Labelling guide of the study "*Asking About Technical Debt: Characteristics and Automatic Identification of Technical Debt Questions on Stack Overflow*"

The goal of this document is to guide the manual identification and labelling of Stack Overflow questions related to Technical debt (TD). The definition of Technical Debt we follow for labelling is the 16162 definition of TD, *i.e.,*
"In software-intensive systems, technical debt is a design or implementation construct that is expedient in the short term, but sets up a technical context that can make a future change more costly or impossible. Technical debt is a contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability." [1]

The labelling task consists of assigning up to three labels (depending on the value of the first label) to each inspected Stack Overflow post. Specifically, we label:

- **Technical-debt-related (TDR):** boolean value. Indicates whether the post focuses on a technical debt topic (value = 1) or not (value = 0);
- **Technical debt type (TD type):** multiple values. Indicates on which category or categories of technical debt the post focuses on (see following section for further detail).
- **Urgency: 6-level scale [0,5].** Indicates the perceived urgency presented in the question. A value of zero indicates that the presented debt item does not possess any urgency, while a value of five indicates that the debt item is of critical urgency.

Further information on how to assign the labels of the three characteristics mentioned above follows.

1. Technical-debt-related

This categorization regards if the considered question is related to technical debt or not. Assigning this label entails evaluating the content of a Stack Overflow question against the 16162 Technical Debt definition [3]. If the question regards TD (as defined in the 16162 TD definition), then it is labelled as technical-debt-related (value = 1), otherwise it is labelled as non-technical-debt-related (value = 0).

Examples of questions related to technical debt is how to fix issues related to TD, how to manage TD, how to measure debt, what decision leads to less TD, etc. Notably, discussions

which regard issues / troubleshooting of specific to tools implementing technical debt analyses, e.g. SonarQube[1] installation problems, will not be labeled as technical debt (as the question regards the use of a tool, rather than a technical debt concept).

An example of a question related to technical debt is:

*"We have some **really shocking code touted as a next generation framework at my current place of employment**.Thing is, there is only one person of this opinion and that is the guy who wrote most of it. The rest of the department are of the impression it is **badly coded, a pita to debug and just a bit naff in general**.The guy that wrote it has a pretty influential position with management so they are on that side of the camp.We have highlighted **(genuine) concerns** to management but obviously they're not willing to put more time in to a project that doesn't directly contribute to the bottom line.There are several applications deployed on this framework so any refactoring will need to encompass those applications.**The whole thing is so intertwined that we cant just rip out an implementation of a particular class and rewrite it that way so even simple changes to core api mean a large project**.It does however have 3 years in live deployment and many bug fixes, corner cases and boundary conditions catered for.Do we rewrite in parts and try to refactor that in given that it would be several large projects, refactor over time which is likely to take another 3 years to get it in to shape or do we just rewrite our specific requirements on top of an existing framework?"*

An example of question not related to technical debt is:

*"I hear a lot of terms which aren't well known amongst programmers (or perhaps the ones I work with at work aren't very good apart from a few), **such as technical debt** (which I studied and even see first hand at work). What other obscure/not-well-known terms are there? This is especially useful to know as interviewers sometimes mention complex terms and if I don't know what they mean, it can screw up the interview as it is in progress. Thanks"*

2. Technical debt type

The classification of posts into one or more technical debt categories follows the classification of debt presented by Li et al. [2] Following a brief description of each category, supported by examples.

Requirements debt refers to the distance between the optimal requirements specification and the actual system implementation, under domain assumptions and constraints. Under this category fall also discussions related to prioritization and management of funcional/non-functional requirements. An example is provided below:

*"I'm at the point with **a project that most of it needs a rewrite**, specifically **one portion needs it very badly for a number of reasons**: it's no longer being used for it's original purpose. **the code has been hacked so badly it's very hard to work with**. **the changes they want will hack it to pieces**. **the original design wasn't the greatest** as we were trying a new technique. The unfortunate part is the piece of code that has been **hacked so badly** is run every minute and **as the site gets used more, the longer it's taking. How do you tell your client that you need to take the time to think it through and rewrite it?"***

[1] https://www.sonarqube.org/

Architectural TD is caused by architecture decisions that make compromises in some internal quality aspects, such as maintain- ability. Under this category fall questions related to the higher level of abstraction of a software-intensive system, such as architectural layers (e.g., databases), architectural elements, libraries, packages, APIs, etc. An example is provided below:

