Pros and Cons of Different options

Options:

1. Each request from a TCN starts a CHC process. Multiple CHC processes at the same time
2. Each request from a TCN sets a flag. Only a single CHC process at the same time

- What this algorithm provides?

1. Each TCN can start only 1 CHC process. This limit can be upgraded by agent designer. However, there should definitely be a limit. Otherwise, a TCN can start as many CHC processes as he wants which is not desirable.
2. TCNs don't have to wait for a CHC process to be completed.
3. TCNs will receive a solution for each CHC process that they are involved in. Each solution they receive will have a time stamp which means the solutions can be sorted from newest to oldest. Also each TCN will be able to view his/her preferences for each CHC solution.

CHC Coordination Protocol
Multiple Processes Simultaneously

Algorithm 1. Request CHC Process

1. **Event**: TCN requests a CHC process from its Agent
2. **Condition**: Its Agent checks the LAKR to see "If it already runs a CHC process as Dedicated agent".
    1. If **YES**,
        1. **Event**: Agent sends a message to TCN telling that "You have already started a CHC process. You can start a new one when it is completed."
    2. If **No**,
        1. **Condition**: Agent checks the LAKR to see "If it is involved in any CHC Process".
            1. If **YES**,
                1. **Event**: You are already involved in a CHC process, do you want to start a new one?
                2. **Condition**: Does the TCN want to start a new CHC process?
                    1. If **YES**,
                        1. **Event**: start **Algorithm 2**: *Initiate CHC Process*
                    2. If **NO,**
                        1. **Abort**
            2. If **NO,**
                1. **Event:** start **Algorithm 2**: *Initiate CHC Process*

Algorithm 2. Initiate CHC Process

1. **Event**: Agent sends a message to TCN telling that "I am starting a CHC process and will inform you when the result is ready".
2. **Event**: start **Algorithm 3**: *Compute CHC*
3. **Event**: receive the output of **Algorithm 3**: *Compute CHC*
4. **Event:** send the output to TCN. Output should include: 1) solution of CHC process and 2) a flag indicating whether the Agent is available to start a new CHC process.

CHC Coordination Protocol
Multiple Processes Simultaneously

Algorithm 3. Compute CHC

1. * TCN's agent acts as Dedicated agent
2. **Event**: Dedicated agent informs WPM that it is "In_Coordination" and populates LAR.
3. **Event**: Dedicated agent sends a message to all agents who are either "Active" or "In_Coordination" state. It requests those agents to return their Personal Info. Receiver agents run **Algorithm 4**: *Personal Info Request*
4. **Condition**: Dedicated agent waits for some amount of time to receive all the Personal info. If any agent doesn't return its info in this time, Dedicated agent proceeds without it.
5. **Event**: Once Dedicated agent collects all Personal info or the time expires, it broadcasts the Complete Personal Info to all participating agents.
6. **Event**: Dedicated agent requests participating agents to return their Similarity values. Receiver agents run **Algorithm 5**: *Similarity Values Request*
7. **Condition**: Dedicated agent waits for some amount of time to receive the Similarity Values from all participating agents. If any agent doesn't return its info in this time, Dedicated agent asks one more time and waits again. If the result is same, then it proceeds without this particular agent.
8. **Action**: Once Dedicated agent collects all Similarity values or the time expires, it runs **Algorithm 6**: *Solve Constrained Clustering Problem* and gives the Asymmetric Similarity Matrix as the input.
9. **Event**: Dedicated agent sends the Output of **Algorithm 6**, to other participating agents and informing them that the CHC solution is computed. Dedicated agent and receiver agents run **Algorithm 7**: *Share CHC Solution with TCN*

Algorithm 4. Personal Info Request

1. **Event**: An agent receives the request to return its Personal Info to Dedicated Agent
2. **Event**: Agent sends its Personal Info to Dedicated Agent
3. **Event**: Agent informs WPM that it is "In_Coordination"

