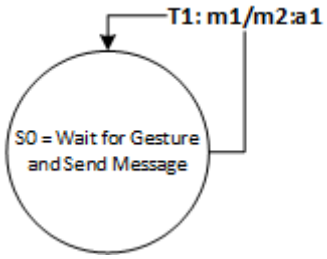
Christopher Thomas
CLT29@PITT.EDU
Software Engineering

Exercise 2.

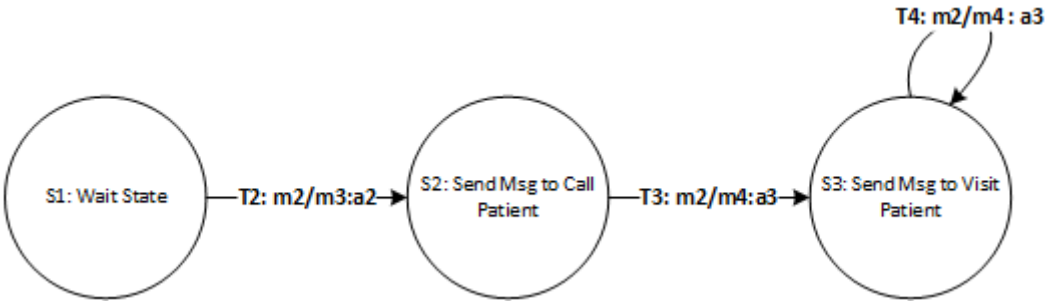
(a.) Draw state-transition diagrams to define graphically the three index cell types.

There are three types of index cells: gesture recognition, emergency manager, and homecare staff index cells. In this question, we are asked to graphically define each of the three index cell types. Thus, state-transition diagrams for the three different index cell types will be given.

The first index cell is the gesture recognition index cell. The gesture recognition index cell is responsible for recognizing the "I need help" gesture from the patient it is monitoring. Thus, we can consider the "I need help" gesture as a message to the GR index cell. Once it recognizes the gesture, the GR index cell, creates a message m2, which states that "Patient Smith needs help." The action taken on receipt of m1 is to send the message "Patient Smith needs help" to the emergency manager index cell. Thus, the GR state-transition diagram is shown below. Note that after sending the message, the gesture recognizer returns to waiting for another gesture.



The second index cell is the emergency manager index cell. The emergency manager index cell receives a message from the gesture recognizer that states patient smith needs help (that was m2). Since this is the first time the EM has received that message, it sends a message (m3) to call patient smith to the homecare staff. Note, the EM remembers that it has already seem one request for patient smith needing help. If it receives a second m2 (patient smith needs help), then it will notify the homecare staff to visit patient smith (m4) because the problem is severe.

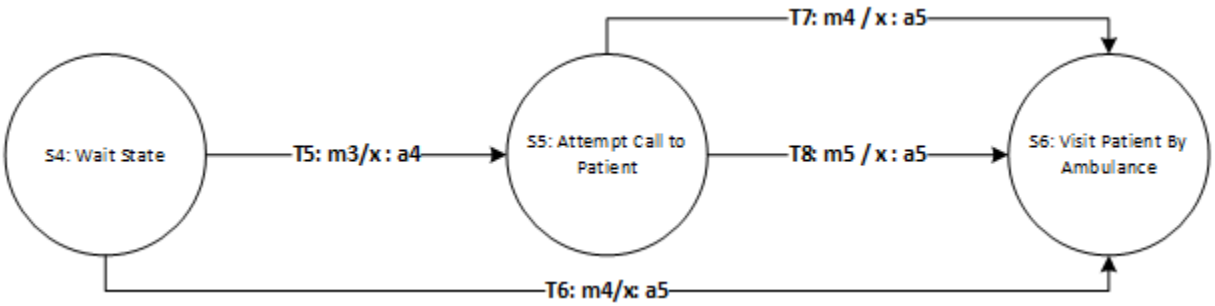


Note, here m2 is the message that Patient Smith needs help. M3 is the message to call Patient Smith. M4 is the message to visit patient smith. Action 2 is to send M3 to the homecare staff index cell. Action 3 is the action to send m4 to the homecare staff index cell.

Note also that S3 would eventually probably need to be reset to S1 after a timeout or after someone resets it. That was not discussed so it is omitted here.