*"There is a lot of good content on SO about **MVC** and getting started with **MVC**, but I'm having trouble finding anything about how best to **implement MVC structure** on a pre-existing, live website. **My site is a nasty mishmash of echos and concatenated HTML that would make any professional programmer throw-up, but it works.** I'd like to spend some time **tackling the mounting technical debt**, however, and that means moving to a much more sane **MVC structure**. If at all possible, I'd like to **avoid a let 'er rip! 100% rewrite and launch approach, and instead take it a section at a time**. But it seems the basic controller's centralized structure is not suitable for such an approach?"*

Design TD refers to technical shortcuts that are taken in detailed design. This consists of the middle-layer of abstraction between *Architectural TD* and *Code TD*. Under this category fall TD discussion which regard the design of relations between classes, dependencies, design patterns [3], etc. This category does not include "inter-class"-related topics (e.g., issues regarding specific lines of code within a class, which do not establish a direct dependency to other classes). An example is provided below:

*"I am in a process of **paying my Technical Debt** and one of **the problem I have facing is the uses of DAO's with CDI injection**. I did write a blog about **DAO injection** (using CDI, of course) and it works wonders and I will use the reference of the blog for my question. I am now in a situation where I have so many database tables (normalized, of course) and **creating DAO's for each tables** is not the issue. **The issue is when a service (let's assume and EJB Session Bean) has "references" to many DAO's**, like in the example below:*

```
@Stateless
public class CustomerServiceEJB implements CustomerServiceLocal, CustomerServiceRemote,
CustomerService {
 private static final Logger LOGGER = Logger.getLogger(CustomerServiceEJB.class.getName());
 @Inject @JPADAO
 private CustomerDAO customerDAO;
 @Inject @JPADAO
 private CustomerAccountDAO customerAccountDAO;
 @Inject @JPADAO
 private CustumerCredentialDAO customerCredentialDAO;
 //And rougly 15 more DAO's.
}
```

*As you can see, **if we need to use various DAO's, the class has "instances" of all DAO's and that means that the CDI container will have to inject all of them prior to being used**. One solution I came up with was to **create a DAOManager** where all references of the DAO's are listed in the manager."*

Code TD is the poorly written code that violates best coding practices or coding rules. Examples include code duplication and over-complex code. Under this category fall TD questions regarding low level implementation details, often manifesting themselves on few

lines of isolated source code (i.e., not establishing relation with other programming elements). An example is provided below:

*"I wrote the following **recursive function** to keep looping through result looking for $result->pages->next and calling out to curl fetching the next page and aggregating the results. Finally it **returns all results as a single object**.*

```
 private function pager($result) {
      static $all_results;

      if(isset($result->pages->next) && !empty($result->pages->next)) {
         $all_results[] = $this->get_curl_request($result->pages->next);
         $this->pager(end($all_results));
      } else {
         return $all_results;
      }
   }
```

However I really don't like using static and it feels poorly implemented and a source of technical debt. What is a more elegant way to do this?

Update

Being called with:

```
return $this->pager($this->get_curl_request("https://api/request/here"));
```

Open to changing how it is called."

Test TD refers to shortcuts taken in testing. An example is lack of tests (e.g., unit tests, integration tests, and acceptance tests). An example is provided below:

*"I'm implementing **unit tests** in a finance system that involves several calculations. **One of the methods, receives an object by parameter with more than 100 properties**, and based on this object's properties, it calculates the return.**In order to implement unit tests for this method, I need to have all of this object filled with valid values**. So...question: today this object is populated via database. **On my Unit Tests (I'm using NUnit), I'd need to avoid the database** and create a mock object of that, to test only the method's return. **How can I efficiently test this method with this huge object? Do I really need to manually fill all of the 100 properties of it?**Is there a way to automatize this object creation using Moq (for emple)?obs: **I'm writing unit tests for a system that is already created. It's not feasible to re-write all the architecture at this moment.**Thanks a million!"*

Build TD refers to flaws in a software system, in its build system, or in its build process that make the build overly complex and difficult. This category includes questions related to build automation tools (e.g., Jenkins, Gradle, Maven, Ant), issues arising at build times (e.g., failed builds), debt present in build scripts, etc. An example is provided below:

'My team has gone years (before I joined) of not adopting sql server database projects. **Primarily because there are items that have circular references therefor can't build. I realize the recommended approach is to factor out or fix the circular references**. However to prove **we can use them without having to fix all that technical debt**, I'd like to **hook my sql server project's build process to remove the items from the Build**

item group. I'm very familiar with msbuild but so far nothing seems to be changing that file being included in the build and failing it.
Setting the file(s) to None instead of Build works fine, until we try to do a schema compare. Where it either sets them back to build, or adds them again with a new file name ending in _1.sql

```
<Target Name="CustomSqlHook"
BeforeTargets="CheckRequiredProperties;BeforeBuild;All;_SetupSqlBuildInputs;SqlBuild;SqlStaticCodeAnaly
sis;ResolveArtifactReferences;Build">
   <Message Text="Build Before: @(Build)" Importance="high" />
   <Message Importance="High" Text="custom sql hooked" />
   <Message Importance="High" Text="build items are @(Build-&gt;Count())" />
   <ItemGroup>
     <Build Remove="project\Views\uvw_survey_history.sql" />
   </ItemGroup>
   <Message Text="-------------------------" />
   <Message Text="Build After: @(Build)" Importance="high" />
   <Message Importance="High" Text="build items are @(Build-&gt;Count())" />
</Target>
```

which shows the count of included items in the build group as going from 2978 to 2977. However the build still fails on the same file.

Documentation TD refers to insufficient, incomplete, or outdated documentation in any aspect of software development. Examples include out-of-date architecture documentation and lack of code comments. Under this category fall also question regarding missing documentation available online, READMEs, problems arising from lack of knowledge of (portion of) a codebase, etc. An example is provided below:

"*I have a bunch of MATLAB script/function files that I and the rest of my team need to work on.* ***We have little to no idea what most of the files do, and little to no idea which ones belong together and which ones are separate.*** *We do know we have a total of 36,000 lines. I'd like to know how many of those lines are comments. Easy, right? Just count how many of them start with the comment start character %.*
Well, no. I don't want to count blocks of code that have been commented out as "comments", since they don't actually tell me anything. And I'd prefer not to count "empty" lines used to make one comment line a "headline"

```
% %%%%%%%%
% headline
% %%%%%%%%
```

like so.
So how can I get a sensible estimate of how many lines of actual informative comments I have? Is there an easy way to distinguish natural language (possibly containing code snippets) from pure code?
Yes, I know code should be self-explanatory as far as is practical, but the code we have inherited clearly is not. Yes, I know we should probably refactor this mess. *The purpose of figuring out how much comments we have is to* ***highlight the technical debt we have here****, so that we can allocate resources to this refactoring."*

Infrastructure TD refers to a sub-optimal configuration of development-related processes, technologies, supporting tools, etc. Such a sub-optimal configuration negatively affects the team's ability to produce a quality product. Under this category fall questions related to the technological environment supporting programming activities (DevOps), such as issues

arising in tools used (e.g., code analyzers, Travis CI, Kubernetes/Docker, etc.), issues related to continuous integration CI, etc. An example is provided below:

*"My Eclipse/Tomcat project is a **tangled mess of slf4j, log4j, JULI, logback, etc.** As a result my **console has an abundance of useless messages from various libraries and it is hard to get the logging output** I want from my own code. I've **resorted to using System.out to get stuff done**. The server **deployment environment is in similar bad shape**. What I'm wondering is, where can I find a reference implementation, or checklist, that outlines what a correct configuration of all these **logging frameworks** is supposed to look like??"*

Versioning TD refers to the problems in source code versioning, such as unnecessary code forks, issues related to the maintenance of multiple branches, pull requests, merges etc. An example is provided below:

*"I'm maintaining a Redis Sentinel library for Laravel in git with **two active development branches**:*
1.x ... o ——— o ——————— o — o — o — o — o ...
↘ ↑ ↑ ↖
2.x (breaking change) — o — o — o — o — o ...
*As illustrated above, **I forked 1.x to accomodate breaking changes** from a newer version of the framework that this library integrates with, but **I find myself cherry-picking commits or manually copying code between branches that update common functionality in both branches**. It seems like I can never merge these branches because they each contain code that is not compatible with the other. What better workflow can I follow to reduce the effort needed to merge common changes between the branches?"*

Defect TD refers to defects, bugs, or failures found in software systems. Under this category prominently fall hard to explain issues present in a software-intensive system. Examples include random crashes, hard to trace defect, etc. An example is not provided, as this TD type does not appear from a preliminary data exploration.

