# Dwa paradygmaty w zarządzaniu projektami: teoria ograniczeń Goldratta i teoria sieci Petriego[1]

Andrzej Blikle

14 czerwca 2011

Podstawowe tezy tej pracy powstały w domu mojego przyjaciela Jacka Padlewskiego położonym w przysiółku Les Aujares na wysokości 1700m w Alpes du Sude we Francji

Teoria ograniczeń Goldratta, która stanowi prawdziwy przełom w zakresie zarządzania projektami, posługuje się dość nieprecyzyjnym językiem, co — zdaniem autora niniejszej pracy — nie pozwala na pełne wykorzystanie jej zalet praktycznych. Niniejsza praca stawia sobie za cel pokazanie, że opisanie idei Goldratta w języku sieci Petriego wyposaża tę pierwszą w narzędzie pozwalające na rozwinięcie jej zalet zarówno na poziomie rozumienia podstawowych pojęć, jak i na poziomie praktycznego zarządzania projektami.

Od czytelnika nie oczekuje się znajomości żadnej z wyżej wymienionych teorii.

# 1 Wstęp

Doświadczenia realizatorów dużych projektów pokazują, że większość projektów przekracza zarówno planowany czas realizacji, jak i planowany budżet. Dla przykładu, w grupie projektów poświęconych budowie systemów informatycznych o łącznej wartości 250 mld USD, 88% projektów przekroczyło planowany czas realizacji lub budżet, średnie przekroczenie planowanych kosztów wyniosło 189%, a średnie przekroczenie czasu realizacji wyniosło 222% (por. [1]).

Teoria ograniczeń Goldratta (theory of constrains, skr. TOC) stanowi przełom w zakresie wiedzy i praktyki o zarządzaniu projektami. Dzięki jej zastosowaniu udaje się nie tylko dotrzymywać terminów realizacji projektów, ale też bardzo poważnie skracać czas ich realizacji (30 do 50 procent), a także sprawiać, że realizacje nie przekraczają planowanych budżetów. Wiele spektakularnych przykładów takich osiągnięć opisano w [5]. Na gruncie polskim przykładem może być zintegrowanie około 100 spółek branży energetycznej w jedną firmę Polska Grupa Energetyczna i wprowadzenie tej firmy na Warszawską Giełdę Papierów Wartościowych. Zadanie to zostało wykonane w rekordowo krótkim czasie.

---

1 

Teoria Goldratta odwołuje się do trzech dość odległych od siebie obszarów wiedzy:

1. logistyki projektów,
2. statystyki
3. psychologii.

W pierwszym z tych trzech zakresów wprowadza istotną innowacyjność polegającą na uwzględnianiu w szeregowaniu zadań nie tylko zależności produktowych (wykonanie całości musi być poprzedzone wykonaniem jej części), ale też zależności wynikających z ograniczoności zasobów narzędziowych, a w tym osobowych. Każdy projekt opisywany jest przed dwa diagramy: jeden, który przypomina sieć PERT i opisuje zależności produktowe (tzw. *precedence view*) i drugi, który przypomina wykres Gantta i opisuje zależności narzędziowe (tzw. resource view).

Przy całej innowacyjności teorii ograniczeń, język jakim posługują się jej autorzy jest bardzo nieprecyzyjny. W rzeczywistości jest to język potoczny uzupełniony rysunkami, co nie pozwala ani na ścisłe zdefiniowanie podstawowych dla tej teorii pojęć, np. pojęcia łańcucha krytycznego, ani też na poddanie ich naukowej analizie.

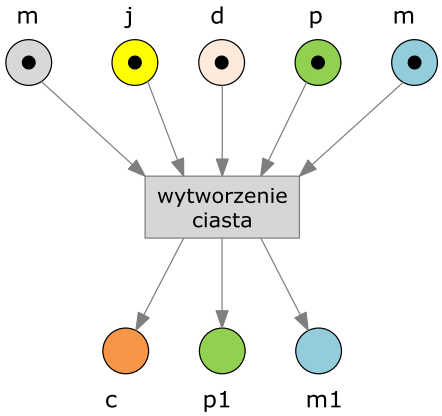
Niniejsza praca stawia sobie za cel zastosowanie teorii sieci Petriego do logistycznych aspektów teorii ograniczeń Goldratta. Dzięki temu udaje się:

1. NA POZIOMIE POJĘCIOWYM:
   1.1. zdefiniować pojęcie sieci projektu,
   1.2. zastąpić dwa diagramy Goldratta jedną siecią,
   1.3. zdefiniować pojęcie łańcucha krytycznego i pokazać, że podawane w literaturze różne wersje definicji tego pojęcia nie są równoważne,
   1.4. uwzględniać niedeterminizm w sieciach projektów,
   1.5. uwzględniać cykle w sieciach projektów,
   1.6. zdefiniować pojęcie blokady w sieci,
   1.7. wykorzystać liczne uogólnienia sieci Petriego
2. NA POZIOMIE NARZĘDZIOWYM:
   2.1. zaproponować systematyczną metodę budowanie sieci projektu,
   2.2. wykorzystać w budowaniu projektu metodę modułowości,
   2.3. wykorzystać istniejące algorytmy wykrywania cykli i blokad,
   2.4. wykorzystać istniejące algorytmy badania wykonalność projektu,
   2.5. wykorzystać liczne uogólnienia sieci Petriego

# 2 Pojęcia podstawowe

Sieci Petriego, których nazwa pochodzi od nazwiska ich twórcy Carla Adama Petri, pojawiły się w matematycznych podstawach informatyki w początku lat1960-tych (por.[7]) jako modele matematyczne programów komputerowych, których elementarne akcje nie muszą być wykonywane sekwencyjnie, tj. jedna po drugiej, i które wykorzystują pewien wspólny zbiór za-

sobów. Dziś sieci Petriego stanowią bogaty dział matematycznych podstaw informatyki opisany w setkach książek i tysiącach prac naukowych.



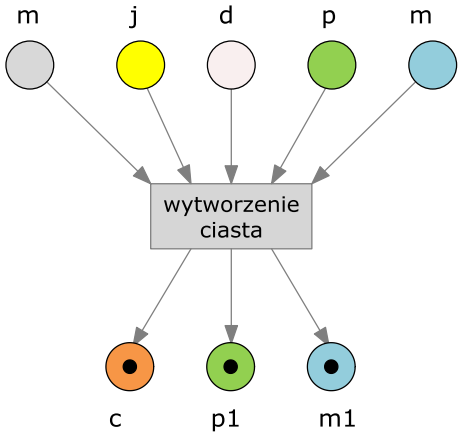

**Rys.2.1 Tranzyt przed odpaleniem**

*Siecią Petriego* nazywamy każdy graf skierowany (krawędzie grafu są strzałkami) o dwóch rodzajach węzłów:

- *miejsca*, które odpowiadają warunkom; oznaczamy je kółkami
- *tranzyty*, które odpowiadają akcjom elementarnym; oznaczamy je prostokątami.

Każda krawędź sieci (strzałka) prowadzi albo od miejsca do tranzytu, albo od tranzytu do miejsca. Każdy tranzyt ma swoje *wejścia* — miejsca, od których strzałki prowadzą do niego, oraz swoje *wyjścia* — miejsca, do których prowadzą strzałki od niego. Zwykle jest tak, że wejścia jednego tranzytu są wyjściami dla innego lub dla innych tranzytów. W ten właśnie sposób miejsca i tranzyty łączą się w sieć.

W każdym z miejsc może stać kropka zwana *znacznikiem*, co oznacza, że warunek związany z danym miejscem jest spełniony. Na Rys.1.1 przedstawiono sieć o jednym tranzycie, którym jest operacja wytworzenia surowego ciasta drożdżowego. Kropki stojące w wejściach tego tranzytu oznaczają dostępność zasobów niezbędnych do wytworzenia ciasta: mąka (m), jajka (j), drożdże (d), pracownik (p) i maszyna (m).




**Rys. 2.2 Tranzyt po odpaleniu**

Aby operacja reprezentowana przez tranzyt mogła zostać wykonana — w języku sieci Petriego mówimy: aby tranzyt mógł zostać *odpalony* — muszą być spełnione dwa warunki:

1. we wszystkich wejściach tranzytu muszą stać znaczniki — wszystkie niezbędne zasoby są dostępne,
2. we wszystkich wyjściach tranzytu znaczników musi nie być — tranzyty następujące po danym tranzycie są gotowe na przyjęcie zasobów.

Po wykonaniu operacji, a więc po odpaleniu tranzytu, znaczniki stojące w wejściach znikają — zasoby zostają wykorzystane — i pojawiają się znaczniki w wyjściach (Rys.1.2) — nowe zasoby zostają wytworzone (ciasto) lub istniejące zostają zwolnione (pracownik i maszyna).

Z punktu widzenia teorii projektów będziemy rozróżniali dwa rodzaje zasobów:

*zasoby produktowe* — zużywane przez tranzyt do wytworzenia nowego produktu (w naszym przypadku mąka, jajka i drożdże); zasoby produktowe po ich wykorzystaniu znikają, a na ich miejsce pojawiają się nowe (w naszym przypadku ciasto),

*zasoby narzędziowe* — wykorzystywane przez tranzyt, ale nie zużywane, a więc zwracane po odpaleniu tranzytu (w naszym przypadku pracownik i maszyna).

Jeżeli z jakiegoś miejsca wychodzi wiele strzałek do różnych tranzytów, to obecność znacznika w tym miejscu pozwala na wykorzystanie go przez tylko jeden z tranzytów. Odpowiada to bowiem sytuacji, w której kilka operacji potrzebuje tego samego zasobu, ale zasób jest tylko jeden, a więc tylko jedna operacja może zostać wykonana[2]. Najczęściej tego typu rozgałęzienie dotyczy miejsc, udostępniających zasoby narzędziowe. Przykład sieci z takimi tranzytami pokazano na Rys.3.3.

W dalszym ciągu pojedynczy tranzyt wraz z jego wejściami i wyjściami będziemy nazywali *atomem sieci*.

# 3 Studium przypadku — tynkowanie i malowania

Rozważmy teraz przykład projektu otynkowania i pomalowania dwóch pokoi. Do realizacji tego projektu przydzielono jednego tynkarza i jednego malarza, przyjęto też dość oczywiste założenie, że pokój musi być wpierw otynkowany, aby mógł zostać pomalowany. Przed narysowaniem sieci Petriego opiszmy projekt w postaci tabelki Tab. 3.1

| tranzyt | opis | zasoby na wejściu | zasoby na wyjściu |
|---|---|---|---|
| T1 | tynkowanie pokoju nr 1 | nieotynkowany pokój nr 1 (n1)<br>tynkarz (t) | otynkowany pokój nr 1 (o1)<br>tynkarz (t) |
| T2 | tynkowanie pokoju nr 2 | nieotynkowany pokój nr 2 (n2)<br>tynkarz (t) | otynkowany pokój nr 2 (o2)<br>tynkarz (t) |
| M1 | malowanie pokoju nr 1 | otynkowany pokój nr 1 (o1)<br>malarz (m) | pomalowany pokój nr 1 (p1)<br>malarz (m) |
| M2 | malowanie pokoju nr 2 | otynkowany pokój nr 2 (o2)<br>malarz (m) | pomalowany pokój nr 2 (p2)<br>malarz (m) |

**Tab.3.1 Tynkowanie i malowanie pomieszczeń**

Każda z operacji tynkowania (T1 i T2) wymaga do wykonania dwóch zasobów:

1. nieotynkowany pokój (n1 lub n2),
2. tynkarz (t)

a po jej wykonaniu pojawiają się zasoby:

1. otynkowany pokój (o1 lub o2)
2. tynkarz (t)

Podobnie jest z operacją malowania. Do jej wykonania potrzebne są dwa zasoby:

1. otynkowany pokój (o1 lub o2)
2. malarz (m)

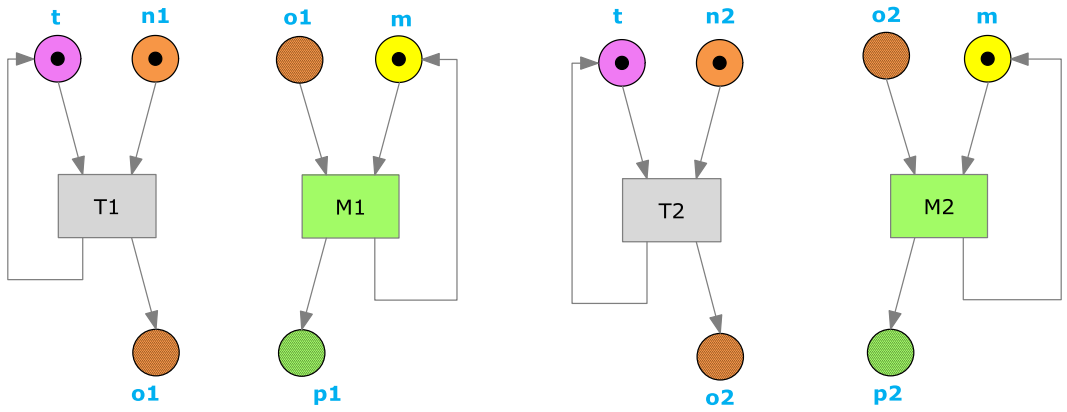
a po jej wykonaniu pojawiają się zasoby:

1. pomalowany pokój (p1 lub p2)

[2] W teorii sieci Petriego rozważa się też sieci dozwalające na pojawienie się więcej niż jednego znacznika w jednym miejscu, a także wprowadza się znaczniki „kolorowane” oraz inne konstrukcje pozwalające na opisywanie złożonych warunków koniecznych do odpalenia tranzytu. W tej pracy zajmiemy się jednak sieciami, które dopuszczają co najwyżej jeden znacznik w każdy miejscu.

