Software Design and Architecture

Engr. Abdul-Rahman Mahmood

DPM, MCP, QMR(ISO9001:2000)

armahmood786@yahoo.com	alphasecure@gmail.com
alphapeeler.sf.net/pubkeys/pkey.htm	http://alphapeeler.sourceforge.net
pk.linkedin.com/in/armahmood	http://alphapeeler.tumblr.com
www.twitter.com/alphapeeler	armahmood786@jabber.org
www.facebook.com/alphapeeler	alphapeeler@aim.com
abdulmahmood-sss alphasecure	mahmood_cubix 48660186
armahmood786@hotmail.com	alphapeeler@icloud.com
http://alphapeeler.sf.net/me	http://alphapeeler.sf.net/acms/

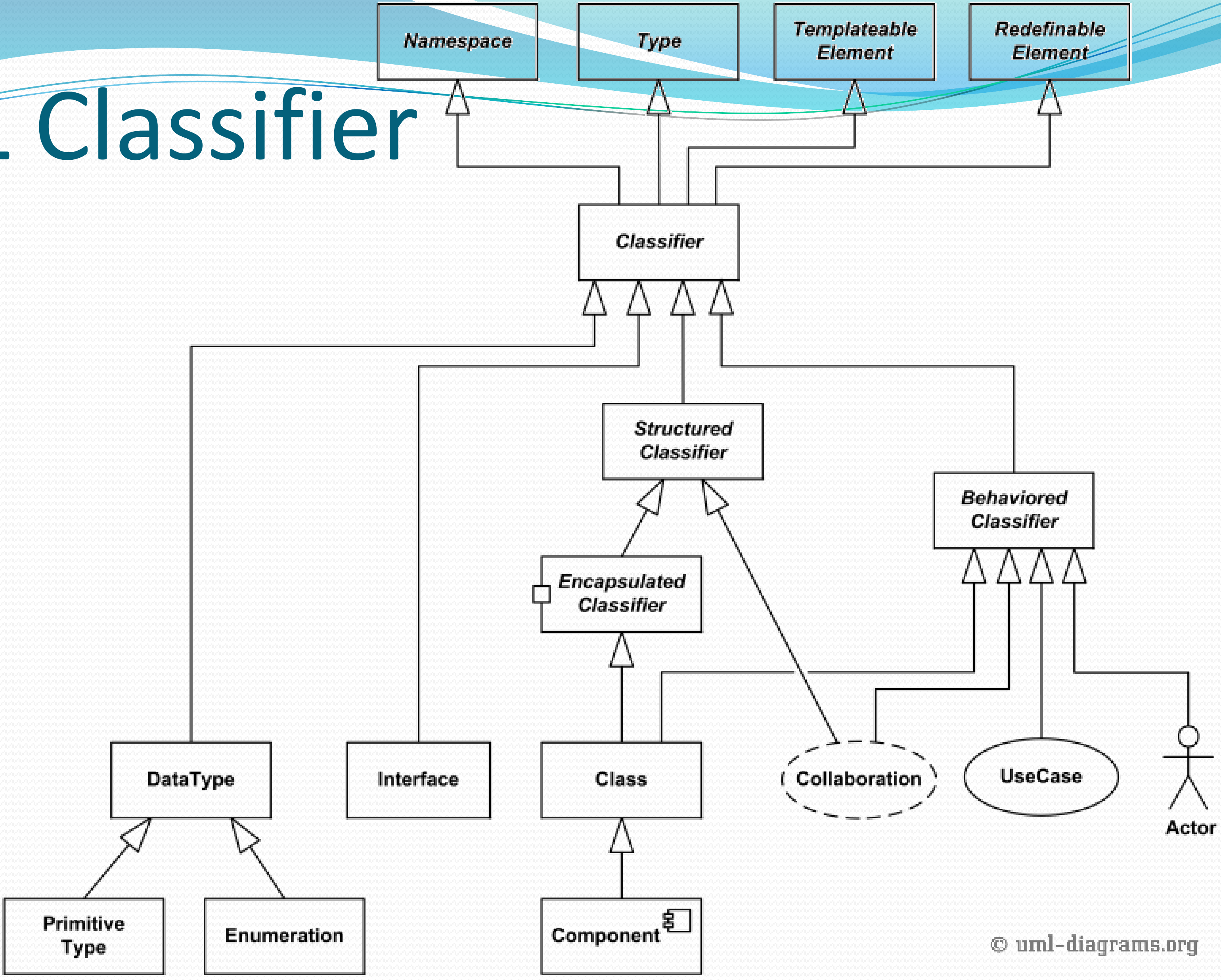
interaction diagrams

Timing Diagrams

Timing Diagrams

- A timing diagram allows you to show the interaction of objects and changes in state for those objects along a time axis.
- A timing diagram provides a convenient way to show active objects and their state changes during their interactions with other active objects and system resources.
- The X-axis of the timing diagram has the time units, while the Y-axis shows the objects and their states.
- Timing diagrams describe behavior of both individual **classifiers** & **interactions of classifiers**, focusing attention on time of events causing changes in the modeled conditions of the lifelines.

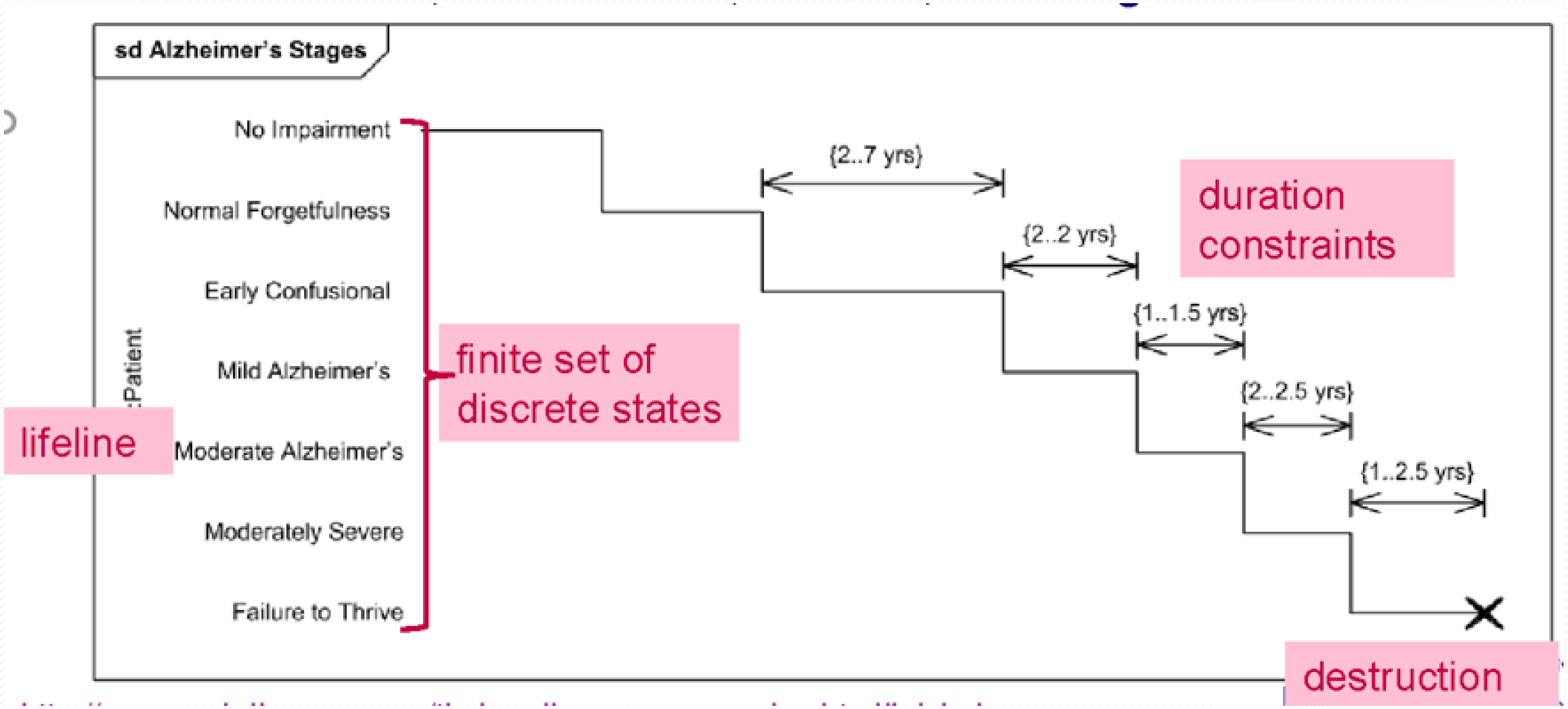
UML Classifier



The most important UML 2.5 classifiers - class, interface, data type, component, collaboration, use case, etc.

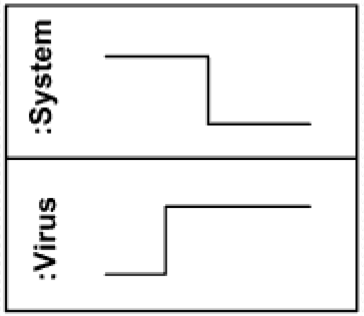
Timing Diagrams

- Timing Diagrams are Interaction diagram for reasoning about time.
- **Basic elements**: lifelines, states, duration/time constraints, destruction, events, messages



Lifeline

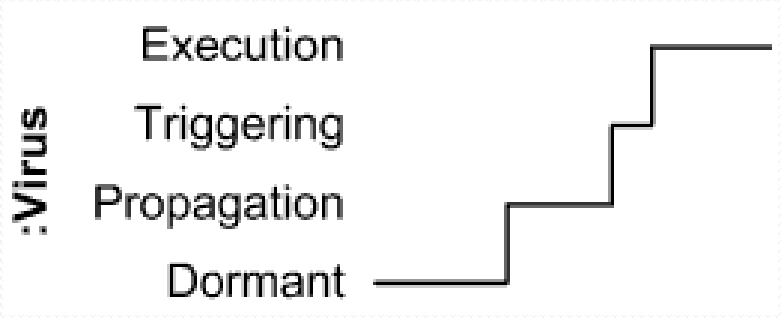
- **Lifeline** is a **named element** which represents an **individual participant** in the interaction. While **parts** and structural features may have multiplicity greater than 1, lifelines represent **only one** interacting entity.
- Lifeline on the timing diagrams is represented by the **name** of classifier or the instance it represents. It could be placed inside diagram frame or a "swimlane".



