# Software development life cycle model

typical Software Development life cycle consists of the following stages

1. Requirement gathering and analysis
2. Design
3. Implementation or coding
4. Testing
5. Deployment
6. Maintenance

<http://blog.rht.com/6-basic-sdlc-methodologies-pros-cons/>

<http://consultingblogs.emc.com/simonevans/archive/2006/04/18/10-Reasons-why-Agile-is-not-Rapid-Application-Development-_2800_RAD_2900_.aspx>

http://en.wikipedia.org/wiki/List\_of\_software\_development\_philosophies

<http://en.wikipedia.org/wiki/Software_development_process>

**software development methodology** (also known as a **system development methodology**, **software development life cycle**, **software development process**, **software process**) is a division of [software development](http://en.wikipedia.org/wiki/Software_development) work into distinct phases (or stages) containing activities with the intent of better planning and management. It is often considered a subset of the [systems development life cycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle). The methodology may include the pre-definition of specific [deliverables](http://en.wikipedia.org/wiki/Deliverable) and artifacts that are created and completed by a project team to develop or maintain an application.[[1]](http://en.wikipedia.org/wiki/Software_development_process#cite_note-CMS08-1)

<http://www.codeproject.com/Articles/124732/Software-Development-Methodologies>

<http://www.itinfo.am/eng/software-development-methodologies/>

## [Approaches](http://en.wikipedia.org/wiki/Software_development_process#Approaches)

* [Waterfall development](http://en.wikipedia.org/wiki/Software_development_process#Waterfall_development)
* [Prototyping](http://en.wikipedia.org/wiki/Software_development_process#Prototyping)
* [Incremental development](http://en.wikipedia.org/wiki/Software_development_process#Incremental_development)
* [Iterative and incremental development](http://en.wikipedia.org/wiki/Software_development_process#Iterative_and_incremental_development)
* [Spiral development](http://en.wikipedia.org/wiki/Software_development_process#Spiral_development)
* [Rapid application development](http://en.wikipedia.org/wiki/Software_development_process#Rapid_application_development)
* [Agile development](http://en.wikipedia.org/wiki/Software_development_process#Agile_development)
* [Code and fix](http://en.wikipedia.org/wiki/Software_development_process#Code_and_fix)
* [Lightweight methodologies](http://en.wikipedia.org/wiki/Software_development_process#Lightweight_methodologies)
* [Other](http://en.wikipedia.org/wiki/Software_development_process#Other)

A software development methodology or system development methodology in software engineering is a framework that is used to structure, plan, and control the process of developing an information system.

There are the following methodologies:

* Agile Software Development
* Crystal Methods
* Dynamic Systems Development Model (DSDM)
* Extreme Programming (XP)
* Feature Driven Development (FDD)
* Joint Application Development (JAD)
* Lean Development (LD)
* **Rapid Application Development** (RAD)
* Rational Unified Process (RUP)
* Scrum
* Spiral
* Systems Development Life Cycle (SDLC)
* Waterfall (a.k.a. Traditional)

## SDLC

<http://en.wikipedia.org/wiki/SDLC>

Though SDLC is academically classified as Software Development life Cycle, the meaning of the process gets limited to a certain boundary beyond which it ceases to provide relevance, whereas System Development Life Cycle is a much broader term which is a superset to the above and is a larger part of the development phase which includes many other approach and end phases”