2. Urgency: 6-point scale [0,6]

"Urgency" indicates the perceived necessity to fix the technical debt item presented in the question. This is expressed via a 5-point intensity Likert scale. The values are described in the following.

None (value 1): The question does not present any urgency. The technical debt item does not not require to be fixed, and with high probability will not be fixed. An example is provided below:

"I want to use a tool like landscape.io to keep track of technical debt that people might be accidentially introducing into an open-source project. Unfortuanately that tool only seems to work with GitHub.Is there a similar tool that offers static code analysis as a hosted service that's also compatible with BitBucket and Mercurial?I'm certain that I could get most of this using a hand-rolled linter running under Jenkins but I'd rather not have to maintain this. It's a nice thing to have not really a core part of the project I want to spend too much time on. In other words I want a ready to roll solution. My project is in Python 3.x"

Very mild (value 2): The question is characterized by a very low degree of urgency. The technical debt item discussed may probably be fixed, but could also be not. An example is provided below:

*"**It's been almost 3 years since I posted this question.** I must say that I have tried exploring this despite the brilliant answers here. Some of the lessons I've learnt so far -More code smell come out by looking at the consumers (Unit tests are best place to look, if you have them). Number of parameters in a constructor are a direct indication of number of dependencies. Too many dependencies = Class is doing too much. Number of (public) methods in a classliSetup of unit tests will almost always give this awayliCode deteriorates over time, unless there is a focused effort to clear technical debt, and refactoring. This is true irrespective of the language.liTools can help only to an extent. But a combination of tools and tests often give enough hints on various smells. It takes a bit of experience to catch them in a timely fashion, particularly to understand each smell's significance and impact."*

Mild (value 3): The question presents some urgency, but the necessity to immediately address the debt item in the question is not considerable. The technical debt item discussed may be fixed, but probably not immediately. An example is provided below:

*"I'm writing my own MVC framework in PHP, just for learning purposes. It wasn't really hard to have a router/dispatcher class to call the right controller/action etc. But now i'm at the part where i'm going to use models. Or actually, the model layer. But there's something that confuses me. Alot of other MVC frameworks have a 'BaseModel'. I've read that this is actually bad practise, because the ""Model"" shouldn't be seen as another class. But as a real 'layer', which can contain things like the 'mapper' pattern or 'repository' pattern etc. But to be honest, i don't see any advantages in that. To me, a 'BaseModel' class seems to be the fastest way to go, with the same results. I can simply do something like: <code_snippet> But if i'd go for a repository pattern then i have to create the following: Repositories Entities DAO Entities have Aggregates to other repositories. So basically i'm manually writing out my entire database scheme to objects. In the end, what's the difference??? **Except that I probably could have saved a lot of time by simply using a BaseModel class...But why is it still considered to be a bad thing then??** It's not that the repo pattern decouples my application more then i'm doing now. Because to me, those patterns mentioned above seem to be highly overrated. It probably would only work in an application that has a shared state; Save objects locally (in the repository) and commit them later on. That's why i think no one can really answer this...But **i'm still hoping to see a decent answer that makes me go: "ahhhh... What was i thinking...."**. But if not, then i'm sure about my case that the BaseModel isn't a bad thing at all and that most bloggers are just a bunch of sheeps :-)"*

Moderate (value 4): The question clearly presents some urgency, but addressing the debt item presented in the question is not of critical importance. The technical debt item discussed will be fixed, but development/maintenance activities can progress regardless. An example is provided below:

"WARNING: This tale of woe contains examples of code smells and poor design decisions, and technical debt.

If you are conversant with SOLID principles, practice TDD and unit test your work, DO NOT READ ON.
Unless you want a good giggle at someone's misfortune and gloat in your own awesomeness knowing that you would never leave behind such a monumental pile of crap for your successors.
So, if you're sitting comfortably then I'll begin.
In this app that I have inherited and been supporting and bug fixing for the last 7 months I have been left with a DOOZY of a balls up by a developer that left 6 and a half months ago. Yes, 2 weeks after I started.
Anyway. In this app we have clients, employees and visits tables.
There is also a table called AppNewRef (or something similar) that ... wait for it ... contains the next record ID to use for each of the other tables. So, may contain data such as :-

