Zarządzanie konfiguracją oprogramowania

Wersja 2.0.0beta1

Zagadnienia zarządzania konfiguracją

- Zarządzanie zmianą
- Zarządzanie wersjami
- Budowanie systemu
- Zarządzanie wydaniami

Zarządzanie konfiguracją (Configuration Management - CM)

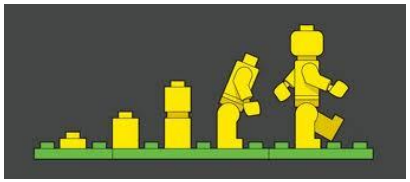
- Oprogramowanie podlega ciągłym zmianom
  - System może być postrzegany jako zbiór wersji, z których każda musi być zarządzana i wspierana.
  - Wersje reprezentując implementacje zatwierdzonych propozycji zmian, adaptacji dla różnego typu sprzętu czy systemów operacyjnych
  - Zarządzanie konfiguracją związane jest ze strategią, procesami oraz narzędziami pozwalającymi na zarządzanie zmieniającym się systemem.
  - Bez zarządzania konfiguracją trudno byłoby odpowiedzieć na takie pytania jak:
    - Jakie zmiany zostały wprowadzone w tej wersji systemu
    - Jakie wersje komponentów są elementami tej wersji systemu

Czynności Zarządzania Konfiguracją (1 z 4)

- **Zarządzanie zmianą**
  - Śledzenie zgłoszeń dotyczących potrzeby zmiany oprogramowania (od klientów, deweloperów, ...).
  - Określanie kosztów oraz wpływu zmiany.
  - Podejmowanie decyzji o implementacji lub odrzuceniu zmiany

Czynności Zarządzania Konfiguracją (2 z 4)

- **Zarządzanie wersjami**
  - Śledzenie zmieniających się wersji komponentów systemu
  - Zapewnianie środowiska, w którym praca różnych deweloperów nie interferuje ze sobą



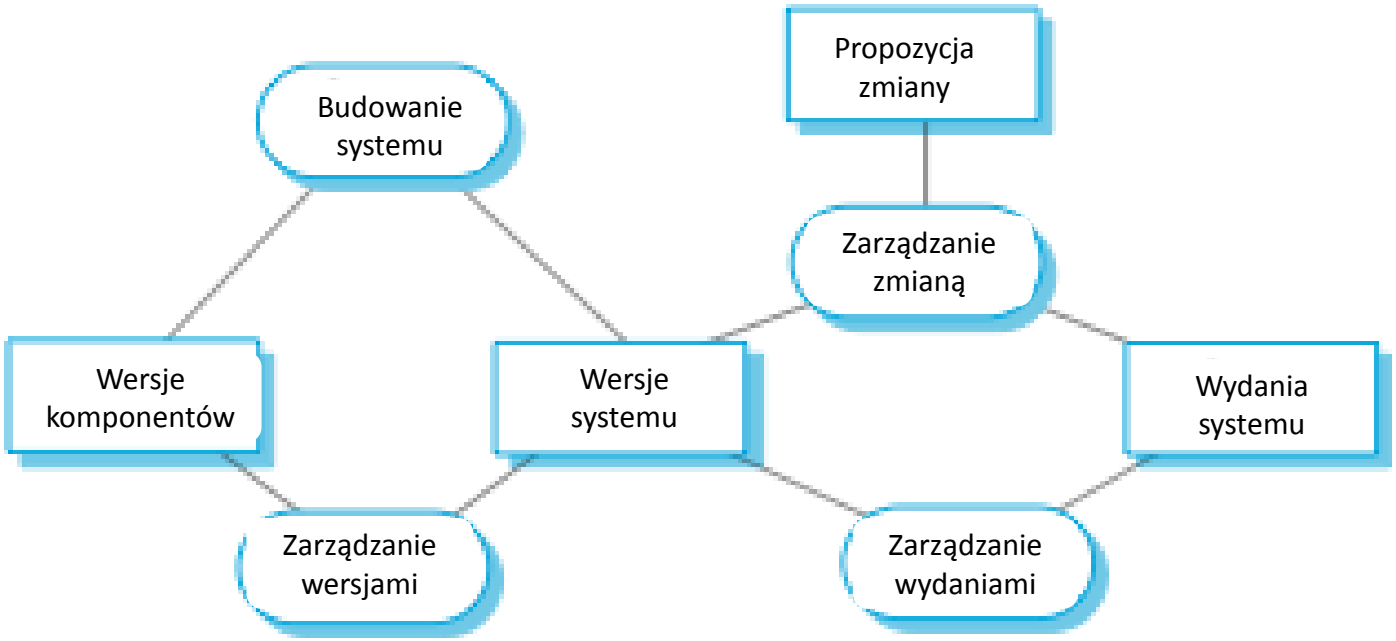
Czynności Zarządzania Konfiguracją (3 z 4)

- **Budowanie systemu**
  - Proces łączenia komponentów programu, danych, bibliotek do postaci wynikowego systemu
  - Konwersja kodu do postaci wynikowej

Czynności Zarządzania Konfiguracją (4 z 4)

- **Zarządzanie wydaniami**
  - Przygotowanie oprogramowania do wydania zewnętrznego
  - Śledzenie wersji systemu, które zostały dostarczone klientom.

Czynności Zarządzania Konfiguracją



Terminologia zarządzania konfiguracją - 1

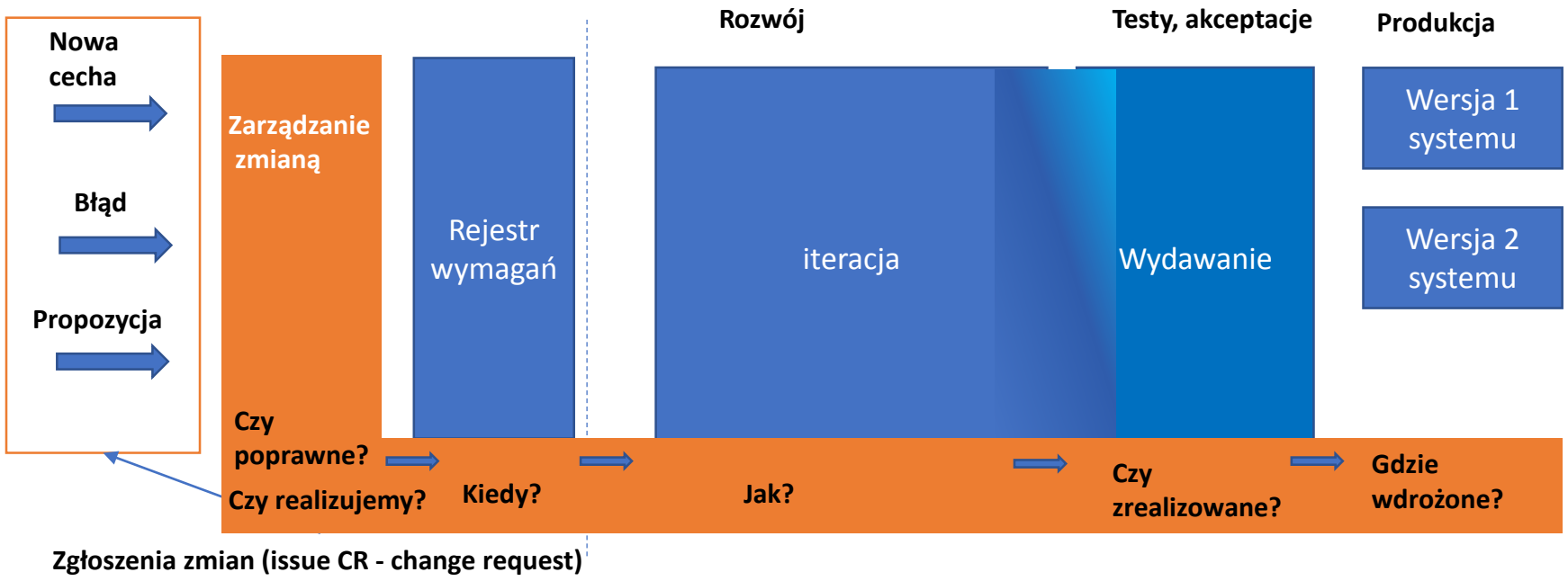
Termin	Wyjaśnienie
Jednostka konfiguracji lub jednostka konfiguracji oprogramowania (ang. configuration item - **CI** software configuration item - **SCI**)	Każdy element powiązany z wytwarzaniem oprogramowania (projekt, kod źródłowy, dane testowe, dokument, itp.) który został poddany kontroli konfiguracji. Jednostki konfiguracji mogą istnieć w różnych wersjach. Każda jednostka konfiguracji ma unikatową nazwę.
Kontrola konfiguracji (ang. Configuration control)	Proces zapewniania, że wersja systemu oraz komponentów została zapisana i jest zarządzana. Oznacza to, że każda wersja systemu oraz jego komponentów jest identyfikowalna i zapisana w sposób trwały (dostępna przez cały czas życia systemu).
Wersja (ang. Version)	Instancja jednostki konfiguracji, różniąca się od innych instancji tej samej jednostki. Wersja zawsze ma unikatowy identyfikator (np. Nazwa jednostki konfiguracji + numer wersji).
Linia bazowa (ang. Baseline)	Zbiór wersji komponentów tworzących system. Linia bazowa jest kontrolowana – wersje komponentów nie mogą zostać zmienione. Oznacza to, że istnieje możliwość ponownego utworzenia linii bazowej z odpowiednich wersji komponentów.
Linia kodu (ang. Codeline)	Zbiór wersji komponentów oprogramowania oraz innych jednostek konfiguracji, od których zależny jest dany komponent.

