 10 + 3 + 4 + 9 = 26$$

$$\omega = c_{p_c} * \omega_{p_c} = c_{p_c} * 26$$

An additional complexity factor is the introduction of a new resource (a snapshot database). This complexity is not accounted for in the current calculations.

A.3.3 Classical CQRS to mCQRS

Table 11: Command process activities matching

Classical CQRS	mCQRS
Create command	Create command
Validate command	Validate command
Route command	Route command

Classical CQRS	mCQRS
Fetch aggregate classical CQRS	Fetch aggregate mCQRS
Update aggregate's state	Update aggregate's state
	Apply events onto aggregate
Save aggregate classical CQRS	Save aggregate mCQRS
Dispatch events	Dispatch events
Route events	Route events
Handle update projection event classical CQRS	Handle update projection event mCQRS
Notify client	Notify client

Fetch an Aggregate

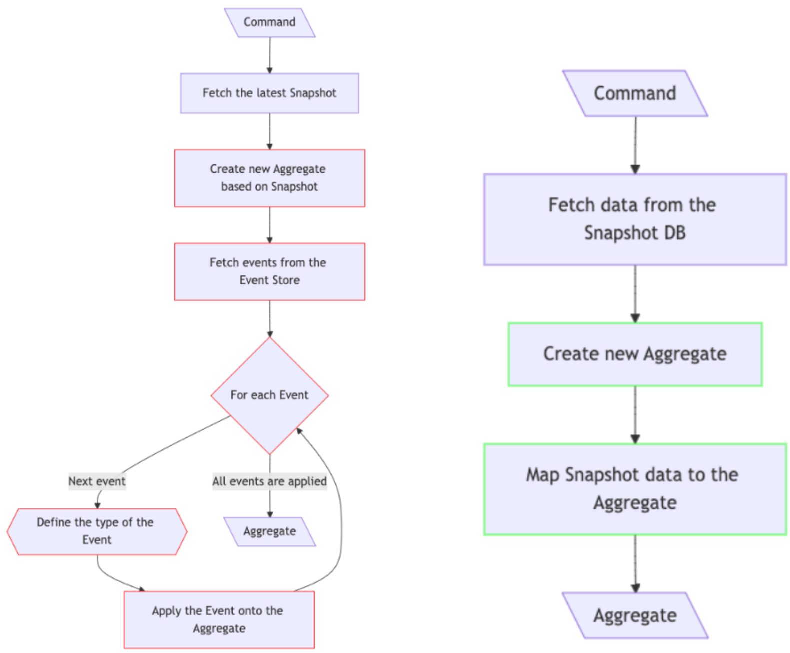

Figure 33: Classical CQRS and mCQRS Fetch an Aggregate Flowchart

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

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

$$W_A = 1 + 1 = 2$$

$$S = S_A + S_R = (1 + 1) * 2 + 5 = 9$$

Apply events onto an Aggregate

There is no Classical CQRS implementation for this activity.