2. malarz (m)

Każdy wiersz Tab.3.1 to jeden atom w sensie sieci Petriego. Te atomy zostały przedstawione na Rys. 3.1.



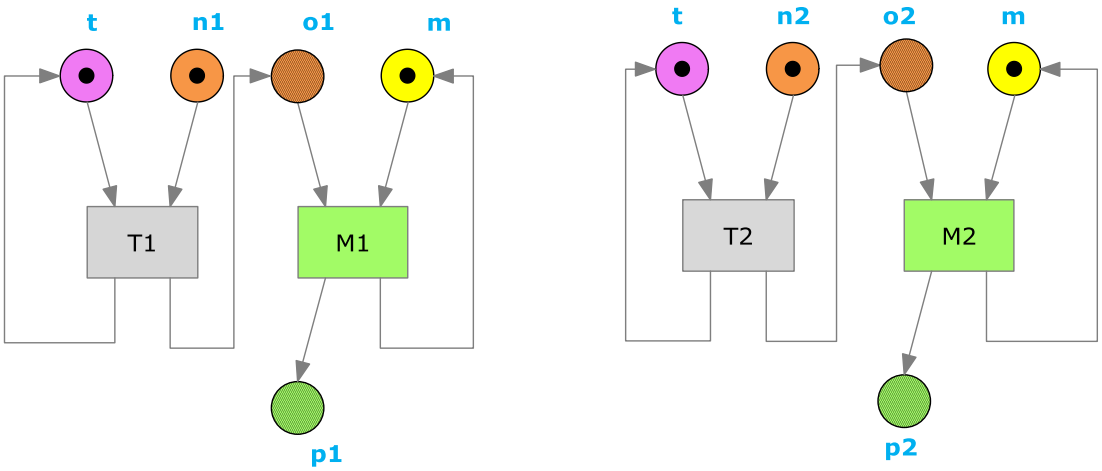

**Rys. 3.1 Cztery atomy**

Każdy z atomów przyjmuje na wejściu jeden zasób produktowy (pokój w odpowiednim stanie wykończenia) i jeden zasób narzędziowy (tynkarz lub malarz). Po wykonaniu zadania każdy atom zwraca odpowiednio przetworzony zasób produktowy oraz uwalnia narzędzie, które używał. Ta sieć realizuje trzy następujące założenia naszego projektu:

1. mamy do wykończenia dwa pokoje,
2. aby pokój mógł być pomalowany, musi być wpierw otynkowany,
3. do tynkowania jest potrzebny i wystarczy jeden tynkarz, a do malowania — jeden malarz.

Zakończenie projektu następuje, gdy w miejscach p1 i p2 pojawią się znaczniki.

Przedstawiona na Rys.3.1 sieć Petriego składająca się z czterech niepołączonych ze sobą atomów nie realizuje naszego projektu, gdyż tranzyty M1 i M2 nigdy nie odpalą, bo w miejscach o1 i odpowiednio o2 brak znaczników. Ponadto, w tej sieci występuje dwóch tynkarzy i dwóch malarzy, a my mamy tylko po jednym.



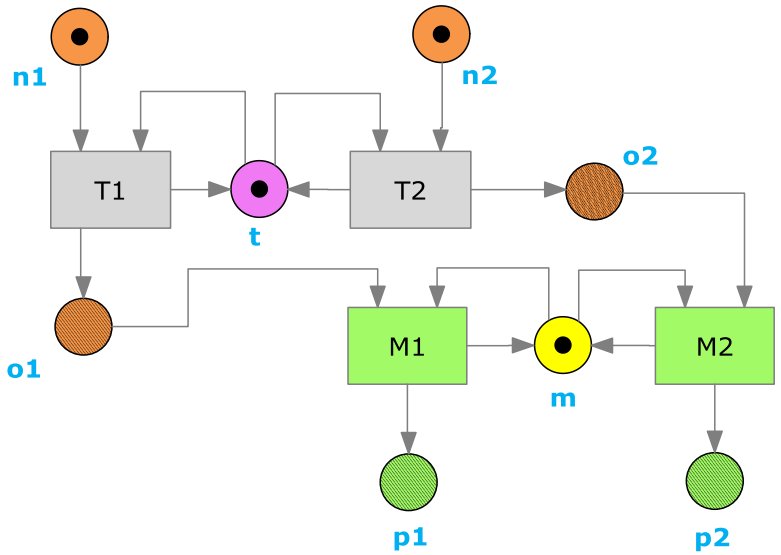

**Rys. 3.2 Atomy po sklejeniu — dwóch tynkarzy i dwóch malarzy**

Aby po tynkowaniu mogło nastąpić malowanie, tranzyt T1 musi po zakończeniu swojej akcji przekazać otynkowany pokój tranzytowi malowania M1 i analogicznie w przypadku T2. Tę cechę naszego projektu zapisujemy w sieci łącząc jednoimienne miejsca o nazwie o1 w pierwszych dwóch atomach i jednoimienne miejsca o2 w drugich dwóch atomach. W ten sposób otrzymujemy sieć Petriego pokazaną na Rys. 3.2. Ta sieć realizuje wymienione wyżej trzy założenia projektowe oraz dodatkowo założenie czwarte:

4. każdy pokój po otynkowaniu zostanie pomalowany

Zauważmy jednak, że nasza sieć nie realizuje założenia projektowego, które mówi, że mamy do dyspozycji tylko jednego tynkarza i jednego malarza. W naszej sieci znaczniki stoją w dwóch miejscach odpowiadających dostępności tynkarza i w dwóch odpowiadających dostępności malarza. Oznacza to, że kierownik projektu ma do dyspozycji dwóch tynkarzy i dwóch malarzy.

Aby spełnić piąte założenie projektowe, tranzyty muszą przekazywać sobie nie tylko produkty (otynkowany pokój), ale też i narzędzia (tynkarza i malarza). Aby zrealizować to założenie projektowe w naszej sieci, trzeba skleić ze sobą oba miejsca oznaczone przez t i oba miejsca oznaczone przez m. Po tej operacji otrzymujemy sieć pokazaną na Rys.3.3.




**Rys.3.3 Jeden tynkarz i jeden malarz — niedeterministycznie**

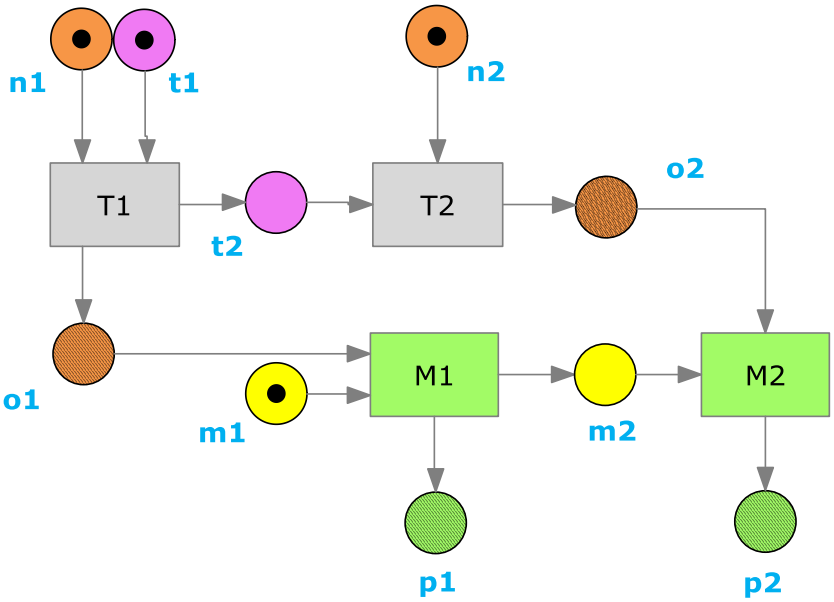
Rozmieszczenie znaczników określa stan początkowy projektu. W tym stanie malarz nie może być użyty, gdyż w miejscach o1 i o2 brak znaczników. Jedyne co możemy zrobić, to otynkować jeden z pokoi. Nasza sieć nie przesądza, który pokój otynkujemy w pierwszej kolejności, i nie powinna, bo założenia projektowe o tym nie mówią. Z miejsca t wychodzą dwie strzałki, co oznacza, że ta decyzja zostanie podjęta na etapie wykonania, a nie na etapie planowania projektu. O sieciach z miejscami, z których wychodzi więcej niż jedna strzałka, mówimy, że są *niedeterministyczne*. O sieciach, które nie są niedeterministyczne, mówimy, że są *deterministyczne*. Np. każda z podsieci na Rys.3.2 jest deterministyczna, a sieć na Rys.3.3 jest niedeterministyczna.

Popatrzmy teraz, jak działa nasza sieć. Przypuśćmy w tym celu, że w pierwszej kolejności otynkowano pokój pierwszy. Po wykonaniu tej czynności zniknie znacznik w n1 i pojawi się w o1, a znacznik odpowiadający tynkarzowi powróci na swoje miejsce do t. Teraz do opale-

nia są gotowe dwa tranzyty: m1 i t2. Można to zrobić w dowolnej kolejności lub równocześnie. Po wykonaniu obu tych czynności pojawią się znaczniki w m i o2 oraz w p1. To pozwoli na odpalenie m2 i spowoduje pojawienie się znacznika w p2. Na tym realizacja projektu zostaje zakończona.

Analogicznie wyglądałaby realizacja projektu, gdyby w pierwszej kolejności otynkowano pokój nr 2.

Przy pomocy sieci Petriego można też opisać projekt, w którym już na etapie planowania zadecydowano, który z pokoi ma być otynkowany w pierwszej kolejności. Gdyby był to pokój nr 1, to sieć projektu wyglądałaby jak na Rys.3.4. Aby wprowadzić do projektu determinizm, trzeba było rozdzielić miejsce t (dostępny tynkarza), na dwa miejsca t1 (tynkarz dostępny do tynkowania pokoju nr 1) i t2 (tynkarz dostępny do tynkowania pokoju nr 2). Podobnie postąpiono z miejscami malarza.




**Rys.3.4 Jeden tynkarz i jeden malarz — deterministycznie**

Rys.3.2, Rys.3.3 i Rys.3.4 przedstawiają trzy przykłady sieci Petriego. W każdym przypadku jest to jedna sieć, choć pierwsza z nich nie jest spójna. Każda z nich odpowiada innej realizacji projektu — innej w sensie dostępnych zasobów narzędziowych lub też czasu podjęcia decyzji o kolejności odpalenia tranzytów.

Rozmieszczenie znaczników w miejscach sieci nazywamy *znakowaniem sieci*. Jeżeli sieć opisuje projekt w sensie Goldratta, to powinno być określone znakowanie, od którego rozpoczyna się wykonanie projektu. To znakowanie nazywamy *znakowaniem początkowym*. Znakowania pokazane na trzech ww. rysunkach są właśnie takimi znakowaniami. Znakowanie początkowe określa zasoby, jakie są dostępne w projekcie w chwili jego rozpoczęcia.

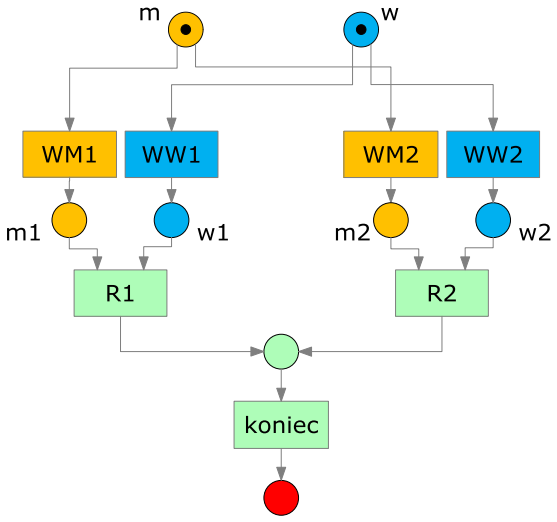
Oczywiście każdy projekt powinien mieć też określone warunki zakończenia projektu. Te warunki, to *znakowanie końcowe*. W naszym trzech przypadkach jest to znakowanie mające znaczniki jedynie w p1 i p2.

# 4 Osiągalność i blokady

Zauważmy, że nie każde znakowanie początkowe pozwala na osiągnięcie znakowania końcowego. Na przykład, gdyby na Rys.3.4 zabrakło znacznika w miejscu m1, tj. gdyby kierownik projektu nie zapewnił dostępności malarza, to projekt nie mógłby być zrealizowany. Zakończyłby się na otynkowaniu obu pomieszczeń (znaczniki w o1 i o2) i nigdy nie osiągnął stanu końcowego, tj. nigdy nie pojawiłby się znacznik w miejscach p1 i p2. W naszym przypadku nieosiągalność stanu końcowego łatwo zauważyć. W przypadkach złożonych sieci sprawa jednak nie musi być wcale prosta. Zagadnienie *osiągalności* jest przedmiotem licznych prac w teorii sieci Petriego. Są też algorytmy sprawdzające, czy z zadanego znakowania da się osiągnąć inne zadane znakowanie.

W opisanym wyżej przykładzie osiągalność nie jest możliwa ze względu na brak zasobów. Może być jednak też tak, że zasobów jest pod dostatkiem, a jednak istnieje zagrożenie, że projekt nie będzie się mógł zakończyć. To zjawisko omówimy na prostym przykładzie projektu wbijania kołków w ścianę.

