SUPERNOVA: Automating Test Selection and Defect Prevention in AAA Video Games Using Risk Based Testing and Machine Learning

Electronic Arts:
Alexander Senchenko,
Naomi Patterson,
Hamman Samuel,
and Dan Ispir

Published in:
2022 IEEE Conference on Software Testing, Verification and Validation (ICST),
August, 2022

s1290042
Yahagi Takuya

Introduction

- Recently, the size of software system (including game) became bigger.
- So, the cost of Quality Assurance (QA) testing became higher.
  (**QA testing**: testing to minimize the defect of software product)

- To solve this problem, they use following two concept.
  1. automating the selection of test cases.
  2. preventive approach
     … Before bug inducing, stop defects from entering software.

Related Work (Automation of test generation)

- **Katalon**: testing product that automates test generation.
  (For web, desktop, API, and mobile environment.)
- **Selenium**
- **TestComplete**

Related Work (Preventative measure for testing)

- **Infer (facebook):** static code analyzer.
- **Clever-Commit (Ubisoft)**

What is SUPERNOVA?

- SUPERNOVA(Selection of tests and Universal defect Prevention in External Repositories for Novel Objective Verification of software Anomalies)
  - Realize end-to-end automation for data science based testing with mathematical models.
    (**end-to-end testing** : test to check if soft works as expected and correctly)
  - These models automate workflows and drive decision making in QA testing.
  - SUPERNOVA needs two central components; end-to-end automation tool and machine or deep learning service.