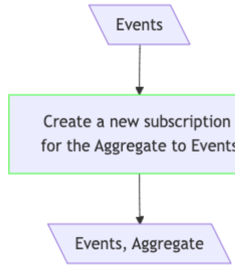

Figure 34: mCQRS Apply events onto an Aggregate Flowchart

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

$$W_A = 1$$

$$S = S_A = (1 + 2) * 1 = 3$$

Save an Aggregate

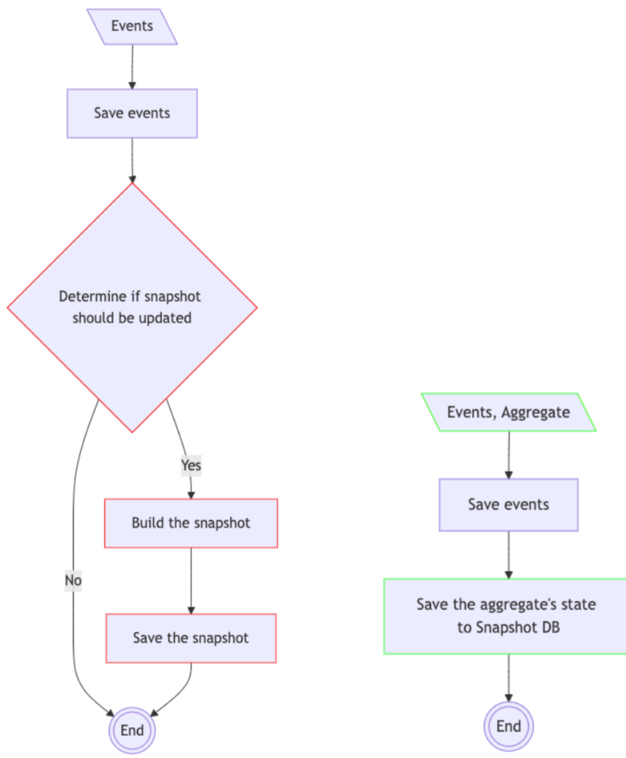

Figure 35: Classical CQRS and mCQRS Save an Aggregate Flowchart

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

$$W_A = 2$$

$$S = S_A + S_R = (2 + 0) * 2 + 3 = 7$$

Handle an Event (Update projection)

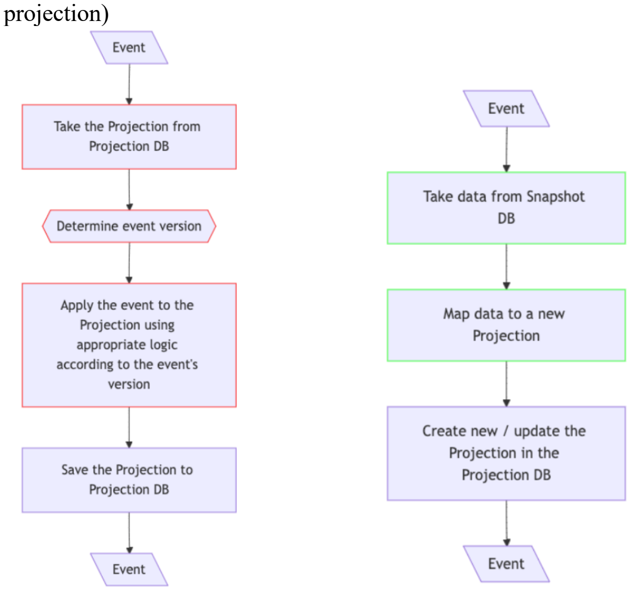

Figure 36: Classical CQRS and mCQRS Handle an Event Flowchart

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

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

$$W_A = 2 + 1 = 3$$

$$S = S_A + S_R = (1 + 1) * 3 + 3 = 9$$

Classical CQRS to mCQRS transition complexity

$$\omega_{p_c} = 9 + 3 + 7 + 9 = 28$$

$$\omega = c_{p_c} * \omega_{p_c} = c_{p_c} * 28$$

A.3.4 mCQRS to Classical CQRS

Table 12: Command process activities matching

mCQRS	Classical CQRS
Create command	Create command
Validate command	Validate command
Route command	Route command

mCQRS	Classical CQRS
Fetch aggregate mCQRS	Fetch aggregate classical CQRS
Update aggregate's state	Update aggregate's state
Apply events onto aggregate	
Save aggregate mCQRS	Save aggregate classical CQRS
Dispatch events	Dispatch events
Route events	Route events
Handle update projection event mCQRS	Handle update projection event classical CQRS
Notify client	Notify client