***Lifelines** representing instances of System and Virus*

State or Condition Timeline

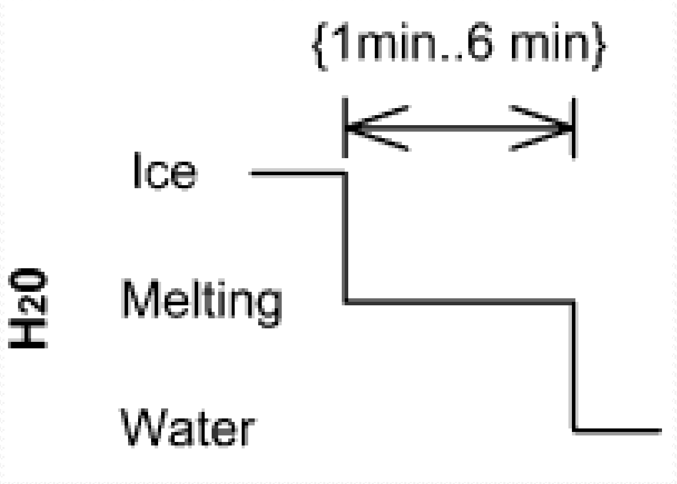
- Timing diagram could show **states** of the participating **classifier** or attribute, or some testable **conditions**, such as a discrete or enumerable value of an attribute.



- *Timeline shows Virus changing its state between Dormant, Propagation, Triggering and Execution state*

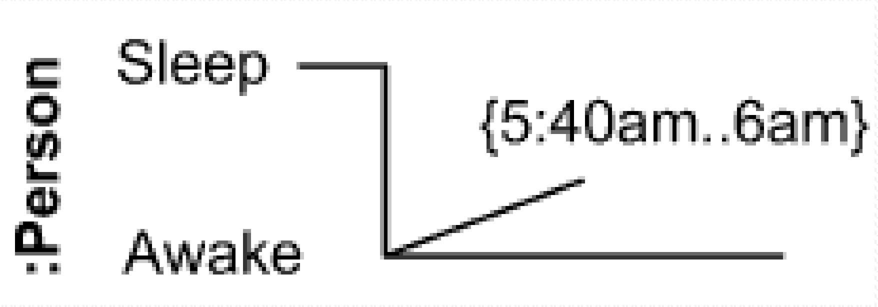
Duration Constraint

- **Duration constraint** is an **interval constraint** that refers to a **duration interval**. The duration interval is duration used to determine whether the constraint is satisfied.
- E.g., *Ice should melt into water in 1 to 6 minutes*



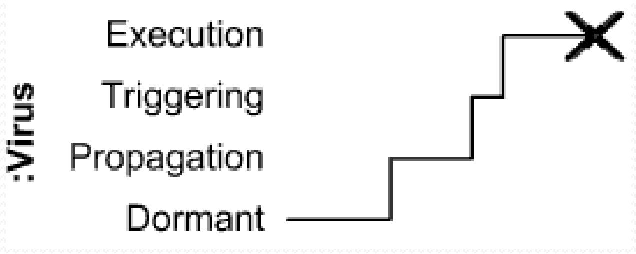
Time Constraint

- **Time constraint** is an **interval constraint** that refers to a **time interval**. The time interval is time expression used to determine whether the constraint is satisfied.
- Typically this graphical association is a small line, e.g., between an occurrence specification and a time interval.
- E.g., *Person should wake up between 5:40 am and 6 am*



Destruction Occurrence

- **Destruction occurrence** is a **message occurrence** which represents the destruction of the instance described by the **lifeline**.
- It may result in the subsequent destruction of other objects that this object owns by **composition**.
- No other occurrence may appear after the destruction event on a given lifeline.
- ***Notation***
- The destruction event is depicted by a cross in the form of an **X** at the end of a timeline.

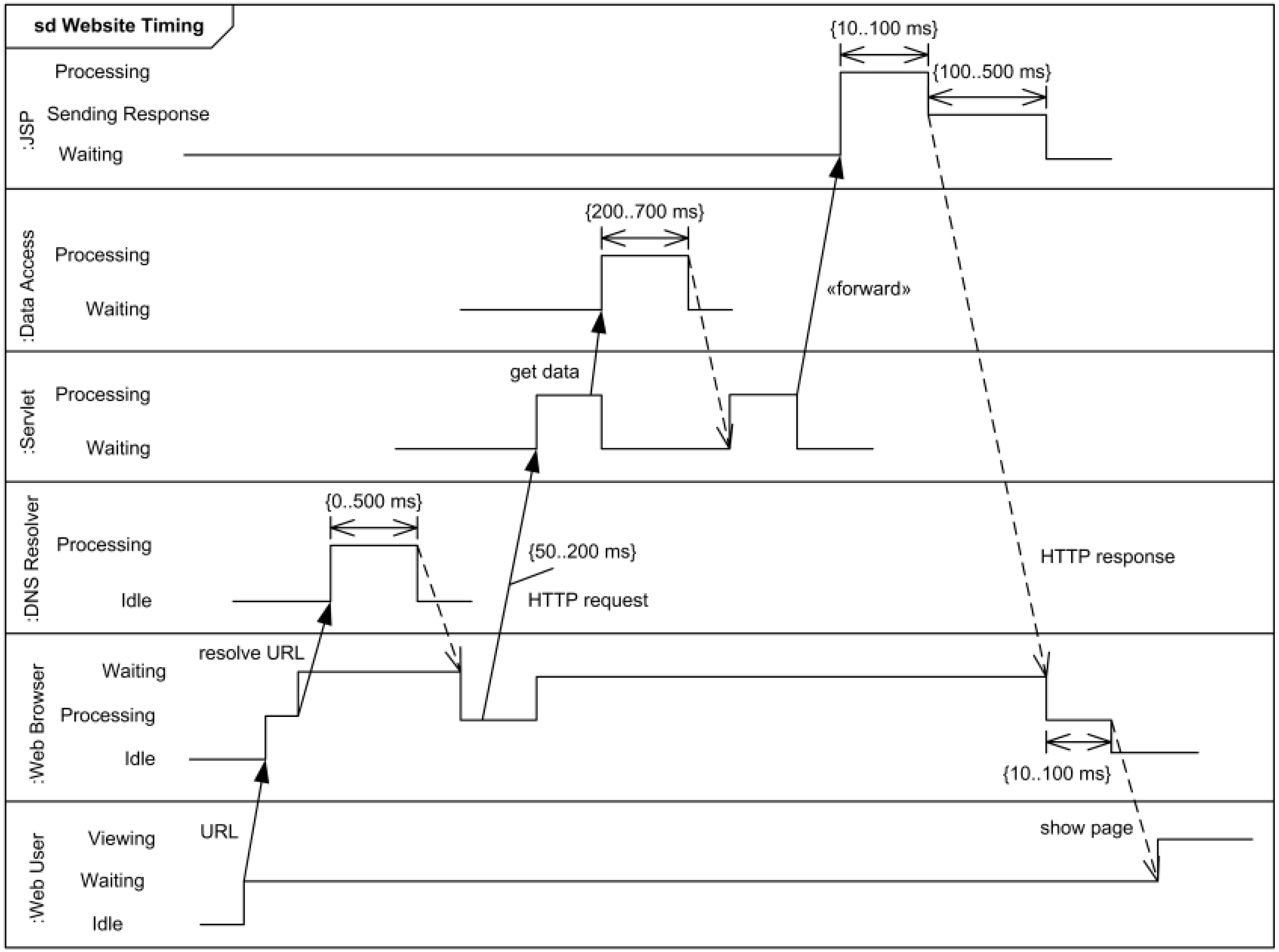


Ex 2: Website Latency

- An example of **timing diagram** which shows some **duration constraints** for a fabricated website to evaluate how long web user should wait to see something rendered on his/her display.
- After web user enters web page URL, the URL should be resolved to some IP address. DNS resolution can add some tangible waiting time to the response latency as perceived by user. Latency delays related to DNS resolution could range from 1 ms (local DNS cache) to several seconds.