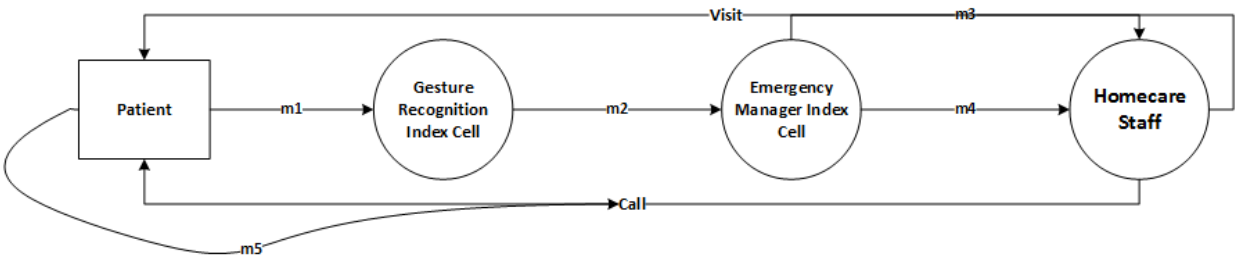
Christopher Thomas
CLT29@PITT.EDU
Software Engineering

We now reach the homecare staff index cell. The homecare staff index cell begins by waiting for a message. If the homecare staff receives a message to visit the patient, immediately an ambulance is dispatched to visit patient. On the other hand, if the homecare staff receive a message to call the patient, they will attempt to call the patient. If the patient cannot be reached, then they will also need to go to the patient's house in an ambulance.



Note, here if there is an item which isn't produced, I represent that with an x. For instance, m3/x:a4, the x means that no message is produced (but an action is still taken).
Here, m3 is the message to call patient Smith. M4 is the message to visit the patient. M5 represents the message that the patient was not reachable by phone. A4 is the action to call the patient on the phone. A5 is the action to visit the patient by ambulance.

Thus, the overall system when put together looks like the below. This is the data-flow diagram, following the convention given in the slides.



(Please see the next page for part B)

Christopher Thomas
CLT29@PITT.EDU
Software Engineering

Part B) Specify the three index cell types formally using mathematical notations ic = (X, Y, S, so, A, tmax, f, g).

Note: d is the dummy message, for consistency with this.
Note that for simplicity in the f function, I omit listing combinations which are not acceptable, i.e. equals 0, such as the empty set and every possible subset of messages crossed with every state in the f function to simplify things. Thus, only those configurations listed in the F function below are accepted, all others are 0. Dr. Chang said we could use notational shortcuts. This will greatly shorten the notation.

The **Gesture Recognition Index Cell** =
X = {d, m1}
Y = {d, m2}
S = {S0}
s0 = S0
A = {a1}
Tmax = infinite
F = f({m1}, s0) = 1. Also f({d, m1}, s0) = 1
G = g({m1}, s0) = (emergency manager,{m2},s0,a1), g ({m1,d}, s0) = (emergency manager,{m2},s0,a1).

Note, all ICs here also accept the dummy message along with the message i.e. {m1,d} is accepted and {m1} is accepted. For purposes of brevity, **I am omitting the d in future steps, as it is just ignored.**

The **Emergency Manager Index Cell** =
X = {m2}
Y = {m3,m4}
S = {S1,S2,S3}
s0 = S1
A = {a2,a3}
Tmax = infinite
F = f({m2},s1) = 1
f({m2},s2) = 1
f({m2},s3) = 1
(again, {m2,d} would also work)… Note that all other combinations from the power set not listed here are NOT accepted (i.e. f({}, s1)=0), so I am only listing accept values as a shorthand
G = g({m2},s1) = (homecare_staff, {m3}, S2, a2)
g({m2},s2) = (homecare_staff, {m4}, S3, a3)
g({m2},s3) = (homecare_staff, {m4}, S3, a3)

