Table 1: SE-Specific Cognitive Strategies Mapped To Software Development Tasks (Total Responses: 29)

Strategies	Responses	% Responses
Requirements Analysis	15	51.7
List System Entities, Classes, Data Inputs and Outputs, Business And System Events	11	37.9
List Corresponding Functions And Attributes For Each System Entity And Class Object	9	31.0
Determine User And System Interactions	8	27.5
List All Data, Variable And Control Dependencies	8	27.5
Create Prototypes To Clarify Requirements	7	24.1
Ask Questions What, When, Where, How, Who About Each System Entity, Class Object, User And Interaction	5	17.2
List Alternate Actions And States For Each System Role And Object	5	17.2
Map Specific Verb Phrases In Project Description with Functions or Events	4	13.7
List Do And Don't , Should And Shouldn't Scenarios	3	10.3
Identify Specific Words That Refer to Data or System Attributes In The Text Such As Speed, Volume, Space	2	6.8
Compare Meeting Notes To Identify Requirement Conflicts	2	6.8
Ask If-Then-Else Questions To Identify Alternate Scenarios For Each Entity, Class Object And User	1	3.4
Design	12	41.3
Compare Strengths and Weaknesses Of Design Alternatives	9	31.0
Conceptually Map Interactions Among System Entities, Class Objects To Clarify Design	8	27.5
Improve Design By Explaining And Prototyping	7	24.1
Determine Action Flows For Each System Role For Designing Interfaces	6	20.6
Create Map Of The Relationship Between Requirement, Scenarios, Contexts To Visualize Information For Design	6	20.6
Determine State Flows For Each Entity, Class Object To Visualize Information For Design	2	6.8
Graph Data And Control Dependencies To Clarify Design	1	3.4
Development	17	58.6
Use Online Forums To Learn Coding Solutions	15	51.7
Improve Knowledge Sharing By Using Collaborative Working Tools	7	24.1
Partition Development Tasks On The Basis Of Functions, Events	4	13.7
Create Tests Before Code	2	6.8
Use Standard Design Patterns	1	3.4
Use Comments To Explain and Understand Code	1	3.4
Develop Mock-Ups To Evaluate Solution	1	3.4
Dry Run Algorithm Before Coding To Check Correctness	1	3.4
Testing	13	44.8
Test Quality Attributes On Minimum, Maximum and In Between Measurements	10	34.4
Use Peer Reviews To Detect Bug Before Coding	8	27.5
Trace Tests With Requirements To Ensure Comprehensive Tests	8	27.5
Use Standards For Test Cases To Manage Test Quality	4	13.7
Perform Root-Cause Analysis To Find Improvement Areas	3	10.3
Describe Pre and Post Conditions For The Scenarios For Testing	1	3.4
Perform Exploratory Testing To Understand System	1	3.4
Project Management	20	68.9
Manage Tasks And Evaluate Performance By Using Scheduling Charts	14	48.2
Break Down Tasks In To Smaller Tasks To Manage Work	14	48.2
Create Task Dependencies For Scheduling	11	37.9
Group All Functionally Related Elements to Simplify Task	10	34.4
Compare Requirements To Assess Complexity	10	34.4
Categorize Tasks On The Basis Of Complexity	9	31.0
Conduct Periodic Meetings To Evaluate Task Accomplishment	8	27.5
List One-line Tests For Estimation	1	3.4
Create Naming Conventions To Manage Document and Code Versions	1	3.4
Use Sample Project Documents To Learn How To Develop Documents	1	3.4
Use Standard Templates For Documentation	1	3.4