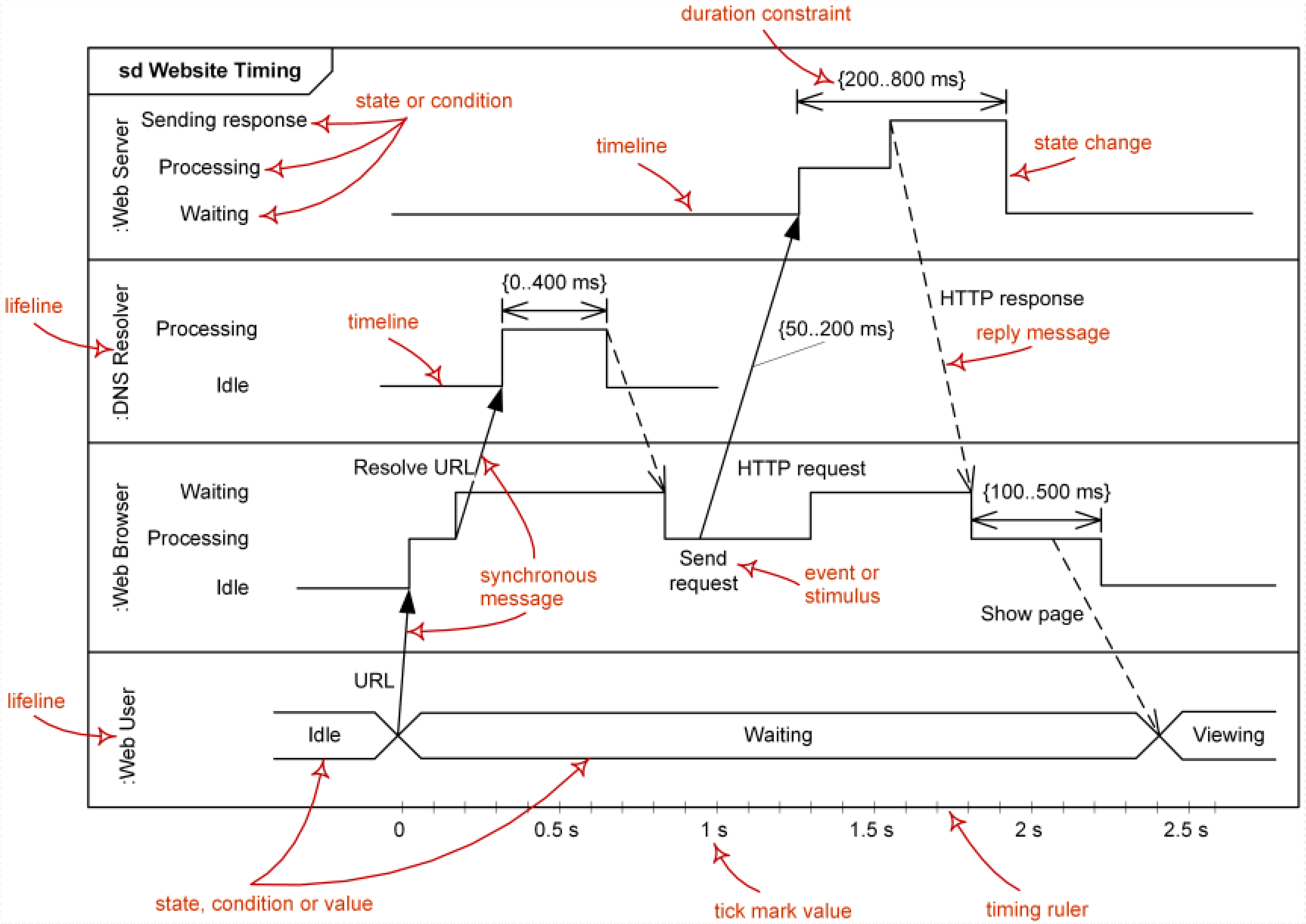
Ex 2: Website Latency

- With simple Model–View–Control (MVC) implementation, Java servlet gets control and requests some data from "model". Communication with data sources usually takes some discernible time. After data is received and processed, servlet forwards request processing to JSP ("view"). Buffered HTTP response is sent back to the browser.
- Web browser takes some time to process HTTP response and HTML page to start rendering the page view to the web client. (Note, that after that it could take even more time for the web browser to request other resources like CSS, JavaScript, images, which is not shown on the diagram.)



sd Website Timing
:JSP
Processing
Sending Response
Waiting
{10..100 ms}
{100..500 ms}
:Data Access
Processing
Waiting
{200..700 ms}
«forward»
:Servlet
Processing
Waiting
get data
:DNS Resolver
Processing
Idle
{0..500 ms}
{50..200 ms}
HTTP request
HTTP response
:Web Browser
Waiting
Processing
Idle
resolve URL
{10..100 ms}
:Web User
Viewing
Waiting
Idle
URL
show page

