Software Development Life Cycle

|  |  |
| --- | --- |
| Key concepts | Explore concepts’ significance and relevance |
| 1 | 2 |
| Software development life cycle is also referred as Application development life cycle  Application development life cycle is the term used in software engineering to describe the process of planning.  Requirement gathering and analysis  Design  implementation or coding  Testing  Deployment  Maintenance  It is a framework that describes the activities performed at each stage of a software development project.  Different approaches to life cycle methodology are: waterfall model, Prototyping Model, Spiral Model, Incremental Model RAD model. | SDLC and its phases contribute to produce high quality system that meet or exceed customer expectations, based on customer requirements, by delivering the system which move through each clearly defined phase, within scheduled time and estimated cost.  The software development lifecycle is not only a great way to ensure your app meets the needs of your business and customers, but it is also essential in supporting the app once it’s published.  By 2020, most SE course students will be working for an IT company. When working on a project, they must recall what kinds of SDLC’s were most feasible to follow and analyses why their company chose the model that they did. It could be an organizational choice, which means they choose the model that they feel is most compatible with their culture and objectives.  Students should remember throughout the lifetime that before doing a task or a project he/she should go through the phases of SDLC which are mentioned to reach ones goal and expectation.  Before starting a project Planning helps us to estimate the cost and time and gives a view of steps to be followed in the entire project. Analysis phase is to determine where the problem is, in an attempt to fix the system. Design phase will describe the new system as a collection of modules or subsystems. Implementation phase is the main implementing (coding) part. After product completion taking care of the system is very important. By following all these |

|  |  |
| --- | --- |
| Establish Relevance and Make Sense and Meaning | |
| Find Real-life contexts | Find Interdisciplinary connections |
| 3 | 4 |
| The systems development life-cycle concept applies to a range of hardware and software configurations, as a system can be composed of hardware only, software only, or a combination of both. Following the assembly line manufacturing concept, an SDLC goals to produce high quality systems that meet customer expectations, based on customer requirements, by delivering systems which move through each clearly well-defined and planned phases, within fixed time schedules and estimated costs.  Setting goals and milestones beforehand or planning ahead increases the chances of success at any given task. Since long term goals are the most difficult to achieve, people seek to short term goals, which will slowly but surely help them to move in the right direction towards their goal.  SDLC helps us apply this idea to system development, giving the team working on the project a process, composed of a number of clearly defined and distinct work phases to adhere to.  This way, the team is organized and their actions well-planned. Even when the going gets tough, they do not lose focus of his goals and objective and have a well-drawn out plan to fall back on. | For example a company has a goal to reach to that specified position in the market, the company then the company first notes it requirements and then analyze its strengthens and weakness and analyzes the same for their opponents. Then design a strategy to upgrade their position in the market and implement that strategy. These phases are not only used in technical activities but also it interconnects to many aspects in the society.  The SDLC process consists of six tailored phases that help manage a wide range of activity to conduct projects or automate House activities with information technology. SDLC is not limited to technical activity but it actually begins with customer needs and evolves through different processes and user requirements to develop a solution to any problem. The primary objective of implementing a standardized SDLC policy is to provide coordinated excellent service, at low costs, to support the activity of customers and users. |

|  |  |
| --- | --- |
| Engage in critical thinking | Technology, Tools, Techniques |
| 5 | 6 |
| Waterfall model: In this model each phase must be completed before we move into another phase. At the end of each phase a review process is schedules to check project is on the right track.  Spiral model: this model is similar to iterative model but it concentrates more on risk analysis. A preliminary design of the system is created.  RAD model: RAD stands for Rapid application development. This model is an incremental software development process model that concentrated more on short development cycle.  Agile model: By breaking the product into cycles, the [agile model](http://istqbexamcertification.com/wp-content/uploads/2012/01/Agile-model.jpg) quickly delivers a working product and is considered a very realistic development approach. | Tools used are HTML, CSS, jQUERY, java Script, bootstrap, php, MySQl.  The frontend has been made using basic html, css , java script and other libraries which are handled on the server side code written on php. |

|  |  |
| --- | --- |
| Plan Project management | Project Specification and Project Brief |
| 7 | 8 |
| We tried to accomplish this project by having joint sittings, planning on the tools, technology, features, functions to be used and the objectives that the project mush accomplish. And organized regular sessions in which we coded, documented, tested, debugged the web app. Major obstacles were planned to be tackled with the help of seniors, course in-charge and online resources. | SDLC phases helped us to show a path to perform the actions before implementation of the code. We first identified and gathered our requirements and specifications and then analyzed them and prepared SRS document and designed , then implemented the code leading us to the goal we wanted. |
|  | |