Terminologia zarządzania konfiguracją - 2

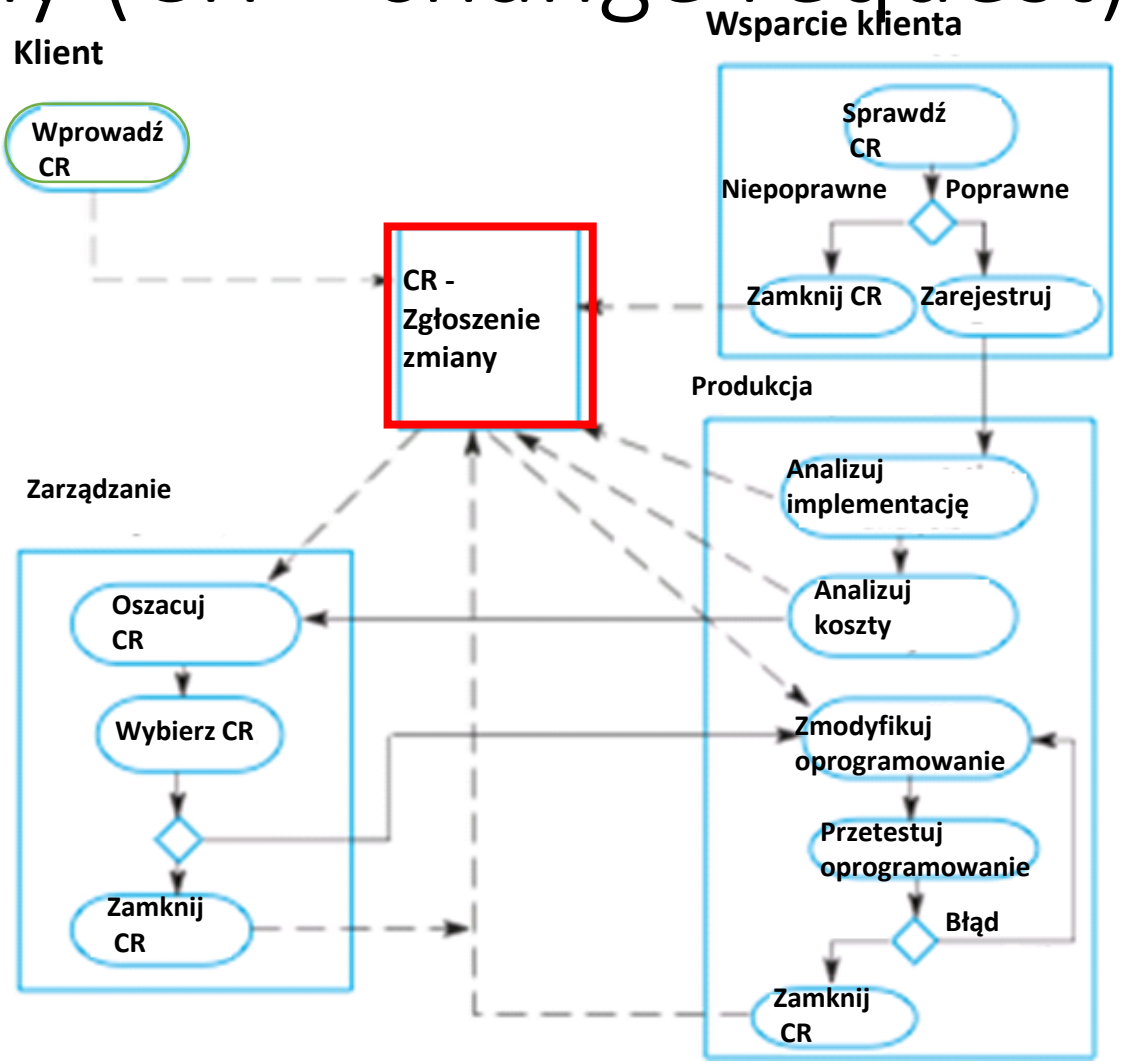
Termin	Wyjaśnienie
Linia główna/rozwojowa (ang. Mainline)	Sekwencja linii bazowych reprezentujących różne wersje systemu.
Wydanie (ang. Release)	Wersja systemu, która została udostępniona do użytku klientom (lub innym użytkownikom w ramach organizacji).
Przestrzeń robocza (ang. Workspace)	Prywatna przestrzeń, w obrębie której dokonanie modyfikacji oprogramowania nie ma wpływu na pracę innych deweloperów, który również pracują na oprogramowaniem.
Utworzenie gałęzi (ang. Branching)	Powołanie do życia nowej linii kodu (ang. codeline) na podstawie wybranej wersji w istniejącej linii. Tak utworzona linia może być rozwijana niezależnie.
Łączenie (ang. Merging)	Utworzenie nowej wersji komponentu oprogramowania poprzez połączenie oddzielnych wersji znajdujących się w różnych liniach kodu. Linie te mogły powstać poprzez uprzednie utworzenie gałęzi.
Budowanie systemu (ang. System building)	Utworzenie wykonywalnego systemu poprzez skompilowanie i połączenie odpowiednich wersji komponentów oraz bibliotek wchodzących w skład systemu.