TypeID	Description	NextRef
1	Employees	804
2	Clients	1708
3	Visits	56783

When the application creates new rows for Employees, it looks in the AppNewRef table, gets the value, uses that value for the ID, and then updates the NextRef column. Same thing for Clients, and Visits and all the other tables whose NextID to use is stored in here.
Yes, I know, no auto-numbering IDENTITY columns on this database. All under the excuse of "when it was an Access app". These ID's are held in the (VB6) code as longs. So, up to 2 billion 147 million records possible.
OK, that seems to work fairly well. (apart from the fact that the app is updating and taking care of locking / updating, etc., and not the database)
So, our users are quite happily creating Employees, Clients, Visits etc. The Visits ID is steady increasing a few dozen at a time. ***Then the problems happen. Our clients are causing database corruptions while creating batches of visits because the server is beavering away nicely, and the app becomes unresponsive. So they kill the app using task manager instead of being patient and waiting. Granted the app does seem to lock up.***
Roll on to earlier this year and developer Tim (real name. No protecting the guilty here) starts to modify the code to do the batch updates in stages, so that the UI remains 'responsive'. Then April comes along, and he's working his notice (you can picture the scene now, can't you ?) and he's beavering away to finish the updates.
End of April, and beginning of May we update some of our clients. Over the next few months we update more and more of them.
Unseen by Tim (real name, remember) and me (who started two weeks before Tim left) and the other new developer that started a week after, the ID's in the ***visit table start to take huge leaps upwards. By huge, I mean 10000, 20000, 30000 at a time. Sometimes a few hundred thousand.***
Here's a graph that illustrates the rapid increase in IDs used.
Take a look at his graph
Roll on November. Customer phones Tech Support and reports that he's getting an error. I look at the error message and ask for the database so I can debug the code. I find that the value is too large for a long. I do some queries, pull the information, drop it into Excel and graph it.
I don't think making the code handle anything longer than a long for the ID's is the right approach, as this app passes that ID into other DLL's and OCX's and ***breaking the interface on those just seems like a whole world of hurt that I don't want to encounter right now.***
One potential idea that I'm investigating is try to modify the ID's so that I can get them down to a lower level. Essentially filling the gaps. Using the ROW_NUMBER function
What I'm thinking of doing is adding a new column to each of the tables that have a Foreign Key reference to these Visit ID's (not a proper foreign key mind, those constraints don't exist in this database). This new column could store the old (current) value of the Visit ID (oh, just to confuse things; on some tables it's called EventID, and on some it's called VisitID). Then, for each of the other tables that refer to that VisitID, update to the new value. ***Ideas ? Suggestions ? Snippets of T-SQL to help all gratefully received.****"*

Severe (value 5): The question is urgent. The technical debt item discussed has to be fixed as soon as possible. An example is provided below:

"I have a Rails application with over 2,000 examples in my RSpec tests. Needless to say, it's a large application and there's a lot to be tested. ***Running these tests at this point is very inefficient and because it takes so long, we're almost at the point of being discouraged from writing them before pushing a new build.*** *I added --profile to my spec.opts to find the longest running examples and there are at least 10 of them that take an average of 10 seconds to run. Is that normal amongst you RSpec experts****?*** *Is 10 seconds entirely too long for one example****?*** *I realize that with 2,000 examples, it will take a non-trivial amount of time to test everything thoroughly - but* ***at this point 4 hours is a bit ludicrous****.What kind of times are you seeing for your longest running examples****?*** *What can I do to troubleshoot my existing specs in order to figure out bottlenecks and help speed things up.* ***Every minute would really help at this point.****"*

Very Severe (Value 6): The question is extremely urgent. The technical debt item discussed is critical, and has to be fixed immediately. An example is provided below:

"I have a large ruby on rails 2.3 which was ***now a disaster because of the slowness and many bugs. I'm the only programmer*** *and every day I've done debugging and* ***tearing my hair off because of this****. The* ***users are already using the product*** *but* ***so many bugs*** *and data are scattered. I was employed without prior knowledge of project development and management. Now* ***I'm suffering of having more overtime and a crisis on my codes to be fixed****. And also I've created this app while learning rails so there are codes there that became stranger to me.* ***What should I do? What are your suggestions?****What books do I need to read about more?****Please I need some help.****Thanks."*

References

[1] Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman. 2016. Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162).
[2] Li, Zengyang, Paris Avgeriou, and Peng Liang. "A systematic mapping study on technical debt and its management." Journal of Systems and Software 101 (2015): 193-220.
[3] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2009). Design Patterns Elements of Reusable Object-Oriented Software. Addison Wesley.