Requirement Analysis and Specification

|  |  |
| --- | --- |
| Key concepts | Explore concepts’ significance and relevance |
| 1 | 2 |
| Requirements analysis in [system](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering)  [engineering](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering) and [software engineering](https://en.wikipedia.org/wiki/Software_engineering), encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting [requirements](https://en.wikipedia.org/wiki/Requirement) of the various [stakeholders](https://en.wikipedia.org/wiki/Stakeholder_(corporate)), analyzing, documenting, validating and managing software or system requirements.  Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.  Approaches for requirement specification are:  SRS  Software design  Feasibility studies  Requirements elicitation and analysis  Requirements validation  Requirements management  Requirement Analysis and Specification concepts  Requirements Analysis: A process of discovery, refinement, modeling, and specification.  During the process, both the developers and customers take an active role.  Communication Techniques:  Initiating the Process  Facilitated Application Specification Techniques  Analysis Principles  Information Domain  Modeling  Partitioning  Software Prototyping  Selecting the Prototyping Approach  Prototyping Methods and Tools  The Software Requirements Specification  Specification Principles  Representation | If requirement analysis phase is not completed properly or in a sluggish manner then there might be inconsistencies in the final product. Let’s check out the possible mistakes.  During requirement gathering, Client is not able to provide more details or not sure exactly what is required. Basic reason is, client comes from a non-technical background and not familiar with technical jargons so they find it difficult to share the exact expectations. If requirements are not proper or not completely covered then there is no point of analysis. This will cause a loop of rework in case of development team.  Irregular Communication between the engaged parties. I know from my personal experience that whenever there are more than one parties involved and they don’t interact regularly at the requirement analysis phase, then there is a possibility of disagreement in later phases of development.  Timelines are not achievable. Client is always in a hurry to see the final product so they directly or indirectly plan unrealistic timelines and finally receive a half-baked product.  When the clients receive their final product, by then, expectations changes and they want to introduce more new features which were not part of earlier documents. This behavior can be expected during the project also which pushes the team back on their schedule.  Improper Documentation is another culprit which enhances the possibility of unachievable tasks.  Requirements are statements of what the system must do, how it must behave, the properties it must exhibit, the qualities it must possess, and the constraints that the system and its development must satisfy. The Institute of Electrical and Electronics Engineers (IEEE) defines a requirement as  a condition or capability needed by a user to solve a problem or achieve an objective  a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document  a documented representation of a condition or capability as in definition 1 or 2 [[IEEE 1990a](http://www.sei.cmu.edu/productlines/frame_report/frame_bib.htm#IEEE1990a)]  Requirements engineering emphasizes the use of systematic and repeatable techniques that ensure the completeness, consistency, and relevance of the system requirements [[Sommerville 1997a](http://www.sei.cmu.edu/productlines/frame_report/frame_bib.htm" \l "Sommerville1997a)]. Specifically, requirements engineering encompasses requirements elicitation, analysis, specification, verification, and management, where |

|  |  |
| --- | --- |
| Establish Relevance and Make Sense and Meaning | |
| Find Real-life contexts | Find Interdisciplinary connections |
| 3 | 4 |
| Requirement analysis is widely used in modeling and communication models and also in Real-Time Control Systems and almost all software engineering projects ranging from designing the control systems of aircrafts to automobiles and other light weight projects. | Requirement analysis would give clear view of and evaluation of the projects there by ensuring a systematic approach to implementations of the software. |

|  |  |
| --- | --- |
| Engage in critical thinking | Technology, Tools, Techniques |
| 5 | 6 |
| * A use case is a structure for documenting the functional requirements for a system, usually involving software, whether that is new or being changed. Each use case provides a set of *scenarios* that convey how the system should interact with a human user or another system, to achieve a specific business goal. Use cases typically avoid technical jargon, preferring instead the language of the [end-user](https://en.wikipedia.org/wiki/End-user) or [*domain expert*](https://en.wikipedia.org/wiki/Domain_expert). Use cases are often co-authored by requirements engineers and stakeholders. * Use cases are deceptively simple tools for describing the behavior of software or systems. A use case contains a textual description of the ways in which users are intended to work with the software or system. Use cases should not describe internal workings of the system, nor should they explain how that system will be implemented. Instead, they show the steps needed to perform a task without sequential assumptions. | * A natural inclination towards writing code can lead to implementation beginning before the requirements analysis is complete, potentially resulting in inelegant refactoring to meet actual requirements once they are known. * Technical personnel and end-users may have different vocabularies. Consequently, they may wrongly believe they are in perfect agreement until the finished product is supplied. * Engineers and developers may try to make the requirements fit an existing system or model, rather than develop a system specific to the needs of the client. * Analysis may often be carried out by engineers or programmers, rather than personnel with the domain knowledge to understand a client's needs properly.   Solutions:   * One attempted solution to communications problems has been to employ specialists in business or system analysis. * Techniques introduced in the 1990s like [prototyping](https://en.wikipedia.org/wiki/Software_prototyping), [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML), [use cases](https://en.wikipedia.org/wiki/Use_case), and [Agile software development](https://en.wikipedia.org/wiki/Agile_software_development) are also intended as solutions to problems encountered with previous methods. * Also, a new class of [application simulation](https://en.wikipedia.org/wiki/Application_Simulation_Software) or application definition tools have entered the market. These tools are designed to bridge the communication gap between business users and the IT organization — and also to allow applications to be 'test marketed' before any code is produced. The best of these tools offer: * [electronic whiteboards](https://en.wikipedia.org/wiki/Electronic_whiteboard) to sketch application flows and test alternatives * ability to capture business logic and data needs * ability to generate high fidelity prototypes that closely imitate the final application * interactivity * capability to add contextual requirements and other comments * ability for remote and distributed users to run and interact with the simulation |

|  |  |
| --- | --- |
| Plan Project management | Project Specification and Project Brief |
| 7 | 8 |
| * This is backed up by controlled experiments in a student context. One possible interpretations is that requirements and design phases do not need to be entirely documentation-driven and theoretical. Instead, performing prototyping as part of the requirement and design phases – which amounts to spending time on and thinking about the requirements – is going to affect project success and implementation time. * There is also plenty of other evidence that points in the same direction: spending time on preparing for implementation pays off in the form of less risk and less chance of schedule overrun due to surprises. Although the question was not about testing, proper preparation positively affects testing, too. * Requirement Gathering/Analysis: 20% UI/UX Design and Prototyping:: 20% Technical Architecture: 10% Main Engineering Effort: 30% SIT/QA/Bugs:: 20% | * Requirement Analysis phases helped us to show a path to perform the actions before implementation of the code. We first identified and gathered our requirements and specifications and then analyzed them and prepared SRS document and designed, then implemented the code leading us to the goal we wanted. * In the real world scenario this project will be beneficial to students who seek value in what they purchase rather than any other aspect . * Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.[[1]](https://en.wikipedia.org/wiki/Software_requirements_specification#cite_note-1) Software requirements specification prevents software projects from failure[[2]](https://en.wikipedia.org/wiki/Software_requirements_specification#cite_note-2) |
|  | |