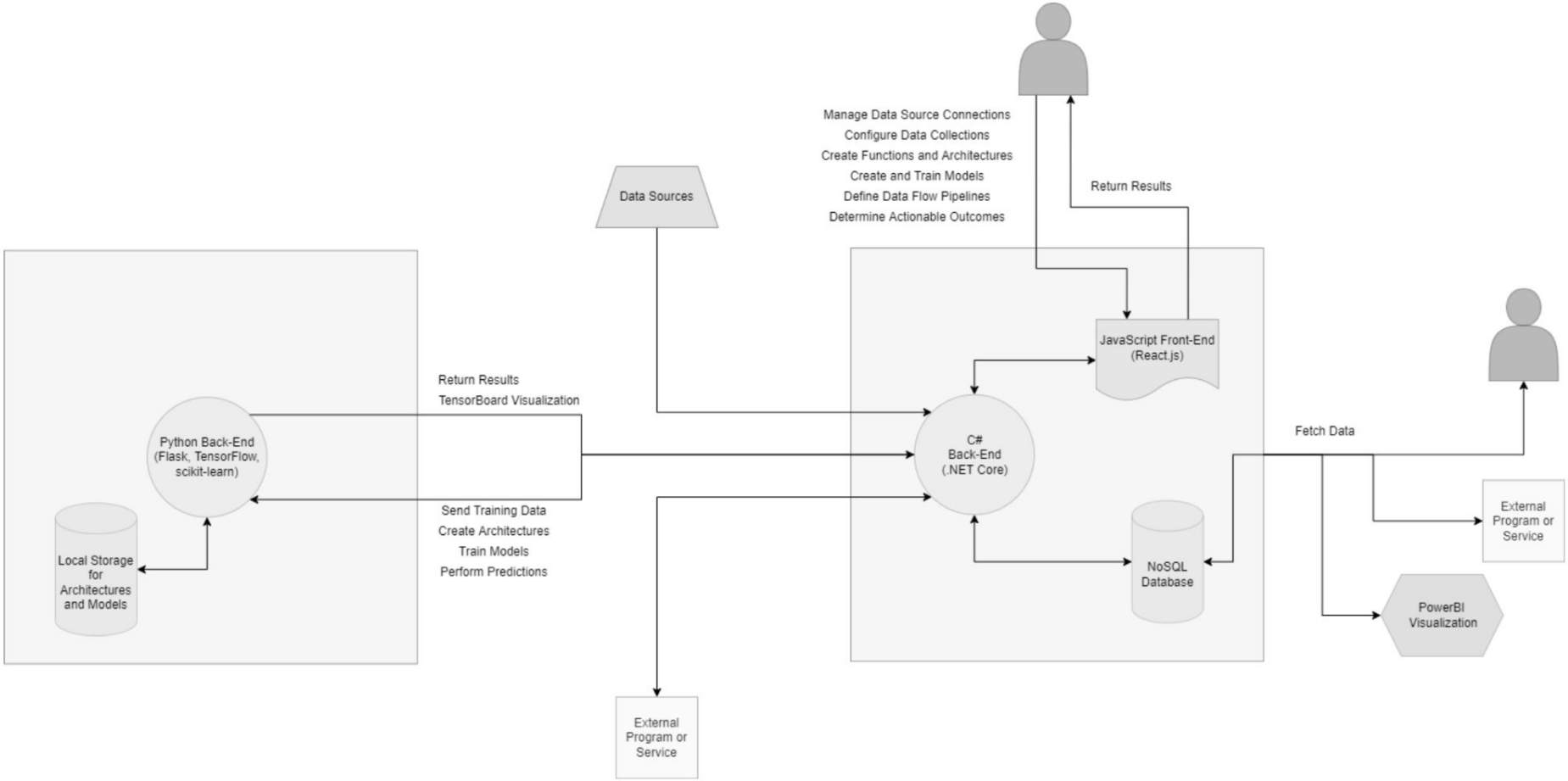


Fig. 1: High level overview of SUPERNOVA

SUPERNOVA's workflow

1. Data source connections and collections
   a. In SUPERNOVA, an interface for all relevant input source structures is provided.
      (ex. status/ level of issue, URL instance, project key, etc…)
2. Data configuration
   a. Combine these data into a unique specialized format.
3. Define functions
   a. With SUPERNOVA's visual programming interface, define evaluation functions that create metrics for describe useful behaviour.
      (ex. consecutive successful test runs, frequency of bug discovery, etc…)
   b. These functions can be represented as decision trees.

