Question-:

Upgrade an existing e-commerce platform to support international expansion. The project involves

adding new features and functionalities in phases, starting with basic localization and payment

options, and gradually expanding to support multiple languages and currencies.

### Task 1.1: Software Development Models

1. Waterfall – Linear, step-by-step (Requirements → Design → Coding → Testing → Deployment).
2. Agile – Iterative, flexible (Sprints, continuous feedback).
3. Spiral – Risk-focused, combines prototyping + iterative cycles.
4. V-Model – Testing at each stage (Verification & Validation).

### Task 1.2: Comparison of Models

| **Model** | **Best For** | **Risk Management** | **Constraints** |
| --- | --- | --- | --- |
| Waterfall | Stable, predictable projects | Low (late risk detection) | Rigid, no changes allowed |
| Agile | Dynamic, evolving projects | High (continuous testing) | Needs customer involvement |
| Spiral | High-risk, complex projects | Best (risk-driven cycles) | Expensive, time-consuming |
| V-Model | Safety-critical systems | Moderate (early testing) | Inflexible, no iterations |

### Task 1.3: Pros & Cons of Each Model

| **Model** | **Advantages** | **Limitations** |
| --- | --- | --- |
| Waterfall | Simple, clear milestones | No flexibility |
| Agile | Fast, adaptive, customer-focused | Requires high collaboration |
| Spiral | Strong risk control, scalable | Complex to manage |
| V-Model | Reliable, strict quality checks | Slow, no mid-project changes |

### Task 2.1: Key Software Development Processes

1. Requirements – Define features (e.g., multi-currency support).
2. Design – Plan architecture (e.g., payment gateway integration).
3. Coding – Build functionalities in phases.
4. Testing – Check for bugs (e.g., currency conversion accuracy).
5. Deployment – Roll out updates incrementally.
6. Maintenance – Fix issues, improve performance.

### Task 2.2: Repeatability & Best Practices

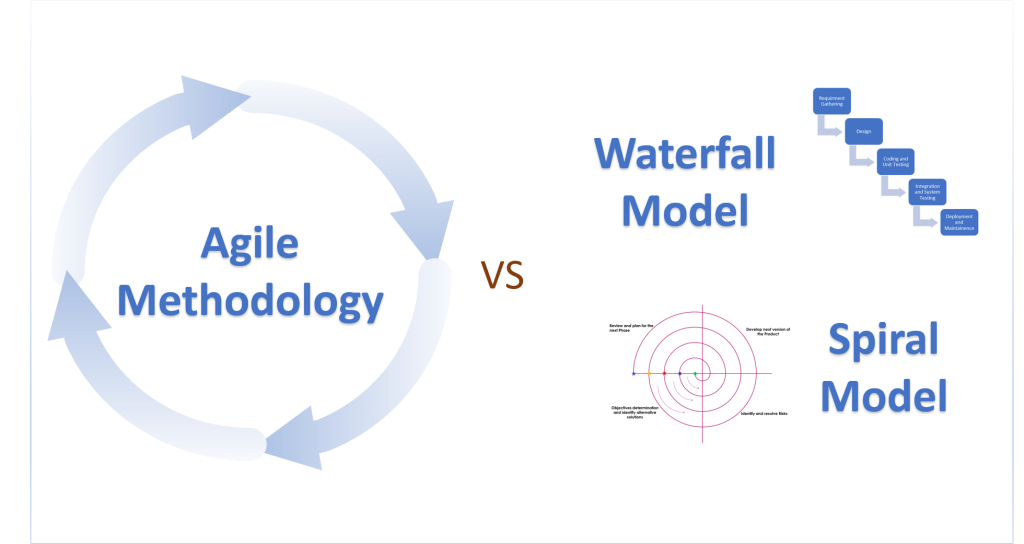
* Repeatability: Use templates (e.g., Agile sprints, test cases).
* Best Practices:
  + Modular coding for easy updates.
  + Automated testing for consistency.
  + CI/CD pipelines for smooth deployments.

### Task 2.3: Process Assurance Methods

* Standards: Follow ISO/IEC 12207 for lifecycle processes.
* Audits: Regular code/process reviews.
* Metrics: Track defects, deployment success rates.

### Recommendation for E-Commerce Upgrade

Use Agile (phased releases, adaptability) + Spiral (risk checks for payment/language integrations). Avoid Waterfall (too rigid) and V-Model (overkill for e-commerce).



### Task 3.1: Verification & Validation (V&V) Techniques

Verification (Checking if software is built correctly):

* Reviews & Inspections – Manual checks of requirements, design, code.
* Static Analysis – Automated code review (e.g., SonarQube).
* Walkthroughs – Team discussions to spot errors early.

Validation (Checking if the right software is built):

* Testing – Unit, integration, system, UAT (User Acceptance Testing).
* Prototyping – Early user feedback on features.
* Beta Testing – Real-world usage before final release.

### Task 3.2: Quality Control Case Studies in Software Models

1. Agile (Spotify) – Frequent testing, CI/CD pipelines, and automated checks ensure quality in rapid releases.
2. Waterfall (NASA’s Space Shuttle Software) – Rigorous V&V at each phase to prevent mission-critical failures.
3. Spiral (Microsoft Windows Development) – Risk-driven prototypes and iterative testing reduced OS crashes.
4. V-Model (Medical Software – FDA Compliance) – Strict validation for safety-critical systems.

### Task 3.3: Applying IEEE 1012-1998 Standards

* Requirements V&V – Traceability matrices to ensure all requirements are testable.
* Design V&V – Architecture reviews for scalability and security.
* Code V&V – Static/dynamic analysis to meet coding standards.
* Testing V&V – Test coverage metrics (e.g., 90%+ for critical modules).

Example: In an e-commerce upgrade, use IEEE 1012 to validate:

* Payment gateways (transaction accuracy).
* Localization (language/currency display correctness).

### Task 4.1: Quality Assurance (QA) Techniques

1. Peer Reviews – Code/design reviews by team members.
2. Automated Testing – Selenium (UI), JUnit (unit tests), LoadRunner (performance).
3. Audits – Compliance checks (e.g., PCI-DSS for payment security).

For E-Commerce Upgrade:

* Automate currency conversion tests.
* Audit third-party payment integrations.

### Task 4.2: Quality Control Checklists

Process Evaluation Checklist:  
 Q.Are requirements traceable to tests?  
 Q.Is code reviewed before merging?  
 Q.Are deployment rollbacks possible?

Product Evaluation Checklist:  
 Q.Do all payment methods work globally?  
 Q.Is multilingual UI error-free?  
 Q.Does the site handle peak traffic?

### Task 4.3: Real-World Quality Management Case Studies

1. Amazon’s Deployment Rollback (2006) – A failed database update caused outages; improved QA now includes phased rollouts.
2. Knight Capital Group (2012) – A software bug lost $460M in 45 mins due to poor testing; now a benchmark for financial QA.
3. Airbnb’s Localization Testing – Used crowdsourced testers to validate translations and regional payment methods.

### Key Takeaways

* Agile + Automation works best for e-commerce upgrades (speed + quality).
* IEEE 1012 ensures structured V&V for compliance.
* Checklists/Audits prevent costly errors in global expansions.

Recommendation: Combine Agile sprints (for phased feature rollouts) + Automated testing (for payment/localization checks) + Peer reviews (to maintain code quality).

n