**Software Project Scheduling**

|  |  |
| --- | --- |
| Key concepts | Explore concepts’ significance and relevance |
| 1 | 2 |
| Requirements analysis in [system](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering)  [engineering](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering) and [software engineering](https://en.wikipedia.org/wiki/Software_engineering), encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting [requirements](https://en.wikipedia.org/wiki/Requirement) of the various [stakeholders](https://en.wikipedia.org/wiki/Stakeholder_(corporate)), analyzing, documenting, validating and managing software or system requirements.  Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.  Approaches for requirement specification are:  SRS  Software design  Feasibility studies  Requirements elicitation and analysis  Requirements validation  Requirements management  Requirement Analysis and Specification concepts  Requirements Analysis: A process of discovery, refinement, modeling, and specification.  During the process, both the developers and customers take an active role.  Communication Techniques:  Initiating the Process  Facilitated Application Specification Techniques  Analysis Principles  Information Domain  Modeling  Partitioning  Software Prototyping  Selecting the Prototyping Approach  Prototyping Methods and Tools  The Software Requirements Specification  Specification Principles  Representation | If requirement analysis phase is not completed properly or in a sluggish manner then there might be inconsistencies in the final product. Let’s check out the possible mistakes.  During requirement gathering, Client is not able to provide more details or not sure exactly what is required. Basic reason is, client comes from a non-technical background and not familiar with technical jargons so they find it difficult to share the exact expectations. If requirements are not proper or not completely covered then there is no point of analysis. This will cause a loop of rework in case of development team.  Irregular Communication between the engaged parties. I know from my personal experience that whenever there are more than one parties involved and they don’t interact regularly at the requirement analysis phase, then there is a possibility of disagreement in later phases of development.  Timelines are not achievable. Client is always in a hurry to see the final product so they directly or indirectly plan unrealistic timelines and finally receive a half-baked product.  When the clients receive their final product, by then, expectations changes and they want to introduce more new features which were not part of earlier documents. This behavior can be expected during the project also which pushes the team back on their schedule.  Improper Documentation is another culprit which enhances the possibility of unachievable tasks.  Requirements are statements of what the system must do, how it must behave, the properties it must exhibit, the qualities it must possess, and the constraints that the system and its development must satisfy. The Institute of Electrical and Electronics Engineers (IEEE) defines a requirement as  a condition or capability needed by a user to solve a problem or achieve an objective  a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document  Requirements engineering emphasizes the use of systematic and repeatable techniques that ensure the completeness, consistency, and relevance of the system requirements.Specifically, requirements engineering encompasses requirements elicitation, analysis, specification, verification, and management, where |

|  |  |
| --- | --- |
| Establish Relevance and Make Sense and Meaning | |
| Find Real-life contexts | Find Interdisciplinary connections |
| 3 | 4 |
| Requirement analysis is widely used in modeling and communication models and also in Real-Time Control Systems and almost all software engineering projects ranging from designing the control systems of aircrafts to automobiles and other light weight projects. | Requirement analysis would give clear view of and evaluation of the projects there by ensuring a systematic approach to implementations of the software. |

|  |  |
| --- | --- |
| Engage in critical thinking | Technology, Tools, Techniques |
| 5 | 6 |
| * A use case is a structure for documenting the functional requirements for a system, usually involving software, whether that is new or being changed. Each use case provides a set of *scenarios* that convey how the system should interact with a human user or another system, to achieve a specific business goal. Use cases typically avoid technical jargon, preferring instead the language of the [end-user](https://en.wikipedia.org/wiki/End-user) or [*domain expert*](https://en.wikipedia.org/wiki/Domain_expert). Use cases are often co-authored by requirements engineers and stakeholders. * Use cases are deceptively simple tools for describing the behavior of software or systems. A use case contains a textual description of the ways in which users are intended to work with the software or system. Use cases should not describe internal workings of the system, nor should they explain how that system will be implemented. Instead, they show the steps needed to perform a task without sequential assumptions. | * A natural inclination towards writing code can lead to implementation beginning before the requirements analysis is complete, potentially resulting in inelegant refactoring to meet actual requirements once they are known. * Technical personnel and end-users may have different vocabularies. Consequently, they may wrongly believe they are in perfect agreement until the finished product is supplied. * Engineers and developers may try to make the requirements fit an existing system or model, rather than develop a system specific to the needs of the client. * Analysis may often be carried out by engineers or programmers, rather than personnel with the domain knowledge to understand a client's needs properly.   Solutions:   * One attempted solution to communications problems has been to employ specialists in business or system analysis. * Techniques introduced in the 1990s like [prototyping](https://en.wikipedia.org/wiki/Software_prototyping), [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML), [use cases](https://en.wikipedia.org/wiki/Use_case), and [Agile software development](https://en.wikipedia.org/wiki/Agile_software_development) are also intended as solutions to problems encountered with previous methods. * Also, a new class of [application simulation](https://en.wikipedia.org/wiki/Application_Simulation_Software) or application definition tools have entered the market. These tools are designed to bridge the communication gap between business users and the IT organization — and also to allow applications to be 'test marketed' before any code is produced. The best of these tools offer: * [electronic whiteboards](https://en.wikipedia.org/wiki/Electronic_whiteboard) to sketch application flows and test alternatives * ability to capture business logic and data needs * ability to generate high fidelity prototypes that closely imitate the final application * interactivity * capability to add contextual requirements and other comments * ability for remote and distributed users to run and interact with the simulation |