<http://en.wikipedia.org/wiki/Computer-aided_software_engineering#Tools>

List of software development philosophies

|  |
| --- |
| [**Software development process**](http://en.wikipedia.org/wiki/Software_development_process) |
| **Core activities** |
| * [Requirements](http://en.wikipedia.org/wiki/Requirements_analysis) * [Design](http://en.wikipedia.org/wiki/Software_design) * [Construction](http://en.wikipedia.org/wiki/Software_construction) * [Testing](http://en.wikipedia.org/wiki/Software_testing) * [Debugging](http://en.wikipedia.org/wiki/Debugging) * [Deployment](http://en.wikipedia.org/wiki/Software_deployment) * [Maintenance](http://en.wikipedia.org/wiki/Software_maintenance) |
| [**Methodologies**](http://en.wikipedia.org/wiki/Software_development_methodology) |
| * [Waterfall](http://en.wikipedia.org/wiki/Waterfall_model) * [Prototype model](http://en.wikipedia.org/wiki/Software_prototyping) * [Incremental](http://en.wikipedia.org/wiki/Incremental_build_model) * [Iterative](http://en.wikipedia.org/wiki/Iterative_and_incremental_development) * [V-Model](http://en.wikipedia.org/wiki/V-Model_(software_development)) * [Spiral](http://en.wikipedia.org/wiki/Spiral_model) * [Scrum](http://en.wikipedia.org/wiki/Scrum_(software_development)) * [Cleanroom](http://en.wikipedia.org/wiki/Cleanroom_software_engineering) * [RAD](http://en.wikipedia.org/wiki/Rapid_application_development) * [DSDM](http://en.wikipedia.org/wiki/Dynamic_systems_development_method) * [UP](http://en.wikipedia.org/wiki/Unified_Process) * [XP](http://en.wikipedia.org/wiki/Extreme_programming) * [Agile](http://en.wikipedia.org/wiki/Agile_software_development) * [Lean](http://en.wikipedia.org/wiki/Lean_software_development) * [Dual Vee Model](http://en.wikipedia.org/wiki/Dual_Vee_Model) * [TDD](http://en.wikipedia.org/wiki/Test-driven_development) * [BDD](http://en.wikipedia.org/wiki/Behavior-driven_development) * [FDD](http://en.wikipedia.org/wiki/Feature-driven_development) * [DDD](http://en.wikipedia.org/wiki/Domain-driven_design) * [MDD](http://en.wikipedia.org/wiki/Model-driven_development) |
| **Supporting disciplines** |
| * [Configuration management](http://en.wikipedia.org/wiki/Software_configuration_management) * [Documentation](http://en.wikipedia.org/wiki/Software_documentation) * [Software Quality assurance (SQA)](http://en.wikipedia.org/wiki/Software_quality_assurance) * [Project management](http://en.wikipedia.org/wiki/Software_project_management) * [User experience](http://en.wikipedia.org/wiki/User_experience) |
| [**Tools**](http://en.wikipedia.org/wiki/Programming_tool) |
| * [Compiler](http://en.wikipedia.org/wiki/Compiler) * [Debugger](http://en.wikipedia.org/wiki/Debugger) * [Profiler](http://en.wikipedia.org/wiki/Profiling_(computer_programming)) * [GUI designer](http://en.wikipedia.org/wiki/Graphical_user_interface_builder) * [Modeling](http://en.wikipedia.org/wiki/UML_tools) * [IDE](http://en.wikipedia.org/wiki/Integrated_development_environment) * [Build automation](http://en.wikipedia.org/wiki/Build_automation) |

▼ [Software development philosophies](http://en.wikipedia.org/wiki/Category:Software_development_philosophies)

▼ [Programming principles](http://en.wikipedia.org/wiki/Category:Programming_principles)

[*Abstraction principle (computer programming)*](http://en.wikipedia.org/wiki/Abstraction_principle_(computer_programming))

[*Code reuse*](http://en.wikipedia.org/wiki/Code_reuse)

[*Cohesion (computer science)*](http://en.wikipedia.org/wiki/Cohesion_(computer_science))

[*Command–query separation*](http://en.wikipedia.org/wiki/Command%E2%80%93query_separation)

[*Defensive programming*](http://en.wikipedia.org/wiki/Defensive_programming)

[*Dependency inversion principle*](http://en.wikipedia.org/wiki/Dependency_inversion_principle)

[*Discoverability*](http://en.wikipedia.org/wiki/Discoverability)

[*Don't repeat yourself*](http://en.wikipedia.org/wiki/Don%27t_repeat_yourself)

[*Fail-fast*](http://en.wikipedia.org/wiki/Fail-fast)

[*GRASP (object-oriented design)*](http://en.wikipedia.org/wiki/GRASP_(object-oriented_design))

[*Information hiding*](http://en.wikipedia.org/wiki/Information_hiding)

[*Interface segregation principle*](http://en.wikipedia.org/wiki/Interface_segregation_principle)

[*Inversion of control*](http://en.wikipedia.org/wiki/Inversion_of_control)

[*KISS principle*](http://en.wikipedia.org/wiki/KISS_principle)

[*Law of Demeter*](http://en.wikipedia.org/wiki/Law_of_Demeter)

*[Liskov substitution principle](http://en.wikipedia.org/wiki/Liskov_substitution_principle)*

[*Loose coupling*](http://en.wikipedia.org/wiki/Loose_coupling)

[*MINASWAN*](http://en.wikipedia.org/wiki/MINASWAN)

[*Open/closed principle*](http://en.wikipedia.org/wiki/Open/closed_principle)

[*Principle of least astonishment*](http://en.wikipedia.org/wiki/Principle_of_least_astonishment)

[*Separation of concerns*](http://en.wikipedia.org/wiki/Separation_of_concerns)

[*Separation of mechanism and policy*](http://en.wikipedia.org/wiki/Separation_of_mechanism_and_policy)