Zarządzanie zmianą oprogramowania

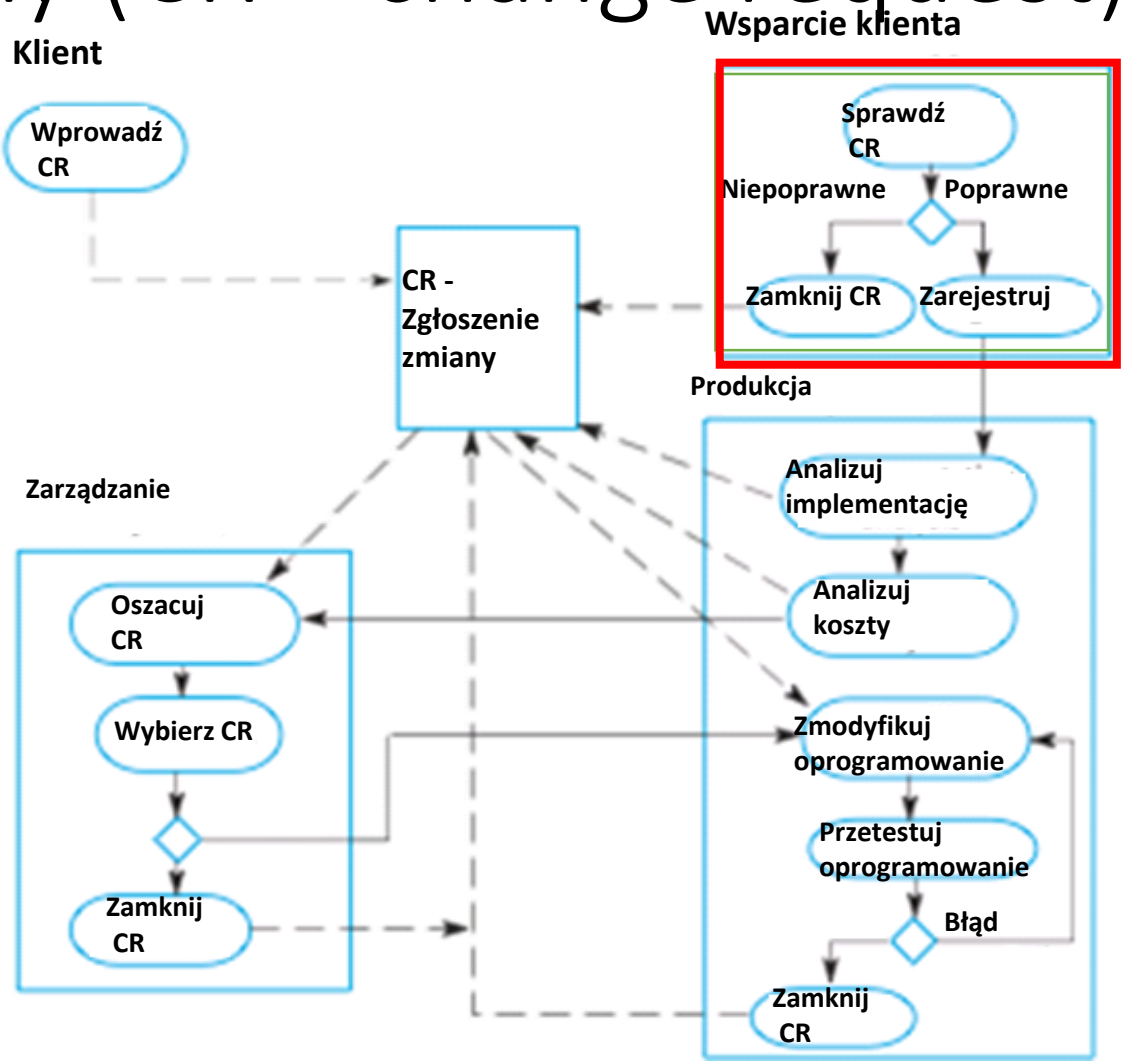
Zarządzanie zmianą za pomocą zgłoszeń



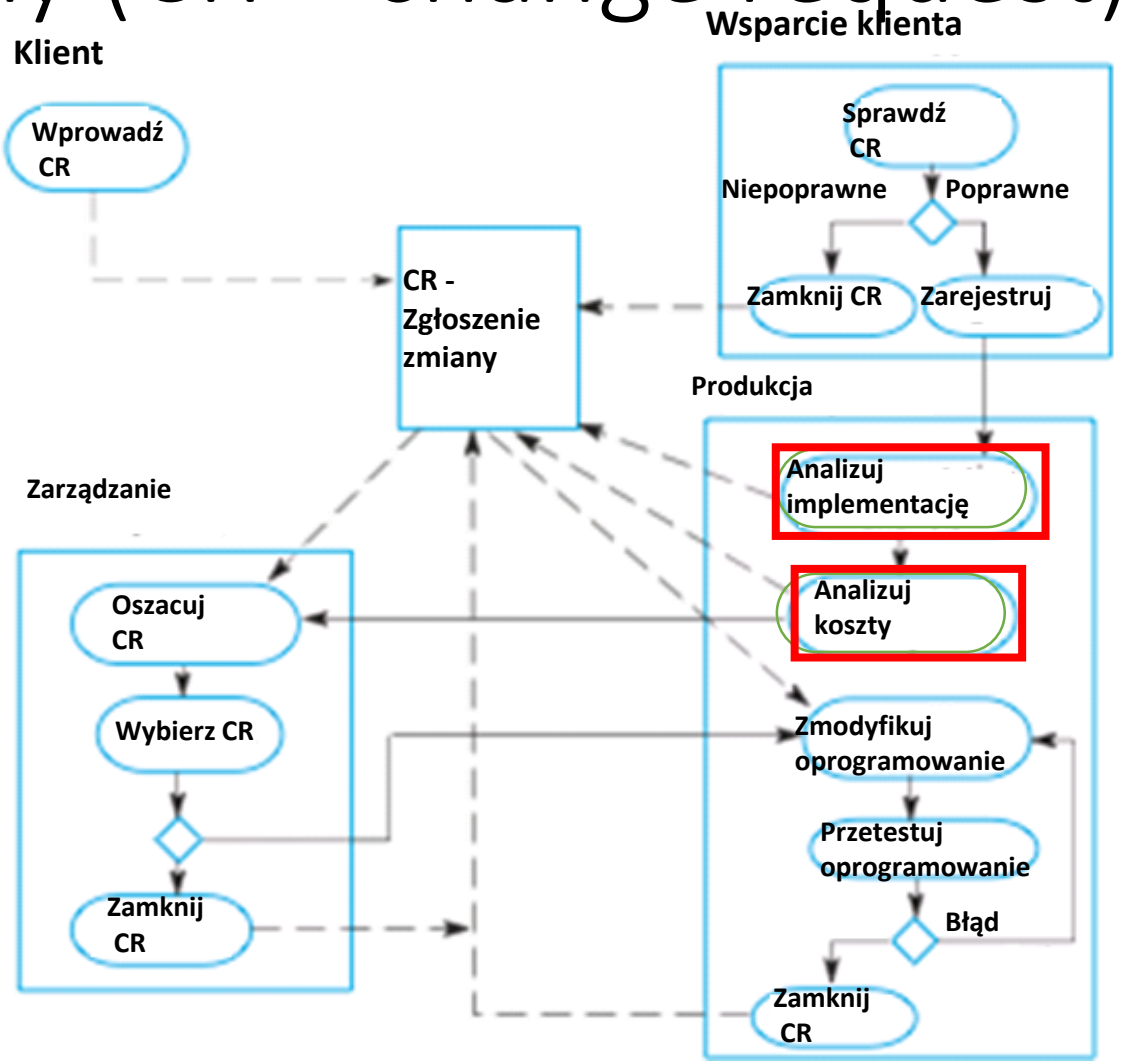
Proces zarządzania zgłoszeniem zmiany (CR – change request) (1)



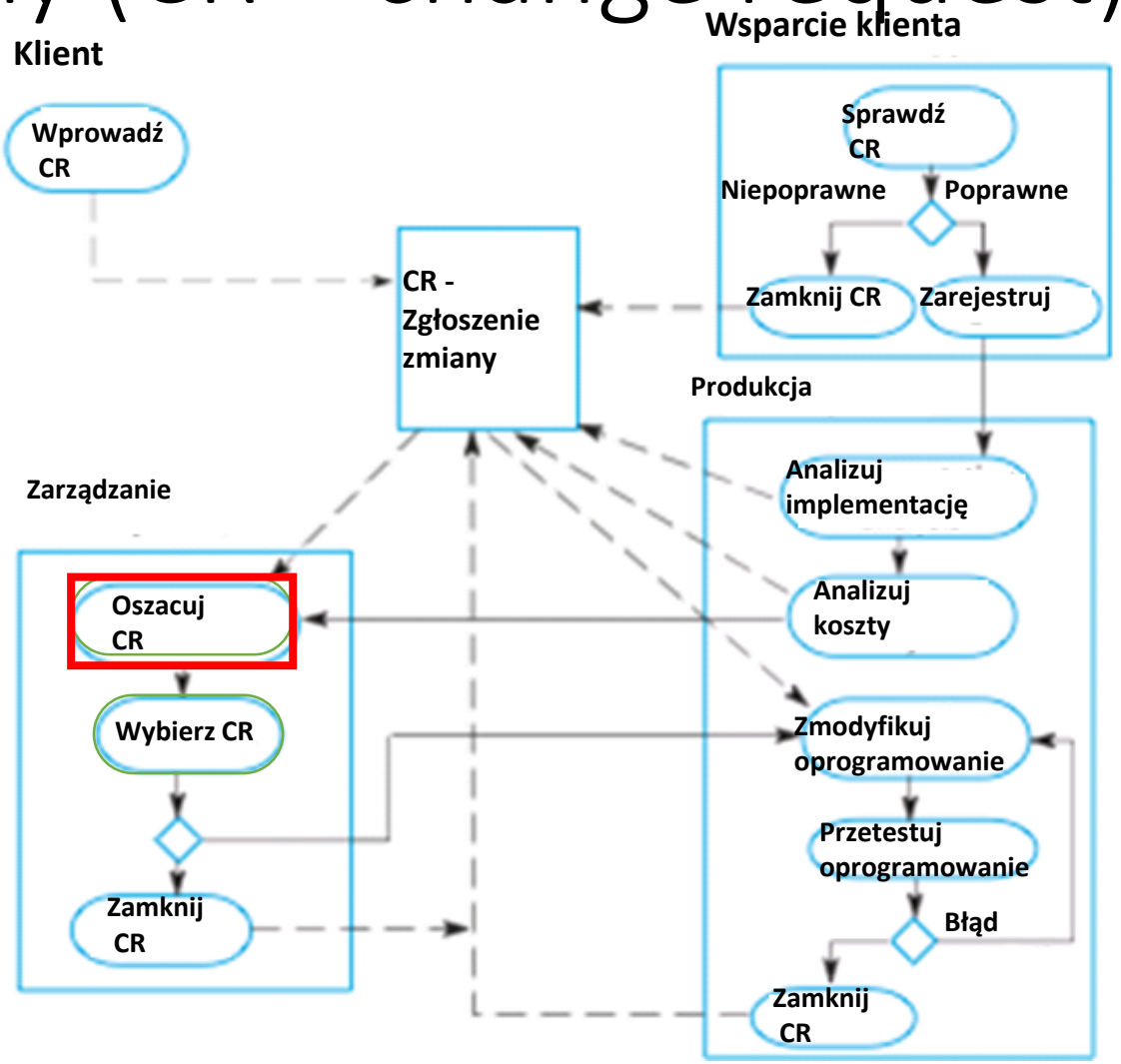
Proces zarządzania zgłoszeniem zmiany (CR – change request) (2)