|  |  |
| --- | --- |
| Plan Project management | Project Specification and Project Brief |
| 7 | 8 |
| This is backed up by controlled experiments in a student context. One possible interpretations is that requirements and design phases do not need to be entirely documentation-driven and theoretical. Instead, performing prototyping as part of the requirement and design phases – which amounts to spending time on and thinking about the requirements – is going to affect project success and implementation time.  There is also plenty of other evidence that points in the same direction: spending time on preparing for implementation pays off in the form of less risk and less chance of schedule overrun due to surprises. Although the question was not about testing, proper preparation positively affects testing, too. | Software Project Scheduling   * phases helped us to show a path to perform the actions before implementation of the code. We first identified and gathered our requirements and specifications and then analyzed them and prepared SRS document and designed, then implemented the code leading us to the goal we wanted. * In the real world scenario this project will be beneficial to students who seek value in what they purchase rather than any other aspect . * Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.[[1]](https://en.wikipedia.org/wiki/Software_requirements_specification#cite_note-1) Software requirements specification prevents software projects from failure[[2]](https://en.wikipedia.org/wiki/Software_requirements_specification#cite_note-2) |
|  | |

Risk Management

|  |  |
| --- | --- |
| Key concepts | Explore concepts’ significance and relevance |
| 1 | 2 |
| Requirements analysis in [system](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering)  [engineering](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering) and [software engineering](https://en.wikipedia.org/wiki/Software_engineering), encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting [requirements](https://en.wikipedia.org/wiki/Requirement) of the various [stakeholders](https://en.wikipedia.org/wiki/Stakeholder_(corporate)), analyzing, documenting, validating and managing software or system requirements.  Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.  Approaches for requirement specification are:  SRS  Software design  Feasibility studies  Requirements elicitation and analysis  Requirements validation  Requirements management  Requirement Analysis and Specification concepts  Requirements Analysis: A process of discovery, refinement, modeling, and specification.  During the process, both the developers and customers take an active role.  Communication Techniques:  Initiating the Process  Facilitated Application Specification Techniques  Analysis Principles  Information Domain  Modeling  Partitioning  Software Prototyping  Selecting the Prototyping Approach  Prototyping Methods and Tools  The Software Requirements Specification  Specification Principles  Representation | If requirement analysis phase is not completed properly or in a sluggish manner then there might be inconsistencies in the final product. Let’s check out the possible mistakes.  During requirement gathering, Client is not able to provide more details or not sure exactly what is required. Basic reason is, client comes from a non-technical background and not familiar with technical jargons so they find it difficult to share the exact expectations. If requirements are not proper or not completely covered then there is no point of analysis. This will cause a loop of rework in case of development team.  Irregular Communication between the engaged parties. I know from my personal experience that whenever there are more than one parties involved and they don’t interact regularly at the requirement analysis phase, then there is a possibility of disagreement in later phases of development.  Timelines are not achievable. Client is always in a hurry to see the final product so they directly or indirectly plan unrealistic timelines and finally receive a half-baked product.  When the clients receive their final product, by then, expectations changes and they want to introduce more new features which were not part of earlier documents. This behavior can be expected during the project also which pushes the team back on their schedule.  Improper Documentation is another culprit which enhances the possibility of unachievable tasks.  Requirements are statements of what the system must do, how it must behave, the properties it must exhibit, the qualities it must possess, and the constraints that the system and its development must satisfy. The Institute of Electrical and Electronics Engineers (IEEE) defines a requirement as  a condition or capability needed by a user to solve a problem or achieve an objective  a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document  Requirements engineering emphasizes the use of systematic and repeatable techniques that ensure the completeness, consistency, and relevance of the system requirements.Specifically, requirements engineering encompasses requirements elicitation, analysis, specification, verification, and management, where |

|  |  |
| --- | --- |
| Establish Relevance and Make Sense and Meaning | |
| Find Real-life contexts | Find Interdisciplinary connections |
| 3 | 4 |
| Requirement analysis is widely used in modeling and communication models and also in Real-Time Control Systems and almost all software engineering projects ranging from designing the control systems of aircrafts to automobiles and other light weight projects. | Requirement analysis would give clear view of and evaluation of the projects there by ensuring a systematic approach to implementations of the software. |