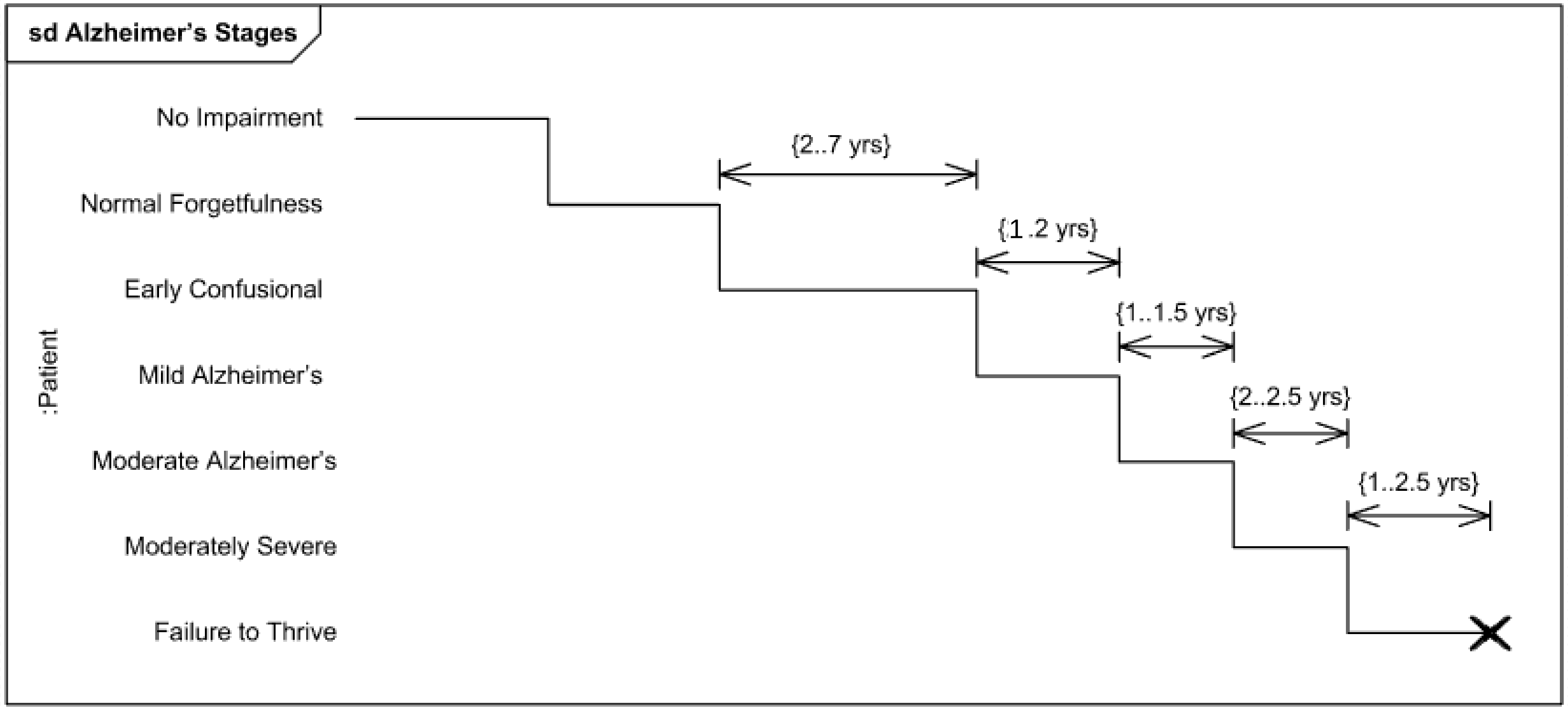
Ex2 : Notations



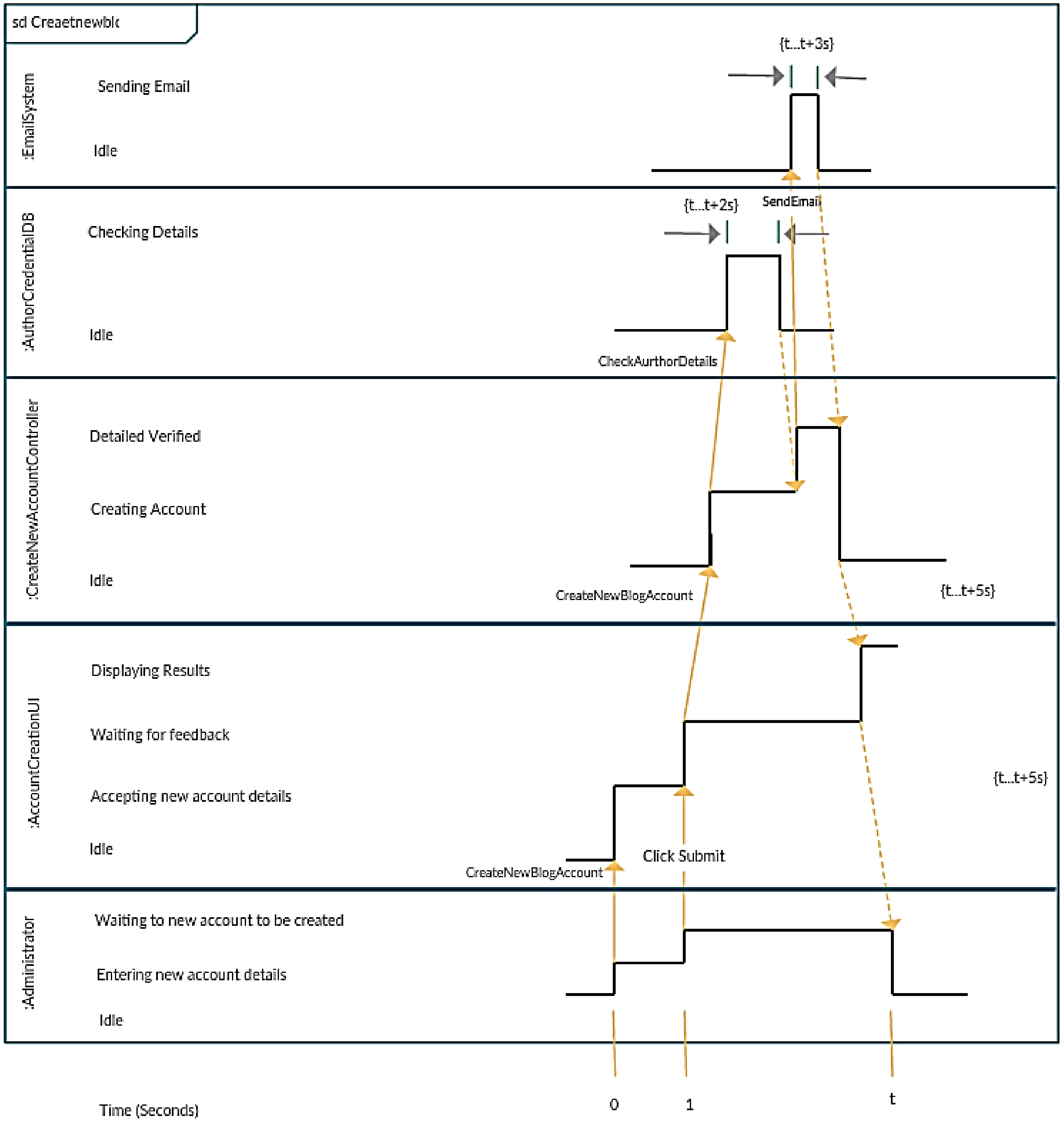
Ex 3: Stages of Alzheimer's Disease

- **Alzheimer's disease** (AD) is a a progressive, ultimately fatal brain disease that causes loss of memory and intellectual abilities. The cause of the disease is unknown. AD has no cure and is one of the leading causes of death in the United States.
- For Alzheimer's disease doctor may use a diagnostic framework with three to seven levels (stages). Progression through these stages may last from 8 to 10 years, and in some cases up to 20 years from the time neuron changes start.
- Example of timing diagram below shows timing for the seven stage framework.

Ex 3: Stages of Alzheimer's Disease

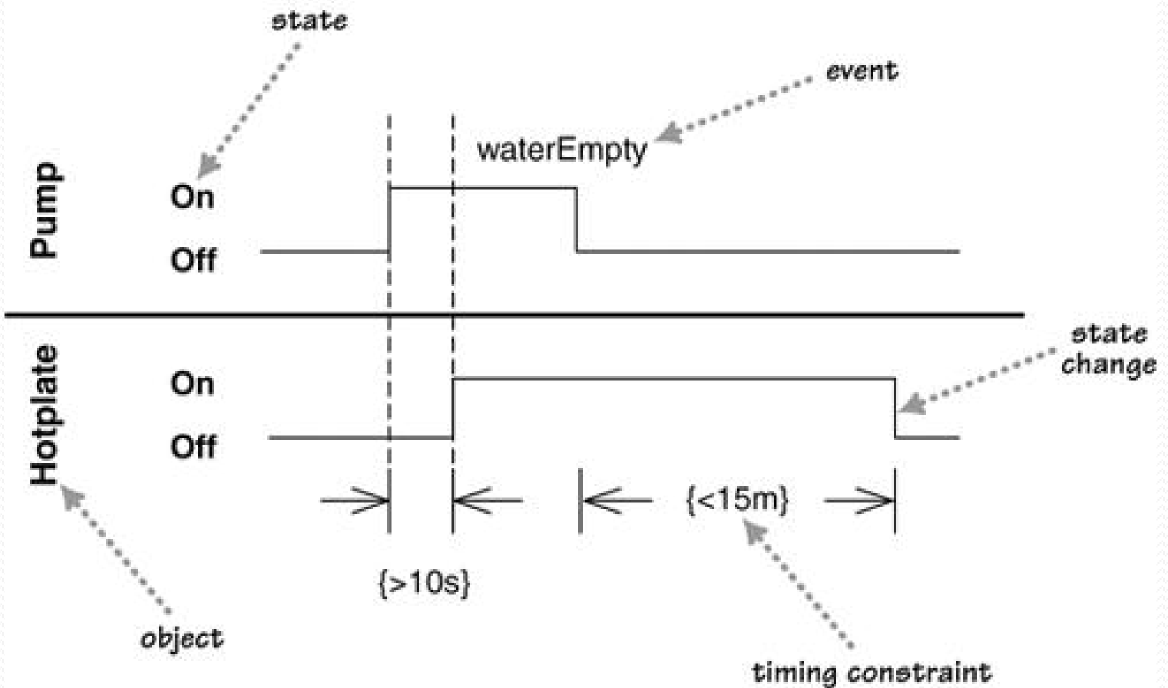


Ex : 4

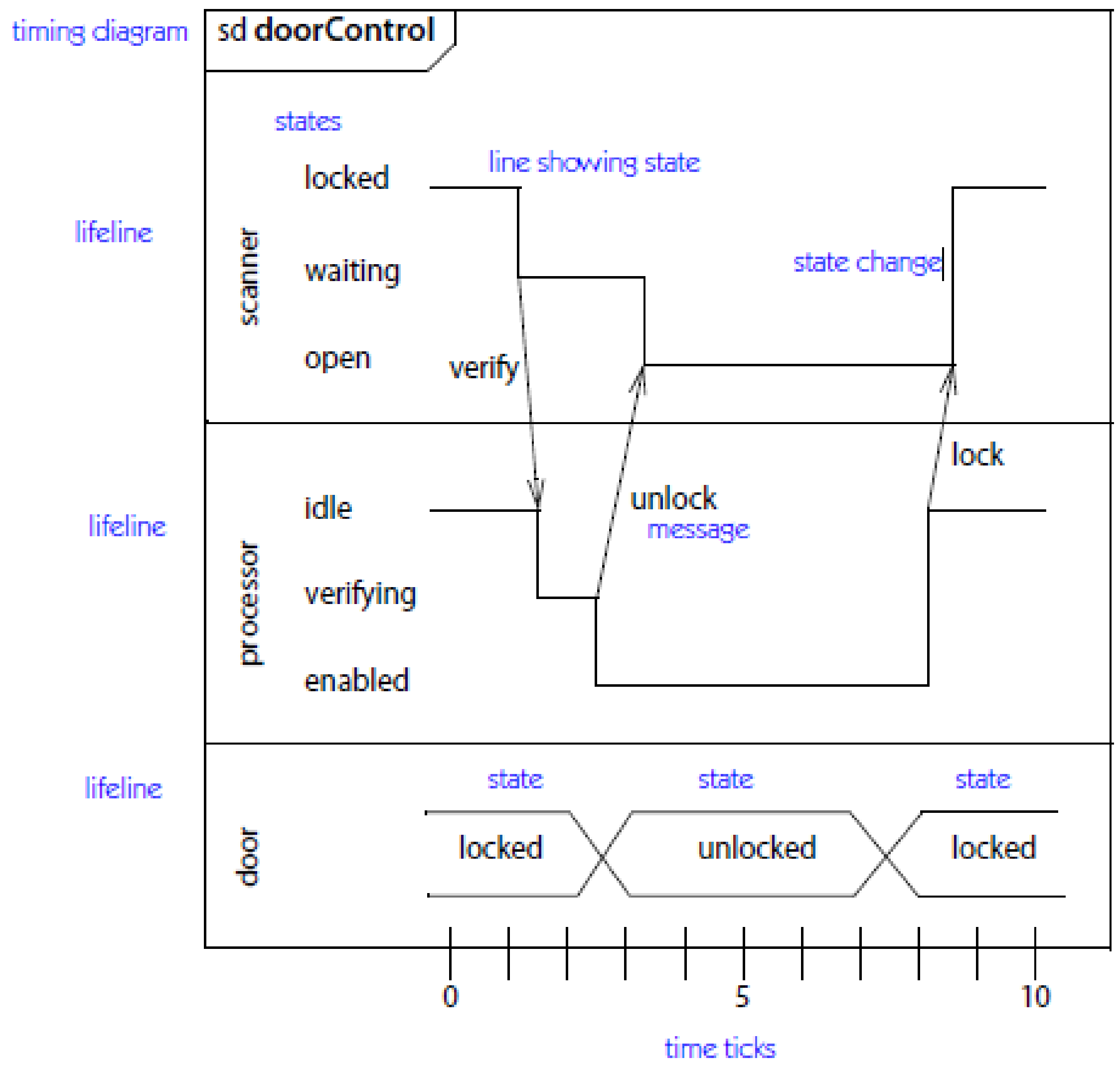

sd Creaetnewblc
:EmailSystem
Sending Email
Idle
{t...t+3s}
:AuthorCredentialDB
Checking Details
Idle
{t...t+2s}
SendEmail
CheckAurthorDetails
:CreateNewAccountController
Detailed Verified
Creating Account
Idle
CreateNewBlogAccount
{t...t+5s}
:AccountCreationUI
Displaying Results
Waiting for feedback
Accepting new account details
Idle
{t...t+5s}
Click Submit
CreateNewBlogAccount
:Administrator
Waiting to new account to be created
Entering new account details
Idle
Time (Seconds)
0
1
t