Algorithm 5. Similarity Values Request

1. **Event**: When an agent is requested to send its Similarity values for a CHC process, it executes the following steps.
2. **Action**: It computes all pairwise similarities by comparing its Individual Preferences with the Personal info of all other participating agents
3. **Event**: It sends the Similarity values to the dedicated agent

Algorithm 6. Solve Constrained Clustering Problem

1. **Event**: When an agent has all necessary pairwise similarity values, it executes the following steps to compute the clusters for the given problem.
2. **Action:** Asymmetric Similarity Matrix (the input) is converted into Symmetric Similarity Matrix
3. **Action:** Symmetric Matrix is given as the input to the HDBSCAN* algorithm
4. **Action:** Algorithm computes and produces the similar clusters of agents (the output)

CHC Coordination Protocol
Multiple Processes Simultaneously

Algorithm 7. Share CHC Solution with TCN

1. **Event**: An agent receives the result of CHC process from Dedicated agent
2. **Event**: The agent shares the corresponding result with its TCN
3. **Event**: Informs WPM that it is not in coordination anymore only if it is not involved in any coordination process.

Multiple Processes Simultaneously
BTs

BT#1: CHCProtocolBT
Event#1: CHCProtocolEvent
Endpoint#1: CHCProtocolEndpoint
BT#2: ExecutingCHCProcessBT
Event#2: ExecuteCHCProcessEvent
BT#3: SharingCHCPersonaInfoBT
Event#3: ShareCHCPersonalInfoEvent
Endpoint#3: ShareCHCPersonalInfoEndpoint
BT#4: ReceivingCHCPersonalInfoBT
Event#4: ReceiveCHCPersonalInfoEvent
Endpoint#4: ReceiveCHCPersonalInfoEndpoint

BT#5: SharingCHCSimilaritiesBT
Event#5: ShareCHCSimilaritiesEvent
Endpoint#5: ShareCHCSimilaritiesEndpoint
BT#6: ReceivingCHCSimilaritiesBT
Event#6: ReceiveCHCSimilaritiesEvent
Endpoint#6: ReceiveCHCSimilaritiesEndpoint
BT#7: SolvingCHCBT
Event#7: SolveCHCEvent
BT#8: CHCFailureBT
Event#8: CHCFailureEvent
BT#9: FinalizingCHCProcessBT
Event#9: FinalizeCHCProcessEvent
Endpoint#9: FinalizeCHCProcessEndpoint

BT#1: CHCProtocolBT

1. **Sequence:**
    1. **HandleEvent**: handle *CHCProtocolEvent*
    2. **Priority**:
        1. **Sequence**:
            1. **Condition**: is agent running a CHC as dedicated agent?
            2. **Message to app**: you are already running a CHC
        2. **Sequence**:
            1. **Condition**: is agent involved in any CHC process?
            2. **Priority**:
                1. **Invert**:
                    1. **Message to app**: you are involved in a CHC process. Do you still want to start a new one?
                2. **Invert**:
                    1. **Condition**: does TCN want to start a CHC as dedicated agent?
        3. **EventProducer**: *ExecuteCHCProcess*Event

