Table 1: Summary of Technical Debt (TD) prioritization approaches

Study	Process
S1	1. Identify an optimal system using an expert's knowledge and the evolution steps required to achieve the optimal system 2. Identify the TD items and the cost to repay each TD item 3. Assign probabilities to each evolution step for the likelihood of including each step in the current iteration 4. Calculate expected savings from addressing each TD item for the full evolution sequence 5. Subtract the expected savings from the cost of each TD item and sort the TD items from largest to smallest based on the result 6. Repay a given TD item if the expected savings exceed the calculated cost
S2	1. Identify TD items and their respective principal, expected interest, interest standard deviation, and correlations with other items 2. Select a component X, which will be significantly worked on in the upcoming release 3. Extract TD items that are associated with component X 4. Adjust estimates for the extracted TD items based on the current release plan 5. Set constraints for the selected portfolio model and a preferred risk level 6. Run the model to generate the optimal portfolio and use the results for prioritization
S3	1. Identify TD items 2. Identify decision criteria 3. Assign weights to each individual criterion 4. Evaluate each TD item based on the decision criteria using Weighted Sum Model (WSM) 5. Sort TD items based on the evaluation score and prioritize based on the sorted list
S4	1. Identify TD items 2. Identify software and/or infrastructure configuration items that are related to each TD item 3. Identify and model business processes based on the identified configuration items 4. Identify and prioritize business process activities, based on urgency and criticality, by consulting business stakeholders 5. Use a business perspective to prioritize TD items
S5	1. Obtain a list of change information from an issue tracking system 2. Perform impact analysis on the system to identify modules, that will most likely be affected by the change information, and calculate the impact probability score of each module 3. Identify TD items in the form of code smells 4. For each TD item, calculate its context relevance index (CRI) value by taking the summation of the impact probability scores of the modules, which match the given TD item 5. Prioritize TD items by their CRI value (with the larger values having higher priority)
S6	1. Identify TD items as defects present in a given system 2. For each TD item, calculate its principal (P), interest cost of defining a workaround (Iw), interest cost for customer support of the workaround (Ic), batch cost (Ip), probability of a requesting a batch (Ipr), and probability of fixing the defect (Ifr) 3. Compute the following cost-benefit ratio: $\frac{P}{Iw+Ic+Ip*Ipr+P*Ifr}$ 4. Prioritize TD items based on the cost-benefit ratio defined above
S7	1. Identify a design best practices for a given system 2. Identify TD as violations of the identified design best practices 3. Calculate a quality index for each violation by assessing each violation using a bench-marking approach 4. Assess the importance of each design best practice 5. Map the design best practices into a portfolio matrix using the quality index and the importance assessment 6. Create a quality design model to assign the design best practices to product parts 7. Assign remediation costs to each product part 8. Define an improvement strategy 9. Taking into account the improvement strategy, business needs, remediation cost, and best practices in the portfolio matrix, prioritize design best practice violations

