Smart-система включає в себе два мікроконтролери: Atmega328 та ESP8266. Задача smart-системи зчитувати інформацію з датчиків та відправляти дані на сервер. З серверу Smart-система отримує команди для виконуючих механізмів. Мобільний додаток, в свою чергу, отримує показники з серверу. Користувач може відправляти команди до Smart-системи через сервер. Таким чином, реалізується моніторинг та автоматизоване управління життєзабезпечення рослини. Передача інформації була виконана за допомогою наступних технологій зв'язку: UART – для зв'язку між контролерами; Wi-Fi – для зв'язку Smart-системи з адаптером, та для зв'язку між мобільним додатком та сервером (таким чином використовується і WiMAX); Ethernet – для доступу до серверу. Мобільний додаток був написаний за допомогою Xamarin.Forms на мові програмування C# для мобільного пристрою на операційній системі Android. В доповіді наведено результати дослідження відмово стійкості як одного із показників надійності створюваної smart-системи.

ДОСЛІДЖЕННЯ ЯКОСТІ ВИМОГ ДО ПЗ ТА ЇЇ ВПЛИВ НА ФУНКЦІОНАЛЬНІСТЬ ФІНАЛЬНОГО ПРОДУКТУ. РОЗРОБКА МЕТОДУ АНАЛІЗУ ТА ОЦІНКИ ЯКОСТІ ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ.

Шевченко В. П.

Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський Авіаційний Інситут», кафедра мехатроніки та електротехніки, спеціальність 152

Однією з основних проблем, з якою стикається управління якістю під час аналізу вимог у програмних проектах, є прийняття рішення про те, чи достатньо якісною є специфікація вимог до програмного забезпечення (SRS).

Існуючі дослідження підтверджують вплив документації на якість програмного забезпечення. Ось деякі ключові висновки:

- Згідно з дослідженням IBM, команди розробників програмного забезпечення, які постійно документують свій код, мають на 15% менше дефектів.

- В опитуванні, проведеному Stack Overflow, розробники назвали читабельність коду та зручність його супроводу найважливішими факторами якості коду. Документація відіграє ключову роль у тому, щоб зробити код читабельним та зручним для обслуговування, що, в свою чергу, може покращити якість програмного забезпечення.

Різні технічні вимоги до програмного забезпечення (SRS) важко порівнювати через унікальність проектів, в рамках яких вони були створені. В рамках даної роботи буде визначено модель якості для SRS та виведені необхідні метрики, використовуючи підхід GQM - «ціль-питання-метрика». На основі цього ми знайдемо поріг якості для успіху проекту.

У цій роботі продемонстровано, що якість може бути виміряна на основі формальних та об'єктивних метрик. Це важливо, оскільки дозволяє оцінити шанси успіху проекту та фінального продукту на основі метрик SRS. Якщо якість є нижчою за певний поріг якості, то проект з більшою ймовірністю може зазнати невдачі. Для того, щоб визначити цей поріг, було досліджено близько 20 проектів на основі методу "ціль-питання-метрика". На основі отриманих

результатів було виявлено два специфічних пороги: Нижчий поріг — проекти, які мають якість SRS нижче цього значення, знаходяться під великою загрозою. Більш високий поріг — проекти, які мають якість SRS вище цього значення, мають більше шансів на успіх.

Щодо взаємозв'язку між якістю SRS та успіхом проекту ми маємо наступні гіпотези:

Гіпотеза 1. Проекти з високим показником якості мають більше шансів на успіх (категорія «++») . Фактори впливу: Взаємозв'язок між формальними аспектами якості та якістю змісту SRS, як зазначено в. Висока якість SRS може також бути ознакою добре організованої команди, яка з більшою ймовірністю досягне успіху у створенні цінного програмного забезпечення. Гіпотеза вважається вірною, якщо буде знайдено верхній поріг, при якому більше 75% проектів, що набрали більше балів, потрапляють в категорію «+» або «++».

Гіпотеза 2. Проекти з дуже низькою оцінкою якості набагато частіше зазнають невдачі (категорія «-») . Фактори впливу: Низька якість SRS - досить погано впливає на проект. Але команди, які впроваджують погану SRS, можуть мати додаткові проблеми. Наприклад, члени команди можуть працювати один проти одного або мати поганий тайм-менеджмент. Ці труднощі можуть збільшуватися по мірі реалізації проекту. Гіпотеза підтверджується, якщо можна знайти нижчий поріг якості, при якому більше 75% проектів отримують нижчу оцінку, то вони провалюються («-» або «--») .

Список використаних джерел

Карл Вігерс: Розробка вимог до програмного забезпечення. Software Requirements. 203-221 (August 15, 2013) // Microsoft Press; 3rd edition

Гауз, Д.С., Вайнберг, Г.М.: Вивчення вимог: Якість перед проектуванням (1989)

IIBA: Посібник з бізнес-аналізу (BABOK Guide). A Guide to the Business Analysis Body of Knowledge // International Institute of Business Analysis; 3rd edition (April 15, 2015)

ВПЛИВ СИСТЕМИ КОНТРОЛЮ НА ПРАЦЕЗДАТНІСТЬ НЕПЕРЕРВНИХ ОБ'ЄКТІВ КОНТРОЛЮ ПРИ ДИСКРЕТНОМУ КОНТРОЛІ З ВІДНОВЛЕННЯМ

Благодарний Микола Петрович, Лома Валерія Юріївна

Національний аерокосмічний університет ім. М. Є. Жуковського

"Харківський авіаційний інститут"

При дискретному контролі стан об'єкту контролю (ОК) контролюється дискретно (в моменти часу $\boldsymbol{t_1, t_2, \ldots, t_l, \ldots}$). За результатами контролю здійснюється управління якістю ОК[1, 2]. ОК ідентифікується як працездатний або такий, що має невиявлені відмови. імовірності. Ймовірність працездатності ОК $\boldsymbol{P(t)}$ після чергового етапу контролю з відновленням будемо визначати за наступним виразом

$$\boldsymbol{P(t_k) = P_0(t_k)P(t_k|1) + (1 - P_0(t_k))P(t_k|2) = 1 \; P_н(t_k)} \qquad (1)$$

де $\boldsymbol{P_0(t_k)}$ – ймовірність працездатності ОК в момент часу $\boldsymbol{t_k, k = 1, 2, \ldots,}$ перед черговим етапом контролю; $\boldsymbol{P(t_k|1)}$ – ймовірність працездатності ОК після обслуговування при відсутності відказу перед контролем (знаходження в працездатному (1-му стані)); $\boldsymbol{P(t_k|2)}$ – ймовірність працездатності ОК після обслуговування при наявності в ньому відказу перед контролем (знаходження в несправному (2-стані)у разі невиявлення відказу); $\boldsymbol{P_н(t_k)}$ – ймовірність невиявлення засобами контролю відказу в момент часу $\boldsymbol{t_k}$.