Załóżmy, że do wbicia kołka w ścianę potrzebne są jednocześnie dwa narzędzia: młotek i wiertarka, a wśród zasobów projektu mamy dwóch robotników, jeden młotek i jedną wiertarkę. Rozważmy teraz sieć przedstawioną na Rys.4.1




**Rys. 4.1 Wbijanie kołków**

WMi oznacza czynność wydania młotka robotnikowi Ri, a WWi — wydania wiertarki robotnikowi Ri. Pojawienie się znacznika w miejscu brązowym oznacza dostępność młotka, a w miejscu błękitnym — dostępność wiertarki. Miejsca m i w symbolizują magazyny narzędziowe. Jeżeli robotnik pierwszy pobierze młotek, a robotnik drugi wiertarkę, to znaczniki pojawią się w miejscach m1 i w2, co spowoduje, że żaden z tranzytów R1 i R2 nie będzie mógł być odpalony. Takie właśnie zjawisko nazwiemy *blokadą*.

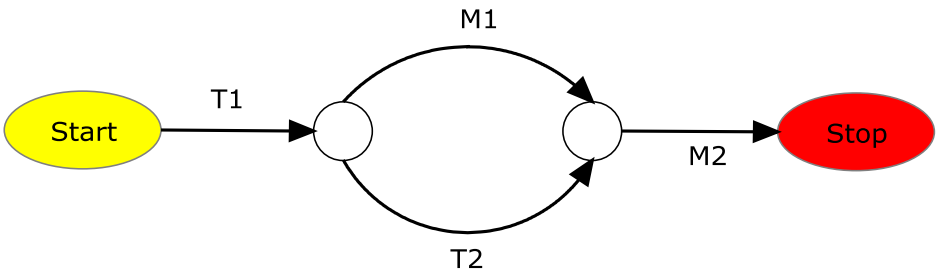
Podobnie jak w poprzednim przypadku, również i ten przykład został pokazany w bardzo prostej sieci, gdzie jest widoczny na pierwszy rzut oka. W rzeczywistości, w rozległej i złożonej sieci, może być on trudny do zauważenia, może też występować w sieciach, które nie są, tak jak nasza, niedeterministyczne. Istnieją jednakże algorytmy, które pozwalają na wykrywanie blokad.

# 5 Tynkowanie i malowanie w innych metodologiach

Przyjrzyjmy się teraz, jak moglibyśmy opisać nasz projekt posługując się każdym z trzech następujących formalizmów:

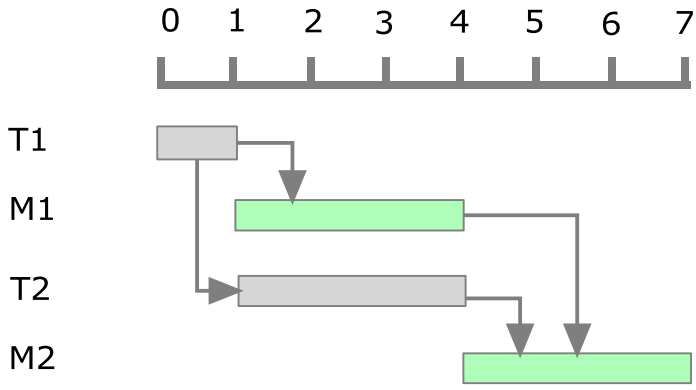
1. sieci CPM (ścieżka krytyczna) lub PERT (ścieżka krytyczna z wariancją czasową),
2. diagramy Gantta,
3. teoria ograniczeń i łańcucha krytycznego Goldratta.

Metody CPM (Critical Path Metod — metoda ścieżki krytycznej) i PERT (Program Evaluation and Review Technique — technika oceny i przeglądu) prowadzą do tej samej sieci następstwa czynności, gdyż różnią się jedynie w zakresie analizy czasu wykonania. Na Rys.5.1 pokazano sieć PERT od razu w wersji uwzględniającej ograniczoność zasobów narzędziowych. Pokazano też deterministyczną wersję projektu, gdyż uwzględnienie niedeterminizmu w sieci PERT, a także w każdym z pozostałych dwóch formalizmów, nie jest możliwe.



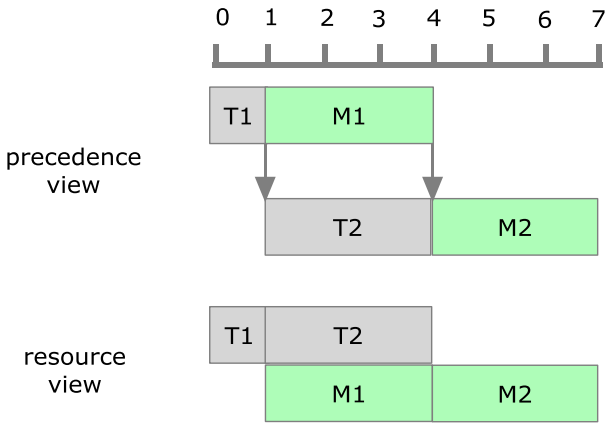

**Rys.5.1 Tynkowanie i malowanie opisane siecią PERT**

Wykres Gantta dla naszego projektu pokazany jest na Rys. 5.2). W tym formalizmie pojawia się skala czasowe, obecna również w schematach Galdratta.




**Rys. 5.2 Tynkowanie i malowanie opisane diagramem Gantta**

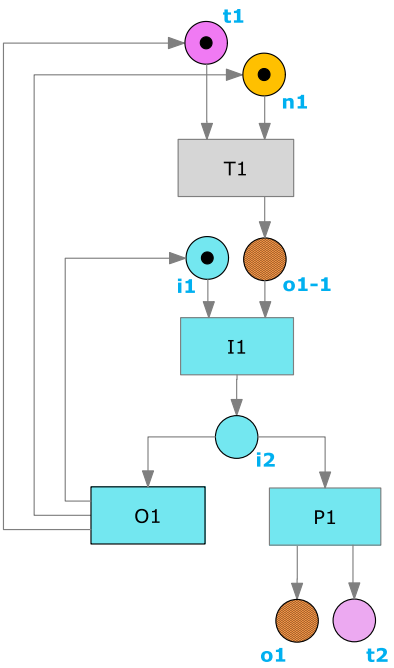
Na Rys.**Błąd! Nie można odnaleźć źródła odwołania.**.3 pokazano wykres projektu w konwencji Goldratta używanej w książce Newbolda [6]. W Roz.11 tej książki pokazane są dwie uzupełniające się techniki graficznego przedstawienia diagramu projektu. Pierwsza z nich pozwala na opisanie jedynie relacji następstwa produktowego pomiędzy działaniami (precedence view) i nie bierze pod uwagę dostępności narzędziowych. Druga z nich (resourceview) opisuje jedynie następstwo narzędziowe i nie bierze pod uwagę następstwa produktowego.




**Rys. 5.3 Tynkowanie i malowanie opisane diagramem Goldratta**

Jak widzimy jest to deterministyczny wariant realizacji projektu odpowiadający sieci Petriego na Rys.3.4. Podobnie jak w obu poprzednich przypadkach, wybór wariantu (T1 przed T2) jest wymuszony przez użyty formalizm już na poziomie definiowania projektu. Może to prowadzić do nieoptymalnych rozwiązań dotyczących czasu wykonania projektu, co zostało pokazane na przykładzie w Roz.10. Oczywiście w przypadku tak prostego projektu jak nasz optymalizację struktury można przeprowadzić w każdym z formalizmów bez większych trudności. Jednakże w przypadku rzeczywistych projektów sprawa może nie być taka prosta. Ponadto — i może jest to jeszcze ważniejsze — możliwość przedstawienia projektu w wersji niedeterministycznej pozwala na odsunięcie w czasie decyzji co do kolejności odpalania tranzytów. W chwili układania planu projektu możemy jeszcze nie znać przyszłych czasów wykonywania tranzytów. Jest to typowa sytuacja przy wykonywaniu remontów złożonych obiektów. Np. przy remoncie budynku dopiero po odsłonięciu fundamentów można oszacować czas wykonania izolacji przeciw wilgoci.

# 6 Cykle




**Rys.6.1 Inspektor nadzoru**

Rozważane w Roz.3 sieci projektów nie zawierały cykli. W rzeczywistości jednak cykle w projektach się zdarzają. Typowym przykładem może być dokonywanie odbioru otynkowanego pokoju przez inspektora nadzoru (Rys.6.1) w projekcie tynkowania i malowania jednego pokoju.

Na początku dostępny jest tynkarz (t1), nieotynkowany pokój (n1) oraz inspektor (i1).

Po otynkowaniu pokoju nr 1 inspektor dokonuje jego inspekcji (I1), po której może albo pracę przyjąć i sporządzić protokół odbioru (P1), albo też (niedeterminizm) odrzucić pracę (O1) i zażądać dokonania poprawek, po których pokój należy ponownie zgłosić do odbioru.

Odesłanie pracy do poprawki uwalnia tynkarza (znacznik powraca do t1), pokój do poprawki (znacznik powraca do n1) oraz inspektora (znacznik powraca do i1). W przypadku przyjęcia pracy znaczniki pojawiają się w o1 (pokój otynkowany) i t2 (tynkarz dostępny do dalszych prac).

W miejscu i2 inspektor podejmuje decyzję, czy praca tynkarza ma być przyjęta, czy też odrzucona. W aktualnym modelu naszego projektu nie określono, jakim kryteriom oceny podlega ta decyzja. Teoria sieci Petriego dopuszcza jednak sytuację, gdzie miejscom rozgałęzień przypisane są warunki jednoznacznie określające, w którą stronę należy przesunąć znacznik. Jest to sytuacja analogiczna do instrukcji warunkowych typu If_Then_Else w językach programowania.

Zauważmy teraz, że ponieważ nasza sieć zawiera cykl, to teoretycznie rzecz biorą jej wykonanie może obejmować dowolną liczbę powtórzeń cyklu, a więc może trwać dowolnie długo. Jest bowiem hipotetycznie możliwe, że inspektor będzie bez końca odsyłał prace do wykonania poprawek. Ten aspekt jest oczywiście bardzo ważny z punktu widzenia oceny czasu wykonania projektu.

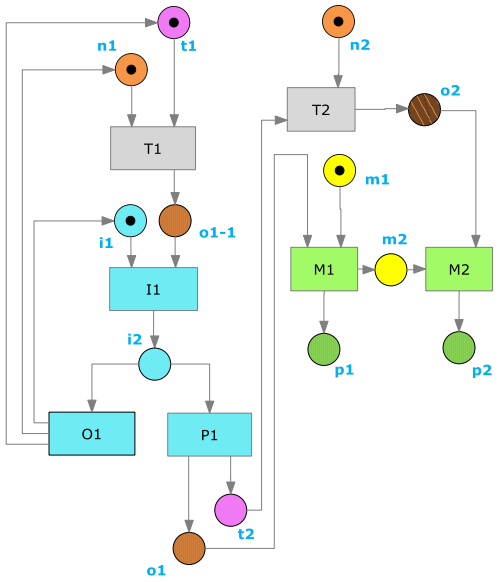
# 7 Zasada modułowości

W czasach, gdy programy komputerowe pisało się ołówkiem na papierze (lata 1960-te) istniała dobra praktyka mówiąca, że każdy program powinien mieścić się na jednej kartce formatu A4, gdyż tylko wtedy programista będzie w stanie „intelektualnie zapanować nad programem, a więc ocenić, czy program wykonuje oczekiwane zadanie. Nie oznaczało to jednak wcale, że programy miały być aż tak krótkie. Już wtedy pisano bowiem programy zawierające setki tysięcy linii kodu. Dla realizacji tej praktyki każdy program dzielono na moduły, które dzielono na podmoduły, które znów dzielono na poppodmoduły i postępowano w ten sposób tak długo, aż moduły nie staną się atomowymi instrukcjami języka programowania. We współczesnych językach programowania praktykę tę stosuje się nadal, a służą po temu narzędzia znane jako procedury i obiekty.

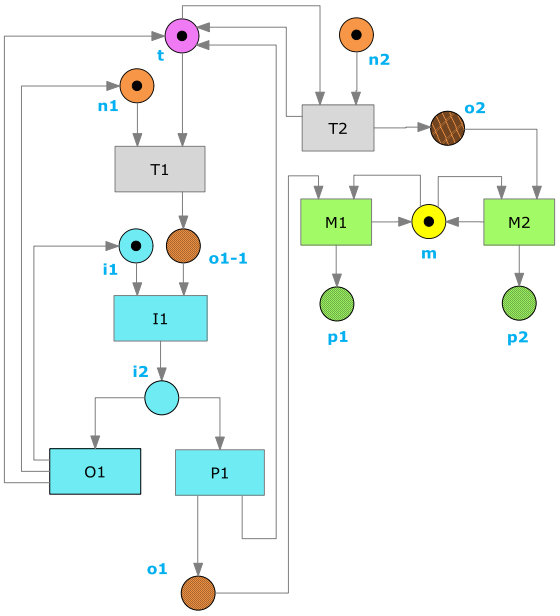
Opisaną metodę, która nazywa się *zasadą modułowości,* daje się z powodzeniem zastosować do sieci Petriego. Modułami w sieci są atomy, a więc pojedyncze tranzyty wraz z ich wejściami i wyjściami, np. takie jakie przedstawiono na Rys. 1.1 i Rys.1.2. Każdy taki atom występujący w sieci możemy zastąpić siecią o tej samej liczbie i rodzaju wejść i wyjść, co dany atom, ale o większej liczbie tranzytów i miejsc wewnętrznych.

Przykładem takiej operacji może być zastąpienie w sieci na Rys.3.4 atomu (n1, t1, T1, o1, t2) siecią przedstawioną na Rys.6.1. Miejsca n1, t1, o1 i t2 tej sieci zastępują miejsca na Rys.3.4 o tych samych nazwach. Otrzymujemy wtedy sieć przedstawioną na Rys.7.1. Postępując analogicznie możemy wstawić sieć inspektora nadzoru do niedeterministycznej sieci naszego projektu z Rys.3.3. W tym drugim przypadku prowadzi to do sytuacji, w której możemy wybierać pomiędzy poprawieniem prac w pokoju nr 1 i otynkowaniem pokoju nr 2. Ten wybór nie oznacza jednak, że możemy w ogóle zrezygnować z poprawki. Gdy po wykonaniu T1 znacznik tynkarza wróci do miejsca t, tranzyt T1 będzie gotów do odpalenia.