Multiple Processes Simultaneously
BTs

BT#2: ExecutingCHCProcessBT

1. **Priority:**
    1. **Sequence:**
        1. **HandleEvent**: *Execute CHC Process*
        2. **Message to app**: "I am starting a CHC process and will inform you when the result is ready"
        3. **Message to WPM**: "I am In_Coordination state"
        4. **EventProducer**: PopulateLAREvent
        5. **Wait**: 5 seconds
        6. **Priority**:
            1. **Condition**: is LAR populated?
            2. **Sequence:**
                1. **EventProducer**: PopulateLAREvent
                2. **Wait**: 5 seconds
                3. **Condition**: is LAR populated?
        7. **Write**: agent builds the CHC Problem instance with the corresponding participants who are either *Active* or *In_Coordination* states, etc.
        8. **Insert**: add the CHC problem id to "isRunningCHC"
        9. **Insert**: extract CHC Personal Info for this agent
        10. **Broadcast**: send message to the *SharingCHCPersonaInfoBT* endpoints of all agents who are participants of the particular problem instance. Specify the problemID in the payload, as well. I.e. trigger **SharingCHCPersonalInfoEvent**
        11. **Insert: "**CHC was broadcasted" to LAKR
        12. **Wait**: 10 seconds
        13. **Priority**:
            1. **Condition**: did all of the participating agents return their Personal info for the particular problemID?
            2. **Update**: update the CHC Problem Instance such that only the agents who returned their Personal info is participant. Remove the agents who didn't return Personal info.
        14. **Broadcast**: send Complete Personal Info to all participating agents and ask them to return their **Similarity values** for the specified problem id. I.e. trigger **SharingCHCSimilaritiesEvent**
        15. **Write:** Compute Similarity values for this agent and write to LAKR
        16. **Wait**: 20 seconds
        17. **Priority**:
            1. **Condition**: did all of the participating agents return their Similarity values for the particular problemID?
            2. **Write**: write 0 similarity values for the agents who didn't send their similarity values
        18. **EventProducer**: **SolveCHCEvent**
    2. **EventProducer:** produce CHCFailureEvent

Multiple Processes Simultaneously
BTs

BT#3: SharingCHCPersonaInfoBT

1. **Sequence**:
    1. **EventHandler**: handle SharingCHCPersonalInfoEvent
    2. **Write:** agent profile from lakr to execution
    3. **Message to Dedicated**: send the Personal Info to the dedicated agent. I.e. trigger **ReceivingCHCPersonalInfoEvent**
    4. **Message to WPM**: send status update to WPM: In_Coordination
    5. **Write**: add CHC problem id to “hasInvolvedInCHC”

BT#4: ReceivingCHCPersonaInfoBT

1. **Sequence**:
    1. **EventHandler**: handle ReceivingCHCPersonalInfo
    2. **Write:** write the Personal Info of the sender agent to Agent Knowledge with the problemID

BT#5: SharingCHCSimilaritiesBT

1. **Sequence**:
    1. **EventHandler**: handle SharingCHCSimilaritiesEvent
    2. **Write**: compute the similarities by comparing its Preferences with the Personal Info of other and write them to agent knowledge
    3. **Message to Dedicated**: send the Similarities to the dedicated agent. i.e. trigger **ReceivingCHCSimilaritiesEvent**

BT#6: ReceivingCHCSimilaritiesBT

1. **Sequence**:
    1. **EventHandler**: handle ReceivingCHCSimilaritiesEvent
    2. **Write:** write the Similarities of the sender agent to Agent Knowledge with the problemID