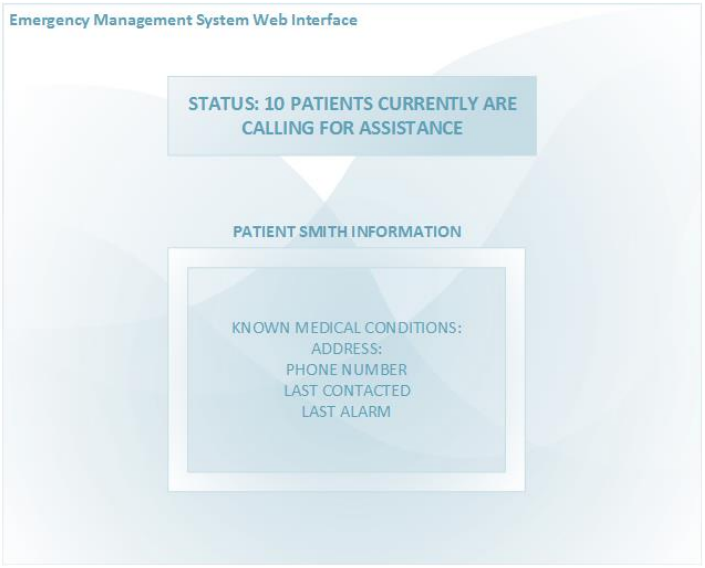
The **Homecare Staff Index Cell =**
X = {m3,m4}
Y = {d} (dummy value, doesn't output anything)
S = {S4,S5,S6}
s0 = S4
A = {a4,a5}
Tmax = infinite
F=f({m3},S4)=1

f({m4},S4)=1
f({m4},S5)=1
f({m5},S5)=1
(again, all other states from the power set and cross product = 0 (except the dummy additions as described above))
G=g({m3},S4)=(null, null, S5, a4)
g({m4},S4)=(null, null, S6, a5)
g({m4},S5)=(null,null,S6,a5)
g({m5},S5)=(null,null,S6,a5)

PART (C) Draw a diagram showing three multimedia interfaces (webpages such as doc-1, ..., doc-3) enhanced with the index cells to illustrate how these index cells work together to form an active index system.

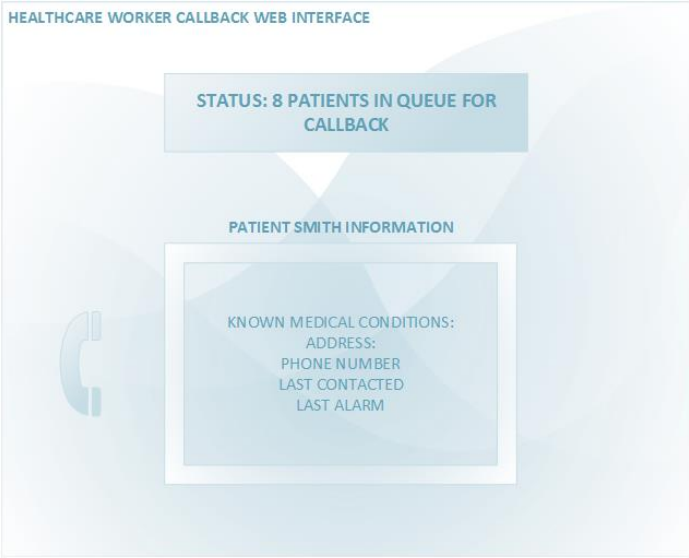
PLEASE SEE NEXT PAGE FOR FULL DIAGRAM.

DOC-1 is the emergency management system web interface, which serves to display the message about patients currently needing assistance. It looks like this:



DOC-2 is the healthcare staff call patient page. It shows how many patients currently need a call and displays their information to the staff. They are able to click and pull up different patients' information. They can also click the phone icon to initiate a call to the patient.