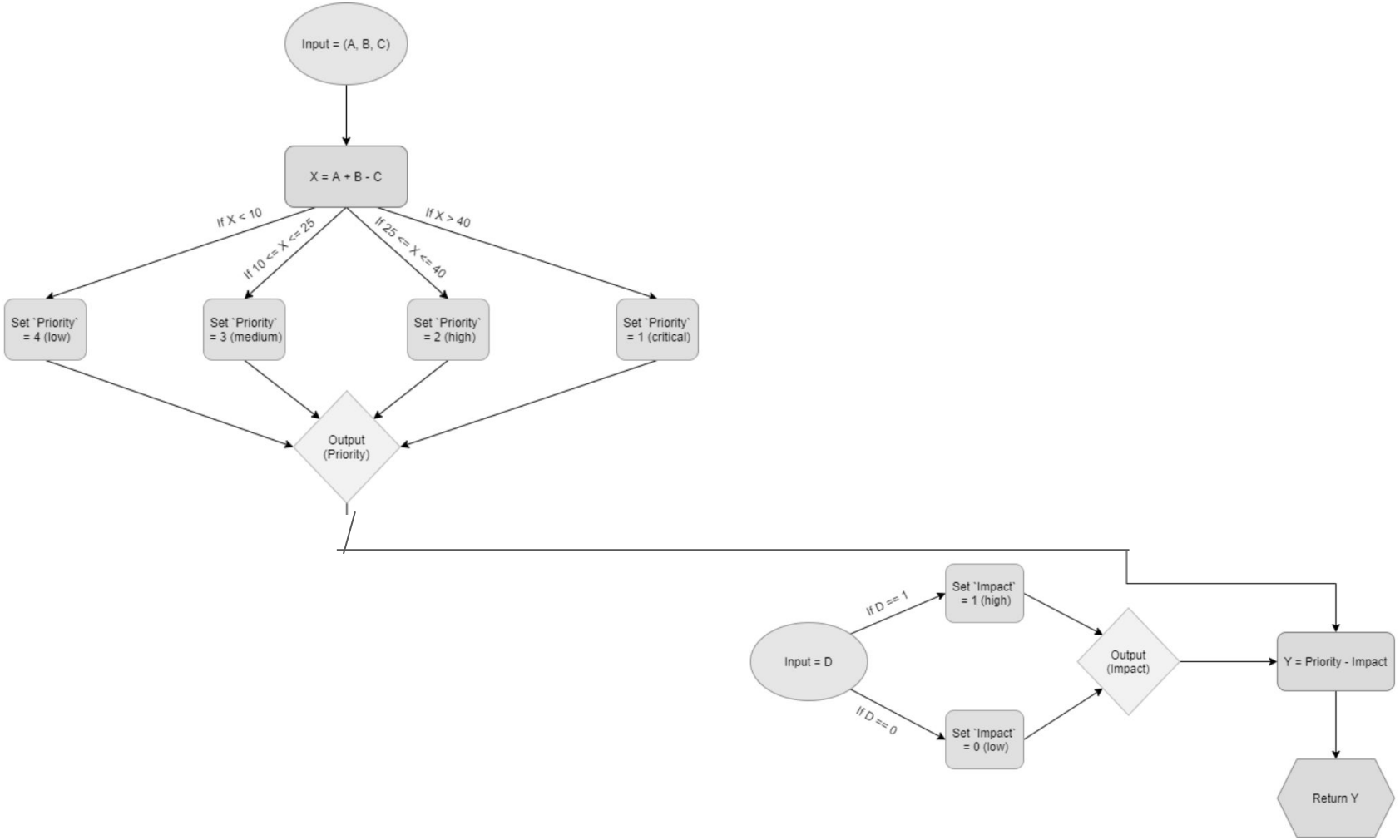


Fig. 3: Example subtree of a parent tree, used in one of EA's

SUPERNOVA's workflow

4. Making models
   There are three types of model on SUPERNOVA:
   a. Probabilistic Risk Based Testing (RBT) Model

   b. Machine Learning Model

   c. (Deep Learning Model)
      This is not used so in this research, I will not describe about it.
5. Decide output action
   It is done by 'pipeline' in SUPERNOVA.
   (ex. sending it as input to another model, exporting the data to a specified format, generating email reports, etc…)

SUPERNOVA's workflow - RBT Model -

- RBT Model
  ... simple mathematical formula.
  R: The risk exposure of a risk item 'a'. [0, 100]
  P: probability of the failure assigned to risk occurs. [0, 10]
  I: impact, the cost or level of failure if it occurs. [0, 10]
  T: time factor, initialized 1.0, if there are no or few bags at risk item 'a', then T(a) will be lower. [0, 1]
  w: weight, weight is valued by hand corresponding to the purpose of project.

$$P(a) = \frac{\sum_{j=1}^{m} p_j \cdot w_j}{\sum_{j=1}^{m} w_j}$$

$$I(a) = \frac{\sum_{j=1}^{n} i_j \cdot w_j}{\sum_{j=1}^{n} w_j}$$

$$R(a) = P(a) \cdot T(a) \cdot I(a)$$

$$T(a) = \frac{\sum_{j=1}^{k} t_j}{k}$$

SUPERNOVA's workflow - Machine Learning Model -

- Machine Learning model
  - uses JSON schema that interface directly with the scikit-learn library.
  - It provide arbitrary algorithms.
  - There are user interface with drop down and text box, so you can use it without skill of coding.

- Example:
  - whether a test found a bug,
  - if a commit is bug inducing.