[*Single responsibility principle*](http://en.wikipedia.org/wiki/Single_responsibility_principle)

[*SOLID (object-oriented design)*](http://en.wikipedia.org/wiki/SOLID_(object-oriented_design))

[*Uniform access principle*](http://en.wikipedia.org/wiki/Uniform_access_principle)

[*Worse is better*](http://en.wikipedia.org/wiki/Worse_is_better)

[*You aren't gonna need it*](http://en.wikipedia.org/wiki/You_aren%27t_gonna_need_it)

▼ [Programming rules of thumb](http://en.wikipedia.org/wiki/Category:Programming_rules_of_thumb)

[*80:20 rule*](http://en.wikipedia.org/wiki/80:20_rule)

[*Amdahl's law*](http://en.wikipedia.org/wiki/Amdahl%27s_law)

[*Code smell*](http://en.wikipedia.org/wiki/Code_smell)

[*Deutsch limit*](http://en.wikipedia.org/wiki/Deutsch_limit)

*[Greenspun's tenth rule](http://en.wikipedia.org/wiki/Greenspun%27s_tenth_rule)*

[*Gustafson's law*](http://en.wikipedia.org/wiki/Gustafson%27s_law)

[*If it ain't broke, don't fix it*](http://en.wikipedia.org/wiki/If_it_ain%27t_broke,_don%27t_fix_it)

[*IIABDFI*](http://en.wikipedia.org/wiki/IIABDFI)

[*MINASWAN*](http://en.wikipedia.org/wiki/MINASWAN)

[*Ninety-ninety rule*](http://en.wikipedia.org/wiki/Ninety-ninety_rule)

[*Rule of three (C++ programming)*](http://en.wikipedia.org/wiki/Rule_of_three_(C%2B%2B_programming))

[*Rule of three (computer programming)*](http://en.wikipedia.org/wiki/Rule_of_three_(computer_programming))

[*Zero one infinity rule*](http://en.wikipedia.org/wiki/Zero_one_infinity_rule)

[*List of software development philosophies*](http://en.wikipedia.org/wiki/List_of_software_development_philosophies)

[*Acceptance test-driven development*](http://en.wikipedia.org/wiki/Acceptance_test-driven_development)

[*After the Software Wars*](http://en.wikipedia.org/wiki/After_the_Software_Wars)

[*Agile Manifesto*](http://en.wikipedia.org/wiki/Agile_Manifesto)

[*Agile software development*](http://en.wikipedia.org/wiki/Agile_software_development)

*[Behavior-driven development](http://en.wikipedia.org/wiki/Behavior-driven_development)*

[*Best practice*](http://en.wikipedia.org/wiki/Best_practice)

[*The Cathedral and the Bazaar*](http://en.wikipedia.org/wiki/The_Cathedral_and_the_Bazaar)

[*Comment programming*](http://en.wikipedia.org/wiki/Comment_programming)

[*Cowboy coding*](http://en.wikipedia.org/wiki/Cowboy_coding)

[*Design-driven development*](http://en.wikipedia.org/wiki/Design-driven_development)

[*Domain-driven design*](http://en.wikipedia.org/wiki/Domain-driven_design)

[*Extreme programming*](http://en.wikipedia.org/wiki/Extreme_programming)

[*Formal methods*](http://en.wikipedia.org/wiki/Formal_methods)

[*Hollywood principle*](http://en.wikipedia.org/wiki/Hollywood_principle)

[*Homesteading the Noosphere*](http://en.wikipedia.org/wiki/Homesteading_the_Noosphere)

[*Integration competency center*](http://en.wikipedia.org/wiki/Integration_competency_center)

[*Iterative and incremental development*](http://en.wikipedia.org/wiki/Iterative_and_incremental_development)

[*Kanban (development)*](http://en.wikipedia.org/wiki/Kanban_(development))

[*KISS principle*](http://en.wikipedia.org/wiki/KISS_principle)

[*Lean integration*](http://en.wikipedia.org/wiki/Lean_integration)

[*Lean software development*](http://en.wikipedia.org/wiki/Lean_software_development)

[*Lightweight methodology*](http://en.wikipedia.org/wiki/Lightweight_methodology)

[*The Magic Cauldron (essay)*](http://en.wikipedia.org/wiki/The_Magic_Cauldron_(essay))

[*Micro-innovation*](http://en.wikipedia.org/wiki/Micro-innovation)

[*Minimalism (computing)*](http://en.wikipedia.org/wiki/Minimalism_(computing))

[*Planning poker*](http://en.wikipedia.org/wiki/Planning_poker)

[*PM Declaration of Interdependence*](http://en.wikipedia.org/wiki/PM_Declaration_of_Interdependence)