|  |  |
| --- | --- |
| Engage in critical thinking | Technology, Tools, Techniques |
| 5 | 6 |
| * A use case is a structure for documenting the functional requirements for a system, usually involving software, whether that is new or being changed. Each use case provides a set of *scenarios* that convey how the system should interact with a human user or another system, to achieve a specific business goal. Use cases typically avoid technical jargon, preferring instead the language of the [end-user](https://en.wikipedia.org/wiki/End-user) or [*domain expert*](https://en.wikipedia.org/wiki/Domain_expert). Use cases are often co-authored by requirements engineers and stakeholders. * Use cases are deceptively simple tools for describing the behavior of software or systems. A use case contains a textual description of the ways in which users are intended to work with the software or system. Use cases should not describe internal workings of the system, nor should they explain how that system will be implemented. Instead, they show the steps needed to perform a task without sequential assumptions. | * A natural inclination towards writing code can lead to implementation beginning before the requirements analysis is complete, potentially resulting in inelegant refactoring to meet actual requirements once they are known. * Technical personnel and end-users may have different vocabularies. Consequently, they may wrongly believe they are in perfect agreement until the finished product is supplied. * Engineers and developers may try to make the requirements fit an existing system or model, rather than develop a system specific to the needs of the client. * Analysis may often be carried out by engineers or programmers, rather than personnel with the domain knowledge to understand a client's needs properly.   Solutions:   * One attempted solution to communications problems has been to employ specialists in business or system analysis. * Techniques introduced in the 1990s like [prototyping](https://en.wikipedia.org/wiki/Software_prototyping), [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML), [use cases](https://en.wikipedia.org/wiki/Use_case), and [Agile software development](https://en.wikipedia.org/wiki/Agile_software_development) are also intended as solutions to problems encountered with previous methods. * Also, a new class of [application simulation](https://en.wikipedia.org/wiki/Application_Simulation_Software) or application definition tools have entered the market. These tools are designed to bridge the communication gap between business users and the IT organization — and also to allow applications to be 'test marketed' before any code is produced. The best of these tools offer: * [electronic whiteboards](https://en.wikipedia.org/wiki/Electronic_whiteboard) to sketch application flows and test alternatives * ability to capture business logic and data needs * ability to generate high fidelity prototypes that closely imitate the final application * interactivity * capability to add contextual requirements and other comments * ability for remote and distributed users to run and interact with the simulation |

|  |  |
| --- | --- |
| Plan Project management | Project Specification and Project Brief |
| 7 | 8 |
| This is backed up by controlled experiments in a student context. One possible interpretations is that requirements and design phases do not need to be entirely documentation-driven and theoretical. Instead, performing prototyping as part of the requirement and design phases – which amounts to spending time on and thinking about the requirements – is going to affect project success and implementation time.  There is also plenty of other evidence that points in the same direction: spending time on preparing for implementation pays off in the form of less risk and less chance of schedule overrun due to surprises. Although the question was not about testing, proper preparation positively affects testing, too. | Risk management   * phases helped us to show a path to perform the actions before implementation of the code. We first identified and gathered our requirements and specifications and then analyzed them and prepared SRS document and designed, then implemented the code leading us to the goal we wanted. * In the real world scenario this project will be beneficial to students who seek value in what they purchase rather than any other aspect . * Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.[[1]](https://en.wikipedia.org/wiki/Software_requirements_specification#cite_note-1) Software requirements specification prevents software projects from failure[[2]](https://en.wikipedia.org/wiki/Software_requirements_specification#cite_note-2) |
|  | |

Software Design Methodologies

|  |  |
| --- | --- |
| Key concepts | Explore concepts’ significance and relevance |
| 1 | 2 |
| Requirements analysis in [system](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering)  [engineering](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering) and [software engineering](https://en.wikipedia.org/wiki/Software_engineering), encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting [requirements](https://en.wikipedia.org/wiki/Requirement) of the various [stakeholders](https://en.wikipedia.org/wiki/Stakeholder_(corporate)), analyzing, documenting, validating and managing software or system requirements.  Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.  Approaches for requirement specification are:  SRS  Software design  Feasibility studies  Requirements elicitation and analysis  Requirements validation  Requirements management  Requirement Analysis and Specification concepts  Requirements Analysis: A process of discovery, refinement, modeling, and specification.  During the process, both the developers and customers take an active role.  Communication Techniques:  Initiating the Process  Facilitated Application Specification Techniques  Analysis Principles  Information Domain  Modeling  Partitioning  Software Prototyping  Selecting the Prototyping Approach  Prototyping Methods and Tools  The Software Requirements Specification  Specification Principles  Representation | If requirement analysis phase is not completed properly or in a sluggish manner then there might be inconsistencies in the final product. Let’s check out the possible mistakes.  During requirement gathering, Client is not able to provide more details or not sure exactly what is required. Basic reason is, client comes from a non-technical background and not familiar with technical jargons so they find it difficult to share the exact expectations. If requirements are not proper or not completely covered then there is no point of analysis. This will cause a loop of rework in case of development team.  Irregular Communication between the engaged parties. I know from my personal experience that whenever there are more than one parties involved and they don’t interact regularly at the requirement analysis phase, then there is a possibility of disagreement in later phases of development.  Timelines are not achievable. Client is always in a hurry to see the final product so they directly or indirectly plan unrealistic timelines and finally receive a half-baked product.  When the clients receive their final product, by then, expectations changes and they want to introduce more new features which were not part of earlier documents. This behavior can be expected during the project also which pushes the team back on their schedule.  Improper Documentation is another culprit which enhances the possibility of unachievable tasks.  Requirements are statements of what the system must do, how it must behave, the properties it must exhibit, the qualities it must possess, and the constraints that the system and its development must satisfy. The Institute of Electrical and Electronics Engineers (IEEE) defines a requirement as  a condition or capability needed by a user to solve a problem or achieve an objective  a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document  Requirements engineering emphasizes the use of systematic and repeatable techniques that ensure the completeness, consistency, and relevance of the system requirements.Specifically, requirements engineering encompasses requirements elicitation, analysis, specification, verification, and management, where |

|  |  |
| --- | --- |
| Establish Relevance and Make Sense and Meaning | |
| Find Real-life contexts | Find Interdisciplinary connections |
| 3 | 4 |
| Requirement analysis is widely used in modeling and communication models and also in Real-Time Control Systems and almost all software engineering projects ranging from designing the control systems of aircrafts to automobiles and other light weight projects. | Requirement analysis would give clear view of and evaluation of the projects there by ensuring a systematic approach to implementations of the software. |