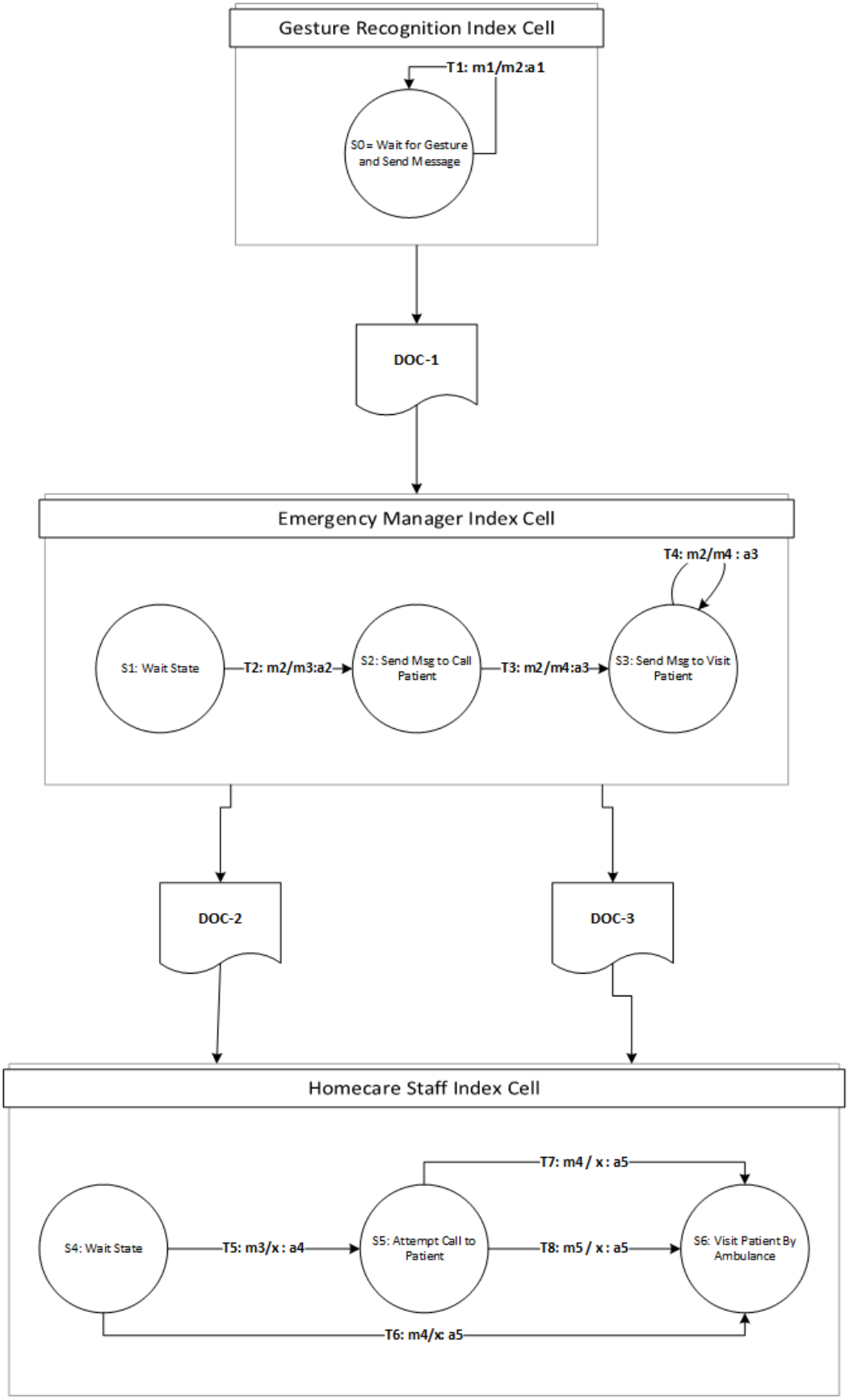
Christopher Thomas
CLT29@PITT.EDU
Software Engineering



DOC-3 is the healthcare staff dispatch ambulance page. It shows how many patients currently need an ambulance sent to their home. The user can scroll through different patients. The healthcare staff can click the ambulance icon to dispatch an ambulance to the user's home.

PLEASE SEE BELOW FOR THE DOCUMENTS PLACED IN THE WHOLE SYSTEM

Christopher Thomas
CLT29@PITT.EDU
Software Engineering



Christopher Thomas
CLT29@PITT.EDU
Software Engineering

Part (D.) Following the discussion on the concept of patterns, define more clearly the pattern(s) you have identified. If you feel the patterns you have identified are lacking in certain respect, you may replace them by some new patterns.

Pattern: Multi-Tiered Client-Server Architectures
Context: A distributed healthcare monitoring system, with components located in different locations.
Problem: Information needs passed to the respective parties, based on prior states.
Solution: A multi-tiered stateful client-server architecture, which passes messages between the gesture recognition interface, the stateful emergency management server, and the stateful homecare staff server.