Multiple Processes Simultaneously
BTs

BT#7: SolvingCHCBT

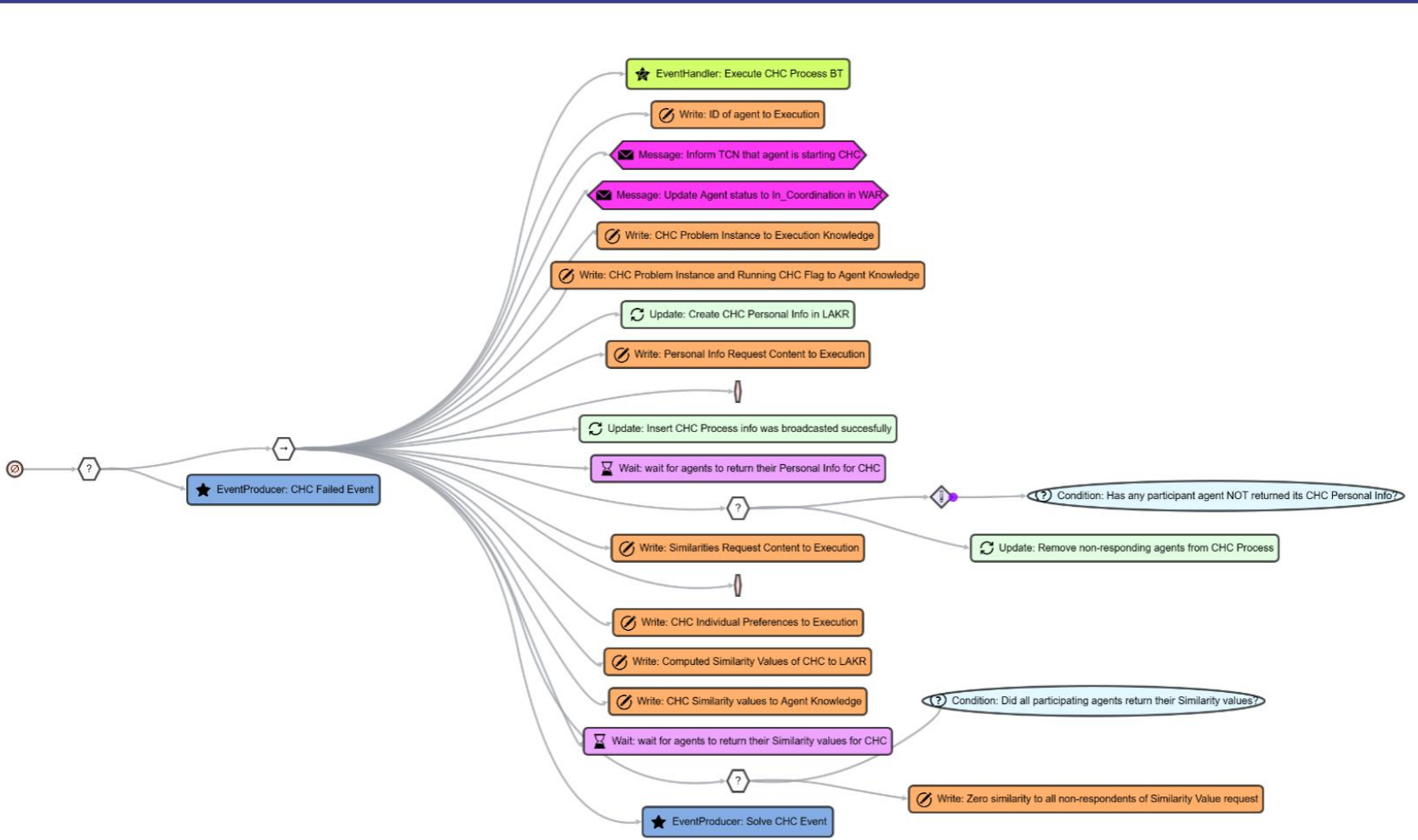
1. **Priority:**
    1. **Sequence**:
        1. **HandleEvent**: handle SolveCHCEvent by writing the asymmetric similarity values to Execution Knowledge for the specified problemID
        2. **Write**: calculate Reciprocal similarities from asymmetric similarities
        3. **HDBSCAN**: compute the distance scores from Reciprocal similarities and pass the distance matrix to HDBSCAN algorithm. Algorithm finds and writes the clusters to Agent Knowledge.
        4. **Broadcast:** send the result of HDBSCAN to other participating agents. i.e. trigger **FinalizeCHCProcessEvent**
        5. **EventProducer:** trigger **FinalizeCHCProcessEvent**
    2. **EventProducer:** produce CHCFailureEvent