|  |  |
| --- | --- |
| Engage in critical thinking | Technology, Tools, Techniques |
| 5 | 6 |
| * A use case is a structure for documenting the functional requirements for a system, usually involving software, whether that is new or being changed. Each use case provides a set of *scenarios* that convey how the system should interact with a human user or another system, to achieve a specific business goal. Use cases typically avoid technical jargon, preferring instead the language of the [end-user](https://en.wikipedia.org/wiki/End-user) or [*domain expert*](https://en.wikipedia.org/wiki/Domain_expert). Use cases are often co-authored by requirements engineers and stakeholders. * Use cases are deceptively simple tools for describing the behavior of software or systems. A use case contains a textual description of the ways in which users are intended to work with the software or system. Use cases should not describe internal workings of the system, nor should they explain how that system will be implemented. Instead, they show the steps needed to perform a task without sequential assumptions. | * A natural inclination towards writing code can lead to implementation beginning before the requirements analysis is complete, potentially resulting in inelegant refactoring to meet actual requirements once they are known. * Technical personnel and end-users may have different vocabularies. Consequently, they may wrongly believe they are in perfect agreement until the finished product is supplied. * Engineers and developers may try to make the requirements fit an existing system or model, rather than develop a system specific to the needs of the client. * Analysis may often be carried out by engineers or programmers, rather than personnel with the domain knowledge to understand a client's needs properly.   Solutions:   * One attempted solution to communications problems has been to employ specialists in business or system analysis. * Techniques introduced in the 1990s like [prototyping](https://en.wikipedia.org/wiki/Software_prototyping), [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML), [use cases](https://en.wikipedia.org/wiki/Use_case), and [Agile software development](https://en.wikipedia.org/wiki/Agile_software_development) are also intended as solutions to problems encountered with previous methods. * Also, a new class of [application simulation](https://en.wikipedia.org/wiki/Application_Simulation_Software) or application definition tools have entered the market. These tools are designed to bridge the communication gap between business users and the IT organization — and also to allow applications to be 'test marketed' before any code is produced. The best of these tools offer: * [electronic whiteboards](https://en.wikipedia.org/wiki/Electronic_whiteboard) to sketch application flows and test alternatives * ability to capture business logic and data needs * ability to generate high fidelity prototypes that closely imitate the final application * interactivity * capability to add contextual requirements and other comments * ability for remote and distributed users to run and interact with the simulation |

|  |  |
| --- | --- |
| Plan Project management | Project Specification and Project Brief |
| 7 | 8 |
| This is backed up by controlled experiments in a student context. One possible interpretations is that requirements and design phases do not need to be entirely documentation-driven and theoretical. Instead, performing prototyping as part of the requirement and design phases – which amounts to spending time on and thinking about the requirements – is going to affect project success and implementation time.  There is also plenty of other evidence that points in the same direction: spending time on preparing for implementation pays off in the form of less risk and less chance of schedule overrun due to surprises. Although the question was not about testing, proper preparation positively affects testing, too. | Software Project Scheduling   * phases helped us to show a path to perform the actions before implementation of the code. We first identified and gathered our requirements and specifications and then analyzed them and prepared SRS document and designed, then implemented the code leading us to the goal we wanted. * In the real world scenario this project will be beneficial to students who seek value in what they purchase rather than any other aspect . * Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.[[1]](https://en.wikipedia.org/wiki/Software_requirements_specification#cite_note-1) Software requirements specification prevents software projects from failure[[2]](https://en.wikipedia.org/wiki/Software_requirements_specification#cite_note-2) |
|  | |

Software Reliability and Quality Assurance

|  |  |
| --- | --- |
| Key concepts | Explore concepts’ significance and relevance |
| 1 | 2 |
| Requirements analysis in [system](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering)  [engineering](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering) and [software engineering](https://en.wikipedia.org/wiki/Software_engineering), encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting [requirements](https://en.wikipedia.org/wiki/Requirement) of the various [stakeholders](https://en.wikipedia.org/wiki/Stakeholder_(corporate)), analyzing, documenting, validating and managing software or system requirements.  Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.  Approaches for requirement specification are:  SRS  Software design  Feasibility studies  Requirements elicitation and analysis  Requirements validation  Requirements management  Requirement Analysis and Specification concepts  Requirements Analysis: A process of discovery, refinement, modeling, and specification.  During the process, both the developers and customers take an active role.  Communication Techniques:  Initiating the Process  Facilitated Application Specification Techniques  Analysis Principles  Information Domain  Modeling  Partitioning  Software Prototyping  Selecting the Prototyping Approach  Prototyping Methods and Tools  The Software Requirements Specification  Specification Principles  Representation | If requirement analysis phase is not completed properly or in a sluggish manner then there might be inconsistencies in the final product. Let’s check out the possible mistakes.  During requirement gathering, Client is not able to provide more details or not sure exactly what is required. Basic reason is, client comes from a non-technical background and not familiar with technical jargons so they find it difficult to share the exact expectations. If requirements are not proper or not completely covered then there is no point of analysis. This will cause a loop of rework in case of development team.  Irregular Communication between the engaged parties. I know from my personal experience that whenever there are more than one parties involved and they don’t interact regularly at the requirement analysis phase, then there is a possibility of disagreement in later phases of development.  Timelines are not achievable. Client is always in a hurry to see the final product so they directly or indirectly plan unrealistic timelines and finally receive a half-baked product.  When the clients receive their final product, by then, expectations changes and they want to introduce more new features which were not part of earlier documents. This behavior can be expected during the project also which pushes the team back on their schedule.  Improper Documentation is another culprit which enhances the possibility of unachievable tasks.  Requirements are statements of what the system must do, how it must behave, the properties it must exhibit, the qualities it must possess, and the constraints that the system and its development must satisfy. The Institute of Electrical and Electronics Engineers (IEEE) defines a requirement as  a condition or capability needed by a user to solve a problem or achieve an objective  a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document  Requirements engineering emphasizes the use of systematic and repeatable techniques that ensure the completeness, consistency, and relevance of the system requirements.Specifically, requirements engineering encompasses requirements elicitation, analysis, specification, verification, and management, where |