[*Quick-and-dirty*](http://en.wikipedia.org/wiki/Quick-and-dirty)

[*Release early, release often*](http://en.wikipedia.org/wiki/Release_early,_release_often)

[*Retrenchment (computing)*](http://en.wikipedia.org/wiki/Retrenchment_(computing))

[*Rule of least power*](http://en.wikipedia.org/wiki/Rule_of_least_power)

[*Scrum pattern*](http://en.wikipedia.org/wiki/Scrum_pattern)

[*Secure by design*](http://en.wikipedia.org/wiki/Secure_by_design)

[*Slow programming*](http://en.wikipedia.org/wiki/Slow_programming)

[*Specification by example*](http://en.wikipedia.org/wiki/Specification_by_example)

[*SWAT Team (process model)*](http://en.wikipedia.org/wiki/SWAT_Team_(process_model))

[*Test double*](http://en.wikipedia.org/wiki/Test_double)

[*Continuous test-driven development*](http://en.wikipedia.org/wiki/Continuous_test-driven_development)

[*Test-driven development*](http://en.wikipedia.org/wiki/Test-driven_development)

[*Test-Driven Development by Example*](http://en.wikipedia.org/wiki/Test-Driven_Development_by_Example)

[*There's more than one way to do it*](http://en.wikipedia.org/wiki/There%27s_more_than_one_way_to_do_it)

[*Top development*](http://en.wikipedia.org/wiki/Top_development)

[*Transformation Priority Premise*](http://en.wikipedia.org/wiki/Transformation_Priority_Premise)

[*Unix philosophy*](http://en.wikipedia.org/wiki/Unix_philosophy)

[*Waterfall model*](http://en.wikipedia.org/wiki/Waterfall_model)

[*Worse is better*](http://en.wikipedia.org/wiki/Worse_is_better)

[*You aren't gonna need it*](http://en.wikipedia.org/wiki/You_aren%27t_gonna_need_it)

## **Miscellany[[edit](http://en.wikipedia.org/w/index.php?title=List_of_software_development_philosophies&action=edit&section=2" \o "Edit section: Miscellany)]**

|  |  |
| --- | --- |
| Ambox rewrite.svg | This section **may be in need of reorganization to comply with Wikipedia's**[**layout guidelines**](http://en.wikipedia.org/wiki/Wikipedia:Layout). Please help by [editing the article](http://en.wikipedia.org/w/index.php?title=List_of_software_development_philosophies&action=edit) to make improvements to the overall structure. *(November 2013)* |