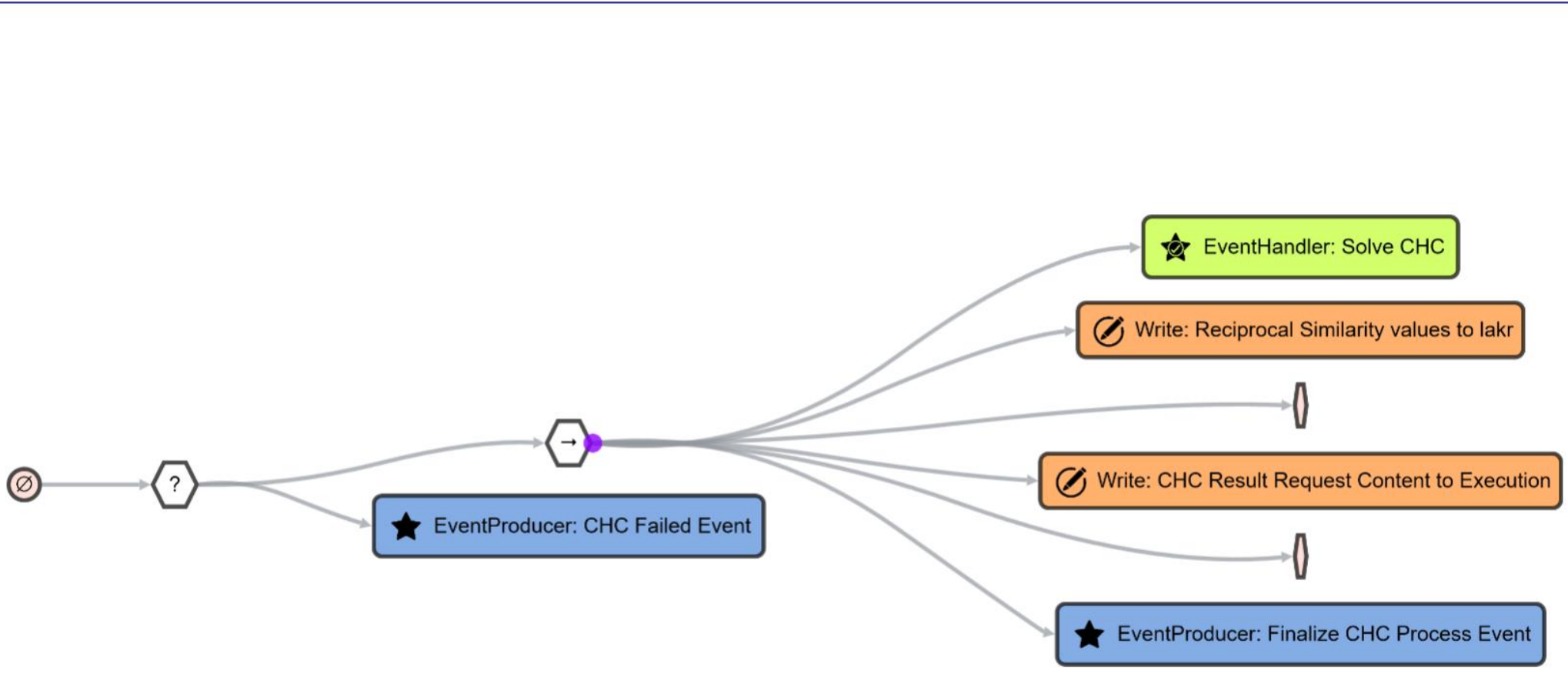
BT#8: CHCFailureBT

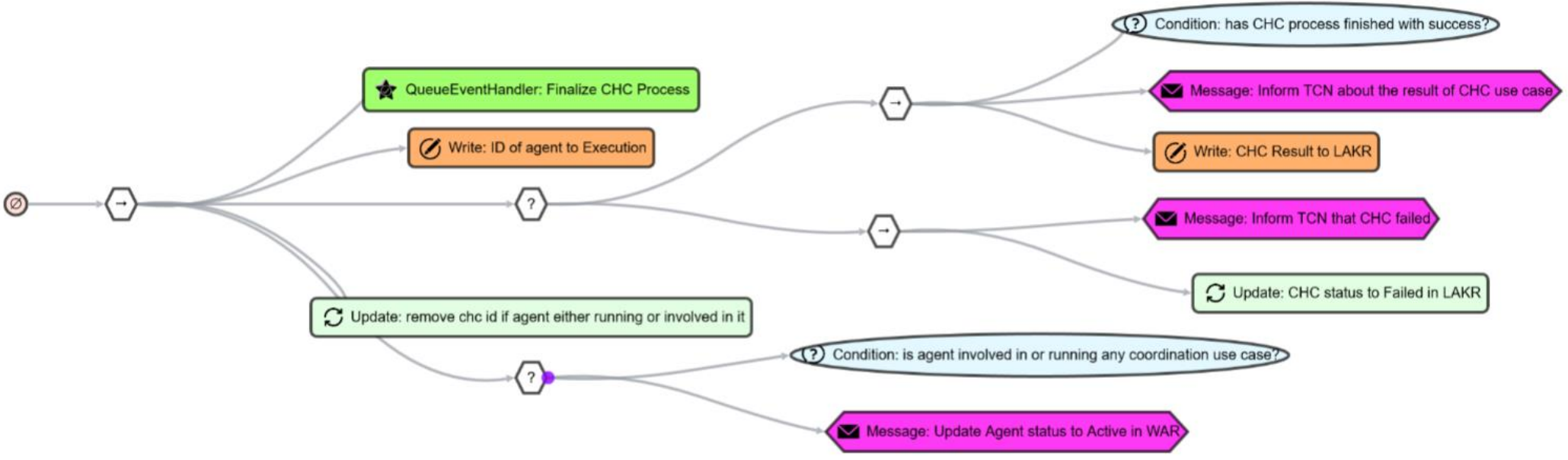
1. **Sequence:**
    1. **HandleEvent:** handle CHCFailureEvent
    2. **EventProducer:** produce FinalizeCHCProcessEvent
    3. **Priority**:
        1. **Invert:**
            1. **Condition**: CHC was built and sent to agents?
        2. **Broadcast :** something went wrong in CHC process. i.e. trigger FinalizeCHCProcessEvent

BT#9: FinalizeCHCProcess

1. **Sequence**:
    1. **HandleEvent**: handle FinalizeCHCProcessEvent
    2. **Priority:**
        1. **Sequence**:
            1. **Condition**: CHC Succeeded?
            2. **Message to app**: send the corresponding CHC result or “Something went wrong” message to TCN
            3. **Write**: chc result to lakr
        2. **Sequence**:
            1. **Message to app**: CHC failed. Try again
            2. **Write**: chc failed to lakr
    3. **Delete**: delete CHC problem id if agent is running or involved in it
    4. **Priority**:
        1. **Condition**: check if involved or running in any coordination use case either LCC or CHC
        2. **Message to WPM**: tell WPM: “coordination ended”


EventHandler: Execute CHC Process BT
Write: ID of agent to Execution
Message: Inform TCN that agent is starting CHC
Message: Update Agent status to In_Coordination in WAR
Write: CHC Problem Instance to Execution Knowledge
Write: CHC Problem Instance and Running CHC Flag to Agent Knowledge