This pattern is commonly seen is distributed systems. For instance, when a user requests a web-page, the user's request may need to be forwarded down some chain of servers before the request can be processed. This pattern also reveals some problems. The first problem is that messages take time to transmit. Thus, by increasing the depth of the architecture we create a situation where the patient's message requires 2 hops over the network to reach homecare staff. This may take time if the servers are overloaded, if the network is congested, or if failures are high in the network. Another problem is that the emergency management server is stateful and the homecare staff is stateful. If the emergency management server fails and reboots and loses the state, it may forget that it has currently seen 1 I need help message. By having an emergency monitoring server in the middle, we increase the nodes that could fail. It may be better to simply communicate directly from the gesture recognizer to homecare staff.

Pattern: Priority Escalation
Context: A home health care monitoring system, which monitors users's gestures for signs of distress.
Problem: Determining patients' needs: for instance, does the patient just need to talk to a nurse over the telephone or is there an actual emergency where the patient immediately needs to the hospital?
Solution: Escalate the response based on the number of messages received by the homecare staff and the inability to reach the patient by telephone.

One pattern I see in this architecture is the concept of escalation. We start out by assuming that the user may have caused a false alarm somehow, or that the gesture recognition made a mistake. To remedy this situation, an attempt is made by the homecare staff to contact the patient. If contact fails, the situation is escalated (i.e. the patient could be passed out or unable to reach the phone). The request is escalated to emergency status and an ambulance is dispatched. Also, if multiple messages for help come in from the gesture recognizer, we are more certain that a problem has occurred because of the multiple messages, so the very receipt of multiple messages for help increases the priority of the request and escalates it.

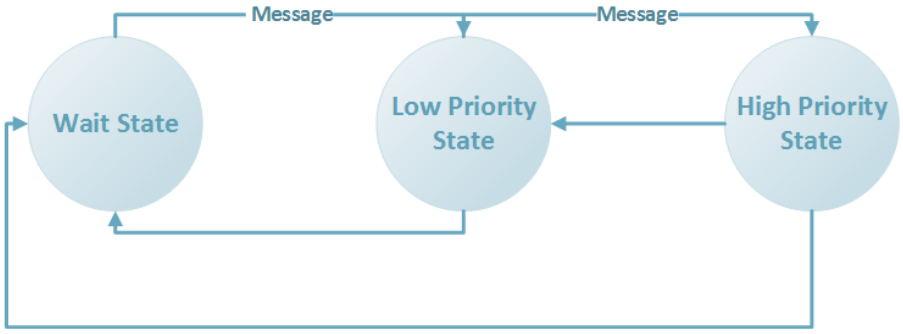
Pattern: Waiting for Event (Daemon systems)
Context: A home health care system, which monitors users for signs of distress and notifies parties of it
Problem: The moment that the user will need service from the system is unknown (i.e. the system could go years without receiving a signal for help)
Solution: If no signal is received, wait and continuously monitor for a signal (both in the gesture monitoring system, the emergency management system, and the healthcare staff system).

Christopher Thomas
CLT29@PITT.EDU
Software Engineering

This pattern illustrates that even when the system is idle, all three components of the system, the gesture recognizer, the emergency management server, and the healthcare staff server must continuously be monitoring and waiting for any available signals to come in. If signals are relatively rare events, a lot of energy / time could be wasted just sitting in the waiting state. However, because of the characteristics of the waiting state, this would allow engineers to design these systems to go into low-power monitoring modes, etc. which may help solve that problem of needing to constantly be available with no down-time.

Part (E) A visual specification of the identified pattern(s) should be included, using for example visual grammar rules. Remember Alexander's dictum: "If you can't draw a picture of it, it isn't a pattern.")

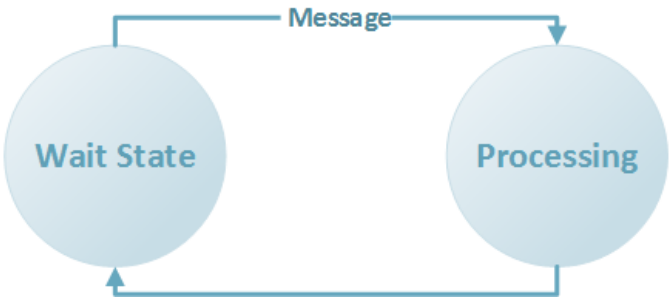
Multi-Tiered Communication (note it takes 2 network hops to reach the healthcare staff)



Escalation Pattern. Note that how the system returns to the wait state will depends on the semantics of the application.

(See next page)

Christopher Thomas
CLT29@PITT.EDU
Software Engineering



Pattern: Waiting for Event