Oczywiście o moduł inspektora można wzbogacić każdy z czterech tranzytów naszej sieci.




**Rys.7.1 Inspektor deterministycznie**




**Rys.7.2 Inspektor niedeterministycznie**

Należy też podkreślić, że zasada modułowości wcale nie oznacza, że mamy komplikować naszą pierwotną sieć zastępując jej moduły kolejnymi sieciami, tak jak w naszych przykładach. Wprost przeciwnie. Zasadę modułowości stosujemy na drodze od ogółu do szczegółu. Postępujemy w tym przypadku tak samo jak przy pisaniu programów. Najpierw budujemy sieć projektu używając do tego celu tylu modułów, aby sieć mieściła się na jednej kartce. Następnie moduły rozpisujemy na sieci, z których każda znów mieści się na jednej kartce i postępujemy tak długo, aż tranzyty odpowiadają akcjom, które w naszym projekcie uznajemy za elementarne. Np. projekt odpowiadający sieci na Rys.7.1 byłby opisany siecią na Rys.3.4 i siecią na Rys.5.1.

# 8 Czas wykonania projektu — co to jest?

Jednym z podstawowych obowiązków kierownika projektu jest zakończenie projektu w planowanym czasie. Temu też zagadnieniu poświęcona jest znaczna część teorii ograniczeń Goldratta. Zastanówmy się więc, jak policzyć czas realizacji projektu opisanego siecią Petriego i jak wyrazić na tym gruncie techniki optymalizacji czasu realizacji. Zaczniemy od zdefiniowania pojęcia *czas wykonania projektu*. Wbrew pozorom nie jest to tak oczywiste, jak by się mogło wydawać.

Najprostsza definicja tego pojęcia wydaje się być następująca:

> *Czas wykonania projektu to czas przejścia sieci projektu do znakowania początkowego do znakowania końcowego.*

Niestety, ta definicja może być zastosowania jedynie w przypadku, gdy projekt spełnia następujące dwa warunki:

A. przejście od znakowania początkowego do znakowania końcowego jest możliwe; w Roz.3 przekonaliśmy się, że ta teza nie musi być prawdziwa,

B. wszystkie możliwe (alternatywne) drogi realizacji projektu trwają tyle samo czasu; w Roz.10 przekonamy się, że i ta teza nie musi być prawdziwa, nawet jeżeli projekt nie zawiera cykli.

Dodatkowo, przy definiowaniu czasu wykonania projektu należy wziąć pod uwagę różne zasady przypisania pojedynczemu tranzytowi czasu wykonania.

Tranzyt nazwiemy *otwartym*, jeżeli nie określono dla niego terminów rozpoczęcia i zakończenia. Tranzyt otwarty jest gotów do odpalenia, gdy tylko zostaną wypełnione jego miejsca wejściowe i opróżnione jego miejsca wyjściowe, a zakończy on działanie i zwolni zasoby (wypełni wyjścia), natychmiast po zakończeniu przypisanej mu akcji. Typowym tranzytem otwartym może być malowane pomieszczenia w przykładzie z Roz.3.

Tranzyt nazwiemy *zamkniętym na wejściu*, gdy przypisano mu termin rozpoczęcia akcji. Jest to termin, przed którym tranzyt nie może odpalić, nawet jeżeli wszystkie jego wejścia zostaną wypełnione. Przykładem takiego tranzytu może być lanie betonu pod fundamenty budynku, które nie może się rozpocząć przed pewną datą gwarantującą, że nie wystąpią już spadki temperatury.

Tranzyt nazwiemy *zamkniętym na wyjściu*, gdy przypisano mu termin zakończenia akcji nie zależny od tego, czy składające się na akcję czynności zostały zakończone wcześniej, czy też zakończone nie zostały. Np. ładowanie samochodów na prom kończy się o określonej godzinie, zarówno wtedy, gdy wszystkie samochody zostały wcześniej załadowane, jak i wtedy, gdy na nadbrzeżu pozostały niezaładowanie samochody.

Tranzyt nazwiemy *obustronnie zamkniętym*, jeżeli jest zamknięty zarówno na wejściu jak i na wyjściu. Ładowanie samochodów na prom jest właśnie przykładem takiego tranzytu.

Sieć projektu nazwiemy *otwartą*, gdy wszystkie jej tranzyty są otwarte.

Niezależnie od tego, jakie czasy i terminy wykonania przypiszemy tranzytom, czas wykonania projektu zależy również o tego, jaką przyjmiemy regułę odpalania tranzytów. Przypomnijmy (Roz.1), że tranzyt może być odpalony, gdy wszystkie jego wejścia są wypełnione, a wszystkie wyjścia puste. Ta reguła nie przesądza jednak, kiedy zostanie odpalony tranzyt, który jest gotowy do odpalenia. Np. może być tak, że ekipa odpowiedzialna za przygotowanie szalunku pod fundamenty zakończyła pracę w piątek wieczorem, ale lanie fundamentów rozpocznie się dopiero w poniedziałek rano. Jeżeli więc chcemy mówić o czasie wykonania projektu, to musimy określić czasową regułę odpalania tranzytów. Jedną z takich reguł jest *reguła niezwłoczności*, która stanowi, że tranzyty są odpalane niezwłocznie po tym, jak staje się to możliwe. Zastosowanie tej reguły oznacza w szczególności, że wszystkie tranzyty, które mogą być realizowane równolegle, będą realizowane tak właśnie. W dalszych rozważaniach nad czasem wykonania projektu będziemy zakładali, że nasze sieci są otwarte, a odpalanie tranzytów rządzi się regułą niezwłoczności. Dla dalszych rozważań nad czasem wykonania projektu konieczne jest podanie definicji sieci projektu.

# 9 Sieć projektu — próba definicji

Wydaje się, że dysponujemy już aparatem pojęciowym pozwalającym na zdefiniowanie pojęcia sieci projektu. W dalszym więc ciągu *siecią projektu* będziemy nazywali taką sieć Petriego, dla której:

1. został określony zbiór produktów i zbiór narzędzi,
2. każdemu miejscu został przypisany produkt lub narzędzie,

3. zostało określone znakowanie początkowe,
4. zostało określone znakowanie końcowe,
5. tranzytom zostały przypisane czasy wykonania i ewentualnie też terminy rozpoczęcia i zakończenia.

W definicji sieci projektu nie zakładamy, jaką regułą rządzi się odpalanie tranzytów. Tę regułę trzeba zawsze określić niezależnie.

Wprowadzimy teraz kilka pojęć pomocniczych, którymi będziemy się posługiwali w dalszej części pracy. Wszystkie one odnoszą się do pojęcia sieci projektu.

Miejsce nazwiemy *produktowym* (*narzędziowym*), gdy przypisano mu produkt (narzędzie).

Miejsce nazwiemy *początkowym*, jeżeli nie jest ono miejscem wyjściowym żadnego tranzytu. Np. na Rys.3.4 miejscami początkowymi sieci są n1, n2, t1 i m1

Powiemy, że tranzyt T1 *poprzedza bezpośrednio* tranzyt T2, lub też że T2 *następuje bezpośrednio* po T1, co oznaczmy przez T1 → T2, gdy istnieje takie miejsce m, które jest wyjściem dla T1 i wejściem dla T2.

Dwa tranzyty nazwiemy *sąsiadującymi*, jeżeli jeden z nich poprzedza bezpośrednio drugi.

*Łańcuchem skierowanym*[3] nazwiemy taki ciąg tranzytów T1,…,Tn, że Ti poprzedza bezpośrednio T(i+1) dla i = 1,…,n-1.

Tranzyt T1 nazwiemy *wcześniejszym* od T2, jeżeli istnieje łańcuch skierowany prowadzący od T1 do T2.

# 10 Studium przypadku — czas wykonania projektu

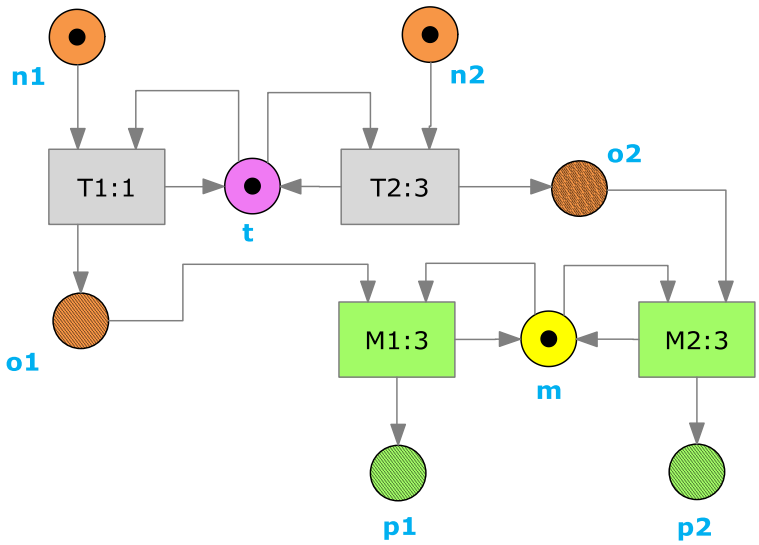
Każdej sieci Petriego, a więc też i każdej sieci projektu, można przypisać zbiór przebiegów[4], z których każdy odpowiada teoriografowemu łańcuchowi tranzytów odpalanych na drodze od znakowania początkowego do znakowania końcowego. W naszym przykładzie niedeterministycznej wersji tynkowania i malowania (Rys. 3.3) będą to dwa przebiegi:

1. T1, (M1 : T2), M2
2. T2, (M2 : T1), M1

gdzie (X : Y) oznacza równoległe wykonanie tranzytów X i Y. Zauważmy, że każdy z tych przebiegów odpowiada jednej z deterministycznych wersji projektu. Zauważmy też, że zrównoleglenie wykonań tranzytów wynika z przyjętej reguły niezwłoczności (Roz.8).

[3] To pojęcie łańcucha jest stosowane w teorii grafów i jest ono różne od pojęcia łańcucha krytycznego używanego w teorii ograniczeń Goltrada.

[4] Na gruncie teorii sieci Petriego wprowadzone przez nas nieformalnie pojęcie przebiegu odpowiada wprowadzonemu przez Antoniego Mazurkiewicza pojęciu *śladu*. Ze względu na matematyczną złożoność tego pojęcia, nie będziemy posługiwać się nim w sposób formalny. Zastąpimy je pewną notacją odwołującą się do intuicji.




**Rys.10.1 Sieć z czasami wykonania tranzytów (czasy po dwukropku)**

Oznaczmy teraz czasy wykonania każdego z tranzytów odpowiednio przez Cz(T1), Cz(M1), Cz(T2) i Cz(M2). Przy tych oznaczeniach — i wobec założenia, że wszystkie tranzyty są otwarte (Roz.8) — czasy realizacji każdego z tych przebiegów wyniosą odpowiednio:

Cz(1) = Cz(T1) + max(Cz(M1),Cz(T2)) + Cz(M2)

Cz(2) = Cz(T2) + max(Cz(M2),Cz(T1)) + Cz(M1)

Okazuje się teraz, że mimo iż w obu przypadkach realizujemy te same zbiory tranzytów (właśnie zbiory, a nie ciągi!), to przy odpowiednio dobranych czasach wykonania tranzytów, czasy wykonań Cz(1) i Cz(2) mogą się różnić. Na przykład, jeżeli przyjmiemy:

Cz(T1) = 1, Cz(T2) = Cz(M1) = Cz(M2) = 3

to

Cz(1) = 1 + max(3,3) + 3 = 7

Cz(2) = 3 + max(3,1) + 3 = 9.

W drugim przypadku zrównoleglenie długo trwającego M2 z krótko trwającym T1 powoduje, że pomieszczenie nr 1 czeka 2 jednostki czasu na malarza, który maluje pomieszczenie nr 2.

Jak można się było spodziewać, niedeterminizm sieci projektu może prowadzić do więcej niż jednego czasu jego wykonania. Jeżeli liczba możliwych wykonań jest skończona, to wśród nich jest jakiś czas minimalny i jakiś maksymalny. Jeżeli jednak sieć zawiera cykl, to zbiór wszystkich wykonań może być nieskończony, nieskończony jest więc też zbiór czasów wykonań. W takim przypadku możemy mówić jedynie o minimalnym czasie wykonania. Czas maksymalny, a więc taki, że każdy inny czas jest od niego krótszy, nie istnieje.