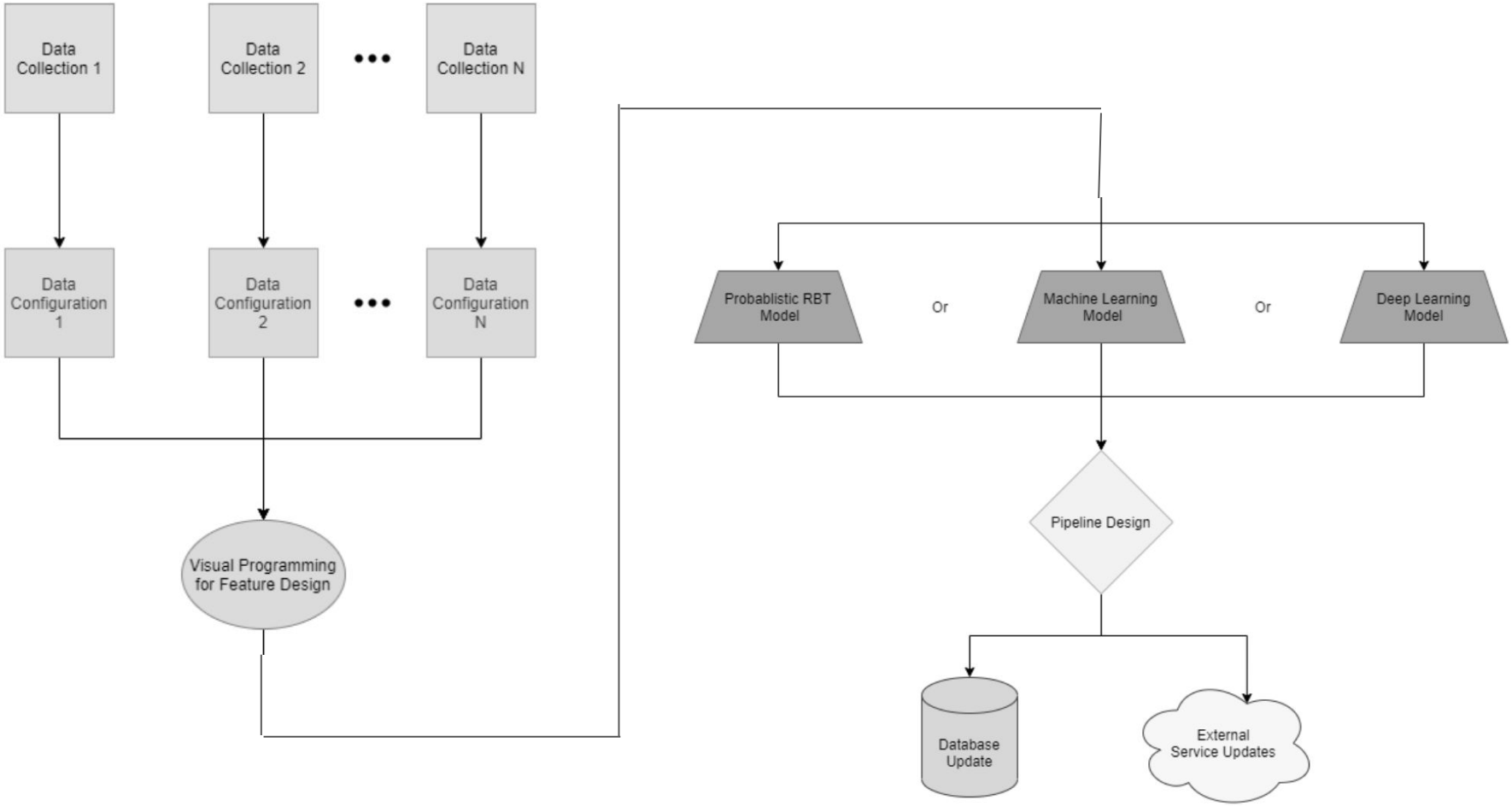


Fig. 2: Example of the SUPERNOVA workflow

Method

1. Automated Test Selection with Risk Based Testing:
   Target: EA's develop team
   Where: Sports Game 2019 (SG19) , and 2020 (SG20)

   For SG19: manual techniques.
   For SG20: SUPERNOVA's automated system.

