# Technical Debt Types Definitions

|  |  |
| --- | --- |
| Architecture  Debt | Refers to the problems encountered in project architecture, for example, violation of modularity, which can affect architectural requirements (performance, robustness, among others). Normally this type of debt cannot be paid with simple interventions in the code, implying in more extensive development activities. |
| Build  Debt | Refers to build related issues that make this task harder, and more time/processing consuming unnecessarily. The build process of a project can contain very unnecessary code to the customer. Moreover, if the build process needs to run ill-defined dependencies, the process becomes unnecessarily slow. When this occurs, one can identify a build debt. |
| Code  Debt | Refers to the problems found in the source code which can affect negatively the legibility of the code making it more difficult to be maintained. Usually, this debt can be identified by examining the source code of the project considering issues related to bad coding practices. |
| Defect  Debt | Software projects may have known and unknown defects in the source code. Defect debt consists of known defects, usually identified by testing activities or by the user and reported on bug track systems, that the CCB agrees should be fixed, but due to competing priorities, and limited resources have to be deferred to a later time. Decisions made by the CCB to defer addressing defects can accumulate a significant amount of technical debt for a product making it harder to fix them later. |
| Design  Debt | Refers to debt that can be discovered by analysing the source code by identifying the use of practices which violated the principles of good object-oriented design (e.g. very large or tightly coupled classes). |
| Documentation  Debt | Refers to the problems found in software project documentation and can be identified by looking for missing, inadequate, or incomplete documentation of any type. Inadequate documentation is those that currently work correctly in the system, but fail to meet certain quality criteria of software projects. |
| Infrastructure  Debt | Refers to infrastructure issues that, if present in the software organization, can delay or hinder some development activities. Some examples of this kind of debt are delaying an upgrade or infrastructure fix. |
| People  Debt | Refers to people issues that, if present in the software organization, can delay or hinder some development activities. An example of this kind of debt is expertise concentrated in too few people, as an effect of delayed training and/or hiring. |
| Process  Debt | Refers to inefficient processes, e.g. what the process was designed to handle may be no longer appropriate. |
| Requirement  Debt | Requirements debt refers to trade-offs made with respect to what requirements the development team need to implement or how to implement them. Some examples of this type of debt are: requirements that are only partially implemented, requirements that are implemented but not for all cases, requirements that are implemented but in a way that doesn’t fully satisfy all the non-functional requirements (e.g. security, performance, etc.). |
| Service  Debt | The need for web service substitution could be driven by business or technical objectives. The substitution can introduce a TD, which needs to be managed, cleared and transformed from liability to value-added. Technical debt can cover several dimensions, which are related to selection, composition, and operation of the service. |
| Test Automation  Debt | Test Automation debt is defined as the work involved in automating tests of previously developed functionality to support continuous integration and faster development cycles. |
| Test  Debt | Refers to issues found in testing activities which can affect the quality of testing activities. Examples of this type of debt are planned tests that were not run, or known deficiencies in the test suite (e.g. low code coverage). |

# Indicators by Type of Technical Debt

|  |  |
| --- | --- |
| Architecture Debt | ACN/PWDR  Betweenness Centrality  Issues in software architecture  Structural Analysis  Structural Dependencies  Violation of Modularity |
| Build Debt | “Dead Flags”  “Zombie Targets”  Dependency  Visibility |
| Code Debt | ASA Issues  Code Metrics  Code outside of standards  Duplicated code  Multithread correctness (ASA)  Slow Algorithm |
| Defect Debt | Uncorrected known defects |
| Design Debt | ASA Issues  Brain Method  Code Metrics  Code Smells  Data Class  Data clumps  Dispersed Coupling  Duplicated Code  God class (or large class)  Grime  Intensive Coupling  Issues in the software design  Refused Parent Bequest  Schizophrenic Class  Structural Analysis |
| Documentation Debt | Documentation does not exist  Incomplete Design Specification  Incomplete Documentation  Insufficient comments in code  Outdated Documentation  Test Documentation |
| Infrastructure Debt |  |
| People Debt |  |
| Process Debt |  |
| Requirement Debt | Requirement Backlog List |
| Service Debt | Selection/Replacement of web service |
| Test Automation Debt |  |
| Test Debt | Incomplete Tests  Low coverage |