# 11 Budowanie sieci projektu

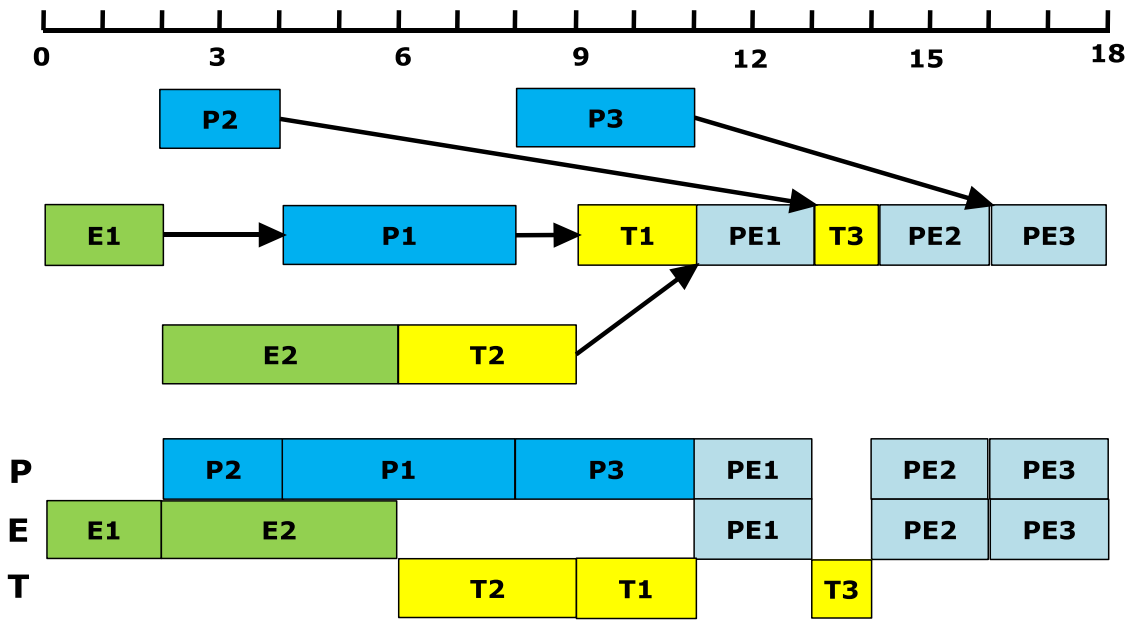
Omówione powyżej studia przypadków pozwalają na zaproponowanie pewnego algorytmu postępowania przy budowaniu sieci Petriego odpowiadającego projektowi:

1. Budowanie sieci rozpoczynamy od stworzenia tabelki na wzór Tab.3.1 opisującej atomy przyszłej sieci, a więc tranzyty wraz z wejściami i wyjściami. W tę tabelkę można też wpisać czasy wykonania tranzytów.
2. Dla każdego wiersza tabelki projektu tworzymy jednoatomową sieć Petriego.
3. Atomy sieci łączmy utożsamiając jednoimienne produktowe wyjścia z wejściami. W ten sposób otrzymujemy sieć, która opisuje produktowe następstwo tranzytów. Wszystkie łańcuchy tej sieci są łańcuchami produktowymi, a opisywane przez nią następstwo jest nienaruszalne, gdyż wynika ono z zasady, że żadna całość nie może powstać przed powstaniem jej części. Na etapie realizowania zależności produktowych nie mamy żadnej możliwości wyboru. Tabelka projektu wyznacza następstwo produktowe jednoznacznie. Newbold [6] nazywa to następstwo *precedence view* (str.82)
4. Teraz przystępujemy do realizowania założeń dotyczących dostępności zasobów narzędziowych. Na tym etapie może pojawić się niedeterminizm związany z kolejnością wykorzystania narzędzi. Możemy pozostawić go w sieci do rozstrzygnięcia na etapie realizacji projektu, lub też podjąć decyzję już na etapie planowania. Newbold nazywa to następstwo *resource view* (str.82).

W dalszym ciągu *projektem* będziemy nazywali zbiór jednoatomowych sieci, który powstaje w kroku drugim, a jego siecią, sieć Petriego, o której mowa w kroku czwartym. Jak już zauważyliśmy na podanych przykładach, jednemu projektowi może odpowiadać wiele realizujących go sieci.

# 12 Studium przypadku — odbiornik radarowy

Wykonajmy teraz ćwiczenie odwrotne do omówionego w Roz.5. Opiszemy w języku sieci Petriego przykład z Roz.11 książki Newbolda [6] przedstawiony tam w postaci diagramu pokazanego na Rys.12.1. Dotyczy on projektu zbudowania odbiornika radarowego składającego się z modułu wewnętrznego (oprogramowane urządzenie elektroniczne) i modułu zewnętrznego (obudowa). Do realizacji projektu oddelegowano inżyniera E, programistę P i technika T.



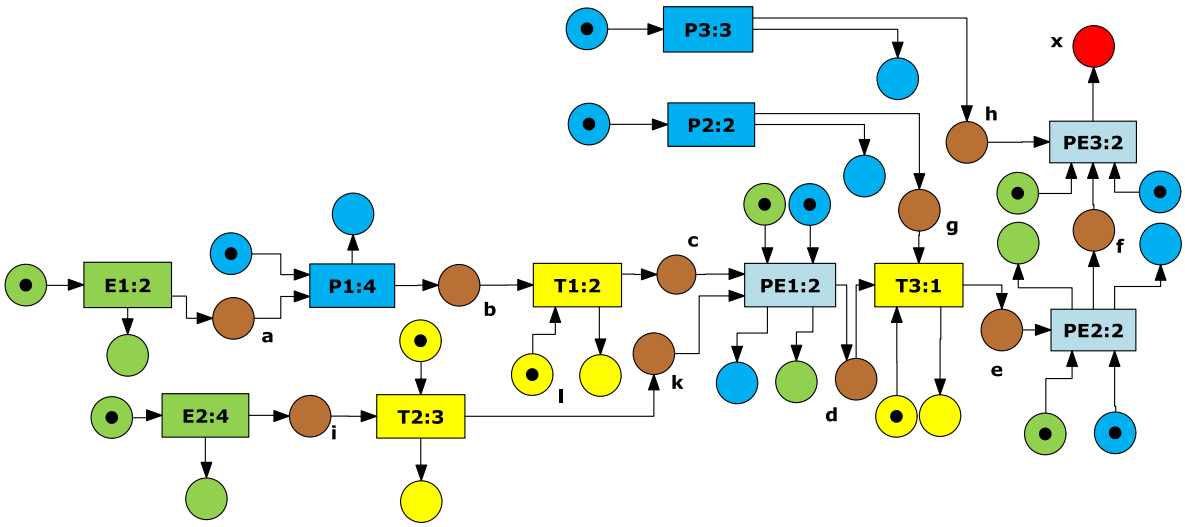

**Rys.12.1 Opis projektu w formalizmie Newbolda**

Opis projektu przy pomocy podobnej jak w Roz.3 tabeli tranzytów i zasobów został zamieszczony w Tab.12.1. Symbole tranzytów są takie same jak w [6], aby łatwo było porównać oba opisy. Litery w nawiasach są symbolami produktów i odpowiadających im miejsc produktowych w sieci. Czasy wykonania liczone są w miesiącach.

| tranzyt | opis | zasoby na wejściu | zasoby na wyjściu | czas |
|---|---|---|---|---|
| E1 | projektowanie modułu wew. | inżynier | projekt modułu wewnętrznego (a)<br>inżynier | 2 |
| P1 | programowanie podstawowe | projekt modułu wewnętrznego (a)<br>programista | oprogramowanie podstawowe (b)<br>programista | 4 |
| T1 | budowa modułu wewnętrznego | oprogramowanie podstawowe (b)<br>technik | moduł wewnętrzny (c)<br>technik | 2 |
| E2 | projektowanie modułu zew. | inżynier | projekt modułu zewnętrznego (i)<br>inżynier | 4 |
| T2 | budowa modułu zewnętrznego | projekt modułu zewnętrznego (i)<br>technik | moduł zewnętrzny (k)<br>technik | 3 |
| PE1 | integracja modułu wewnętrznego z zewnętrznym | moduł zewnętrzny (k)<br>moduł wewnętrzny (c)<br>inżynier<br>programista | zintegrowane urządzenie (d)<br>inżynier<br>programista | 2 |
| P2 | programowanie autotestu | programista | autotest (g)<br>programista | 2 |
| P3 | programowanie testu akceptacji | programista | test akceptacji (h)<br>programista | 3 |
| T3 | scalenie urządzenia | zintegrowane urządzenie (d)<br>technik | scalone urządzenie (e)<br>technik | 1 |
| PE2 | integracja po scaleniu | scalone urządzenie (e)<br>inżynier<br>programista | scalone i zintegrowane urządzenie (f)<br>inżynier<br>programista | 2 |
| PE3 | ostateczne testowanie | scalone i zintegrowane urządzenie (f)<br>test akceptacji (h)<br>programista<br>inżynier | koniec projektu (x) | 2 |

**Tab.12.1 Tabela projektu budowy odbiornika radarowego**

Wydaje się, że tranzyt T1 powinien otrzymać na wejściu dwa produkty: oprogramowanie podstawowe i projekt modułu wewnętrznego. Nie wprowadzamy jednak tej korekty do tabeli, aby zachować zgodność z przykładem opisanym przez Newbolda.




**Rys. 12.2 Budowa odbiornika radarowego — następstwo produktowe**

Dla uproszczenia przykładu opuszczamy drugi krok algorytmu opisanego w Roz.11 (zbiór atomów) i od razu przechodzimy do etapu sieci uwzględniającej wszystkie następstwa produktowe. Ta sieć jest pokazana na Rys.12.2. Na Rys.12.1 odpowiada ona górnej części diagramu, która przez Newbolda jest nazywana *precedence view*. Liczby stojące w opisach tranzytów po dwukropkach określają czasy wykonania. Obecne w sieci znaczniki odpowiadają znakowaniu początkowemu, a znakowanie końcowe (zakończenie projektu), to pojawienie się znacznika w miejscu x (czerwone).

Teraz mamy do przeprowadzenia ostatni i najtrudniejszy krok budowy sieci projektu polegający na uwzględnieniu w sieci ograniczoności zasobów narzędziowych: jeden inżynier, jeden programista i jeden technik. Zauważmy, że realizacja projektu zgodnie z siecią na Rys.12.2 wymagałaby dostępności sześciu programistów, pięciu inżynierów i trzech techników, tyle bowiem znaczników stoi w miejscach narzędziowych.

Technicznie najprostszym do wykonania krokiem byłoby połączenie wszystkich miejsc o wspólnym kolorze w jedno miejsce (podobnie jak w przykładzie tynkowania i malowania pokoi na Rys.3.3). W ten sposób powstałaby sieć niedeterministyczna opisująca projekt, w którym decyzje o narzędziowym następstwie tranzytów byłyby podejmowane w takcie realizacji projektu. Taka sieć miałaby wiele różnych realizacji, z których wiele byłoby zapewne nieefektywnymi czasowo. Np. wykonanie na początku E1, P3, P2, P1 w tej właśnie kolejności umożliwiłoby odpalenie T1 dopiero w 9 miesięcy po zakończeniu E1, podczas, gdy kolejność P1, P2, P3 skróciłaby ten czas do czasu wykonania P1, a więc do 4 miesięcy.

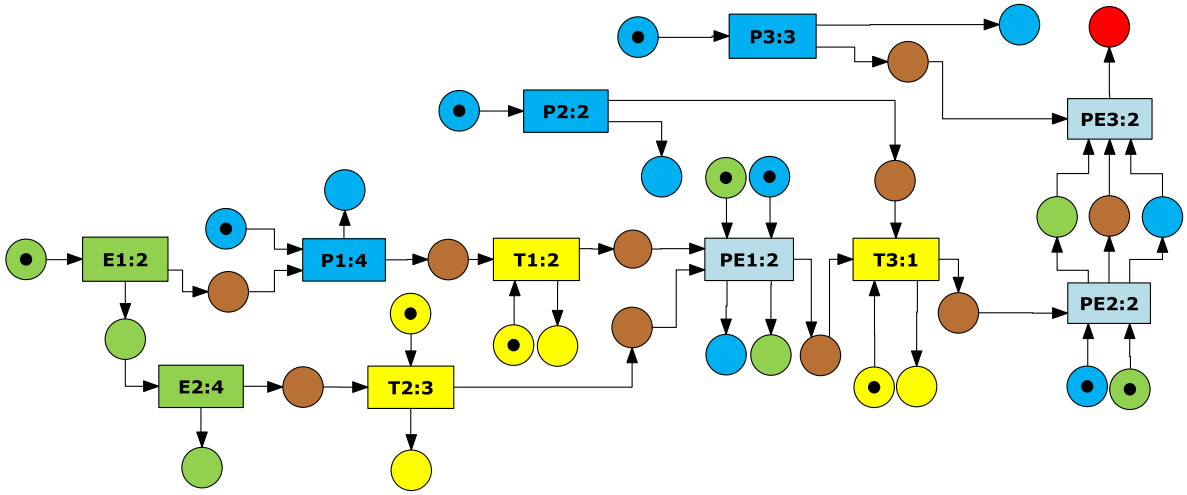
Aby wprowadzić do naszej sieci z góry zadane (przez planistę) narzędziowe następstwa tranzytów, należy w pierwszym rzędzie zdecydować, jakie cele mają one realizować. Przypuśćmy, że mamy do osiągnięcia dwa często występujące wspólnie cele optymalizacyjne:

1. odpalanie każdego tranzytu najpóźniej, jak to jest tylko możliwe; dlaczego „najpóźniej”, a nie „najwcześniej” zastanowimy się w Roz.15,
2. minimalizacja czasu wykonania projektu.

Założymy też, że w przypadku konfliktu kryteriów, kryterium pierwsze jest ważniejsze. Możemy teraz zastosować algorytm Newbolda polegający na tym, że w pierwszej kolejności rozważamy ostatni tranzyt projektu, później bezpośrednio go poprzedzające itd. Ostatnim tranzytem projektu jest PE3. Aby mógł on zostać odpalony, wcześniej muszą być odpalone bezpośrednio poprzedzające go tranzyty P3 i PE2. W tej sprawie nie mamy żadnego wyboru. Jednakże „wcześniej" nie oznacza „tuż przed". Zauważmy też, że w naszym przypadku P3 i PE2 nie mogą być odpalone równocześnie, gdyż oba korzystają z jedynego w projekcie programisty. Należy więc zdecydować o ich kolejności.

W tym miejscu Newbold stosuje heurystyczną zasadę mówiącą, że później powinien zostać odpalony ten tranzyt, który stoi na końcu czasowo dłuższego łańcucha (str.84). W naszym przypadku to oczywiście PE2, co prowadzi do połączenia wyjść PE2 z wejściami PE3.