|  |  |
| --- | --- |
| Establish Relevance and Make Sense and Meaning | |
| Find Real-life contexts | Find Interdisciplinary connections |
| 3 | 4 |
| Requirement analysis is widely used in modeling and communication models and also in Real-Time Control Systems and almost all software engineering projects ranging from designing the control systems of aircrafts to automobiles and other light weight projects. | Requirement analysis would give clear view of and evaluation of the projects there by ensuring a systematic approach to implementations of the software. |

|  |  |
| --- | --- |
| Engage in critical thinking | Technology, Tools, Techniques |
| 5 | 6 |
| * A use case is a structure for documenting the functional requirements for a system, usually involving software, whether that is new or being changed. Each use case provides a set of *scenarios* that convey how the system should interact with a human user or another system, to achieve a specific business goal. Use cases typically avoid technical jargon, preferring instead the language of the [end-user](https://en.wikipedia.org/wiki/End-user) or [*domain expert*](https://en.wikipedia.org/wiki/Domain_expert). Use cases are often co-authored by requirements engineers and stakeholders. * Use cases are deceptively simple tools for describing the behavior of software or systems. A use case contains a textual description of the ways in which users are intended to work with the software or system. Use cases should not describe internal workings of the system, nor should they explain how that system will be implemented. Instead, they show the steps needed to perform a task without sequential assumptions. | * A natural inclination towards writing code can lead to implementation beginning before the requirements analysis is complete, potentially resulting in inelegant refactoring to meet actual requirements once they are known. * Technical personnel and end-users may have different vocabularies. Consequently, they may wrongly believe they are in perfect agreement until the finished product is supplied. * Engineers and developers may try to make the requirements fit an existing system or model, rather than develop a system specific to the needs of the client. * Analysis may often be carried out by engineers or programmers, rather than personnel with the domain knowledge to understand a client's needs properly.   Solutions:   * One attempted solution to communications problems has been to employ specialists in business or system analysis. * Techniques introduced in the 1990s like [prototyping](https://en.wikipedia.org/wiki/Software_prototyping), [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML), [use cases](https://en.wikipedia.org/wiki/Use_case), and [Agile software development](https://en.wikipedia.org/wiki/Agile_software_development) are also intended as solutions to problems encountered with previous methods. * Also, a new class of [application simulation](https://en.wikipedia.org/wiki/Application_Simulation_Software) or application definition tools have entered the market. These tools are designed to bridge the communication gap between business users and the IT organization — and also to allow applications to be 'test marketed' before any code is produced. The best of these tools offer: * [electronic whiteboards](https://en.wikipedia.org/wiki/Electronic_whiteboard) to sketch application flows and test alternatives * ability to capture business logic and data needs * ability to generate high fidelity prototypes that closely imitate the final application * interactivity * capability to add contextual requirements and other comments * ability for remote and distributed users to run and interact with the simulation |

|  |  |
| --- | --- |
| Plan Project management | Project Specification and Project Brief |
| 7 | 8 |
| This is backed up by controlled experiments in a student context. One possible interpretations is that requirements and design phases do not need to be entirely documentation-driven and theoretical. Instead, performing prototyping as part of the requirement and design phases – which amounts to spending time on and thinking about the requirements – is going to affect project success and implementation time.  There is also plenty of other evidence that points in the same direction: spending time on preparing for implementation pays off in the form of less risk and less chance of schedule overrun due to surprises. Although the question was not about testing, proper preparation positively affects testing, too. | * Software Reliability and Quality Assurance phases helped us to show a path to perform the actions before implementation of the code. We first identified and gathered our requirements and specifications and then analyzed them and prepared SRS document and designed, then implemented the code leading us to the goal we wanted. * In the real world scenario this project will be beneficial to students who seek value in what they purchase rather than any other aspect . * Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.[[1]](https://en.wikipedia.org/wiki/Software_requirements_specification#cite_note-1) Software requirements specification prevents software projects from failure[[2]](https://en.wikipedia.org/wiki/Software_requirements_specification#cite_note-2) |
|  | |