* [Abstraction principle (programming)](http://en.wikipedia.org/wiki/Abstraction_principle_(programming))
* [Alignment of authority and responsibility](http://en.wikipedia.org/w/index.php?title=Alignment_of_authority_and_responsibility&action=edit&redlink=1)
* [Big Design Up Front](http://en.wikipedia.org/wiki/Big_Design_Up_Front) (BDUF)
* [Black box engineering](http://en.wikipedia.org/wiki/Black_box_engineering)
* [Brooks's law](http://en.wikipedia.org/wiki/Brooks%27s_law)
* [Cathedral and the Bazaar](http://en.wikipedia.org/wiki/The_Cathedral_and_the_Bazaar) (see also [Release early, release often](http://en.wikipedia.org/wiki/Release_early,_release_often) (RERO))
* [Chief programmer team](http://en.wikipedia.org/wiki/Chief_programmer_team)
* [CMMI](http://en.wikipedia.org/wiki/CMMI)
* [Code and fix](http://en.wikipedia.org/wiki/Code_and_fix)
* [Cone of Uncertainty](http://en.wikipedia.org/wiki/Cone_of_Uncertainty)
* [Continuous integration](http://en.wikipedia.org/wiki/Continuous_integration)
* [Control tables](http://en.wikipedia.org/wiki/Control_table)
* [Convention over configuration](http://en.wikipedia.org/wiki/Convention_over_configuration)
* [Conway's Law](http://en.wikipedia.org/wiki/Conway%27s_Law)
* [Cowboy coding](http://en.wikipedia.org/wiki/Cowboy_coding)
* [Dependency injection](http://en.wikipedia.org/wiki/Dependency_injection)
* [Design by Contract](http://en.wikipedia.org/wiki/Design_by_contract) (DbC)
* [Design for Test](http://en.wikipedia.org/wiki/Design_for_Test) (DFT)
* [Don't repeat yourself](http://en.wikipedia.org/wiki/Don%27t_repeat_yourself) (DRY) or Duplication is Evil (DIE) or Once and Only Once (OAOO), [Single Point of Truth](http://en.wikipedia.org/wiki/Single_Point_of_Truth) (SPoT), [Single Source Of Truth](http://en.wikipedia.org/wiki/Single_Source_of_Truth) (SSOT)
* [Easier to Ask Forgiveness than Permission](http://en.wikipedia.org/wiki/Python_syntax_and_semantics#Exceptions) (EAFP)
* [Encapsulation (computer science)](http://en.wikipedia.org/wiki/Encapsulation_(computer_science))
* [Evolutionary prototyping](http://en.wikipedia.org/wiki/Evolutionary_prototyping)
* [Free software license](http://en.wikipedia.org/wiki/Free_software_license)
* [General Responsibility Assignment Software Patterns](http://en.wikipedia.org/wiki/GRASP_(object-oriented_design)) (GRASP)
* [Hofstadter's law](http://en.wikipedia.org/wiki/Hofstadter%27s_law)
* [Hollywood Principle](http://en.wikipedia.org/wiki/Hollywood_Principle)
* [Information Hiding](http://en.wikipedia.org/wiki/Information_Hiding)
* [Interface (computer science)](http://en.wikipedia.org/wiki/Interface_(computer_science))
* [Interface (object-oriented programming)](http://en.wikipedia.org/wiki/Interface_(object-oriented_programming))
* [Inversion of control](http://en.wikipedia.org/wiki/Inversion_of_control)
* [Joint application design](http://en.wikipedia.org/wiki/Joint_application_design), aka JAD or "Joint Application Development"
* [Kaizen](http://en.wikipedia.org/wiki/Kaizen)
* [Literate Programming](http://en.wikipedia.org/wiki/Literate_Programming)
* [Microsoft Solutions Framework](http://en.wikipedia.org/wiki/Microsoft_Solutions_Framework) (MSF)
* [Model-driven architecture](http://en.wikipedia.org/wiki/Model-driven_architecture) (MDA)
* [MoSCoW Method](http://en.wikipedia.org/wiki/Moscow_Method)
* [Naked Objects](http://en.wikipedia.org/wiki/Naked_objects)
* [Open source](http://en.wikipedia.org/wiki/Open_source)
* [Principle of least astonishment](http://en.wikipedia.org/wiki/Principle_of_least_astonishment) (POLA/PLA)
* [Principle of good enough](http://en.wikipedia.org/wiki/Principle_of_good_enough) (POGE)
* [Project triangle](http://en.wikipedia.org/wiki/Project_triangle)
* [Program optimization](http://en.wikipedia.org/wiki/Program_optimization)
* [Protocol (object-oriented programming)](http://en.wikipedia.org/wiki/Protocol_(object-oriented_programming))
* [Quick-and-dirty](http://en.wikipedia.org/wiki/Quick-and-dirty)
* [Rapid prototyping](http://en.wikipedia.org/wiki/Rapid_application_development)
* [Refactoring](http://en.wikipedia.org/wiki/Code_refactoring)
* [Release early, release often](http://en.wikipedia.org/wiki/Release_early,_release_often) (RERO) - see also [The Cathedral and the Bazaar](http://en.wikipedia.org/wiki/The_Cathedral_and_the_Bazaar)
* [Responsibility-driven design](http://en.wikipedia.org/wiki/Responsibility-driven_design) (RDD)
* the [Right thing](http://en.wikipedia.org/wiki/Right_Thing_(software_development)), or the MIT approach, as contrasted with the New Jersey style, [Worse is better](http://en.wikipedia.org/wiki/Worse_is_better).
* [Secure by design](http://en.wikipedia.org/wiki/Secure_by_design)
* [Separation of concerns](http://en.wikipedia.org/wiki/Separation_of_concerns) (SoC)
* [Service-oriented modeling](http://en.wikipedia.org/wiki/Service-oriented_modeling)
* [Software craftsmanship](http://en.wikipedia.org/wiki/Software_craftsmanship)
* [Software System Safety](http://en.wikipedia.org/wiki/Software_System_Safety)
* [Spiral model](http://en.wikipedia.org/wiki/Spiral_model)
* [Stepwise Refinement](http://en.wikipedia.org/wiki/Stepwise_Refinement)
* [Team Software Process](http://en.wikipedia.org/wiki/Team_Software_Process) (TSP)
* [Type-Generic-Profile](http://c2.com/cgi/wiki?TgpMethodology) (TGP) definition at Wards Wiki
* [Ubuntu philosophy](http://en.wikipedia.org/wiki/Ubuntu_(philosophy)#In_popular_culture)
* [Unix philosophy](http://en.wikipedia.org/wiki/Unix_philosophy)
* [V-Model](http://en.wikipedia.org/wiki/V-Model_(software_development))
* [Wheel and spoke model](http://en.wikipedia.org/wiki/Wheel_and_spoke_model)
* [The Zen of Python](http://en.wikipedia.org/wiki/The_Zen_of_Python)

