Гломозда Дмитро Костянтинович, магістр, аспірант кафедри інформатики Національного університету „Києво-Могилянська Академія”.

Адреса для листування: 03037, Київ, вул. Максима Кривоноса, д. 14, кв. 7, e-mail: glomozda@voliacable.com

ТОТАЛЬНІСТЬ АЛГОРИТМУ РОБОТИ КООРДИНАЦІЙНОГО МЕХАНІЗМУ СИСТЕМИ ПРОГРАМНОЇ ПІДТРИМКИ МЕРЕЖНОЇ СПІВПРАЦІ В РАЗІ ВИНИКНЕННЯ ПОМИЛКИ

Вступ

В роботі продовжується розгляд питань, пов’язаних з моделюванням дистанційної взаємодії в мережі Інтернет. В попередніх роботах [1; 2] було описано загальну структуру моделі колаборативного середовища (далі КС). Але досі ми розглядали лише безвідмовне середовище, в якому не виникає жодних позаштатних ситуацій. Реальне середовище такої властивості немає, тому наступним етапом буде моделювання поведінки системи в разі виникнення помилок, викликаних як діями окремих користувачів, так і збоями обладнання чи програмного забезпечення системи.

Моделювання роботи КС в разі виникнення помилки

В загальному випадку мережа Петрі, яка моделює роботу координаційного механізму колаборативної системи, до складу якої входять N користувачів (позначимо їх U), M сеансів (S) та L рівнів (F), складатиметься з:

- $3 * N$ позицій та $4 * N$ переходів, що описують поведінку користувачів;
- $N * L$ переходів типу „U_i хоче створити ресурс R_k” ($i \in 1…N; k \in 1…L$);
- $4 * M$ позицій та M переходів, що описують роботу контролера сеансу;
- $9 * L$ позицій, що відповідають контролерам рівнів;
- $3 * L$ позицій та $3 * L$ переходів, що описують роботу контролера рівня та ресурсу, що йому відповідає;
- $3 * N * M$ позицій та $4 * N * M$ переходів, що регламентують взаємовиключне створення сеансів;
- $5 * N * L$ позицій та $10 * N * L$ переходів, що відповідають зв’язкам між контролерами рівнів та користувачами.

У випадку із 2 користувачами, 1 рівнем і 1 ресурсом матимемо мережу, що складатиметься з 38 позицій та 42 переходів. В праці [1] було доведено, що ця мережа обмежена та активна.

Розглянемо, як зміниться мережа після додавання до неї позицій та переходів, що моделюватимуть реакцію системи на позаштатні ситуації. Спершу розглянемо її реакцію на помилки, зумовлені діями користувачів (наприклад, спробу порушити роботу системи чи доступитися до закритої інформації). Вважатимемо, що користувач скоює недозволені дії, маючи контроль над ресурсом. Маємо два можливих варіанти: а) він одночасно є керівником сеансу, б) не є керівником сеансу.

Почнемо з найпростішого випадку, коли порушник — простий учасник сеансу. В цьому випадку реакція системи обмежиться примусовим від’єднанням порушника від системи та звільнення зайнятого ним ресурсу. Змоделюємо цю реакцію за допомогою переходів „U_i здійснив недозволену дію, не будучи керівником жодного з сеансів та займаючи при цьому F_k”, ($i \in 1…N; k \in 1…L$). Таким чином, всього нам знадобиться $N * L$ таких переходів. Вхідними позиціями цих переходів для фіксованих користувача U_x і рівня F_y будуть:

- всі позиції виду „U_x не є керівником S_j” ($j \in 1…M$);
- позиція „U_x є утримувачем F_y”;
- позиція „F_y активно використовується U_x”;
- позиція „Ресурс R_y використовується”;
- позиція „U_x перебуває в системі”.

Вихідними позиціями будуть:

- „U_x знаходиться поза системою”;
- „F_y вільний”;
- „Ресурс R_y вільний”.

В результаті спрацювання цих переходів користувач опиниться поза системою, а зайняті ним рівень та ресурс будуть вільними та готовими до використання іншими користувачами.