Ex 5: coffee pot

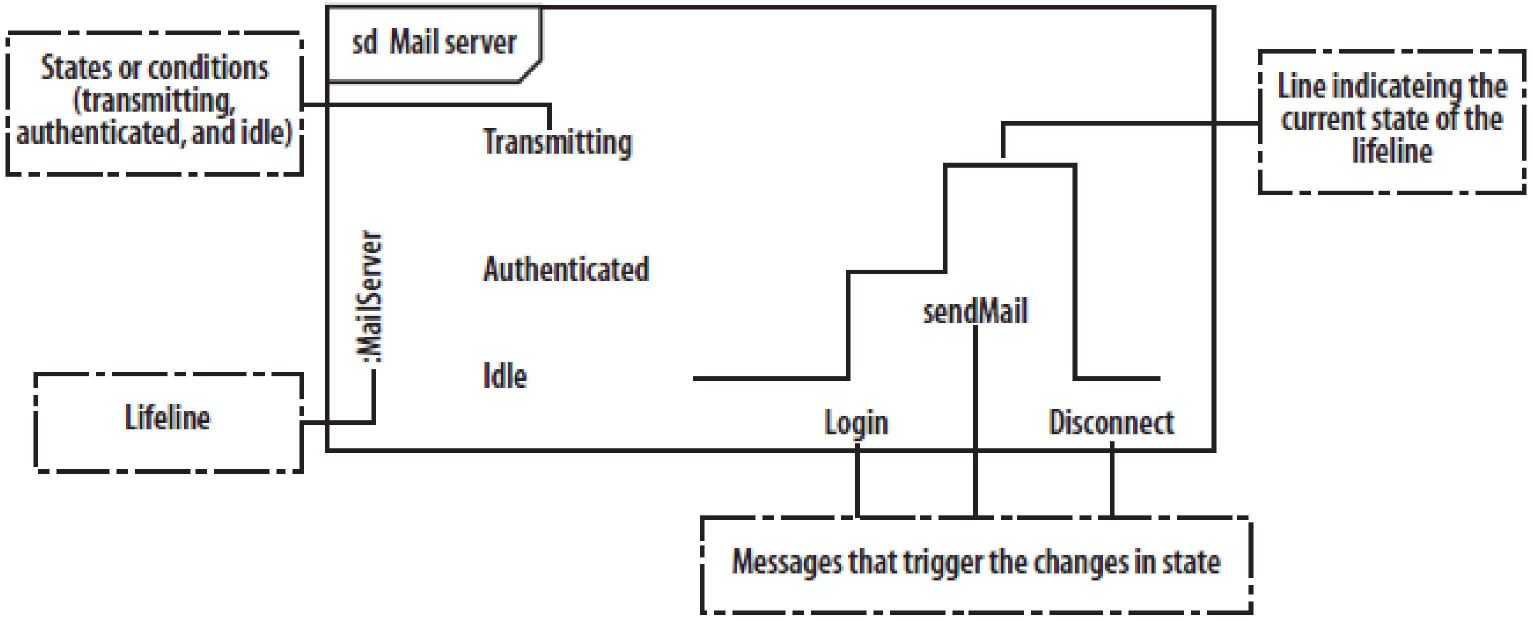
- Let's take a simple scenario based on the pump and hotplate for a coffee pot. Let's imagine a rule that says that at least 10 seconds must pass between the pump coming on and the hotplate coming on. When the water reservoir becomes empty, the pump switches off, and the hotplate cannot stay on for more than 15 minutes more.



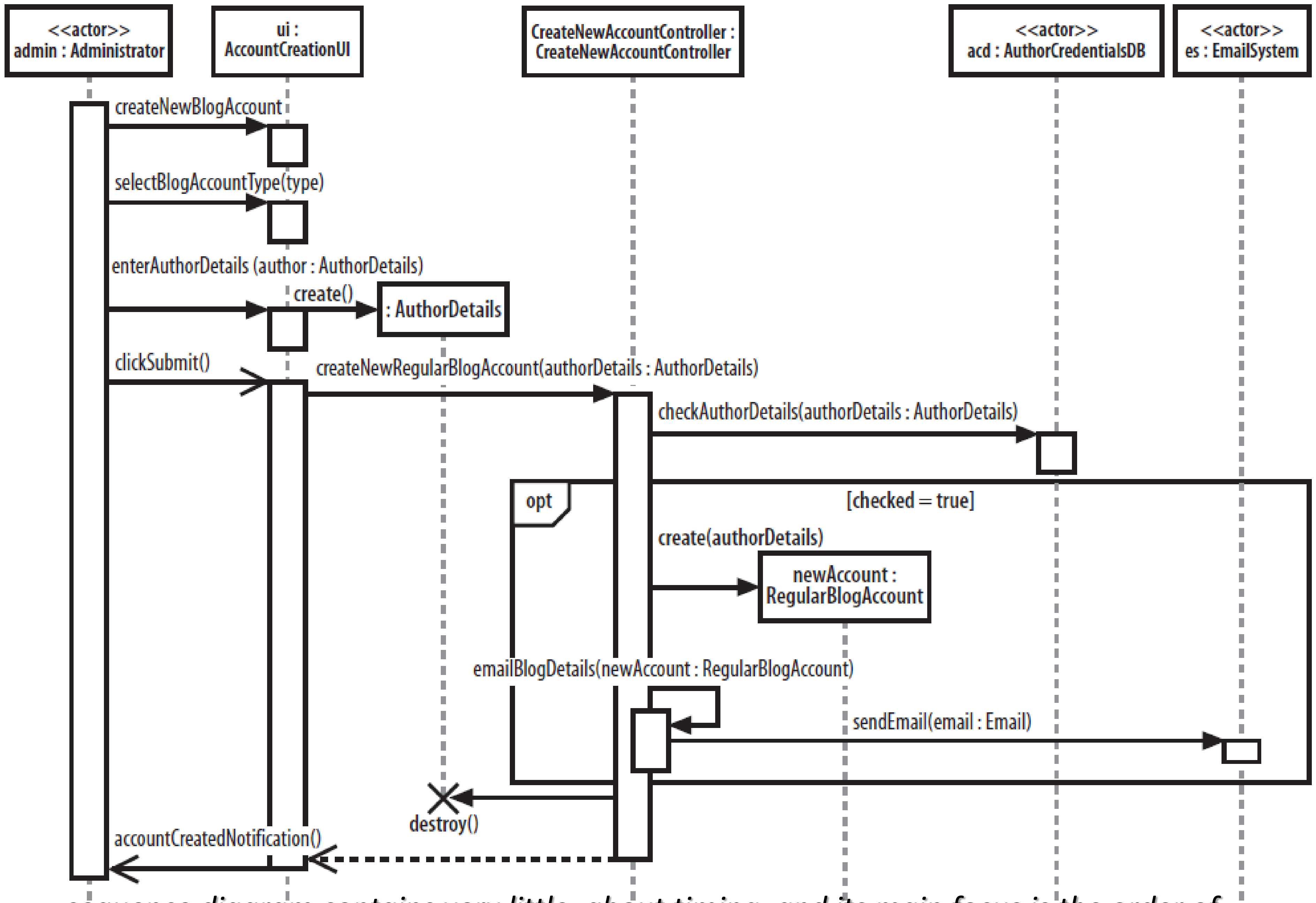
Ex 6:


timing diagram
sd doorControl
states
line showing state
lifeline
scanner
locked
waiting
open
state change
verify
lifeline
processor
idle
verifying
enabled
unlock
message
lock
lifeline
door
state
state
state
locked
unlocked
locked
0
5
10
time ticks

Ex 7: timing diagram for a mail server


States or conditions (transmitting, authenticated, and idle)
sd Mail server
Transmitting
Line indicateing the current state of the lifeline
:MailServer
Authenticated
sendMail
Idle
Lifeline
Login
Disconnect
Messages that trigger the changes in state

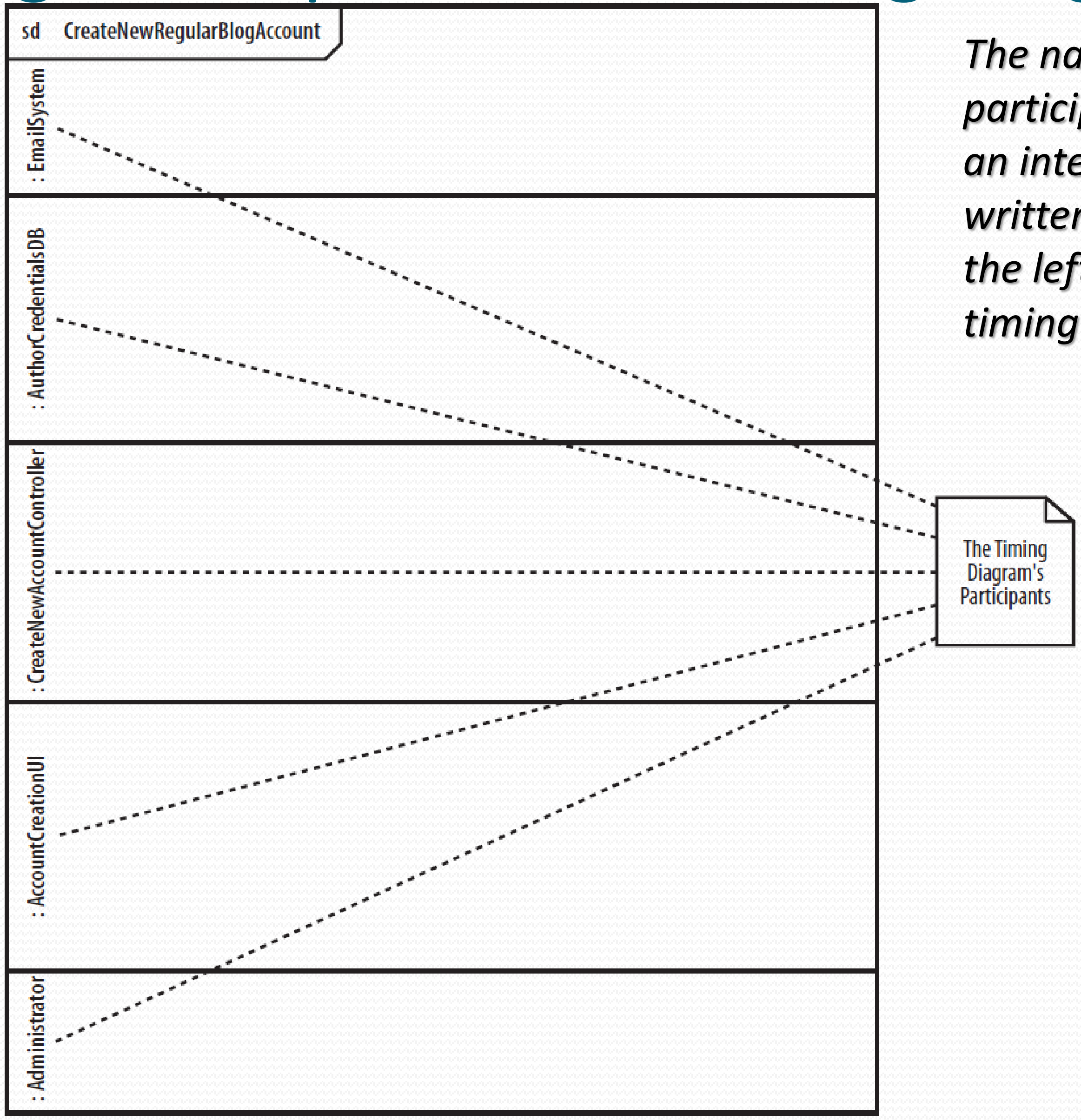
Building a Timing Diagram from a Sequence Diagram