W tym kroku możemy też rozstrzygnąć konflikt pomiędzy E1 i E2. E1 ustawiamy na pierwszej pozycji, gdyż czasowa długość łańcucha do PE1 jest od E1 dłuższa, niż od E2. To oznacza, że narzędziowe wyjście E1 łączmy z narzędziowym wejściem E2. W rezultacie otrzymujemy diagram przedstawiony na Rys.12.2.




**Rys.12.2 Pierwsze przybliżenie następstwa narzędziowego**

Teraz należy ustalić wzajemną kolejność czterech tranzytów korzystających z programisty: P1, P2, P3 i PE1. Stosując opisaną poprzednio regułę PE1 powinien wyprzedzać P2 i P3, jednakże Newbold zauważa, że w takim przypadku programista po wykonaniu P1 czekałby bezużytecznie dwie jednostki czasu, lepiej więc te dwie jednostki wykorzystać na wykonanie P2. To nas prowadzi do kilku możliwych kolejności:

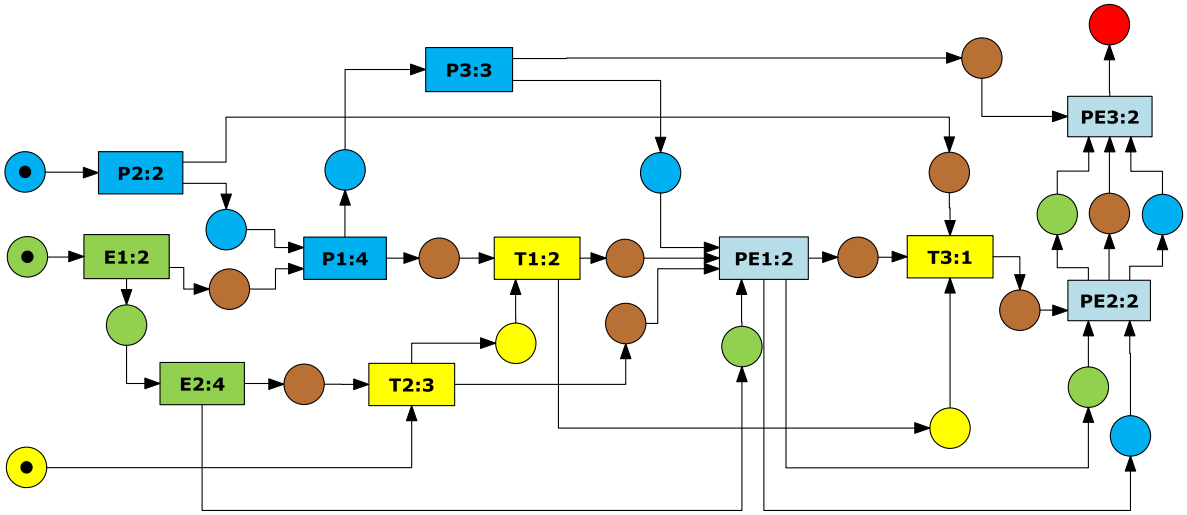
P1 → P2 → PE1 → P3 lub

P2 → P1 → PE1 → P3 lub

P1 → P2 → P3 → PE1 lub

P2 → P1 → P3 → PE1 lub

Za Newboldem wybieramy trzecie rozwiązanie, tym bardziej, że wszystkie trzy są równoważne z punktu widzenia czasu całego projektu. Po ustaleniu dość oczywistych dróg przekazywania technika i inżyniera otrzymujemy diagram przedstawiony na Rys.12.3.




**Rys.12.3 Sieć projektu**

To już jest ostateczna sieć projektu, oczywiście nie jedyna, która realizuje specyfikację przedstawioną w Tab.12.1. Ze względu na dalsze rozważania, wybrany został wariant zgodny z rozwiązaniem zaproponowanym przez Newbolda.

# 13 Łańcuch krytyczny — co to jest?

Zajmiemy się teraz pojęciem łańcucha krytycznego, centralnym pojęciem teorii ograniczeń Goldratta.

W literaturze przedmiotu trudno jest znaleźć definicję tego pojęcia, lub też — ujmując rzecz inaczej — jest ich zbyt wiele. Znajdujące się tam określenia, bo trudno nazwać je definicjami, nie są wzajemnie równoważne, wydaje się też, że ich autorzy nie do końca zdają sobie z tego sprawę. Wynika to po części stąd, że używany przez nich język nie pozwala na precyzję wypowiedzi. Oto określenia łańcucha krytycznego, które udało mi się znaleźć:

1. *The set of tasks that determine, when a project can finish is called Critical Chain. They are critical because an improvement anywhere in the chain means the project can get done earlier*. ([6] str. 57). Zważywszy że autor najprawdopodobniej (choć nie pisze tego wyraźnie) rozważa jedynie tranzyty otwarte w sensie zdefiniowanym w Roz.8, tę „definicję" można by rozumieć tak, że czas wykonania całego projektu może być przyspieszony, jeżeli przyspieszymy wykonanie dowolnego tranzytu łańcucha krytycznego. To jednak oznaczałoby, że każdy podzbiór łańcucha krytycznego jest łańcuchem krytycznym, co nie jest zgodne z intencją autora, gdyż na str.64 znajdujemy następujące uściślenie:
2. *Improvements along the critical chain will likely result in improvements to the project as a whole; improvements elsewhere will not*. (podkreślenie AB). Z części zdania po średniku można więc wnosić, że łańcuch krytyczny zawiera wszystkie tranzyty o tej własności, że przyspieszenie dowolnego z nich spowoduje przyspieszenie całego projektu.
3. *The Critical Chain is the set of tasks which determine the overall duration of the project, taking into account both precedence and resource-dependencies.* ([6] str. 85).

4. *No task can be pushed to the future without pushing the completion date. The Critical Chain tasks are those that also can't be pushed to the past (...). To identify the Critical Chain, we need to compute how far to the past each task could be moved, without pushing anything before the horizon start.* ([6] str. 85). Zwróćmy uwagę, że "pushed to the future" oznacza opóźnienie zakończenia, podobnie jak "pushed to the past" oznacza przyspieszenie zakończenia. To nieco nienaturalny sposób mówienia i opóźnieniach i przyspieszeniach, wynikający być może z proponowanej przez autora metody ustalania łańcucha krytycznego.
5. *...the constraint to a single Project is a critical chain, defined as the longest path through the network after resource leveling.* ([5] str. 65). Przez „resource levelling" autor rozumie uwzględnienie w planie projektu następstwa narzędziowego.
6. *With unlimited resources, the critical chain is the same path as the critical path.* ([5] str. 77).

Obok tych wszystkich określeń znajduje się też uwaga, która po części wyjaśnia, dlaczego autorzy nie dbają zbytnio o jednoznaczne ustalenie pojęcia łańcucha krytycznego:

*You might have noticed that the beginning of this section is entitled „Determine the Critical Chain", rather than „Identify the Critical Chain". This is an important distinction. The Critical Chain will determine, to a large degree, what we will focus on. We can't let an algorithm decide by itself, so now we must consider: is this where we want the Critical Chain to be? This is an important strategic decision, for the project and for the company.* ([6] str. 87)

Jak stąd wynika, łańcuch krytyczny nie jest pojęciem zdefiniowanym w sposób algorytmiczny lub choćby funkcyjny — dla danej sieci projektu łańcuchem krytycznym nazywamy to a to — lecz pojęciem, które podlega opartemu na intuicji wyborowi dokonywanemu przez kierownika projektu.

Choć z formalnego punktu widzenia można by na tym rozważania o łańcuchu krytycznym zakończyć, to z punktu widzenia pragmatycznego warto się zastanowić wśród jakich zbiorów tranzytów w sieci warto szukać kandydata na łańcuch krytyczny pochodzący ostatecznie z wybory planisty. Warto też zadać sobie pytanie, czy łańcuch krytyczny ma być „łańcuchem skierowanym" w sensie teorii grafów (definicja w Roz.9), czy też po prostu zbiorem tranzytów nie koniecznie układających się w spójną sieć. Aby odpowiedzieć na te pytania wprowadzimy trzy pojęcia:

*Łańcuchem krytycznym typu P* (przyspieszenie) nazwiemy zbiór tych wszystkich tranzytów sieci, że każde przyspieszenie wykonania jednego z nich powoduje przyspieszenie wykonania całego projektu. To pojęcie odpowiada cytowanym wyżej określeniom 1 i 2 oraz częściowo 4.

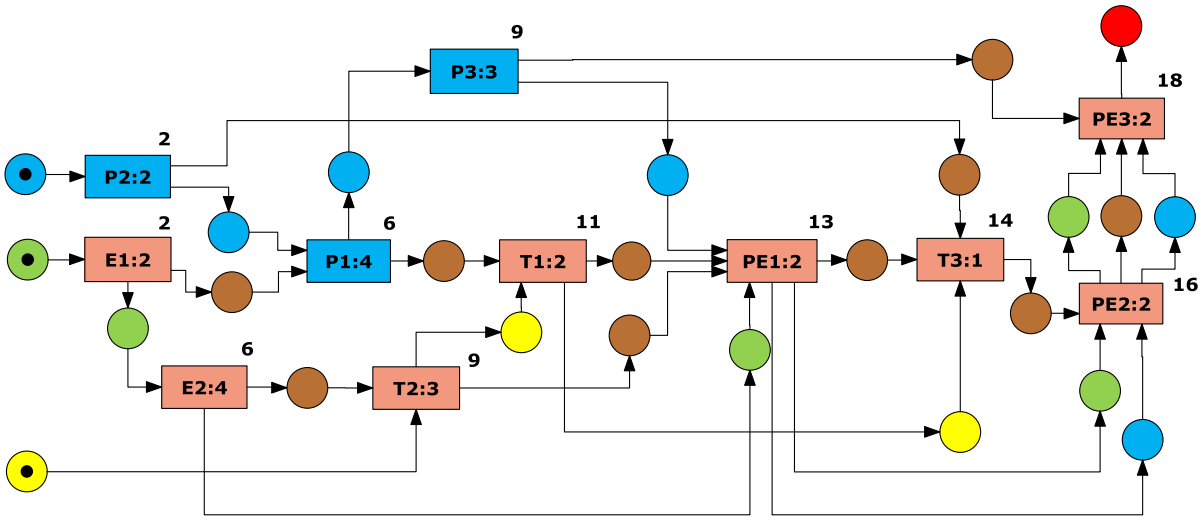
*Łańcuchem krytycznym typu O* (opóźnienie) nazwiemy zbiór tych wszystkich tranzytów sieci, że każde opóźnienie wykonania jednego z nich powoduje opóźnienie wykonania całego projektu. To pojęcie odpowiada drugiej części określenia 4.

*Łańcuchem krytycznym typu PO* nazwiemy łańcuch, który jest jednocześnie typu P i typu O (określenie 4).

*Łańcuchem krytycznym typu M* (maksymalny czas) nazwiemy najdłuższy czasowo skierowany łańcuch tranzytów (w sensie Roz. 9).

Zajmiemy się teraz analizą każdego z tych pojęć. W dalszym ciągu będziemy rozważać jedynie sieci otwarte (Roz.8), tj. takie, których tranzyty odpalają niezwłocznie, gdy tylko staje się to możliwe.

Na Rys.13.1 widzimy sieć projektu budowy odbiornika radarowego, w której dla każdego tranzytu pokazano liczony od wystartowania projektu czas po jakim zostanie osiągnięte zakończenie wykonania tego tranzytu. Te czasy są naniesione nad prawym górnym rogiem prostokąta symbolizującego tranzyt.

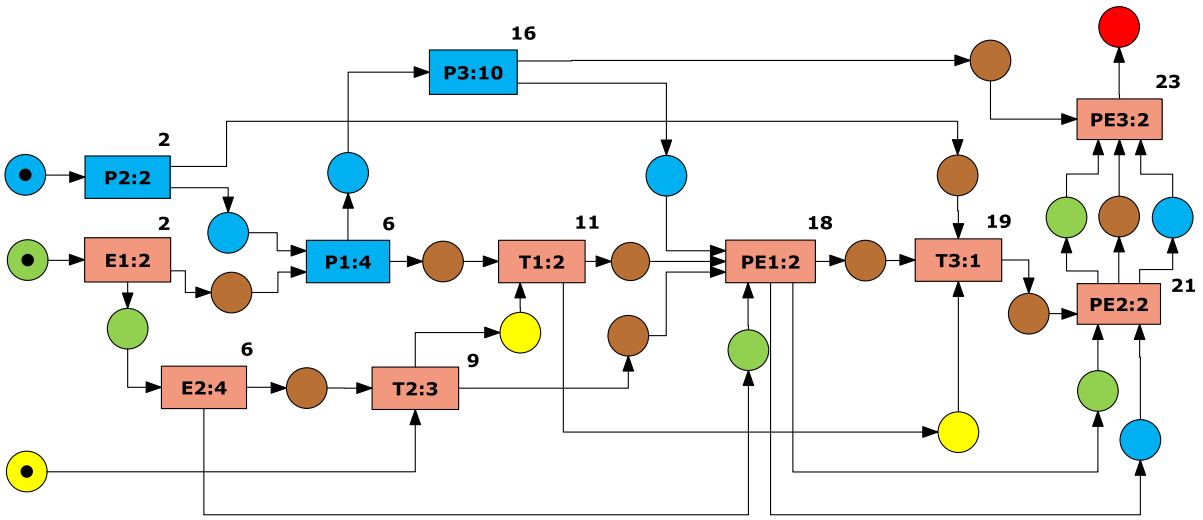



**Rys. 13.1 Odbiornik radarowy z czasami wykonania i jednym łańcuchem krytycznym**

W naszym przykładzie łańcuch krytyczny typu P jest jednocześnie łańcuchem krytycznym typu O, a więc też łańcuchem krytycznym typu PO. Jest też łańcuchem typu M i składa się z wymienionych niżej tranzytów, które tworzą łańcuch skierowany w sensie zdefiniowanym w Roz.9 (na rysunku zaznaczony na czerwono):

E1 → E2 → T2 → T1 → PE1 → T3 → PE2→ PE3

Nie zawsze jednak musi być tak, że łańcuchy typu P, O i M są jednakowe. Przypuśćmy, że w naszej sieci wykonanie tranzytu P3 trwa nie 3 ale 10 miesięcy.