Update: Create CHC Personal Info in LAKR
Write: Personal Info Request Content to Execution
Update: Insert CHC Process info was broadcasted succesfully
Wait: wait for agents to return their Personal Info for CHC
Condition: Has any participant agent NOT returned its CHC Personal Info?
Update: Remove non-responding agents from CHC Process
EventProducer: CHC Failed Event
Write: Similarities Request Content to Execution
Write: CHC Individual Preferences to Execution
Write: Computed Similarity Values of CHC to LAKR
Write: CHC Similarity values to Agent Knowledge
Condition: Did all participating agents return their Similarity values?
Wait: wait for agents to return their Similarity values for CHC
Write: Zero similarity to all non-respondents of Similarity Value request
EventProducer: Solve CHC Event


EventHandler: Solve CHC
Write: Reciprocal Similarity values to Iakr
Write: CHC Result Request Content to Execution
EventProducer: Finalize CHC Process Event
EventProducer: CHC Failed Event


QueueEventHandler: Finalize CHC Process
Write: ID of agent to Execution
Update: remove chc id if agent either running or involved in it
Condition: has CHC process finished with success?
Message: Inform TCN about the result of CHC use case
Write: CHC Result to LAKR
Message: Inform TCN that CHC failed
Update: CHC status to Failed in LAKR
Condition: is agent involved in or running any coordination use case?
Message: Update Agent status to Active in WAR

Thanks!