Method

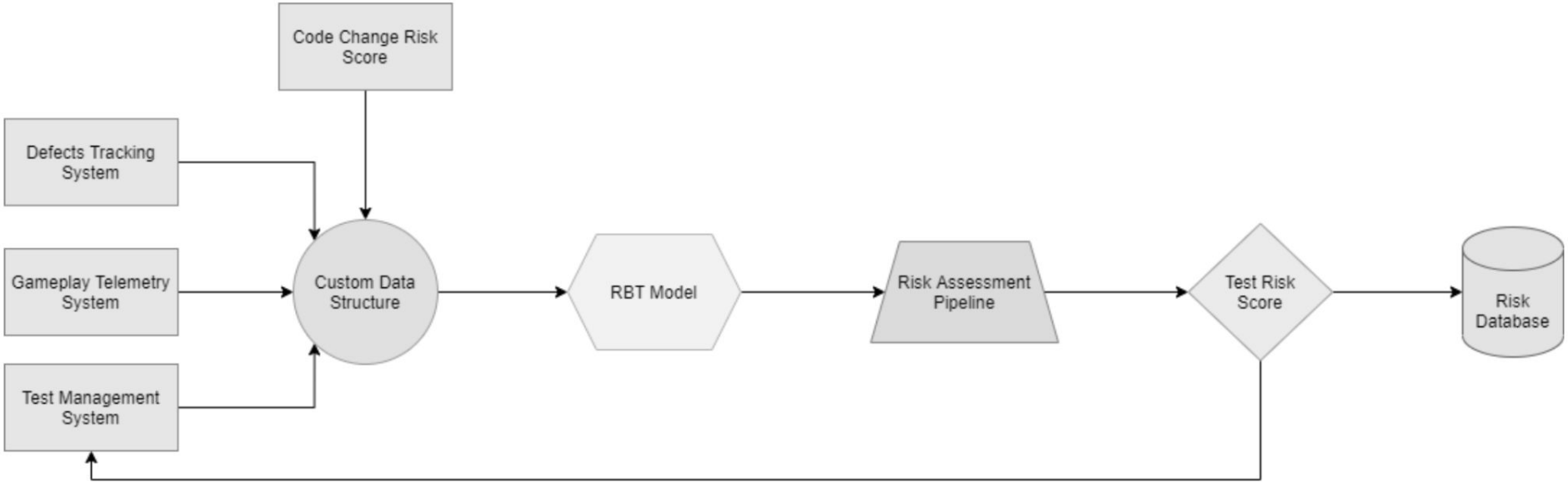


Fig. 4: Example of the RBT workflow

Method

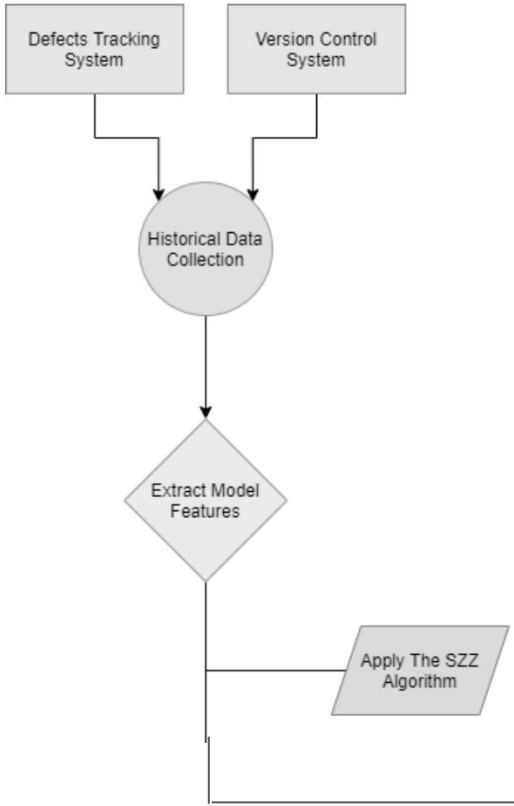
Name	Type	Description
Open Unaddressed Defects	Probability	Open unaddressed defects per test case.
Addressed Change Requests	Probability	Change requests can impact the game in unknown ways (given that these are requests that are not planned or accounted for). Knowing how many issues will be assessed will give us an indication of risk.
Defect to Change Ratio	Probability	The ratio of reported defects per feature change.
Script Failure Rate	Probability	The ratio of the positive and negative classes for each test case TC. If a TC has 3 passed, 4 failed and 3 blocked tasks, then its failure rate will be 70%.
Average Distribution	Impact	The average distribution is an automatically computed measure from game telemetry to understand which game modes were most significant to our players.
Average Stickiness	Impact	Stickiness is an automatically computed measure to understand the retention of a game mode.
QX Final Target	Impact	Expected quality milestone target, which captures production ready feature quality.
QX Target vs Current Target	Time	Semi-automatically weighted metric to assess the target vs expected quality milestone. Weights are applied against returned values.
Testing Hours in the Last T Days	Time	This is a automatically computed metric aggregating the total elapsed time for the past T days per test case (where the date is reflective off the tested-on date).
Days Since last tested	Time	Each task will have a tested-on date, which is subtracted from the current date to calculate the number of days since lasted tested.
Dev Changes in the last T days	Probability $\times$ Time	The amount of changes developers make will influence the risk and where we test. As changes increase in a given period, so will the risk associated with a particular task. We put a window on it to keep it timely and relevant.