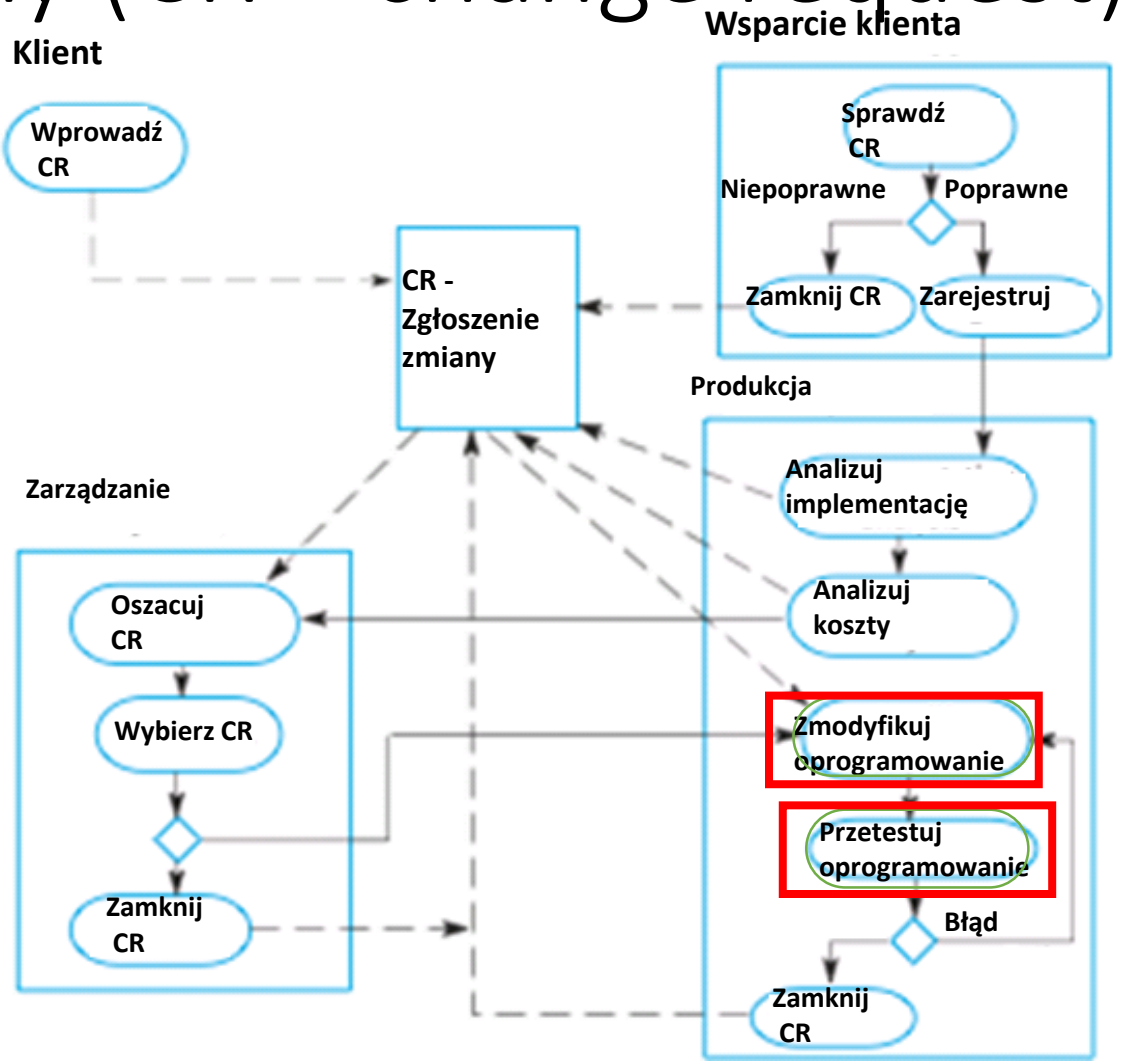
Proces zarządzania zgłoszeniem zmiany (CR – change request) (3)



Proces zarządzania zgłoszeniem zmiany (CR – change request) (4)



Proces zarządzania zgłoszeniem zmiany (CR – change request) (5)



Formularz zgłoszenia zmiany

Change Request Form

Project: SICSA/AppProcessing **Number:** 23/02
Change requester: I. Sommerville **Date:** 20/01/09
Requested change: The status of applicants (rejected, accepted, etc.) should be shown visually in the displayed list of applicants.

Change analyzer: R. Looek **Analysis date:** 25/01/09
Components affected: ApplicantListDisplay, StatusUpdater

Associated components: StudentDatabase

Change assessment: Relatively simple to implement by changing the display color according to status. A table must be added to relate status to colors. No changes to associated components are required.

Change priority: Medium
Change implementation:
Estimated effort: 2 hours
Date to SGA app. team: 28/01/09 **CCB decision date:** 30/01/09
Decision: Accept change. Change to be implemented in Release 1.2
Change implementor: **Date of change:**
Date submitted to QA: **QA decision:**
Date submitted to CM:
Comments:

JDK 8 Update 151 - JavaFX - setOnAction - Crash in glass.dll when triggered

Details

Type:	Bug	Status:	CLOSED
Priority:	P3	Resolution:	Won't Fix
Affects Version/s:	8u151	Fix Version/s:	8-pool
Component/s:	javafx		
Labels:	10-na 9-na dcsprm regression reproducer-hard sustaining webbug		
Subcomponent:	graphics		
CPU:	x86		
OS:	windows_xp		

Description

FULL PRODUCT VERSION :
I have reproduced it with both 1.8.0_151 and 1.8.0_152 (both 32-bit). The problem does NOT occur on 1.8.0_131, 1.8.0_141, or 1.8.0_144.

FULL OS VERSION :
Microsoft Windows XP [Version 5.1.2600]

EXTRA RELEVANT SYSTEM CONFIGURATION :
The user of my program who reported the problem was running an AMD CPU with an ATI graphics card; I have reproduced it on an Intel/nVIDIA configuration. I do not believe it to be GPU-dependent.

A DESCRIPTION OF THE PROBLEM :
With Java 8 Update 151 (but not previous updates) on Windows XP, the setOnAction method in many JavaFX classes (ButtonBase

People

Assignee:	Unassigned
Reporter:	Webbug Group
Votes:	0 Vote for this issue
Watchers:	6 Start watching this issue

Dates

Created:	2017-11-21 14:29
Updated:	2018-04-01 21:54
Resolved:	2017-12-04 02:41

Czynniki brane pod uwagę w oszacowaniu zmiany

1. Konsekwencje nie wprowadzenia zmiany
2. Zyski wynikające z wprowadzenia zmiany
3. Liczba użytkowników, na których zmiana będzie miała wpływ
4. Koszt wprowadzenia zmiany
5. Wpływ zmiany na cykl wydań produktu

Zarządzanie zmianą w metodykach zwinnych (ang. agile)

- W niektórych metodykach klienci są bezpośrednio zaangażowani w zarządzanie zmianą (np. XP).
- Proponują oni zmianę w wymaganiach a następnie współpracują z zespołem w celu określenia wpływu zmiany oraz decydują czy zmiana powinna być bardziej priorytetowa niż cechy zaplanowane w kolejnym przyroście (inkrementacji) systemu.
- Zmiany związane udoskonalaniem systemu są po stronie programistów.
- Refaktoryzacja, w ramach której oprogramowanie jest udoskonalane w sposób ciągły, nie jest postrzegana jako dodatkowe obciążenie, ale jako niezbędna część procesu rozwoju systemu.