<<actor>> admin : Administrator
ui : AccountCreationUI
CreateNewAccountController : CreateNewAccountController
<<actor>> acd : AuthorCredentialsDB
<<actor>> es : EmailSystem
createNewBlogAccount
selectBlogAccountType(type)
enterAuthorDetails (author : AuthorDetails)
create()
: AuthorDetails
clickSubmit()
createNewRegularBlogAccount(authorDetails : AuthorDetails)
checkAuthorDetails(authorDetails : AuthorDetails)
opt
[checked = true]
create(authorDetails)
newAccount : RegularBlogAccount
emailBlogDetails(newAccount : RegularBlogAccount)
sendEmail(email : Email)
destroy()
accountCreatedNotification()

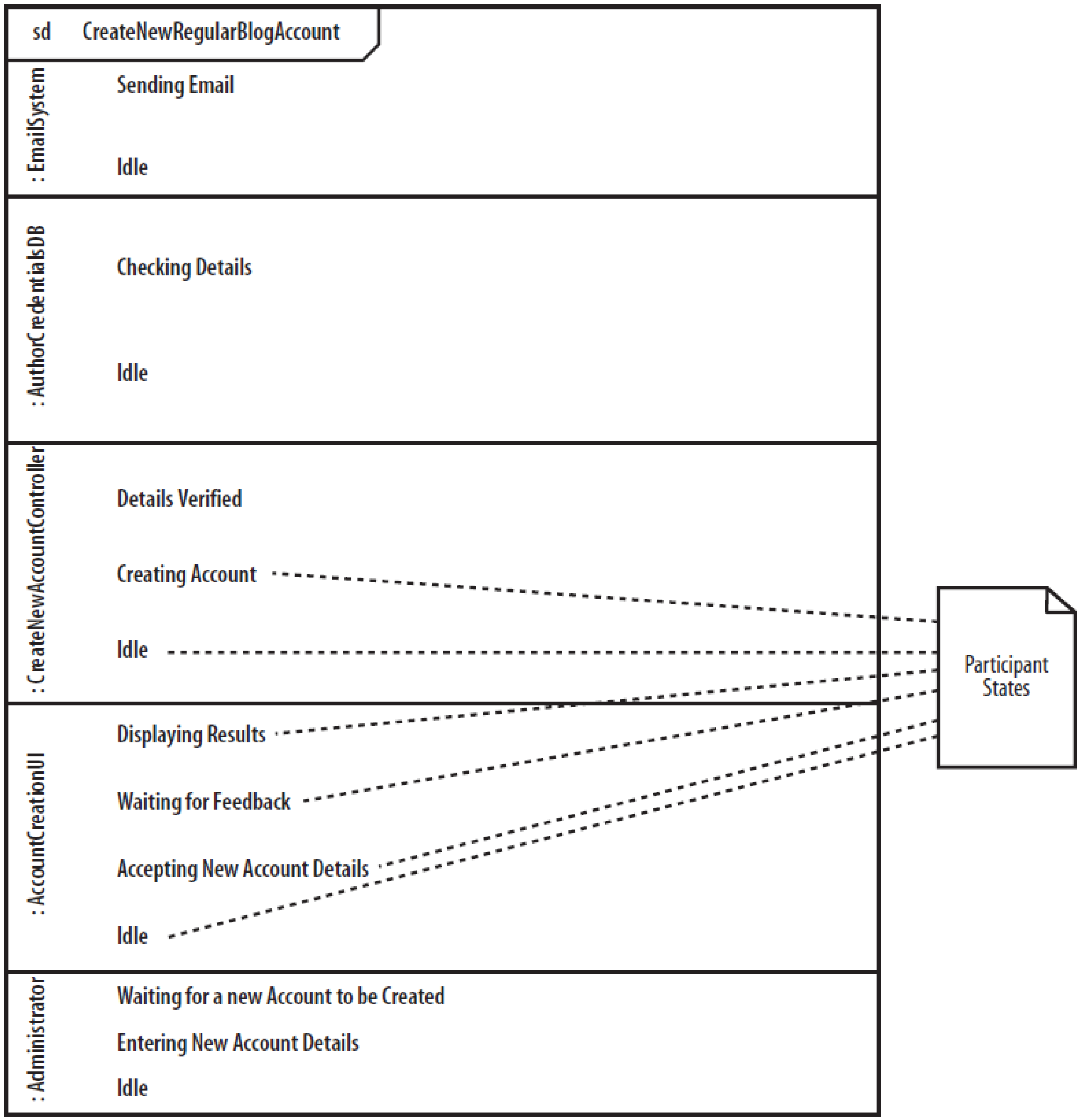
sequence diagram contains very little, about timing, and its main focus is the order of events within an interaction

Applying Participants to a Timing Diagram


sd CreateNewRegularBlogAccount
: EmailSystem
: AuthorCredentialsDB
: CreateNewAccountController
: AccountCreationUI
: Administrator
The Timing Diagram's Participants

The names of the main participants involved in an interaction are written vertically on the lefthand side of a timing diagram

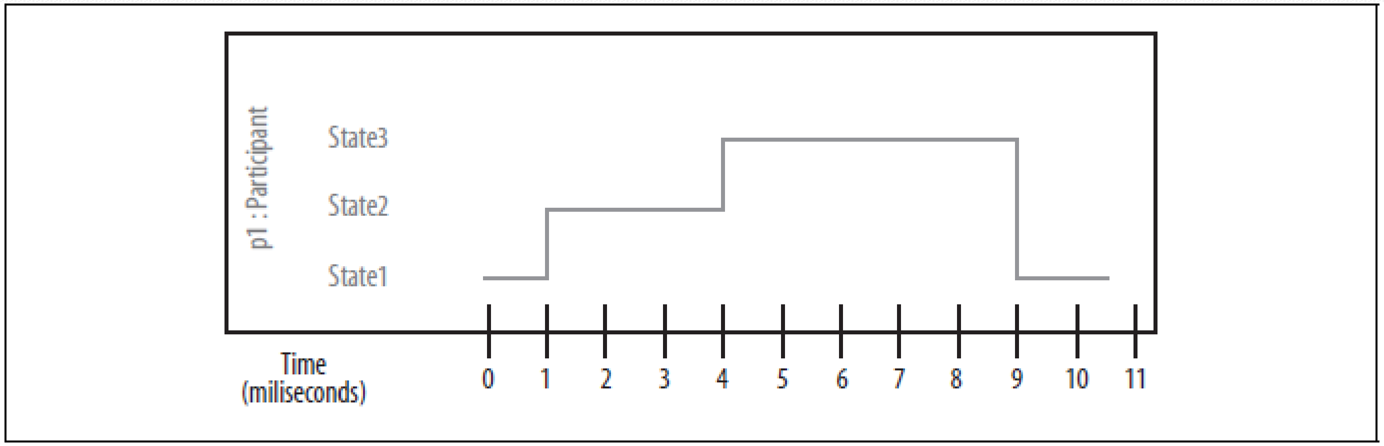
States



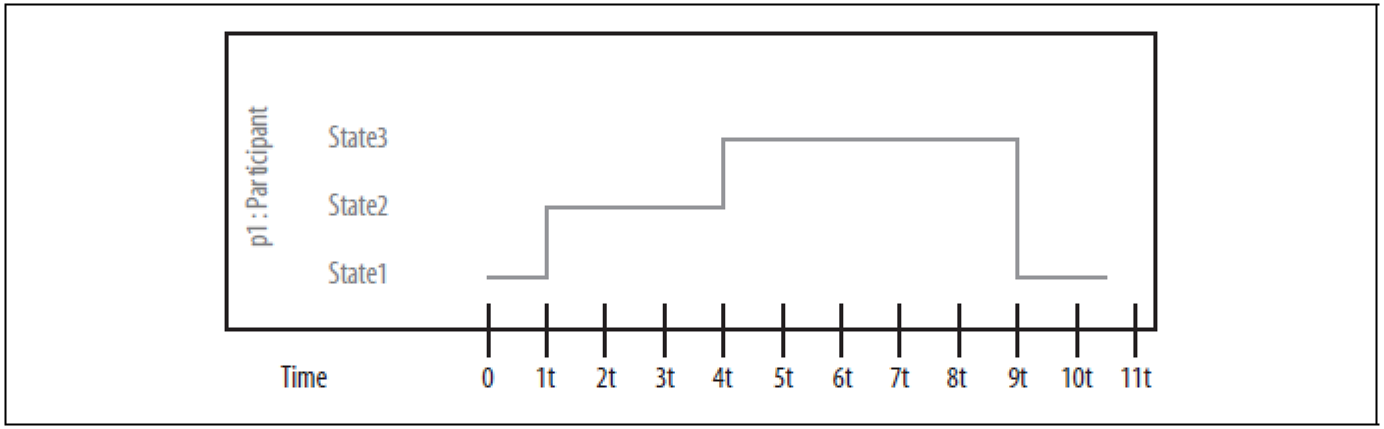
sd CreateNewRegularBlogAccount
: EmailSystem
Sending Email
Idle
: AuthorCredentialsDB
Checking Details
Idle
: CreateNewAccountController
Details Verified
Creating Account
Idle
: AccountCreationUI
Displaying Results
Waiting for Feedback
Accepting New Account Details
Idle
: Administrator
Waiting for a new Account to be Created
Entering New Account Details
Idle
Participant States