**Rys. 13.2 Sieć z trzema łańcuchami krytycznymi**

W tej sytuacji mamy trzy różne łańcuchy:

Łańcuch typu P: P1, P3, PE3

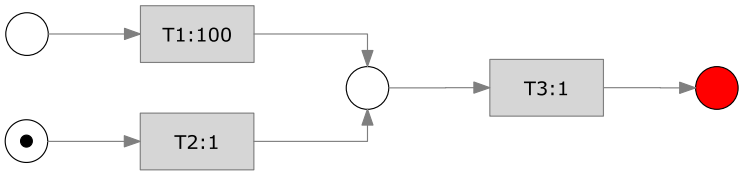
Łańcuch typu O: P2, P1, P3, PE1, T3, PE2, PE3

Łańcuch typu M: E1 → E2 → T2 → T1 → PE1 → T3 → PE2→ PE3

W tym przypadku czas wykonania wszystkich tranzytów łańcucha typu M, to oczywiście czas wykonania projektu.

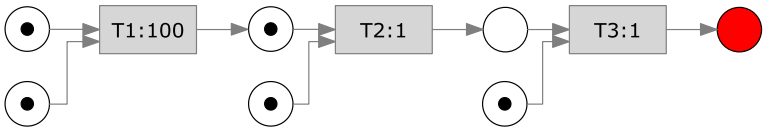
# 14 Łańcuchy krytyczne — rozważania ogólne

Zastanowimy się teraz, jakie własności ogólne mają łańcuchy krytyczne. Zaczniemy od zbadania hipotezy, która przewija się między wierszami w obu cytowanych książkach [5] i [6], że czas wykonania łańcucha krytycznego typu M jest czasem wykonania projektu. W obu wersjach projektu omawianego w Roz.13 jest tak właśnie. Okazuje się jednak, że nie zawsze musi tak być. Na Rys.14.1 widzimy sieć, której łańcuch typu M ma długość czasową 101, ale czas wykonania projektu wynosi jedynie 2. Tranzyt T1 przez brak znacznika w jego wejściu, nigdy nie zostanie odpalony. I w rzeczywistości jego odpalenie wcale nie jest do zrealizowania projektu potrzebne. Taki tranzyt będziemy nazywali *martwym*. Ogólnie *tranzytem martwym* będziemy nazywali taki tranzyt, że wypełnienie znacznikami jego miejsc wejściowych nie jest osiągalne ze znakowania początkowego.




**Rys.14.1 Sieć z martwym tranzytem**

Zastanówmy się więc, czy dla sieci projektu, która nie ma martwych tranzytów, nasza hipoteza jest zawsze spełniona. I znów okazuje się, że nie jest, a przykład widzimy na Rys.14.2. Łańcuch typu M tej sieci składa się z tranzytów T1, T2 i T3 (zakończenie projektu), a jego długość czasowa wynosi 102, natomiast do zakończenie projektu wystarczy odpalenie T2 i T3, a więc czas wykonania projektu wynosi 2. Choć tranzyt T1 nie jest martwy, to jednak jego odpalenie nie jest potrzebne do osiągnięcia znakowania końcowego. Można go z sieci usunąć nie pozbawiając jej atrybutu osiągalności. Taki tranzyt będziemy nazywali *zbędnym*. Ogólnie *tranzytem zbędnym* będziemy nazywali taki tranzyt, którego odpalenie nie jest konieczne dla przejścia sieci od znakowania początkowego do końcowego. Zauważmy, że każdy tranzyt martwy jest zbędny, ale nie na odwrót.




**Rys.14.2 Sieć z tranzytem zbędnym**

Oczywiście można zadać pytanie, czy ktoś przy zdrowych zmysłach zaplanuje projekt z tranzytem martwym lub zbędnym? I oczywiście odpowiedź będzie, że nikt. Jednakże przy rozległej sieci projektu, która na dodatek była wielokrotnie modyfikowana (a to się zdarza!) taka sytuacja może powstać w sposób nie tylko przez planistę niezamierzony, ale też i dla niego niewidoczny. W takim przypadku planista fałszywie oszacuje czas wykonania całości zadania.

Opisane wyżej sytuacje, jeżeli się zdarzą, powinny zostać wyeliminowane. Jak to zrobić w sposób algorytmiczny, zastanowimy się później. Teraz postaramy się uzupełnić definicję sieci

projektu podaną w Roz.9 o warunki, które powinna spełniać „sensowna" sieć. Te warunki są następujące:

1. w sieci projektu istnieje jeden tranzyt stojący na końcu wszystkich łańcuchów skierowanych (Roz.9); ten tranzyt nazywamy *tranzytem końcowym*, a jego odpalenie oznacza zakończenie projektu; innymi słowy znakowanie końcowe, to pojawienie się znaczników w miejscach wyjściowych tranzytu końcowego i tylko w tych miejscach,
2. dla zadanego w projekcie znakowania początkowego, znakowanie końcowe jest osiągalne,
3. żaden tranzyt sieci nie jest zbędny,
4. sieć nie ma cykli.

Sieć projektu spełniające te cztery warunki będziemy nazywali *siecią znormalizowaną*. Można udowodnić, że:

*każdą sieć projektu, który ma sens praktyczny,*
*można sprowadzić do sieci znormalizowanej*

Dowód tej tezy jest następujący:

Warunek pierwszy oznacza, że projekt kończy się wspólnym dla całej sieci tranzytem ostatecznego przyjęcia projektu. W projektach rzeczywistych jest to protokolarny odbiór wytworzonego produktu, który zwykle ma miejsce, a nawet gdyby nie miał, można taki tranzyt do sieci dodać. Bez tego założenia komplikują się definicje czasu zakończenia projektu i łańcuchach krytycznych. Ten warunek gwarantuje również, że sieć projektu jest spójna w tym sensie, że jest to jedna sieć, a nie kilka sieci, tak jak np. na Rys.3.1 i Rys. 3.2.

Warunek drugi oznacza, że projekt da się zrealizować. Gdyby sieć projektu nie spełniała tego warunku, oznaczałoby to, że w planie realizacji projektu są błędy, które należy usunąć. Np. gdyby w sieci projektu odbiornika radarowego zabrakło znacznika na wejściu tranzytu P2 (projektowi nie udostępniono programisty), to pojawienie się znacznika na wyjściu tranzytu PE3 nie byłoby możliwe, co oznaczałoby, że projekt nie może być ukończony. W ogólnej sieci Petriego znane są algorytmy sprawdzania, czy z danego znakowania początkowego jest osiągalne dane znakowanie końcowe.

Warunek trzeci oznacza, że w projekcie nie ma tranzytów zbędnych. Przed zatwierdzeniem sieci do realizacji należy więc sprawdzić ją pod kontem obecności tranzytów zbędnych i jeżeli takie zostaną odnalezione, to należy je usunąć. Również to zadanie można wykonać w sposób algorytmiczny.

Warunek czwarty nie zawsze jest spełniony, gdyż, jak pokazano w Roz.5, cykle w projektach mogą się zdarzać. Jeżeli jednak w projekcie występuje cykl, to pojęcie czasu wykonania projektu, a także każde z pojęć łańcucha krytycznego, nie dają się określić, bo projekt może trwać dowolnie długo. W takim przypadku, aby móc w ogóle optymalizować projekt pod kątem czasowym, trzeba postawić jakąś górną granicę na liczbę wykonania każdego z cykli, co na gruncie teorii sieci Petriego oznacza, że każdy z cykli zastępujemy kilkoma (niedeterministycznie rozgałęziającymi się) skończonymi łańcuchami. W teorii sieci Petriego znane są algorytmy ustalania, czy w sieci istnieje cykl.

Udowodnimy teraz dwa proste twierdzenia o znormalizowanych sieciach projektów, które mogą być pomocne przy zarządzania projektami.

**Twierdzenie 13.1** Dla każdej znormalizowanej sieci projektu (teoriomnogościowa) suma łańcuchów krytycznych typu M tworzy łańcuch krytyczny typu O.

**Dowód**.

Przypuśćmy, że tranzyt T należy do łańcucha typu M. Z faktu nieistnienia w sieci tranzytów zbędnych wynika, że tranzyt T musi być odpalony w procesie osiągania znakowania końcowego. Skoro tak, to dowolne wydłużenie czasu wykonania tranzytu T spowoduje wydłużenie czasu wykonania łańcucha, do którego tranzyt należy, a więc i wydłużenie trwania całego projektu.

Przypuśćmy teraz, że tranzyt T nie należy do żadnego łańcucha typu M. Skoro tak, to należy do łańcucha o czasowej długości mniejszej od czasu wykonania każdego z łańcuchów typu M, a więc wydłużenie czasu jego wykonania o nie więcej niż różnica pomiędzy czasem wykonania projektu, a czasem wykonania tego łańcucha, nie zwiększy czasu wykonania projektu. Zatem tranzyt T nie należy do łańcucha typu O.

c.b.d.o.

**Twierdzenie 13.2** Dla każdej znormalizowanej sieci projektu, która zawiera tylko jeden łańcuch krytyczny typu M, ten łańcuch jest równocześnie łańcuchem typu O i typu P.

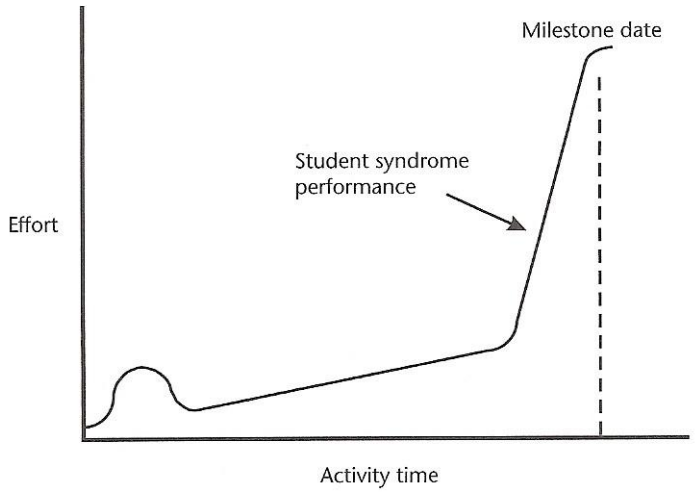
**Dowód.**

Pierwsza część dowodu wynika wprost z Tw.13.1. Jeżeli natomiast łańcuch krytyczny typu M jest w sieci tylko jeden, to wszystkie pozostałe łańcuchy kończące się tranzytem terminalnym są czasowo krótsze, a więc skrócenie czasu wykonania dowolnego tranzytu należącego do łańcucha typu M, skróci czas wykonania całego projektu.

c.b.d.o.

# 15 Psychologiczne i statystyczne aspekty zarządzania projektami

Dotychczasowe rozważania związane z czasem wykonania projektu prowadziliśmy przy założeniu, że każdy tranzyt ma pewien z góry określony czas wykonania, który przy realizacji projektu, ani nie może być skrócony, ani też nie zostanie przedłużony. W rzeczywistości jednak tak właśnie nie jest, o czym świadczy statystyka opóźnień przytoczona w Roz.1. Choć czas wykonania projektu rzadko bywa skrócony, to jego wydłużenie jest w zasadzie regułą. Dzieje się tak z kilku powodów.




**Rys. 15.1 Syndrom studenta**

Pierwszy powód, to tzw. *syndrom studenta* przedstawiony na pochodzącym z [5] Rys.15.1, gdzie na osi pionowej mierzymy intensywność pracy, a na poziomej — czas. Przerywana linia pionowa wskazuje termin egzaminu, co w naszym przypadku odpowiada terminowi zakończenia zadania. Jak widzimy, włożony wysiłek koncentruje się w pobliżu tej daty, co pozwala na wyciągnięcie dwóch wniosków:

1. wydłużenie czasu nie zmieni zapewne faktu, że zadanie i tak nie zostanie wykonane w terminie,
2. przy zmianie rozkładu intensywności pracy możliwe jest poważne skrócenie czasu wykonania zadania bez obawy, że nie zostanie ono wykonane na czas.

Zdaniem praktyków zarządzania projektami, większość zespołów projektowych pracuje właśnie w ten sposób. Co więcej, opierając się na własnych doświadczeniach związanych z opóźnianiem się zadań, stara się zawsze żądać więcej czasu na wykonanie zadania, niż w rzeczywistości potrzebuje. Uzyskuje w ten sposób poczucie, że chroni się przed nieprzewidzianymi okolicznościami, gdy w rzeczywistości tak nie jest, ponieważ niezależnie od planowanej daty zakończenia, gros pracy jest wykonywane w ostatniej chwili.

Drugie źródło opóźnień, to *prawo Murphiego*, które mówi, że jeżeli coś może się nie udać, to z pewnością tak właśnie się stanie. To prawo również powoduje, że planujemy więcej czasu niż potrzebujemy, jednakże syndrom studenta uniemożliwia nam skorzystanie z tego zabezpieczenia. Jeżeli dodatkowo jesteśmy zaangażowani w więcej niż jeden projekt na raz, to wkraczając w nowy projekt będziemy w pierwszym rzędzie odrabiać zaległości w już biegnącym kosztem nowego.