TABLE I: Example criteria for a RBT model. These metrics capture coverage and/or yield across many different categories, which allow for robust RBT testing.

Method

2. Defect Prevention with Machine Learning
   Target: EA's Game (had not been published)
   Method: SZZ algorithm, semi supervised machine learning.
   machine learning is used for a binary classifier, whether a commit is bug inducing.

Method



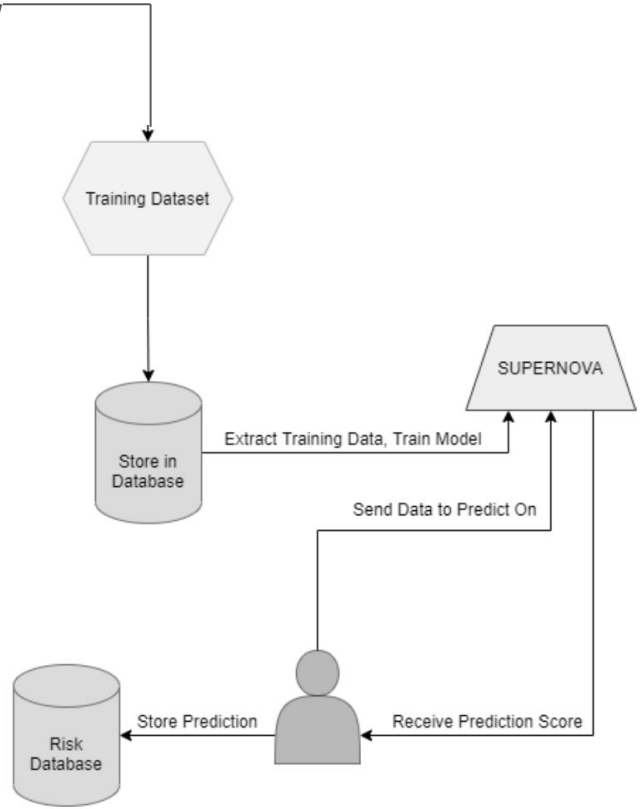
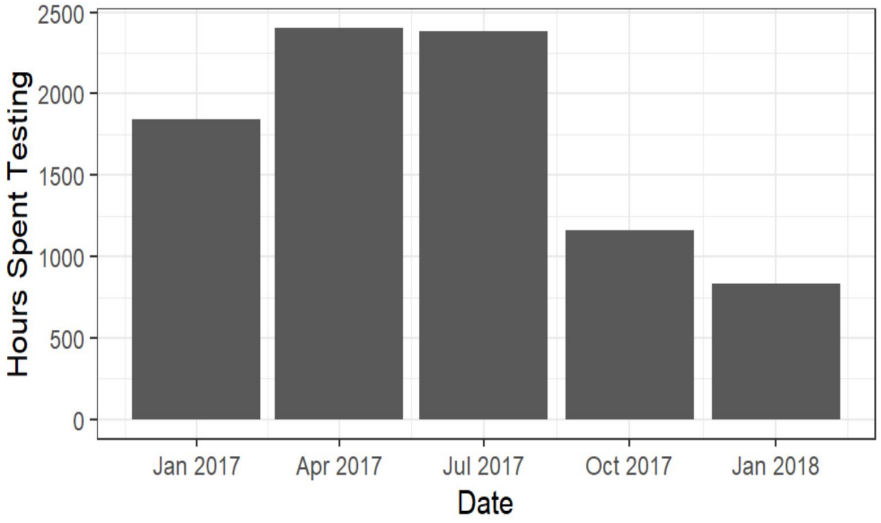
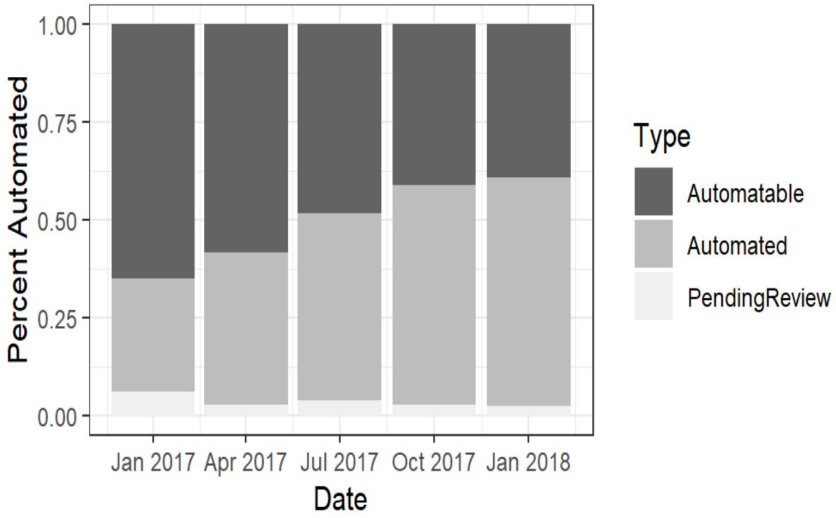