Fetch an Aggregate

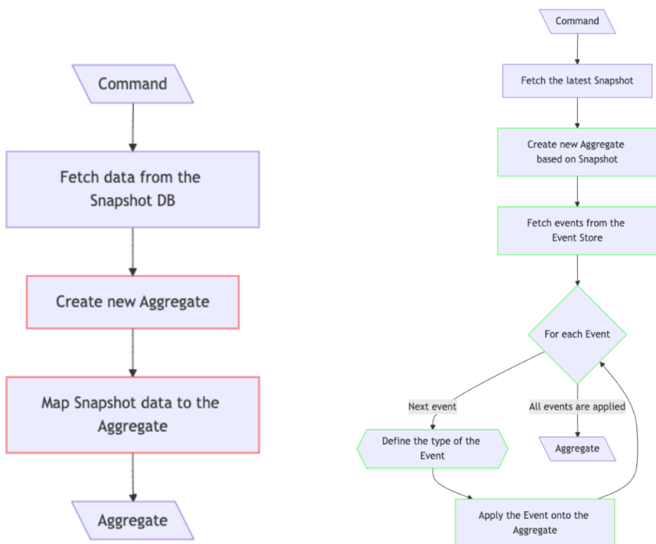

Figure 37: mCQRS and Classical CQRS Fetch an Aggregate Flowchart

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

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

$$BCS_3(\text{iteration}): \quad W_3 = 3$$

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

$$BCS_5(\text{fsequence}): \quad W_5 = 1$$

$$W_A = 1 + 2 + 3 + 2 + 1 = 9$$

$$S = S_A + S_R = (1 + 1) * 9 + 2 = 20$$

Apply events onto an Aggregate

There is no Classical CQRS implementation for this activity.

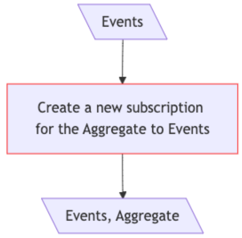

Figure 38: mCQRS Apply events onto an Aggregate Flowchart

$$S = S_R = 1$$

Save an Aggregate

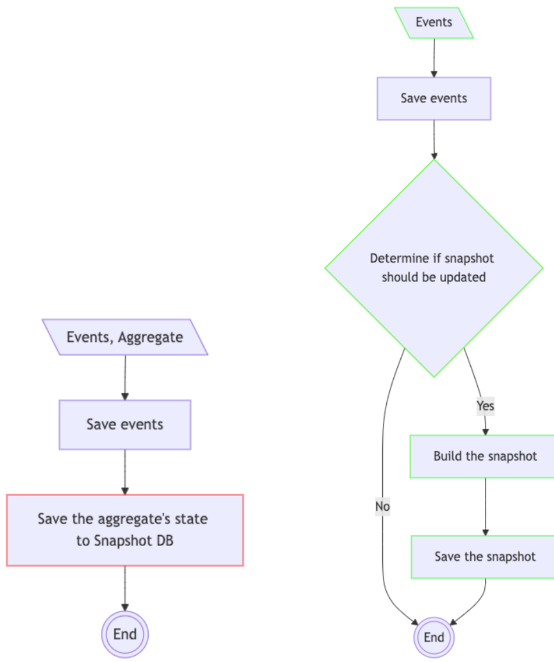

Figure 39: mCQRS and Classical CQRS Save an Aggregate Flowchart

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

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

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

$$W_A = 2 + 1 + 2 = 5$$

$$S = S_A + S_R = (1 + 0) * 5 + 1 = 6$$

Handle an Event (Update projection)

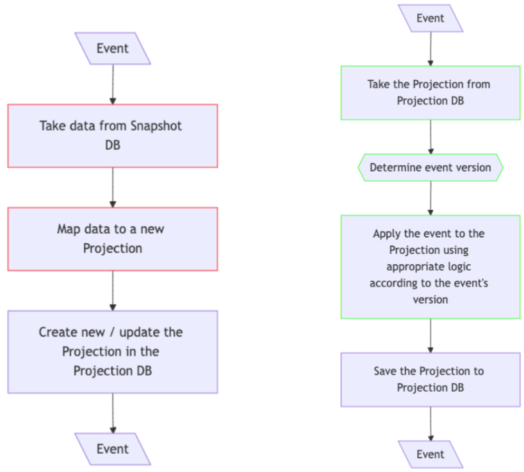

Figure 40: mCQRS and Classical CQRS Handle an Event Flowchart

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

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

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

$$W_A = 2 + 2 + 1 = 5$$

$$S = S_A + S_R = (1 + 1) * 5 + 2 = 12$$

Classical CQRS to mCQRS transition complexity

$$\omega_{p_c} = 20 + 1 + 6 + 12 = 39$$

$$\omega = c_{p_c} * \omega_{p_c} = c_{p_c} * 39$$

A.3.4 Summary

Table 13: Command process activities matching

Transition	Command process transition complexity (CWU)
Pure CQRS to Classical CQRS	12
Pure CQRS to mCQRS	26
Classical CQRS to mCQRS	28
mCQRS to Classical CQRS	39