Тепер розглянемо випадок, коли користувач, який здійснив нелегальну дію, був не лише утримувачем рівня, а й керівником сеансу. В цьому випадку при його аварійному відключенні треба прослідкувати, щоб воно якомога менше відбилося на роботі інших учасників цього сеансу. В залежності від реалізації системи та особливості ситуації, що склалася, можна або призначити нового керівника сеансу з-поміж його учасників, або коректно від'єднати їх. Змоделюємо другий сценарій. Для цього нам знадобляться $N * M * L$ переходів, позначених „U_i здійснив недозволену дію, будучи керівником S_j та займаючи при цьому F_k", ($i \in 1…N$; $j \in 1…M$; $k \in 1…L$). Вхідними позиціями для фіксованих користувача U_x, рівня F_y та сеансу S_z будуть всі ті самі позиції, що й в попередньому випадку, крім однієї, „U_x не є керівником S_z", місце якої займе „U_x є керівником S_z", плюс позиція „S_z активний". До переліку вихідних позицій додадуться „S_z вимикається" та нова спеціальна позиція „U_x примусово від'єднується від системи" і перехід від неї до позиції „U_x знаходиться поза системою". Вона потрібна для більш чіткої синхронізації переходу користувача у стан „поза системою" з переходом сеансу у стан „вимкнений". Отже, в підсумку маємо $N * M * L + N$ додаткових переходів та N додаткових позицій.

Тепер змоделюємо поведінку системи в разі виникнення критичної помилки, яка унеможливлює її подальшу роботу. Система повинна подолати таку ситуацію з мінімальними втратами і можливістю відновити роботу в повному обсязі. В термінах мережної моделі це означатиме повернення до початкового стану, коли всі користувачі знаходяться поза системою, сеанси неактивовані, рівні вільні. Розглянемо помилку, що відбувається у найнесприятливіший момент, коли всі користувачі активно працюють у системі чи то в якості утримувачів рівнів, чи в якості керівників сеансів. Щоб охопити якнайбільше користувачів, припустимо, що $N = M + L$, і що одні M користувачів є контролерами сеансів, а інші L — займають рівні. Щоб змоделювати поведінку такої системи мережею Петрі, нам знадобиться множина переходів виду „U_1 є керівником S_1, U_2 є керівником S_2, …, U_M є керівником S_M, U_{M+1} є утримувачем F_1, U_{M+2} є утримувачем F_2, …, U_N є утримувачем F_L" до „U_N є керівником S_1, U_{N-1} є керівником S_2, …, U_{N-M} є керівником S_M, U_{N-M-1} є утримувачем F_1, U_{N-M-2} є утримувачем F_2,…, U_1 є утримувачем F_L". Всього таких переходів буде N!.

У випадку двох користувачів, одного сеансу та одного рівня мережа (позначимо її M) складатиметься з 40 позицій та 52 переходів. Використовуючи графічний редактор PM Editeur [3] та Petri Nets Toolbox [4], переведемо мережну модель в матричний вигляд, що оброблятиметься програмою MATLAB. В результаті обробки MATLAB'ом отримуємо дерево досяжності для нашої мережі.

Теорема 1. Мережа моделі колаборативної системи M обмежена і активна.

Доведення. Проаналізуємо дерево досяжності, побудоване за допомогою програми MATLAB. Воно містить 186 можливих варіантів маркування, тому M обмежена (в іншому випадку маркувань було б нескінченно багато). Крім того, вона активна, оскільки дерево досяжності не містить термінальних вершин. Доведено.

Тотальність алгоритму роботи КС в разі виникнення помилки

Розглянемо алгоритм реагування КС на помилку з точки зору тотальності. Тотальним називається такий алгоритм мережної взаємодії, згідно з яким ухвалення рішення потребує участі всіх процесів в мережі [5]. Поняття тотальності алгоритмів є важливим для розробки алгоритмів керування мережею, оскільки, по-перше, алгоритми розв'язання багатьох важливих задач мережної взаємодії є необхідно тотальними, а по-друге, для розв'язання цих задач можна застосувати будь-який тотальний алгоритм.