## **Programming paradigm[[edit](http://en.wikipedia.org/w/index.php?title=List_of_software_development_philosophies&action=edit&section=3" \o "Edit section: Programming paradigm)]**

* [Agent-oriented programming](http://en.wikipedia.org/wiki/Agent-oriented_programming)
* [Aspect-oriented Programming](http://en.wikipedia.org/wiki/Aspect-oriented_Programming) (AOP)
* [Modular programming](http://en.wikipedia.org/wiki/Modular_programming)
* [Object-Oriented Programming](http://en.wikipedia.org/wiki/Object-Oriented_Programming) (OOP)
* [Functional Programming](http://en.wikipedia.org/wiki/Functional_Programming) (FP)

## **Software development methodology[[edit](http://en.wikipedia.org/w/index.php?title=List_of_software_development_philosophies&action=edit&section=4" \o "Edit section: Software development methodology)]**

* [Agile Unified Process](http://en.wikipedia.org/wiki/Agile_Unified_Process) (AUP)
* [Dynamic Systems Development Method](http://en.wikipedia.org/wiki/Dynamic_Systems_Development_Method) (DSDM)
* [Constructionist design methodology](http://en.wikipedia.org/wiki/Constructionist_design_methodology) (CDM)
* [Crystal Clear](http://en.wikipedia.org/wiki/Crystal_Clear_(software_development))
* [Extreme Programming](http://en.wikipedia.org/wiki/Extreme_Programming) (XP)
* [Iterative and incremental development](http://en.wikipedia.org/wiki/Iterative_and_incremental_development)
* [Kanban](http://en.wikipedia.org/wiki/Kanban_(development))
* [Lean software development](http://en.wikipedia.org/wiki/Lean_software_development)
* [Open Unified Process](http://en.wikipedia.org/wiki/OpenUP)
* [Pair programming](http://en.wikipedia.org/wiki/Pair_programming)
* [Rapid application development](http://en.wikipedia.org/wiki/Rapid_application_development) (RAD)
* [Rational Unified Process](http://en.wikipedia.org/wiki/IBM_Rational_Unified_Process) (RUP)
* [Scrum](http://en.wikipedia.org/wiki/Scrum_(development))
* [Structured Systems Analysis and Design Method](http://en.wikipedia.org/wiki/Structured_Systems_Analysis_and_Design_Method) (SSADM)
* [Unified Process](http://en.wikipedia.org/wiki/Unified_Process) (UP)

## **Software development processes[[edit](http://en.wikipedia.org/w/index.php?title=List_of_software_development_philosophies&action=edit&section=5" \o "Edit section: Software development processes)]**

* [Behavior-driven development](http://en.wikipedia.org/wiki/Behavior-driven_development) (BDD)
* [Design-driven development](http://en.wikipedia.org/wiki/Design-driven_development) (D3)
* [Domain-Driven Design](http://en.wikipedia.org/wiki/Domain-Driven_Design) (DDD)
* [Feature Driven Development](http://en.wikipedia.org/wiki/Feature_Driven_Development) (FDD)
* [Test-driven development](http://en.wikipedia.org/wiki/Test-driven_development) (TDD)
* [User-centered design (UCD)](http://en.wikipedia.org/wiki/User-centered_design)

## **Software metrics[[edit](http://en.wikipedia.org/w/index.php?title=List_of_software_development_philosophies&action=edit&section=6" \o "Edit section: Software metrics)]**

* [Cohesion (computer science)](http://en.wikipedia.org/wiki/Cohesion_(computer_science))
* [Coupling (computer programming)](http://en.wikipedia.org/wiki/Coupling_(computer_programming))

## **References[[edit](http://en.wikipedia.org/w/index.php?title=List_of_software_development_philosophies&action=edit&section=7" \o "Edit section: References)]**

* [Don't Make Me Think](http://en.wikipedia.org/wiki/Don%27t_Make_Me_Think) (book by [Steve Krug](http://en.wikipedia.org/wiki/Steve_Krug) about [human computer interaction](http://en.wikipedia.org/wiki/Human_computer_interaction) and [web usability](http://en.wikipedia.org/wiki/Web_usability))

## **See also[[edit](http://en.wikipedia.org/w/index.php?title=List_of_software_development_philosophies&action=edit&section=8" \o "Edit section: See also)]**