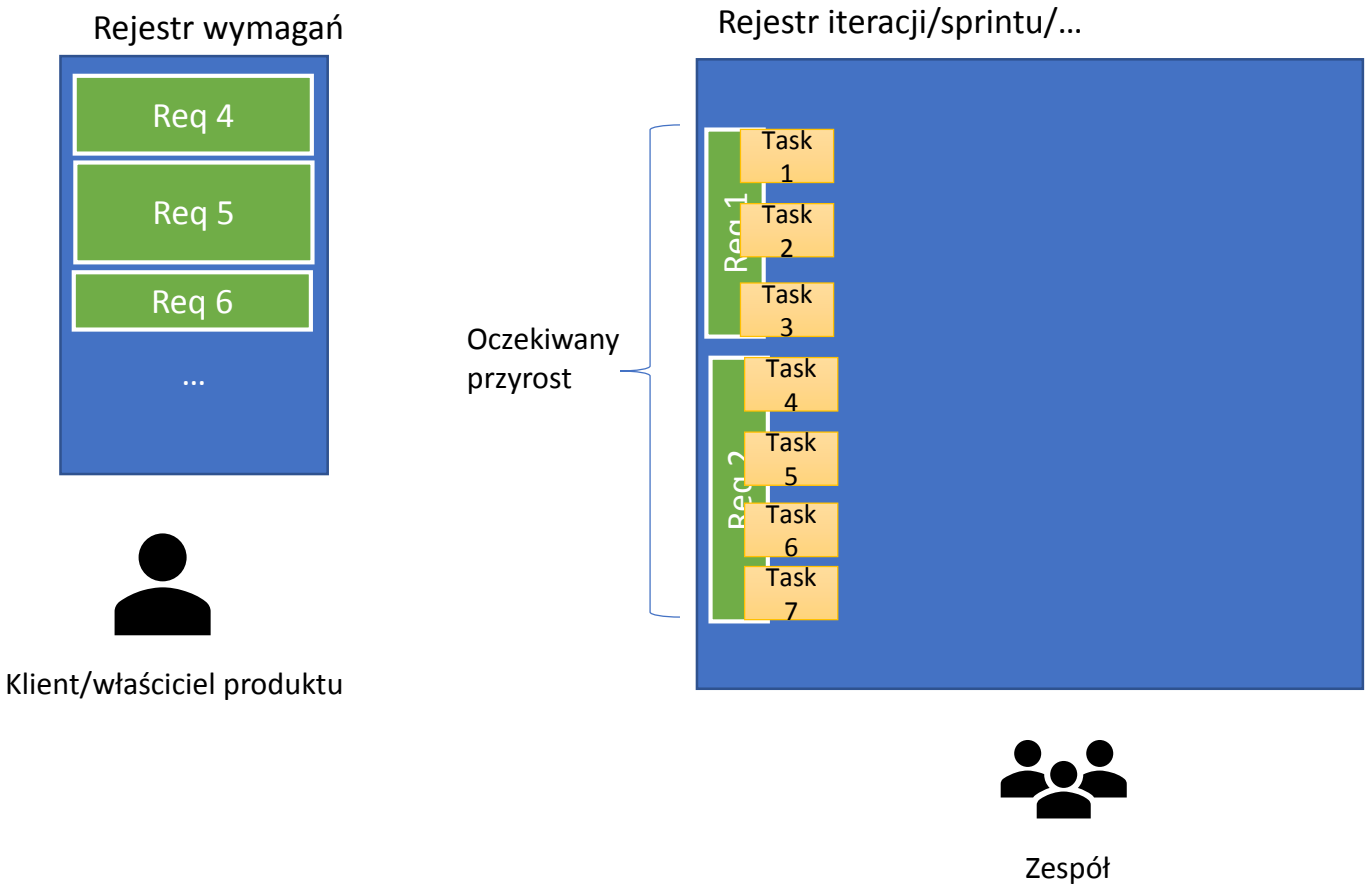
Narzędzia zarządzania zmianą

- Systemy śledzenia zgłoszeń
  - ***bug/defect tracking, issue-tracking systems***
  - Bugzilla, Trac, Redmine, JIRA, Zentrac, Pivotal Tracker, FogBugz, Lighthouse

Zarządzanie zmianą a zarządzanie wersjami

Rejestry

„Własność” rejestrów



Początek iteracji

Rejestr wymagań

Req 4

Req 5

Req 6

...

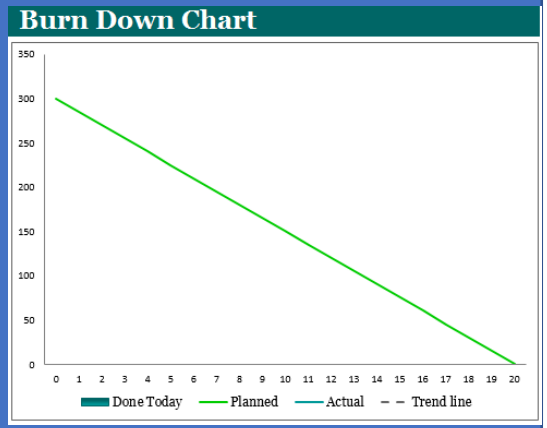
Klient/właściciel produktu

Rejestr iteracji/sprintu/...

Todo | Doing | Done

Req 1: Task 1, Task 2, Task 3

Req 2: Task 4, Task 5, Task 6, Task 7



Zespół

W trakcie iteracji

Rejestr wymagań

Req 5

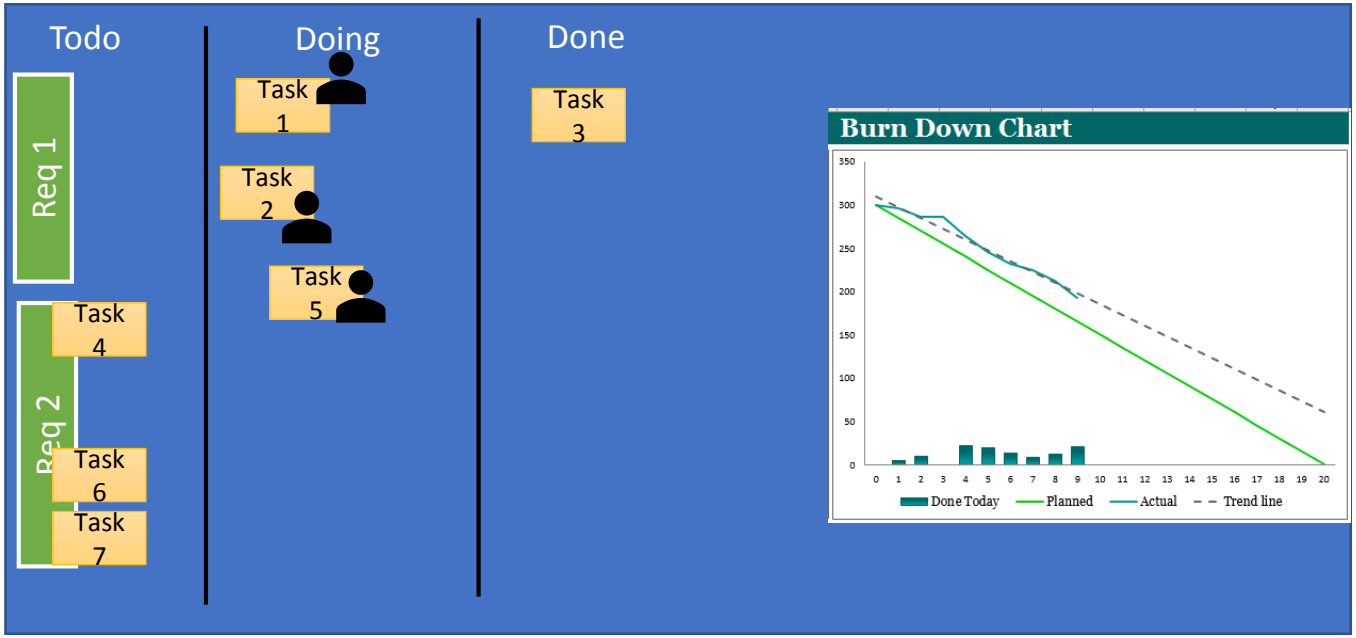
Req 7

Req 6

...

Klient/właściciel produktu

Rejestr iteracji/sprintu/...