Table 1: Summary of TD prioritization approaches

Study	Process
S8	Using JSpIRIT and expert knowledge: 1. Identify TD items in the form of code smells 2. For each TD item: (a) Calculate its component criterion by determining the set of classes affected by the given TD item (b) Count the number of components of blueprints affected by the number of classes and normalize the resulting score values (c) Calculate the concern criterion using developer's knowledge to identify the relationship between a given TD item and the architectural concern provided by the architecture blueprint (d) Count the number of concerns the TD item affects and normalize the score (e) Calculate the scenario criterion by taking the sum of the importance value of those identified and normalizing their values (f) Apply the voting criterion by defining individual thresholds for each of the criteria provided, and if the normalized score of a TD item is above the threshold, consider the TD item as potentially critical (g) If a TD item is considered potentially critical by more than one criterion, then take the average of those criteria 3. Prioritize the TD item based on their voting criterion value
S9	1. Identify TD items 2. Estimate cost, which is based on principal, and value, which is based on the multiplication of interest amount and interest probability, for each TD item 3. Categorize each TD item based on its value estimate of high/medium/low 4. Select a specific software component and extract a related list of TD items 5. Reevaluate the original high/medium/low value estimates for extracted items 6. Restrict extracted list of TD items to one of only high value items 7. Compare each item's cost and benefit in the restricted list and eliminate items of which their cost do not outweigh their value 8. Add up the estimated cost of the remaining items, if total cost can not be observed in the current release a decision should be made to eliminate some of the TD items from the current repayment
S10	1. Identify TD items in the form of defect TD 2. Identify the state of the software by calculating its degree of maintenance, which is calculated by dividing the accumulation of code churn of defect fixing activities by the initial software size 3. Study historical data to identify the major release that has the largest code churn 4. Divide historical data into two parts, historical data (before the major release) and testing data (after the major release) 5. Use testing data to apply the Markov chain model to simulate decisions at each release planning (RP) 6. At each RP, use the testing data up to the given RP time to determine the software state 7. At each RP, apply the model to estimate defect TD, principal, and interest 8. Use the principal and interest as inputs for Real-options Analysis (ROA) to prioritize defect TD
S11	1. Identify TD items 2. JSpIRIT uses predefined criteria or added criteria by a practitioner 3. Assign weights to each criterion 4. A score for each TD item is calculated using WSM, which can be altered by the practitioner
S12	1. Using expert knowledge, create a quality model by making a list of non-functional requirements, that define the "right" code 2. Using expert knowledge, develop an estimation model that estimates the remediation cost to refactor non-compliant code 3. Map each requirement to a Software Quality Assessment Based on Lifecycle Expectations (SQALE) quality characteristic that would be affected in the case of a requirement violation 4. Follow the SQALE indicator pyramid to prioritize TD repayment from the bottom to the top
S13	1. Analyze historical data to identify frequently refactored classes and categorize such classes as refactoring prone classes 2. From the identified refactoring prone classes, select classes that contain code smells that have a direct relationship with architectural problems and categorize them as TD items 3. For each TD class, calculate its class score, using the frequency score of a class (F), severity score of a class (S), severity score of a particular code smell $S(x_i)$, and the number of instances a code smell is present in a given class $I(x_i)$, as follows: $ClassScore = F * S * \Sigma(S(x_i) * I(x_i))$ 4. Prioritize TD classes based on class score in decreasing order