* [Anti-pattern](http://en.wikipedia.org/wiki/Anti-pattern)
* [Design pattern](http://en.wikipedia.org/wiki/Design_pattern_(computer_science))
* [Programming paradigm](http://en.wikipedia.org/wiki/Programming_paradigm)
* [Software development methodology](http://en.wikipedia.org/wiki/Software_development_methodology)
* [Software development process](http://en.wikipedia.org/wiki/Software_development_process)

[**Software engineering**](http://en.wikipedia.org/wiki/Software_engineering) is the application of a systematic, disciplined, quantifiable approach to the design, development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

*The main article for this*[*category*](http://en.wikipedia.org/wiki/Help:Categories)*is*[***Software engineering***](http://en.wikipedia.org/wiki/Software_engineering)*.*

See [Category:Systems engineering](http://en.wikipedia.org/wiki/Category:Systems_engineering" \o "Category:Systems engineering) for such topics as: **Requirements analysis, Systems analysis, Systems architecture, System life cycle,** and many more.

|  |  |
| --- | --- |
| http://upload.wikimedia.org/wikipedia/commons/thumb/5/57/Sub-arrows.svg/40px-Sub-arrows.svg.png | **Pages in this category should be moved to subcategories where applicable.** This category may require frequent maintenance to avoid becoming too large. It should directly contain very few, if any, articles and should mainly contain subcategories. |

## **Subcategories**

This category has the following 23 subcategories, out of 23 total.

|  |  |  |
| --- | --- | --- |
| A  * ► [Software architecture](http://en.wikipedia.org/wiki/Category:Software_architecture)‎ (9 C, 194 P)  B  * ► [Software engineering books](http://en.wikipedia.org/wiki/Category:Software_engineering_books)‎ (1 C, 10 P)  C  * ► [Computer programming](http://en.wikipedia.org/wiki/Category:Computer_programming)‎ (26 C, 127 P) * ► [Software engineering conferences](http://en.wikipedia.org/wiki/Category:Software_engineering_conferences)‎ (13 P)  D  * ► [Software design](http://en.wikipedia.org/wiki/Category:Software_design)‎ (6 C, 72 P)  E  * ► [Software engineers](http://en.wikipedia.org/wiki/Category:Software_engineers)‎ (6 C, 43 P)  F  * ► [Formal methods](http://en.wikipedia.org/wiki/Category:Formal_methods)‎ (18 C, 94 P)  L  * ► [Programming language topics](http://en.wikipedia.org/wiki/Category:Programming_language_topics)‎ (12 C, 65 P) | M  * ► [Software maintenance](http://en.wikipedia.org/wiki/Category:Software_maintenance)‎ (35 P) * ► [Software metrics](http://en.wikipedia.org/wiki/Category:Software_metrics)‎ (45 P)  O  * ► [Software engineering organizations](http://en.wikipedia.org/wiki/Category:Software_engineering_organizations)‎ (1 C, 19 P)  P  * ► [Programming rules of thumb](http://en.wikipedia.org/wiki/Category:Programming_rules_of_thumb)‎ (13 P) * ► [Software engineering publications](http://en.wikipedia.org/wiki/Category:Software_engineering_publications)‎ (3 C, 11 P)  Q  * ► [Software quality](http://en.wikipedia.org/wiki/Category:Software_quality)‎ (6 C, 52 P)  R  * ► [Software requirements](http://en.wikipedia.org/wiki/Category:Software_requirements)‎ (1 C, 54 P) * ► [Revision control](http://en.wikipedia.org/wiki/Category:Revision_control)‎ (2 C, 23 P) | S  * ► [Software development](http://en.wikipedia.org/wiki/Category:Software_development)‎ (17 C, 17 P) * ► [Software for modeling software](http://en.wikipedia.org/wiki/Category:Software_for_modeling_software)‎ (1 C, 4 P) * ► [Specification languages](http://en.wikipedia.org/wiki/Category:Specification_languages)‎ (8 C, 47 P)  T  * ► [Software engineering terminology](http://en.wikipedia.org/wiki/Category:Software_engineering_terminology)‎ (1 C, 16 P) * ► [Software testing](http://en.wikipedia.org/wiki/Category:Software_testing)‎ (7 C, 246 P)  W  * ► [Web development](http://en.wikipedia.org/wiki/Category:Web_development)‎ (15 C, 89 P)  Σ  * ► [Software engineering stubs](http://en.wikipedia.org/wiki/Category:Software_engineering_stubs)‎ (3 C, 211 P) |

## **Pages in category "Software engineering"**

The following 52 pages are in this category, out of 52 total. This list may not reflect recent changes ([learn more](http://en.wikipedia.org/wiki/Wikipedia:FAQ/Categories#Why_might_a_category_list_not_be_up_to_date.3F)).