Software Maintenance

|  |  |
| --- | --- |
| Key concepts | Explore concepts’ significance and relevance |
| 1 | 2 |
| Requirements analysis in [system](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering)  [engineering](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering) and [software engineering](https://en.wikipedia.org/wiki/Software_engineering), encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting [requirements](https://en.wikipedia.org/wiki/Requirement) of the various [stakeholders](https://en.wikipedia.org/wiki/Stakeholder_(corporate)), analyzing, documenting, validating and managing software or system requirements.  Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.  Approaches for requirement specification are:  SRS  Software design  Feasibility studies  Requirements elicitation and analysis  Requirements validation  Requirements management  Requirement Analysis and Specification concepts  Requirements Analysis: A process of discovery, refinement, modeling, and specification.  During the process, both the developers and customers take an active role.  Communication Techniques:  Initiating the Process  Facilitated Application Specification Techniques  Analysis Principles  Information Domain  Modeling  Partitioning  Software Prototyping  Selecting the Prototyping Approach  Prototyping Methods and Tools  The Software Requirements Specification  Specification Principles  Representation | If requirement analysis phase is not completed properly or in a sluggish manner then there might be inconsistencies in the final product. Let’s check out the possible mistakes.  During requirement gathering, Client is not able to provide more details or not sure exactly what is required. Basic reason is, client comes from a non-technical background and not familiar with technical jargons so they find it difficult to share the exact expectations. If requirements are not proper or not completely covered then there is no point of analysis. This will cause a loop of rework in case of development team.  Irregular Communication between the engaged parties. I know from my personal experience that whenever there are more than one parties involved and they don’t interact regularly at the requirement analysis phase, then there is a possibility of disagreement in later phases of development.  Timelines are not achievable. Client is always in a hurry to see the final product so they directly or indirectly plan unrealistic timelines and finally receive a half-baked product.  When the clients receive their final product, by then, expectations changes and they want to introduce more new features which were not part of earlier documents. This behavior can be expected during the project also which pushes the team back on their schedule.  Improper Documentation is another culprit which enhances the possibility of unachievable tasks.  Requirements are statements of what the system must do, how it must behave, the properties it must exhibit, the qualities it must possess, and the constraints that the system and its development must satisfy. The Institute of Electrical and Electronics Engineers (IEEE) defines a requirement as  a condition or capability needed by a user to solve a problem or achieve an objective  a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document  Requirements engineering emphasizes the use of systematic and repeatable techniques that ensure the completeness, consistency, and relevance of the system requirements.Specifically, requirements engineering encompasses requirements elicitation, analysis, specification, verification, and management, where |

|  |  |
| --- | --- |
| Establish Relevance and Make Sense and Meaning | |
| Find Real-life contexts | Find Interdisciplinary connections |
| 3 | 4 |
| Requirement analysis is widely used in modeling and communication models and also in Real-Time Control Systems and almost all software engineering projects ranging from designing the control systems of aircrafts to automobiles and other light weight projects. | Requirement analysis would give clear view of and evaluation of the projects there by ensuring a systematic approach to implementations of the software. |

|  |  |
| --- | --- |
| Engage in critical thinking | Technology, Tools, Techniques |
| 5 | 6 |
| * A use case is a structure for documenting the functional requirements for a system, usually involving software, whether that is new or being changed. Each use case provides a set of *scenarios* that convey how the system should interact with a human user or another system, to achieve a specific business goal. Use cases typically avoid technical jargon, preferring instead the language of the [end-user](https://en.wikipedia.org/wiki/End-user) or [*domain expert*](https://en.wikipedia.org/wiki/Domain_expert). Use cases are often co-authored by requirements engineers and stakeholders. * Use cases are deceptively simple tools for describing the behavior of software or systems. A use case contains a textual description of the ways in which users are intended to work with the software or system. Use cases should not describe internal workings of the system, nor should they explain how that system will be implemented. Instead, they show the steps needed to perform a task without sequential assumptions. | * A natural inclination towards writing code can lead to implementation beginning before the requirements analysis is complete, potentially resulting in inelegant refactoring to meet actual requirements once they are known. * Technical personnel and end-users may have different vocabularies. Consequently, they may wrongly believe they are in perfect agreement until the finished product is supplied. * Engineers and developers may try to make the requirements fit an existing system or model, rather than develop a system specific to the needs of the client. * Analysis may often be carried out by engineers or programmers, rather than personnel with the domain knowledge to understand a client's needs properly.   Solutions:   * One attempted solution to communications problems has been to employ specialists in business or system analysis. * Techniques introduced in the 1990s like [prototyping](https://en.wikipedia.org/wiki/Software_prototyping), [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML), [use cases](https://en.wikipedia.org/wiki/Use_case), and [Agile software development](https://en.wikipedia.org/wiki/Agile_software_development) are also intended as solutions to problems encountered with previous methods. * Also, a new class of [application simulation](https://en.wikipedia.org/wiki/Application_Simulation_Software) or application definition tools have entered the market. These tools are designed to bridge the communication gap between business users and the IT organization — and also to allow applications to be 'test marketed' before any code is produced. The best of these tools offer: * [electronic whiteboards](https://en.wikipedia.org/wiki/Electronic_whiteboard) to sketch application flows and test alternatives * ability to capture business logic and data needs * ability to generate high fidelity prototypes that closely imitate the final application * interactivity * capability to add contextual requirements and other comments * ability for remote and distributed users to run and interact with the simulation |

|  |  |
| --- | --- |
| Plan Project management | Project Specification and Project Brief |
| 7 | 8 |
| This is backed up by controlled experiments in a student context. One possible interpretations is that requirements and design phases do not need to be entirely documentation-driven and theoretical. Instead, performing prototyping as part of the requirement and design phases – which amounts to spending time on and thinking about the requirements – is going to affect project success and implementation time.  There is also plenty of other evidence that points in the same direction: spending time on preparing for implementation pays off in the form of less risk and less chance of schedule overrun due to surprises. Although the question was not about testing, proper preparation positively affects testing, too. | * Software Maintenance assurance phases helped us to show a path to perform the actions before implementation of the code. We first identified and gathered our requirements and specifications and then analyzed them and prepared SRS document and designed, then implemented the code leading us to the goal we wanted. * In the real world scenario this project will be beneficial to students who seek value in what they purchase rather than any other aspect . * Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.[[1]](https://en.wikipedia.org/wiki/Software_requirements_specification#cite_note-1) Software requirements specification prevents software projects from failure[[2]](https://en.wikipedia.org/wiki/Software_requirements_specification#cite_note-2) |
|  | |