Exact Time and Relative Time Indicators

- Exact time measurements



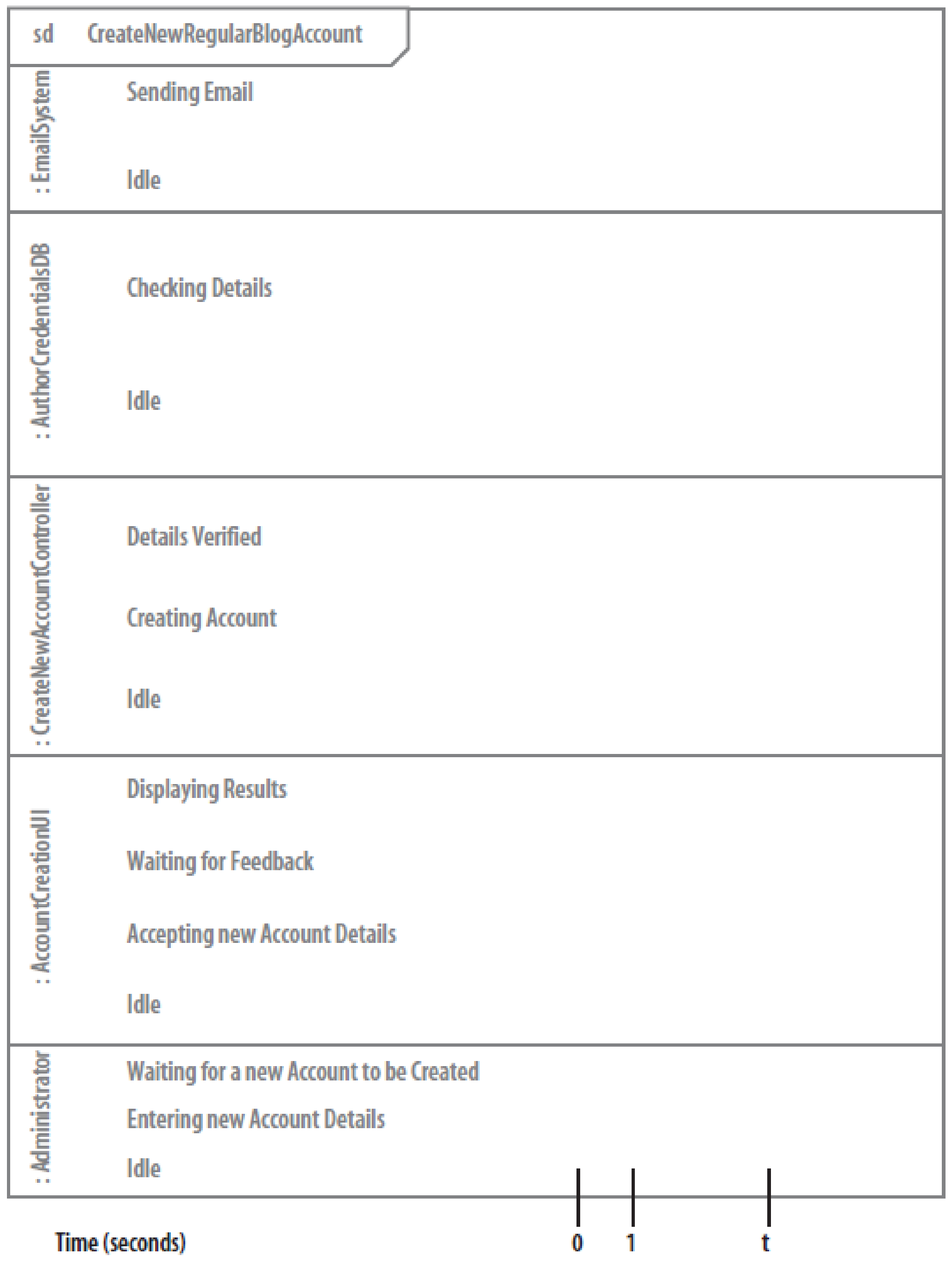
- Relative time indicators



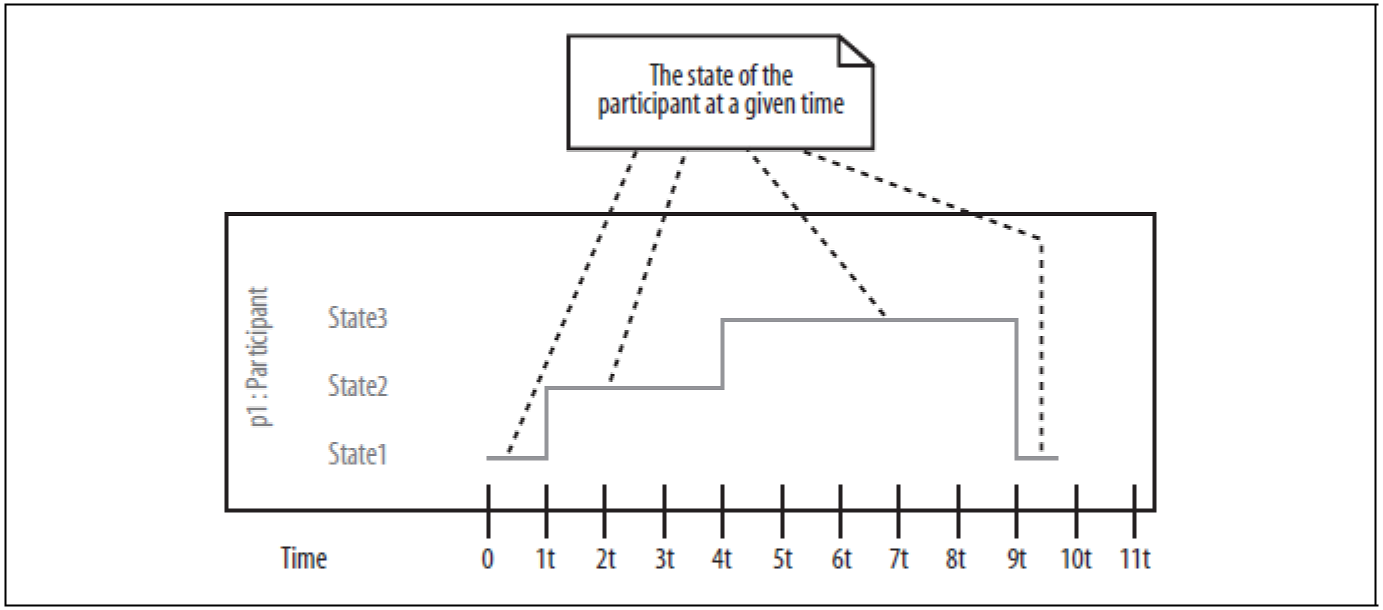
  - Relative time indicators are particularly useful when you have timing considerations such as "ParticipantA will be in State1 for half of the time that ParticipantB is in State2

Exact Time Measurements and Relative Time Indicators

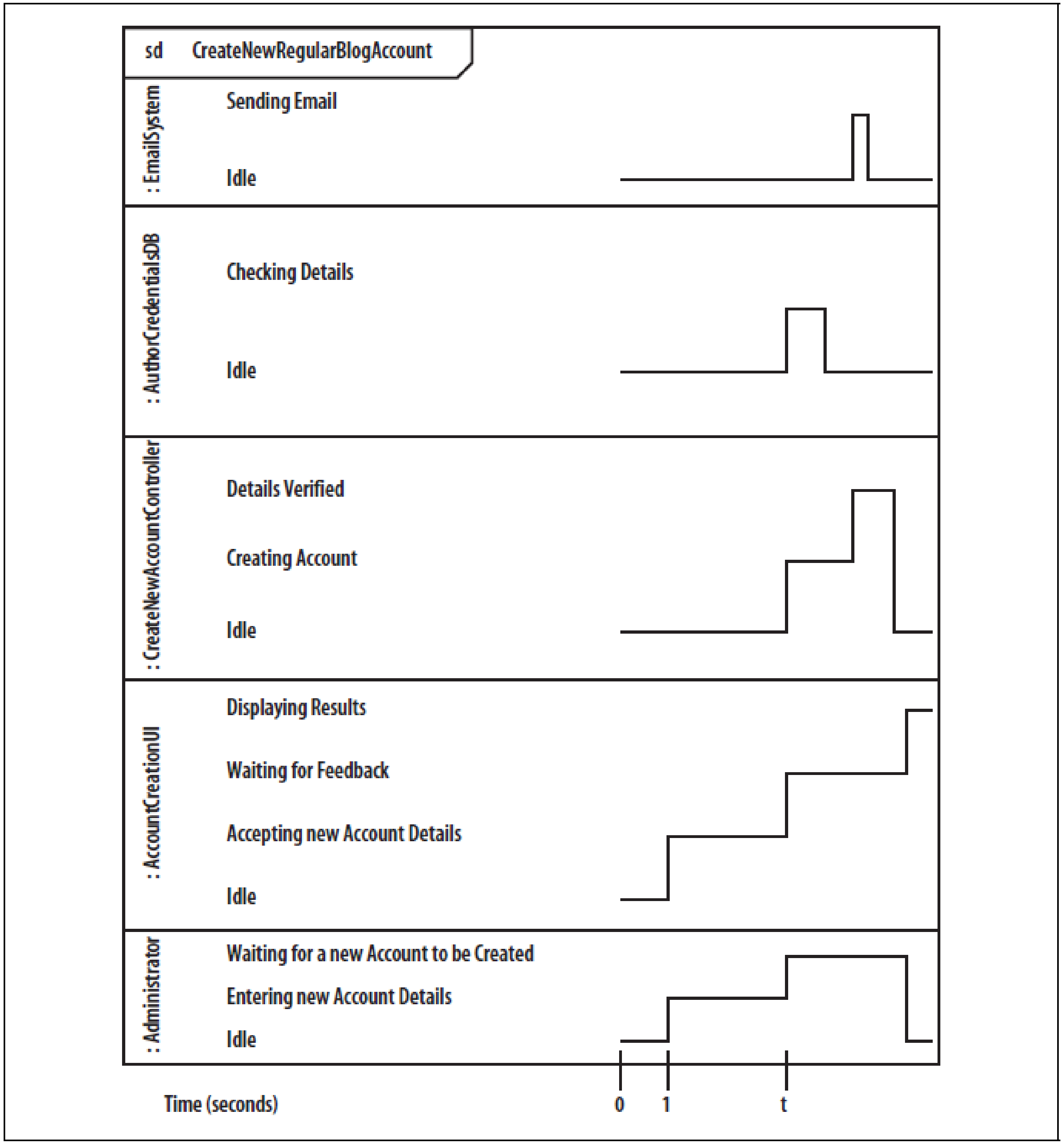
- In a timing diagram, t represents a point in time that is of interest. You don't know exactly when it will happen because it may happen in response to a message or event, but t is a way to refer to that moment without knowing exactly when it is. With t as a reference, you can then specify time constraints relative to that point t.
- Adding time to the diagram we've been putting together so far is tricky because we don't have any concrete timing information in the original requirement.
- Figure 9-8. The timing constraints are a blend of exact and relative timings -> next page