Покажемо, що алгоритм відкату системи після помилки в початковий стан тотальний. Ми можемо це припустити, якщо порівняємо його з розглянутим в праці [5] алгоритмом ресинхронізації, який полягає в тому, що спершу всі процеси в мережі переводяться до стану *synch*, а потім — до стану *normal*, при чому процес може перейти до стану *normal* лише після того, як в певний момент часу всі процеси одночасно перебувають в стані *synch*. В нашій моделі станом *synch* для процесів-користувачів буде стан „U_i перебуває поза системою" ($i \in 1…N$), для процесів-контролерів сеансу — „S_j вимкнений" ($j \in 1…M$), для процесів-контролерів рівнів — „F_k вимкнений" ($k \in 1…L$), для ресурсів — „R_k відсутній" ($k \in 1…L$). Стану *normal* відповідатиме стан користувача „U_i намагається увійти в систему" ($i \in 1…N$). Вводити стани-аналоги *normal* для інших компонентів системи немає потреби, оскільки, згідно з нашою моделлю, вони не зможуть перейти в жоден інший стан без команд

користувачів. Тому в нашому випадку точкою ухвалення системою рішення вважатимемо перехід всіх користувачів в стан *normal*.

Доведемо це. Спершу наведемо формальне визначення тотальності.

Означення 1. Подія *a передує* події *b*, якщо:

1) $a = b$, або *a* та *b* відбуваються в одному й тому самому процесі, і *a* відбувається раніше за *b*;
2) *a* — подія надсилання повідомлення, а *b* — відповідна подія одержання повідомлення;
3) існує така подія *c*, що $a \to c$ та $c \to b$.

Означення 2. Виконання алгоритму *тотальне*, якщо принаймні один процес *p* ухвалює рішення, і для кожного $q \in P$ і кожного *p*, що ухвалює рішення, $e_q \to d_p$ (де *P* — множина всіх процесів, e_q — перша подія процесу *e*, d_p — подія ухвалення рішення процесом *p*). Алгоритм *тотальний* тоді, коли всі можливі його виконання тотальні.

Для доведення тотальності алгоритму роботи КС в разі виникнення помилки використаємо теорему [5]:

Теорема 2. Нехай $a_1 \ldots a_k$ — виконання деякого алгоритму А, та нехай $a_{\sigma(1)} \ldots a_{\sigma(k)}$ — така перестановка подій, що з $a_{\sigma(1)} \to a_{\sigma(k)}$ випливає, що $i \le j$. Тоді $a_{\sigma(1)} \ldots a_{\sigma(k)}$ — також можливе виконання алгоритму А.

Таким чином, маємо:

Теорема 3. Будь-який алгоритм роботи КС в разі виникнення помилки тотальний.

Доведення. Нехай А — нетотальний алгоритм роботи КС в разі виникнення помилки, тобто існує таке виконання алгоритму А, в якому для деякого процесу *q* e_q не передує ухваленню рішення. З теореми 2 випливає, що можна побудувати таке виконання, в якому ухвалення рішення передує переходу *q* в стан *synch*, тому А — некоректний. Доведено. ■

Теорема 4. Для реалізації поведінки КС в разі виникнення помилки можна застосувати будь-який тотальний алгоритм.

Доведення. Нехай А — тотальний алгоритм. Змінимо його таким чином: кожний процес *q* змінює свій стан одразу після залучення в А, тобто в момент e_q, а кожний процес *p* (в нашому випадку, користувач) змінює стан на *normal* тоді, коли ухвалює рішення, тобто в момент d_p. Оскільки А тотальний, отриманий в результаті алгоритм роботи КС в разі виникнення помилки буде коректним. Доведено. ■

Висновки

Результатом роботи є подальший розвиток та уточнення загальної моделі системи програмної підтримки мережної співпраці. Доведення теорем 3 і 4 дозволяє пов'язати нашу модель з одержаними раніше важливими теоретичними результатами, що полегшить подальшу роботу в цьому напрямку.

ЛІТЕРАТУРА

1. *Гломозда Д.К., Глибовець М.М.* Формальна модель функціонування колаборативного середовища // Тези доповідей Третьої Міжнародної конференції „Теоретичні та прикладні аспекти побудови програмних систем" (TAAPSD'2006). — Київ (Україна). — 2006. — С. 225—230.
2. *Глибовец Н.Н., Гломозда Д.К.* Сложность задачи верификации координационного механизма системы программной поддержки совместной сетевой работы // Кибернетика и системный анализ. — 2008. — № 4. — С. 15—19.
3. PM Editeur (http://www.cremona.polimi.it/dispense/Automazione%20Industriale/pm_us.zip).
4. *Svadova M., Hanzalek Z.* Matlab Toolbox for Petri Nets // INOVACE 2000. — Praha. — 2000. — P. 15.
5. *Tel G.* Total algorithms // Proc. International Conf. on Concurrency "Concurrency-'88". — Hamburg (Germany). — 1988. — P. 277—291.