|  |  |  |
| --- | --- | --- |
| * [Outline of software engineering](http://en.wikipedia.org/wiki/Outline_of_software_engineering) * [Software engineering](http://en.wikipedia.org/wiki/Software_engineering)  \*  * [Index of software engineering articles](http://en.wikipedia.org/wiki/Index_of_software_engineering_articles)  E  * [Experimental software engineering](http://en.wikipedia.org/wiki/Experimental_software_engineering)  F  * [Fundamental theorem of software engineering](http://en.wikipedia.org/wiki/Fundamental_theorem_of_software_engineering)  H  * [History of software engineering](http://en.wikipedia.org/wiki/History_of_software_engineering)  I  * [ISO 29110](http://en.wikipedia.org/wiki/ISO_29110) * [ISO/IEC 15504](http://en.wikipedia.org/wiki/ISO/IEC_15504) * [ISO/IEC JTC 1/SC 7](http://en.wikipedia.org/wiki/ISO/IEC_JTC_1/SC_7)  L  * [List of software development philosophies](http://en.wikipedia.org/wiki/List_of_software_development_philosophies)  M  * [Meta-process modeling](http://en.wikipedia.org/wiki/Meta-process_modeling) * [Mining Software Repositories](http://en.wikipedia.org/wiki/Mining_Software_Repositories) * [Mixed criticality](http://en.wikipedia.org/wiki/Mixed_criticality) * [MockServer](http://en.wikipedia.org/wiki/MockServer)  O  * [Object-orientation](http://en.wikipedia.org/wiki/Object-orientation) * [Overhead (computing)](http://en.wikipedia.org/wiki/Overhead_(computing))  P  * [Physical Data Flow](http://en.wikipedia.org/wiki/Physical_Data_Flow) * [Prosa Structured Analysis Tool](http://en.wikipedia.org/wiki/Prosa_Structured_Analysis_Tool) | Q  * [Quality Engineering](http://en.wikipedia.org/wiki/Quality_Engineering)  R  * [Reference model](http://en.wikipedia.org/wiki/Reference_model) * [Round-trip engineering](http://en.wikipedia.org/wiki/Round-trip_engineering) * [Runtime error detection](http://en.wikipedia.org/wiki/Runtime_error_detection)  S  * [Schema migration](http://en.wikipedia.org/wiki/Schema_migration) * [Search-based software engineering](http://en.wikipedia.org/wiki/Search-based_software_engineering) * [Service virtualization](http://en.wikipedia.org/wiki/Service_virtualization) * [Service-oriented software engineering](http://en.wikipedia.org/wiki/Service-oriented_software_engineering) * [Social software engineering](http://en.wikipedia.org/wiki/Social_software_engineering) * [Software analyst](http://en.wikipedia.org/wiki/Software_analyst) * [Software bloat](http://en.wikipedia.org/wiki/Software_bloat) * [Software configuration management](http://en.wikipedia.org/wiki/Software_configuration_management) * [Software construction](http://en.wikipedia.org/wiki/Software_construction) * [Software development process](http://en.wikipedia.org/wiki/Software_development_process) * [Software diagnosis](http://en.wikipedia.org/wiki/Software_diagnosis) * [Software engineer](http://en.wikipedia.org/wiki/Software_engineer) * [Software engineering demographics](http://en.wikipedia.org/wiki/Software_engineering_demographics) * [Software engineering professionalism](http://en.wikipedia.org/wiki/Software_engineering_professionalism) | S cont.  * [Software map](http://en.wikipedia.org/wiki/Software_map) * [Software portability](http://en.wikipedia.org/wiki/Software_portability) * [Software system](http://en.wikipedia.org/wiki/Software_system) * [Software system safety](http://en.wikipedia.org/wiki/Software_system_safety) * [Software visualization](http://en.wikipedia.org/wiki/Software_visualization) * [Specification language](http://en.wikipedia.org/wiki/Specification_language) * [Structural synthesis of programs](http://en.wikipedia.org/wiki/Structural_synthesis_of_programs) * [System appreciation](http://en.wikipedia.org/wiki/System_appreciation) * [System context diagram](http://en.wikipedia.org/wiki/System_context_diagram) * [System requirements specification](http://en.wikipedia.org/wiki/System_requirements_specification) * [Systems development life cycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) * [Systems modeling](http://en.wikipedia.org/wiki/Systems_modeling)  T  * [Task-oriented information modelling](http://en.wikipedia.org/wiki/Task-oriented_information_modelling) * [Traceability](http://en.wikipedia.org/wiki/Traceability)  V  * [View model](http://en.wikipedia.org/wiki/View_model)  W  * [Web engineering](http://en.wikipedia.org/wiki/Web_engineering) |