Koniec iteracji

Rejestr wymagań

Req 5

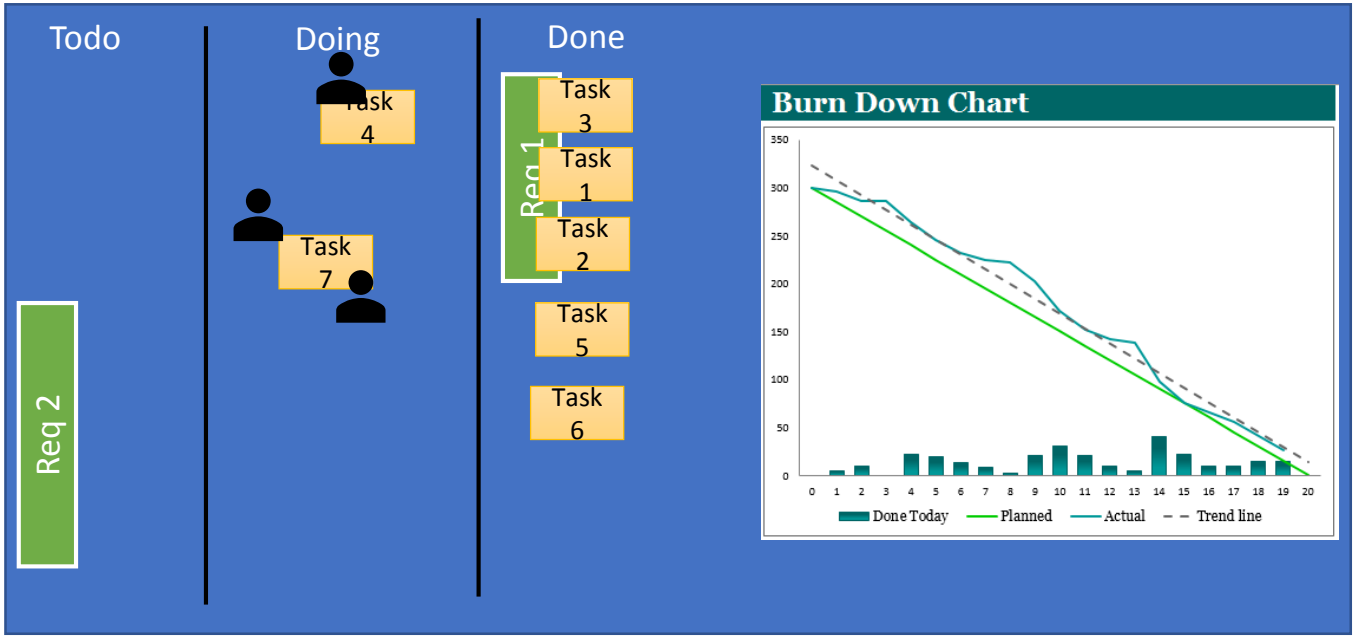
Req 7

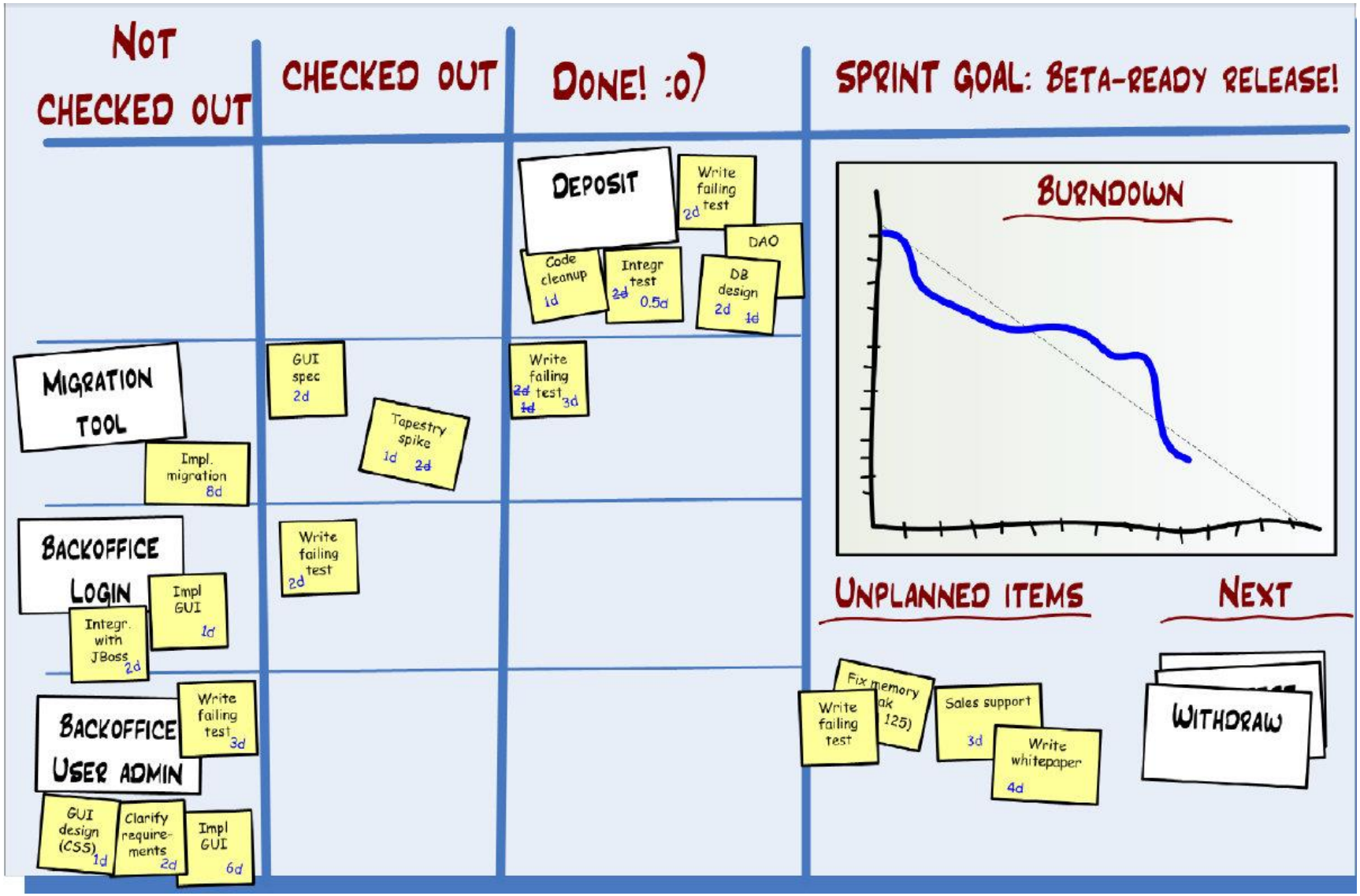
Req 6

...

Klient/właściciel produktu

Rejestr iteracji/sprintu/...





Źródło: http://www.infoq.com/minibooks/scrum-xp-from-the-trenches

Realizacja zadań w ramach iteracji

- Zespół

  +

- Środowisko
  - Kontrola wersji
    - zmiany wersji kodu źródłowego
  - Budowanie oprogramowania
    - całościowy efekt pracy – paczka (ang. build)
  - Ciągła integracja
    - Zmiana wersji kodu -> paczka

Zarządzanie wersjami

… jednostek konfiguracji oprogramowania
(na przykładzie systemu Git)

Zarządzanie wersjami

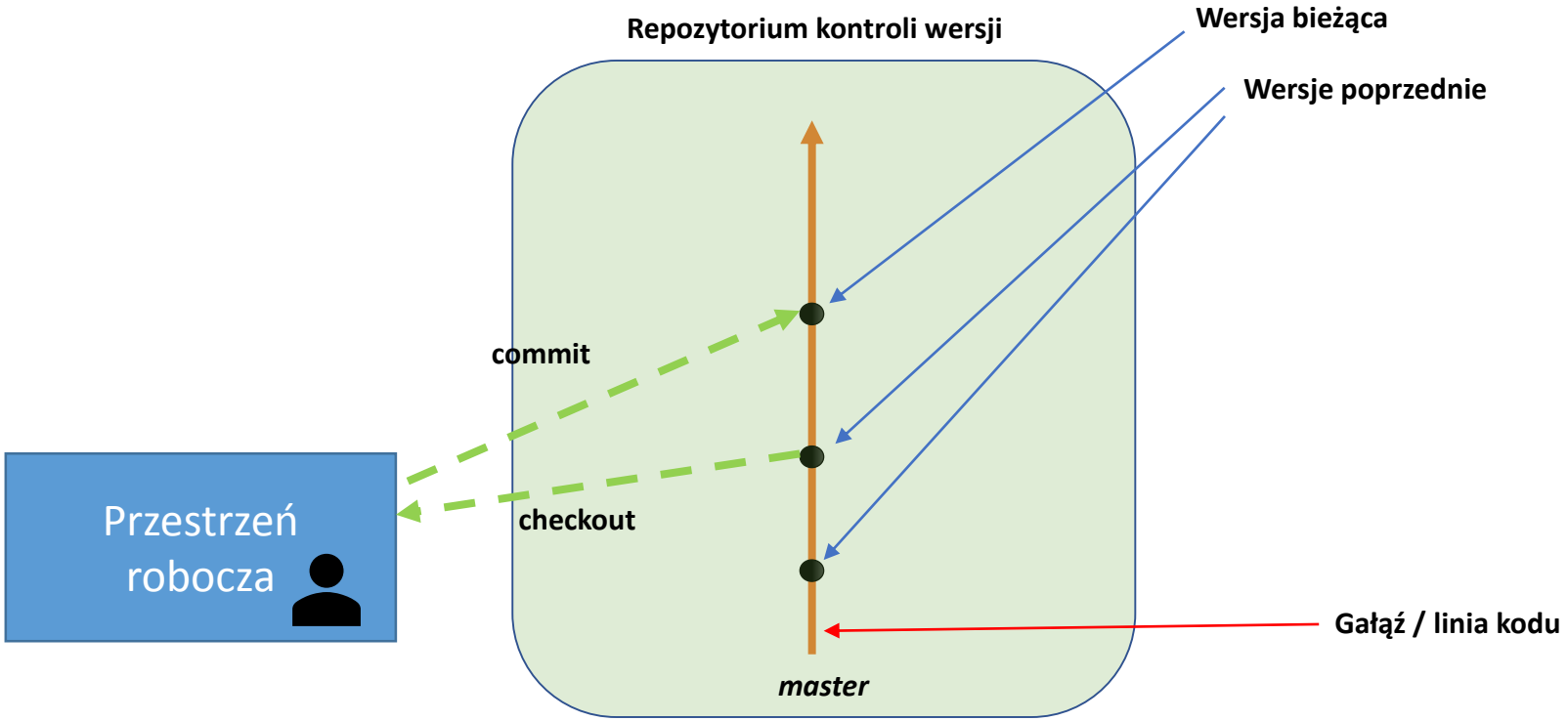
- Proces śledzenia różnych wersji komponentów oprogramowania lub innych jednostek konfiguracji oraz systemów, w których zostały wykorzystane.