Table 1: Summary of TD prioritization approaches

Study	Process
S14	1. Identify self-admitted technical debt (SATD) items 2. Categorize each SATD item as a major or minor task, based on urgency, seriousness, and significance 3. Analyze the gap between a given task's significance and complexity 4. Categorize the tasks as expected or expedited tasks 5. Categorize tasks as "vital few" or "trivial many" 6. Identify plausible causes of key (i.e., buggy prone) SATD item tasks and calculate the rework effort for each task 7. Use the rework effort estimation metric as a guide in the prioritization process
S15	Using CodeScene: 1. Identify TD items as refactoring candidate files 2. Using a machine learning algorithm, which is based on technical and social factors, prioritize and visualize the candidate files 3. Using complexity trend analysis, determine whether the candidates files will continue to degrade in code quality 4. Using X-Ray analysis, prioritize individual functions inside the candidate files
S16	1. Identify TD items in the form of god classes 2. Calculate the WMC, ATC, and ATFD metrics for each god class and thresholds, as proposed by Marinescu [1] 3. For each class and for each metric, assign a rank (the closer a class is to a threshold, the lower the rank it is given) 4. Calculate the sums of the three given ranks (i.e., one for each metric) for each god class 5. Rank the god classes based on the calculated total sum, which indicates the cost of repaying each god class 6. Calculate the change likelihood and the defect likelihood for each god class 7. For each god class and for each likelihood, assign a separate rank 8. For each god class, calculate the sums of the two obtained ranks 9. Create a separate ranking for the god classes based on the calculated total sums of the likelihoods, which indicates the benefits of repaying each god class 10. Compute a final ranking for each god class by computing a profitability measure by subtracting the cost from the value
S17	1. Identify TD tables (i.e., tables below 4th normal form) 2. Using the database monitoring system, determine the TD tables' growth rate 3. For TD tables with high growth rate, calculate their I/O cost 4. For tables with high growth rate, calculate their values of the portfolio model variables ($expected\,return = \frac{1}{(I/O\,cost)}$ and $risk = table\,growth\,rate$) 5. Run the model on the data to produce the optimal portfolio of the TD tables
S18	1. Identify TD items in the form of classes that are defect and change prone 2. For each identified class, extract class level metrics for defect and change proneness 3. Using a Bayesian based prediction model, determine the TD proneness of each class 4. Using a predefined classification scheme, categorize classes, based on the TD proneness probability, into high, medium, or low 5. Apply Analytic Hierarchy Process (AHP) to generate a prioritized list of defect and change prone classes based on the result of the prediction model and a set of predefined prioritization criteria
S19	1. Identify TD in the form of defect TD items 2. Identify each TD principal as the difference between the defect reporting time and the resolving time 3. For each defect, calculate its interest amount as $(real\,fixing\,time - principal) * severity$ 4. Formalize the TD problem as a reinforcement learning problem as follows: (a) The goal is to eliminate all defect TD or maximize the amount of saved interest (b) The agents are the developers (c) The states are points in time in the total time remaining before the upcoming release (d) The set of actions is whether to repay TD (x) in the upcoming release or not (e) The reward equals the amount of saved interest resulting from paying TD (x) 5. Apply reinforcement learning, where the output policy is the prioritization of the defect TD
S20	1. Set a threshold on chosen metrics for detecting TD items in the form of code smells and identify TD items 2. Assign an Intensity Value (IV) to each metric used in the TD detection strategy 3. Calculate the Intensity Index (II) of each TD item by taking the average of each metric's IV 4. Calculate the exceeding ratio for each metric by dividing the metric value by the threshold value for each metric 5. Sum the ratios of each metric and round the value to obtain an index representing the II 6. Prioritize based on the II and refer to the exceeding ratio-based index if ties occur as an additional criterion

Table 1: Summary of TD prioritization approaches

Study	Process
S21	1. Identify TD items 2. Calculate the impact, defect, and change likelihood for each TD item 3. Calculate total benefit by multiplying the raw benefit (i.e., saved future effort) with the characteristics mentioned above 4. Calculate Return on Investment (ROI) of each TD item by dividing benefit by cost and prioritize TD items based on ROI
S22	1. Identify TD items in the form of requirement TD 2. Formulate the problem as a binomial valuation model as follows: (a) For each TD item, assign it an initial value in terms of its product value in product-driven software development or its market payoff in market-driven software development (b) Calculate the potential increase and decrease of the initial value to form the upward and downward branches (c) For every terminal node, calculate its net value (d) Recursively fold back the tree to calculate the present value of each node (e) Calculate the net present value of each node by subtracting the given option's exercise cost from the option's present value 3. Prioritize the TD items based on the output of the model
S23:	
AHP	1. Identify TD items 2. Identify the criteria related to each TD item (e.g., principal and interest) 3. Assign weights and scales to each criteria based on a series of pairwise comparisons 4. Perform a series of pairwise comparisons between the alternatives against the various criteria
ROA	1. Identify TD items and represent them as investment decisions 2. Identify the short term cost of each decision 3. Identify uncertain long term benefits of each decision 4. Apply options theory to determine the optimal timing of paying off each TD item

References

[1] Radu Marinescu. Detection strategies: Metrics-based rules for detecting design flaws. In *20th IEEE International Conference on Software Maintenance, 2004. Proceedings.*, pages 350–359. IEEE, 2004.