Streszczenie

Metoda wykrywania konfliktów zasobowych w aplikacjach wielowątkowych

mgr inż. Damian GIEBAS

Prawie każdy z powszechnie dzisiaj stosowanych typów procesorów wspiera technologię wielowątkowości. Dzięki tej technologii programiści otrzymali możliwość równoległego wykonywania zadań w ramach jednej aplikacji. Aby do tego doszło należało jednak rozszerzyć istniejące języki programowania o odpowiednie biblioteki, dzięki którym programista może wykorzystać wspomnianą technologię. Skutkiem tego wiele języków, w tym język C, umożliwia tworzenie aplikacji wielowątkowych, pomimo że języki te nie zaprojektowano w tym celu. Efektem takiego stanu rzeczy jest pojawienie się nowej kategorii błędów. Polega ona na możliwości tworzenia struktur w kodzie programu, których skutkiem są wzajemne (zwykle nieprzewidziane) interakcje między jednostkami logicznymi nazywanymi wątkami. Struktury te powodują zwiększenie kosztów wytwarzania oprogramowania, a to oznacza, że tworzenie aplikacji wielowątkowych staje się w wielu przypadkach nieopłacalne. W takiej sytuacji poszukiwana jest odpowiedź na pytanie: czy zadana aplikacja wielowątkowa napisana w języku C z użyciem biblioteki *pthread* jest wolna od błędów strukturalnych prowadzących do konfliktów zasobowych w szczególności: szkodliwej rywalizacji, zakleszczenia, naruszenia niepodzielności i naruszenia porządku?

W celu lokalizowania błędów prowadzących do konfliktów zasobowych opracowano wiele metod i narzędzi. Wiele z tych rozwiązań wspiera lokalizowanie od jednego do trzech typów konfliktów zasobowych, a więc ich użycie nie pozwala odpowiedzieć na wcześniej postawione pytanie. Powodem tego są m. in. pierwotne założenia istniejących rozwiązań (np. wykorzystanie prostych modelów), które powodują, że nie są one na tyle elastyczne, aby można było wykorzystać je w celu lokalizowania czterech typów konfliktów zasobowych w aplikacjach wielowątkowych pisanych za pomocą języka C z użyciem biblioteki *pthread*. Innym istotnym czynnikiem, który nie spełniało żadne inne rozwiązanie jest możliwość jego wykorzystania on-line tzn. czas potrzebny na lokalizowanie błędów wynosił nawet kilka godzin. Oznacza to brak rozwiązań wspierających programistów języka C w procesie wytwarzania kodu źródłowego aplikacji wielowątkowych, w sposób umożliwiający szybką analizę kodu źródłowego, w celu zlokalizowania ww. błędów strukturalnych prowadzących do konfliktów zasobowych.

Analiza literatury tematu wykazała, że aby osiągnąć cel jakim jest opracowanie metody umożliwiającej lokalizowanie konfliktów zasobowych w trybie on-line należy wykorzystać metody analizy statycznej. Aby osiągnąć ten cel należało opracować model umożliwiający wyznaczanie warunków pozwalających na wykrywania błędów prowadzących do konfliktów zasobowych pozwalających na obniżenie kosztów wykorzystania metod analizy statycznej. Opracowany model kodu źródłowego aplikacji wielowątkowej zorientowany na lokalizowanie czterech typów konfliktów zasobowych zakłada wykorzystanie języka C i biblioteki *pthread* w trybie on-line. Model ten pozwala na analizę kodu źródłowego w ujęciu strukturalno-czasowym.

W dysertacji przedstawiono problem lokalizacji błędów w aplikacjach wielowątkowych. Przegląd literatury tematu pozwolił doprecyzować założenia, składające

się na podstawy modelu kodu źródłowego i warunki wykrywania błędów. Umożliwiło to opracowanie metody, jej zaimplementowanie w prototypowym narzędziu i przeprowadzenie eksperymentów w celu weryfikacji opracowanej metody.