- Zapewnienie, że zmiany dokonane w tych samych komponentach przez różnych deweloperów nie wpływają na siebie nawzajem.

- Proces zarządzania wersjami jest procesem zarządzania liniami kodu oraz liniami bazowymi.

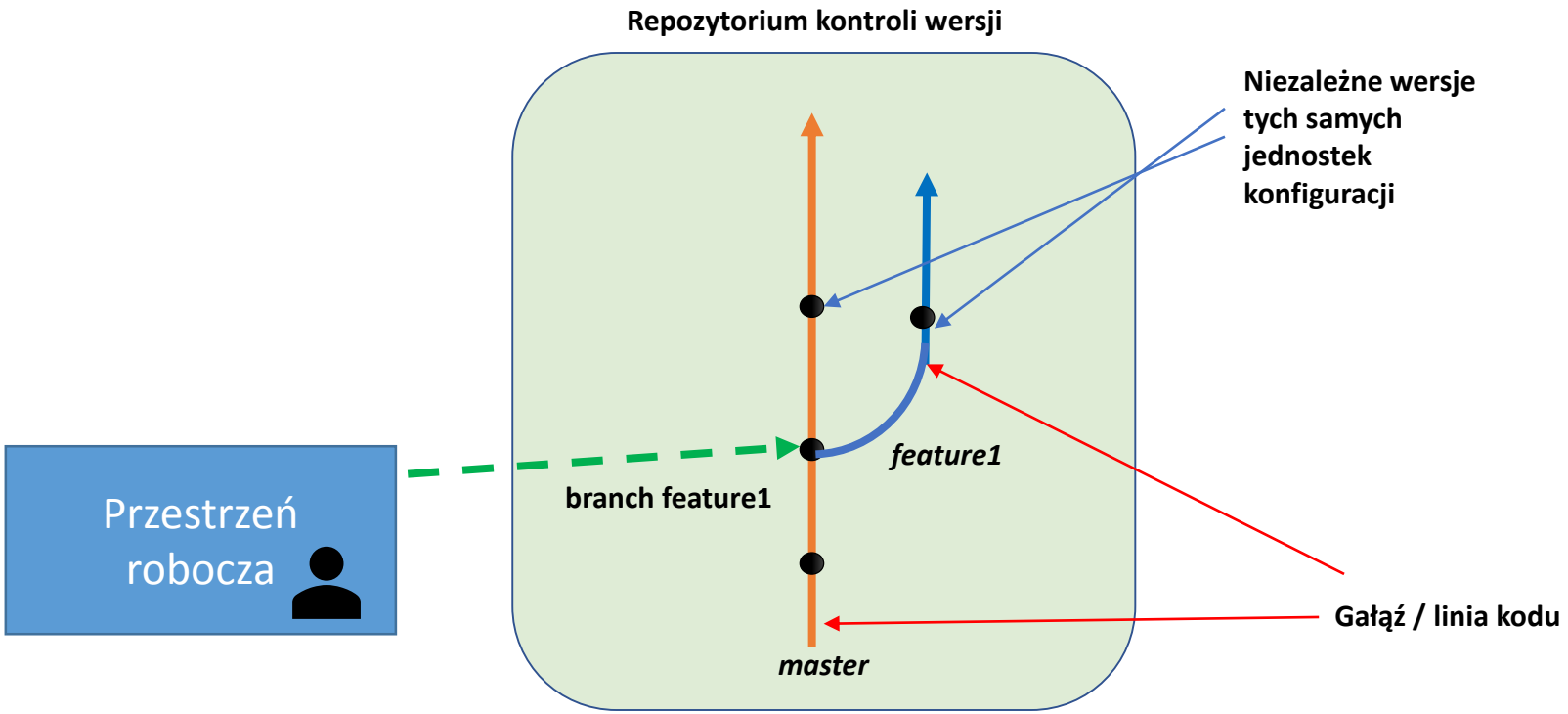
System kontroli wersji

- W systemie kontroli wersji jednostki konfiguracji oprogramowania wersjonowane są w ramach linii kodu reprezentowanych przez gałęzie (ang. branch):



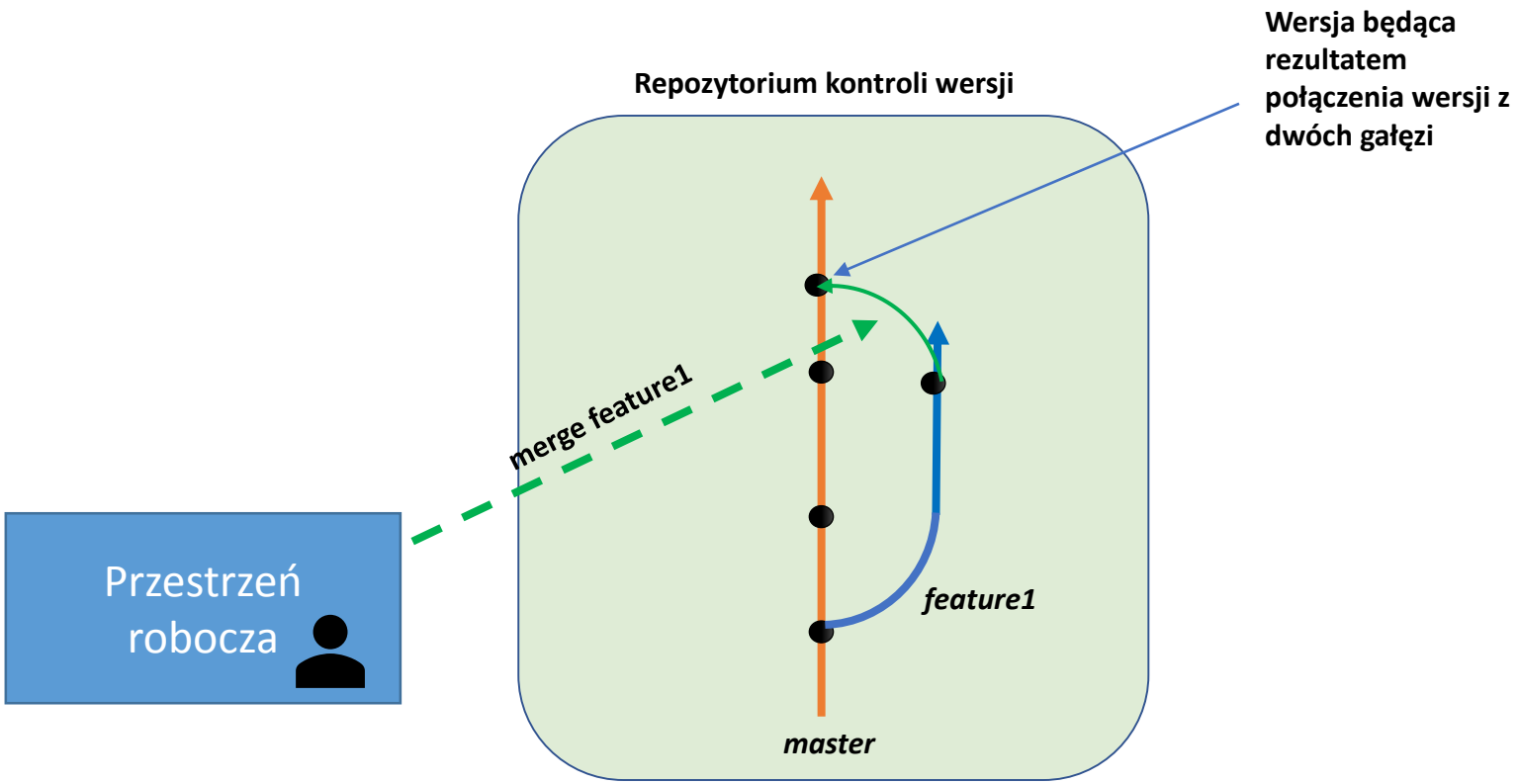
Kontrola wersji

- Gałęzie mogą być tworzone :