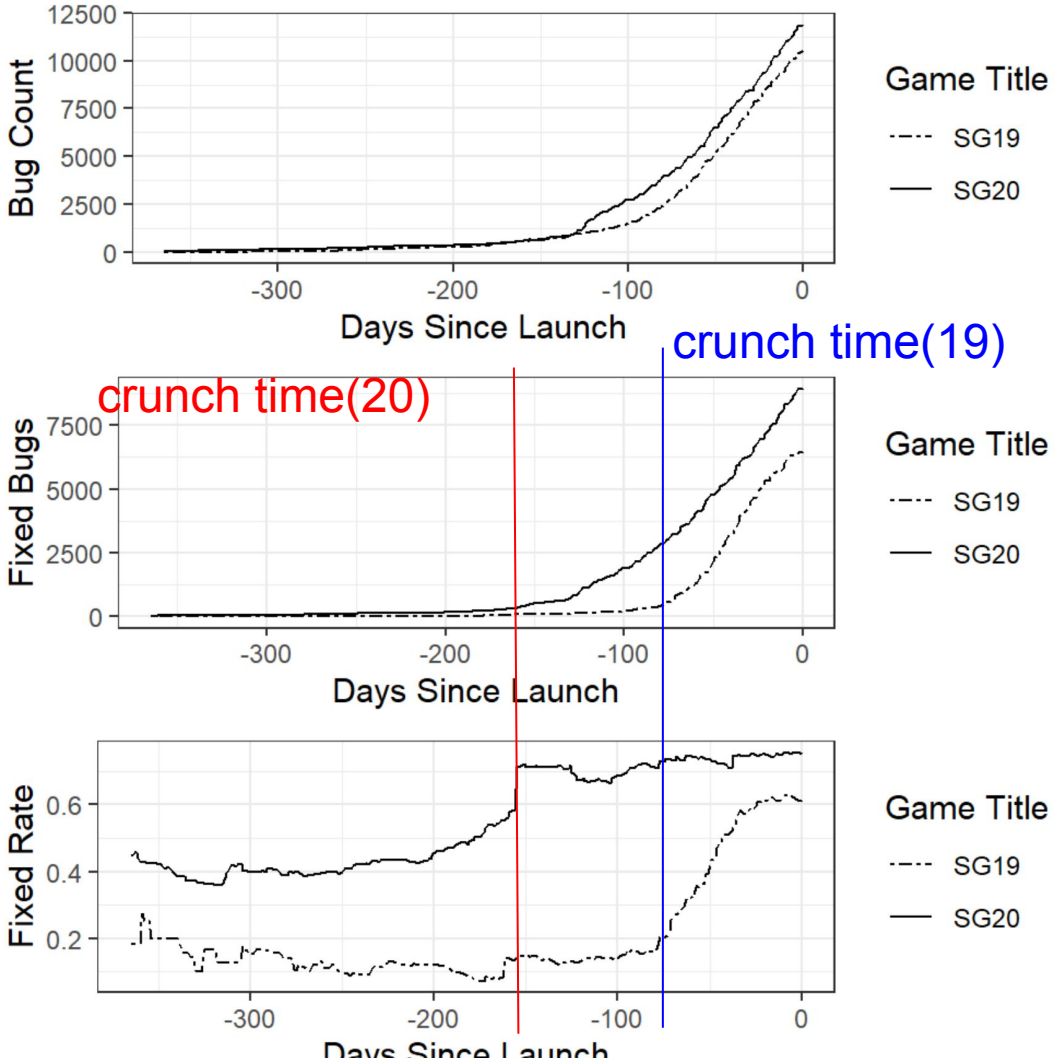
Fig. 5: Example of the defect prevention workflow

Result - Automated Test Selection with RBT-

After introducing SUPERNOVA, the change of hours spent testing and percent automated.






Bug Count
12500
10000
7500
5000
2500
0
-300
-200
-100
0
Days Since Launch
Game Title
SG19
SG20
crunch time(19)
crunch time(20)
Fixed Bugs
7500
5000
2500
0
-300
-200
-100
0
Days Since Launch
Game Title
SG19
SG20
Fixed Rate
0.6
0.4
0.2
-300
-200
-100
0
Game Title
SG19
SG20

Result - Automated Test Selection with RBT-

Another result:

	SG19	SG20	
Test planning hour	489	12 (for maintenance of SUPERNOVA)	97.5% cut cost
average bugs found per hour	0.186	0.227	-
overall testing time (hour)	48800	9776	reduce 80%
overall bugs found	10577	11909	

etc...

Result - Improvements from Defect Prevention -

- The model's overall macro average:
  Precision: 71%, (how it is correct)
  Recall: 77%, (how it imitate dataset)
  F1 score: 74% (the balance of precision and recall)
- Minority class performance (positive detection of more risky commits):
  Precision: 49%,
  Recall: 66%,
  F1 score: 56%,
- Majority class performance:
  Precision: 94%,
  Recall: 88%,
  F1 score: 91%

Discussions

- They show that by automating test selection, they can gain many merits. (ex, staff hours, cost, efficiency, and consistency.)
- SUPERNOVA can be tips to full testing automation.
- However, it should be viewed as a way to transition from old testing methods to new one.
- In future work, they attempt to include generation of test cases, as well as machine or deep learning.

Conclusion

- SUPERNOVA is able developer to reduce costs of test planning.
- about defect prevention, it need to improve SZZ algorithm. (more correctly)
- If defect prevention technology are improved, it increases efficiencies and is more reliable than traditional method.

Thanks for Listening!