sd CreateNewRegularBlogAccount
: EmailSystem
Sending Email
Idle
: AuthorCredentialsDB
Checking Details
Idle
: CreateNewAccountController
Details Verified
Creating Account
Idle
: AccountCreationUI
Displaying Results
Waiting for Feedback
Accepting new Account Details
Idle
: Administrator
Waiting for a new Account to be Created
Entering new Account Details
Idle
Time (seconds)
0
1
t

A Participant's State-Line



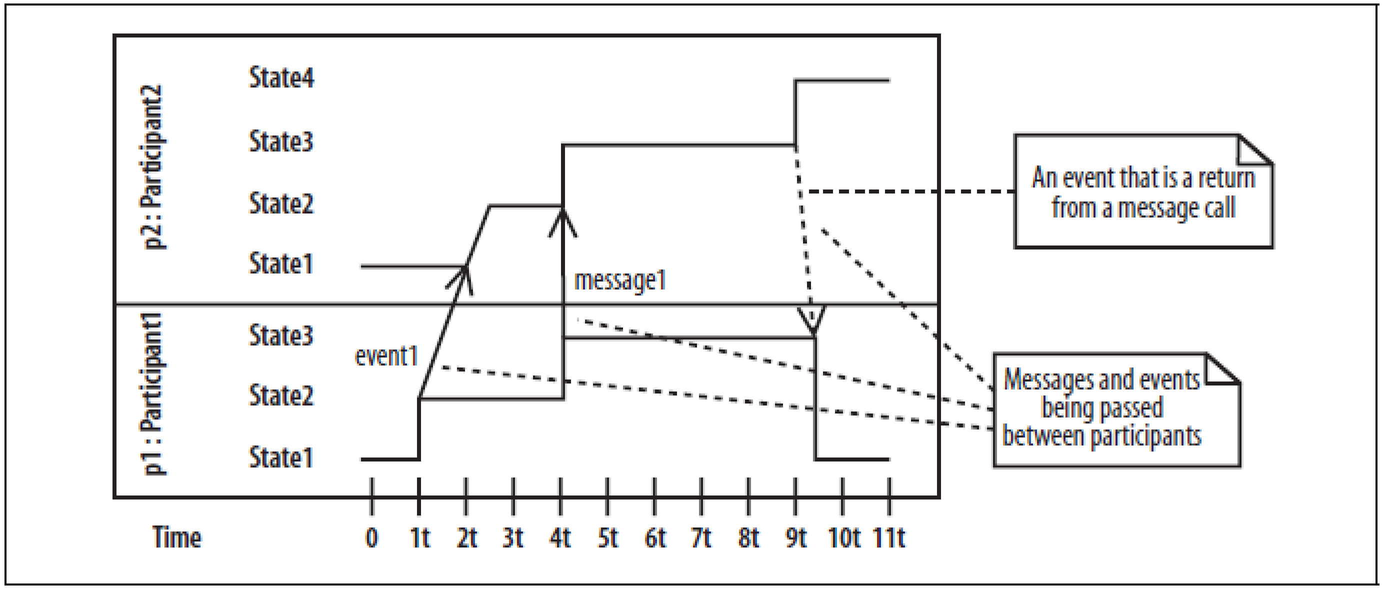
- *In this example, p1:Participant's state-line indicates that it is in State1 for 1 unit of time, State2 for three units of time, and State3 for roughly five units of time (before returning to State1 at the end of the interaction)*
- *Each of the participants needs to have a corresponding state-line to indicate their state at any given point in time -> next page*

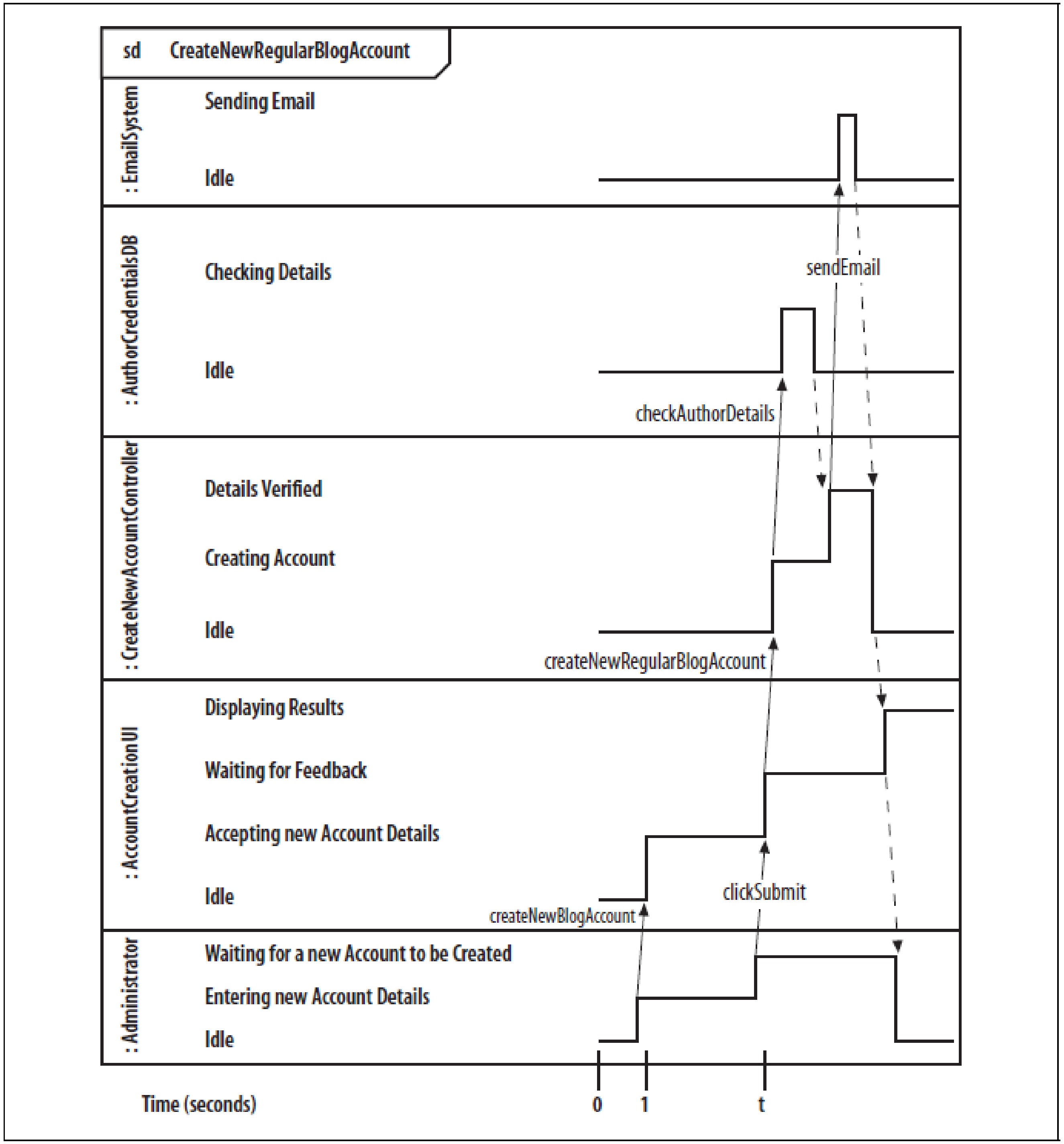

sd CreateNewRegularBlogAccount
: EmailSystem
Sending Email
Idle
: AuthorCredentialsDB
Checking Details
Idle
: CreateNewAccountController
Details Verified
Creating Account
Idle
: AccountCreationUI
Displaying Results
Waiting for Feedback
Accepting new Account Details
Idle
: Administrator
Waiting for a new Account to be Created
Entering new Account Details
Idle
Time (seconds)
0
1
t

Events and Messages

- The distinction between messages and events is not as important on a timing diagram as it is on sequence diagrams. The important thing to remember is that ***whatever the event is, it is shown on a timing diagram to trigger a change in the state*** of a participant.

- *Events on a timing diagram can even have their own durations, as shown by event1 taking 1 unit of time from invocation by p1:Participant1 and reception by p2:Participant2*




sd CreateNewRegularBlogAccount
: EmailSystem
Sending Email
Idle
sendEmail
: AuthorCredentialsDB
Checking Details
Idle
checkAuthorDetails
: CreateNewAccountController
Details Verified
Creating Account
Idle
createNewRegularBlogAccount
: AccountCreationUI
Displaying Results
Waiting for Feedback
Accepting new Account Details
Idle
createNewBlogAccount
clickSubmit
: Administrator
Waiting for a new Account to be Created
Entering new Account Details
Idle
Time (seconds)
0
1
t