Trzecie źródło opóźnień to zasada, którą nazwiemy tu *nigdy przed terminem*. Jeżeli zespół zadaniowy wykona zadanie przed terminem, to najczęściej nie zgłosi tego kierownikowi projektu gdyż:

1. każde zadanie zawsze można wykonać „jeszcze lepiej”,
2. jak zgłoszę zakończenie zadania, to dostanę następne,
3. w przyszłości dadzą mi mniej czasu niż zaplanuję,
4. zaoszczędzony czas mogę wykorzystać na redukcję innych opóźnień.

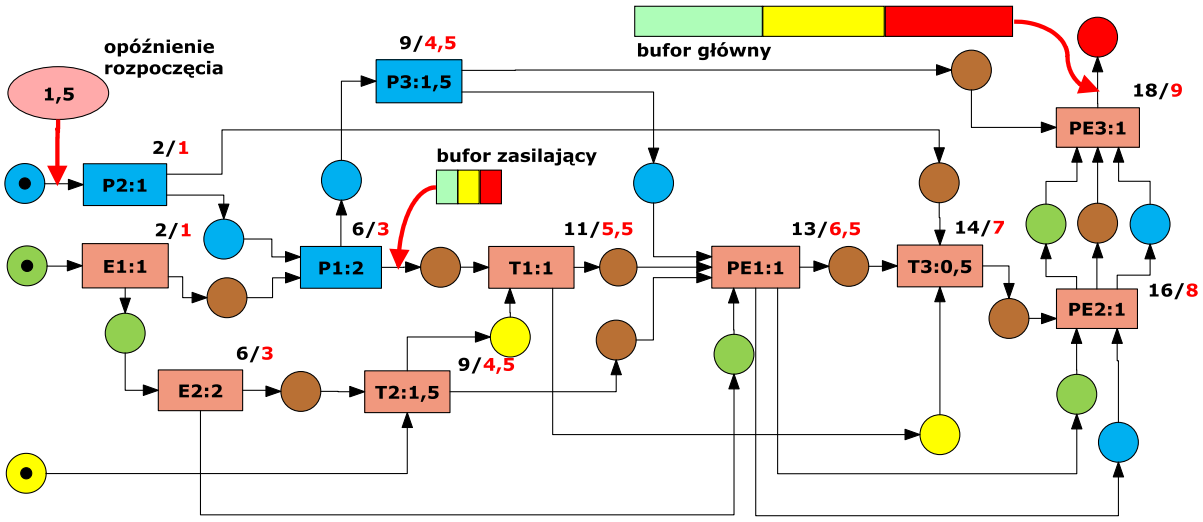
Ze statystycznego punktu widzenia, ze względu na różne nieprzewidziane okoliczności, część zadań w projekcie może się opóźnić, a część może być wykonanych wcześniej. Niestety, zasada *nigdy przed terminem* spowoduje, że wcześniejsze zakończenia niektórych zadań nie zredukuje efektu opóźnień innych.

# 16 Buforowanie projektu

Buforowanie projektu to jedna z ważniejszych technik w metodologii Goldratta. Polega ona na wykonaniu dwóch przekształceń sieci projektu:

1. poważne ograniczenie czasów danych na wykonanie poszczególnych tranzytów, np. do 50% czasu planowanego pierwotnie,
2. wprowadzenie do sieci buforów zawierających rezerwę czasową pochodzącą ze skrócenia czasu wykonania tranzytów.

Jeżeli tranzyty projektu podlegają syndromowi studenta, to powinno dać się je wykonać w czasie znacznie krótszym od pierwotnie planowanego. Gdyby jednak pojawiło się zagrożenie opóźnieniem, to osoba odpowiedzialna za działanie może zgłosić kierownikowi projektu potrzebę dodatkowego czasu na wykonanie zadania. Ten czas zostanie przydzielony i jednocześnie odjęty od bufora.




**Rys.16.1 Sieć projektu budowy odbiornika radarowego z buforami**

Na Rys.16.1 pokazana jest znana nam sieć, w której wszystkie czasy przypisane tranzytom zostały skrócone o połowę. Nad tranzytami widzimy czasy osiągnięcia ich zakończenia w wersji pierwotnej (na czarno) i w wersji po skróceniu czasów (na czerwono). Tranzyty w kolorze czerwonym wskazują łańcuch krytyczny.

Jak widać, planowany czas wykonania całego projektu został skrócony z 18 do 9 miesięcy. Zaoszczędzony w ten sposób czas umieszczono w buforze i podzielono na trzy równe odcinki — po trzy miesiące każdy. W czasie realizacji projektu jego kierownik otrzymuje informacje o stanie wykonania wszystkich zadań. Te informacje wpływają do niego bardzo często, nawet codziennie, i wskazują liczbę jednostek czasu, jaka jest potrzebna do zakończenia zadania[5].

[5] Jak uczy doświadczenie, taka właśnie forma raportowania pozwala najskuteczniej wykrywać ewentualne zagrożenia czasu wykonania projektu. Skądinąd często stosowana forma alternatywna polega-

Jeżeli kierownik danego tranzytu uzna, że istnieje realne zagrożenie przekroczenia czasu wykonania tranzytu, zgłasza się do kierownika projektu z prośbą o przydział dodatkowego czasu. Musi jednak przedstawić dobre uzasadnienie swojej prośby.

Każde przydzielenie dodatkowego czasu powoduje odjęcie go od bufora. Dopóki pierwsza tercja bufora (zielona) nie zostanie wyczerpana, kierownik projektu nie ogłasza alarmu. Gdy jednak tak się stanie, włączone zostają specjalne procedury, a po wyczerpaniu drugiej tercji (żółtej) jest ogłaszany stan alarmowy. Jak się okazuje, ta stosunkowo prosta strategia pozwala nie tylko na utrzymanie pierwotnego czasu wykonania projektu (w naszym przykładzie 18 miesięcy), ale często na jego skrócenie.

Obok bufora głównego często wprowadza się do projektu tzw. *bufory zasilające* (ang. *feeding buffers*) instalowane na „dopływach" do łańcucha krytycznego. W naszym przypadku wprowadzono taki bufor po tranzycie P1. Jego pojemność to 1,5 miesiąca będąca różnicą pomiędzy datami zakończenia T2 i P1. Taka różnica powstałaby, gdyby P2 i E1 odpalono równocześnie. Wtedy, po zakończeniu działania P1, tranzyt T1 czekałby 1,5 miesiąca na zakończenie T2. Aby pozbawić P2 i P1 tej komfortowej sytuacji (nadmiar czasu), kierownik projektu postanawia, że wykonanie P2 rozpocznie się w 1,5 miesiąca po odpaleniu E1 i te 1,5 miesiąca umieszcza w buforze zasilającym.

Z teorio-sieciowego punktu wiedzenia wszystkie bufory, a także opóźnienia, można potraktować jako tranzyty, które nie wykonują żadnych czynności, a jedynie konsumują czas. W ten sposób sieć z buforami staje się zwykłą siecią Pietriego.

Jeżeli na czas spojrzymy jako na zasób konsumowany przez tranzyty, to natychmiast nasuwa się idea buforowania innych zasobów, np. narzędziowych lub finansów. Ta idea jest również znana praktykom metodologii Goldratta. W szczególności rozważa się bufory narzędziowe, które monitorują zbliżanie się daty, w której dane narzędzie będzie potrzebne. Ta data jest komunikowana menadżerowi zasobów z odpowiednim wyprzedzeniem pozwalającym na zagwarantowanie dostępności narzędzia, gdy stanie się ono wymagane.

# 17 Uogólnienia idei sieci Petriego

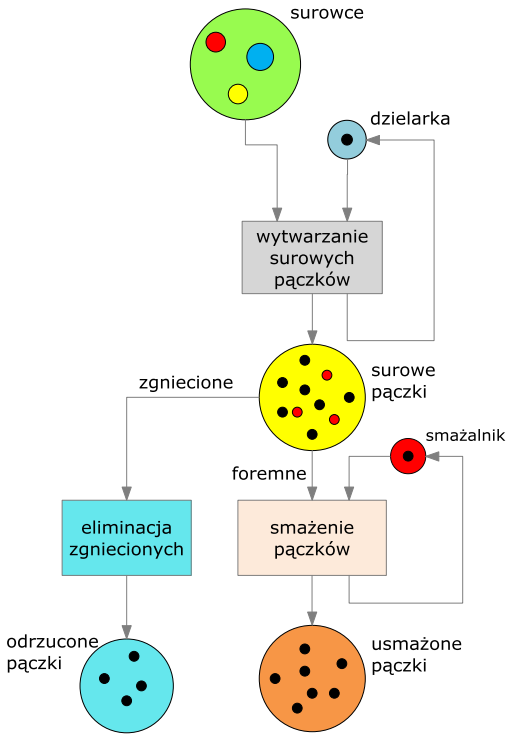
Sieci Petriego, którymi posługiwaliśmy się w naszych rozważaniach, to ich najprostsza wersja określona przez następujące ograniczenia:

1. w każdy miejscu może stać co najwyżej jeden znacznik,
2. warunek przypisany miejscu, to jedynie sprawdzenie, czy znacznik jest, czy też go nie ma,
3. znaczniki nie niosą żadnej informacji poza rodzajem zasobu, jaki reprezentują.

W ogólnej teorii sieci Petriego rozważa się przypadki wychodzące poza te ograniczenia. Taki właśnie przypadek jest pokazany na Rys.17.1. Sieć opisuje fragment procesu produkcji pączków.

---

jąca na ocenie, jaki procent zadania został już wykonany, okazuje się bardzo zawodna, gdyż wykonawcy mają tendencję do zawyżania swoich tak wyrażanych ocen.




**Rys. 17.1 Uogólniona sieć Petriego**

Miejsce oznaczone jako surowce zawiera trzy różne znaczniki odpowiadające trzem różnym surowcom. Różne kolory reprezentują różne surowce, a różne wielkości — odpowiednie ilości tych surowców. W tym przypadku, aby tranzyt mógł zostać odpalony, surowce muszą być dostępne zarówno co do rodzajów, jak i co do ilości każdego z rodzajów.

Miejsce oznaczone jako surowe pączki zawiera wiele znaczników, z których każdy jest jednego z dwóch rodzajów: znaczniki czarne to pączki foremne, a czerwone, to pączki pogniecione. Aby odpalenie tranzytu smażenie pączków było możliwe, w jego miejscu wejściowym musi znajdować się co najmniej jeden czarny znacznik, a odpalenie powoduje zniknięcie tego znacznika.

Analogicznie, dla odpalenia tranzytu eliminacja zgniecionych, potrzebny jest znacznik czerwony.

Z miejsca surowe pączki wychodzą dwie strzałki, co do tej pory oznaczało niedeterminizm. Tym razem jednak sytuacja jest deterministyczna, gdyż zakładamy, że każdy znacznik niesie informację o stanie surowego pączka.

Inne przyjęte przez nas ograniczenie idei sieci Petriego, to stały i z góry znany czas wykonania tranzytu. W rzeczywistości taki czas zwykle nie jest z góry znany, a dodatkowo na tranzyt narzucane są ograniczenia typu: „odpalenie nie wcześniej (lub później) niż” i „zakończenie nie wcześniej (lub później) niż”. Rozważa się też sieci, w których czasy wykonania określane są z zadanym prawdopodobieństwem.

# 18 Stała zmienność

Ktoś kiedyś powiedział, że jedyne, co w dzisiejszych czasach jest stałe, to fakt, że wszystko się zmienia. Tak dzieje się też w projektach. Założone na początku czasy wykonania w miarę realizacji projektu ulegają zmianie, zasoby pojawiają się i znikają, pojawiają się też nowe zadania (tranzyty), których nie było w chwili rozpoczęcia projektu, lub znikają te, które początkowo były planowane. To wszystko powoduje, że pojawia się konieczność zarządzania zmianami. Czy sieci Petriego mogą przyczynić się do łatwiejszego zarządzania zmianami w projekcie? Pewne prace w tym kierunku zostały już poczynione.

# 19 Literatura

Niżej podana literatura obejmuje trzy pozycje poświęcone teorii Goldratta ([4][5][6]) i pozostałe poświęcone zastosowaniom sieci Petriego w różnych ich wariantach. Pozycji [7] raczej nie polecam do czytania — podałem ją jedynie jako referencję historyczną. Natomiast moi koledzy matematycy wskazują na pozycję [8], jako bardzo dobry wstęp do teorii sieci Petriego napisany dla nie-matematyków.

[1] Aalst Wil van der i Hee Kees van, *Workflow management: models, methods, and systems* (Cooperative Information Systems), MIT Press 2004

[2] Bause Falko i Kritzinger Pieter S. , *Stochastic Petri Nets (2nd Edition) — - An Introduction to the Theory*, Vieweg Verlag 2. Aufl. 2002, ISBN: 3-528-15535-3, do pobrania w formacie pdf na http://www4.cs.uni-dortmund.de/~Bause/spnbook2.html

[3] David René i Alla Hassane, *Discrete, continuous, and hybrid Petri Nets*, Springer-Verlag Berlin Heidelberg 2005

[4] Kowalczyk Marek, *Jak kończyć projekty na czas*, Mandarine Project Partners, prezentacja wykładu wygłoszonego na konwersatorium TQM w dniu 25 lutego 2010.

[5] Leach Lawrence P., *Critical Chain Project Management*, Artech House, Norwood 2005.

[6] Newbold Robert C., *Project Management in the Fast Lane — Applying the Theory of Constrains*, St . Lucie Press 1998.

[7] Petri Carl Adam, *Kommunikation mit Automaten*, Schriften des Institutes für Instrumentelle Mathematik, Bonn 1962

[8] Wolfgang Reisig, *A Primer in Petri Net Design,* Springer Compass International 1992