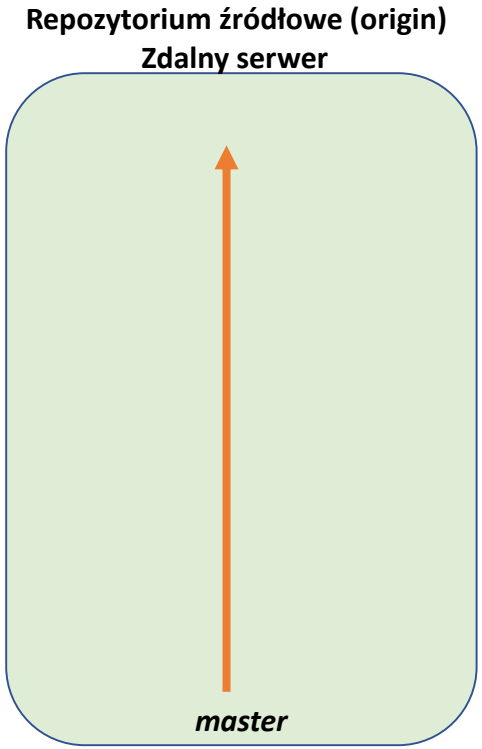


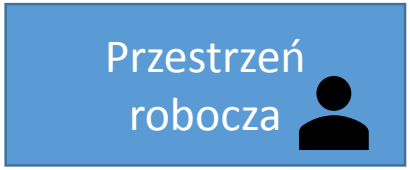
Kontrola wersji

- i łączone (ang. merge):



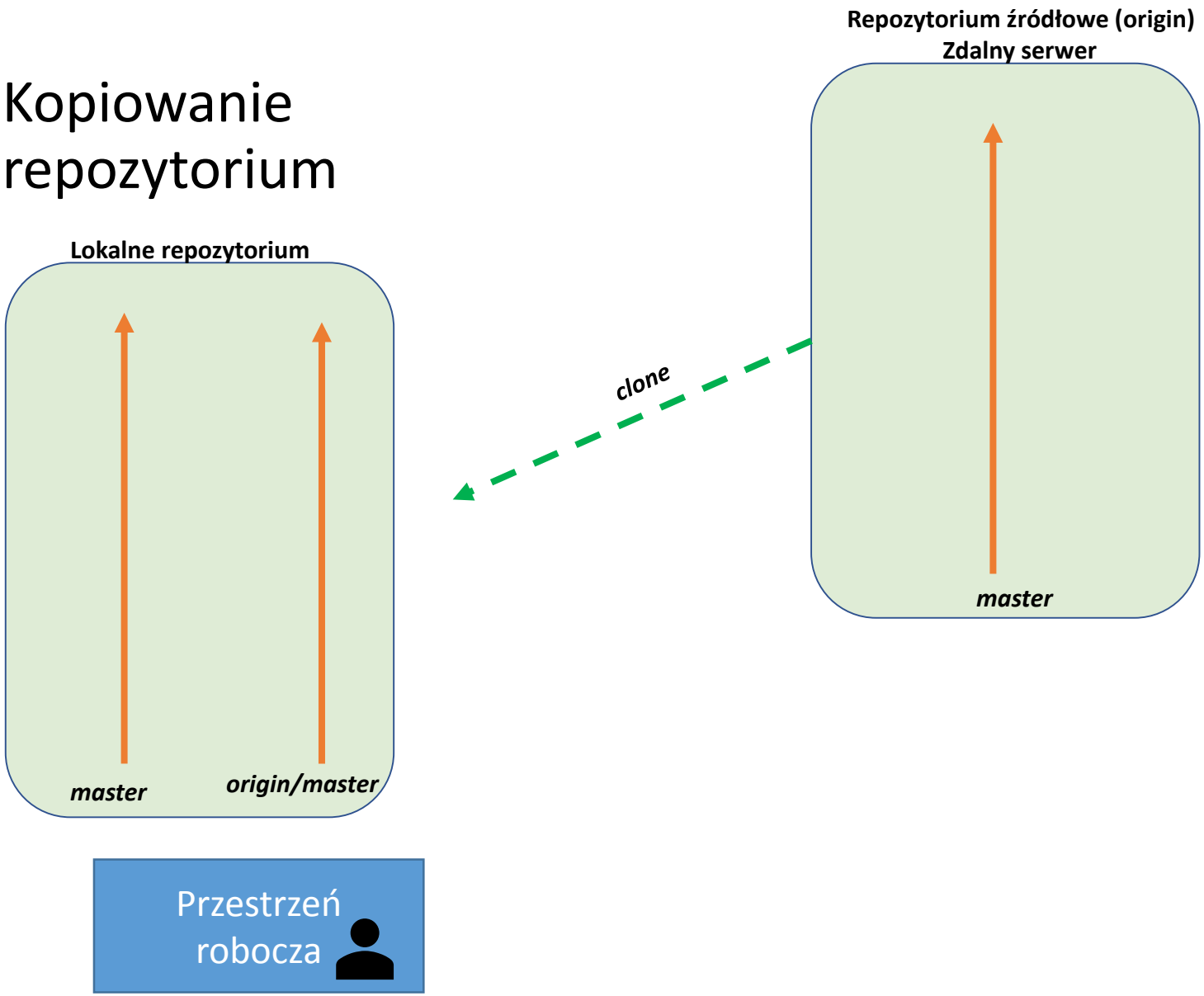
Rozproszona kontrola wersji





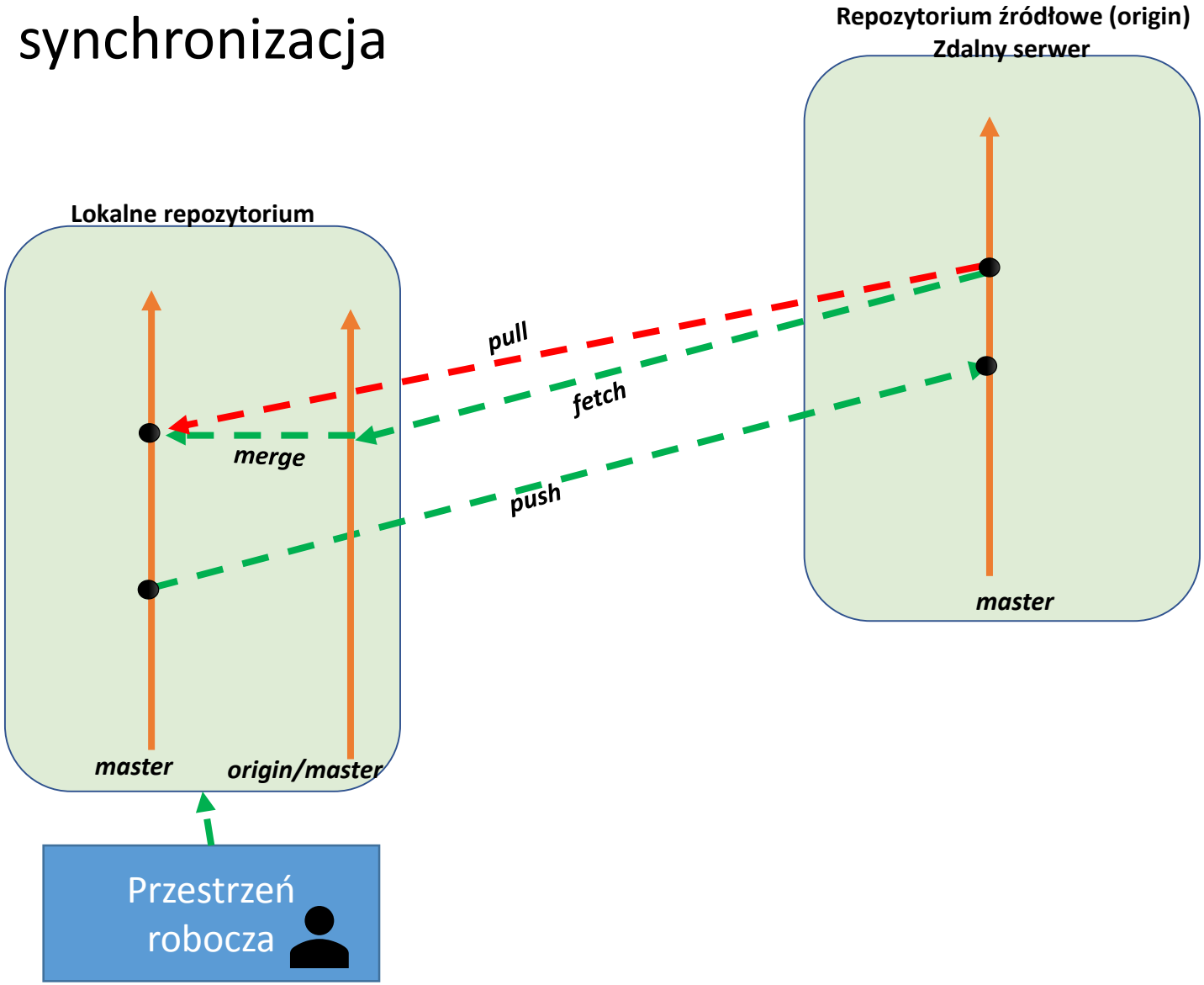
Rozproszona kontrola wersji

- Kopiowanie repozytorium



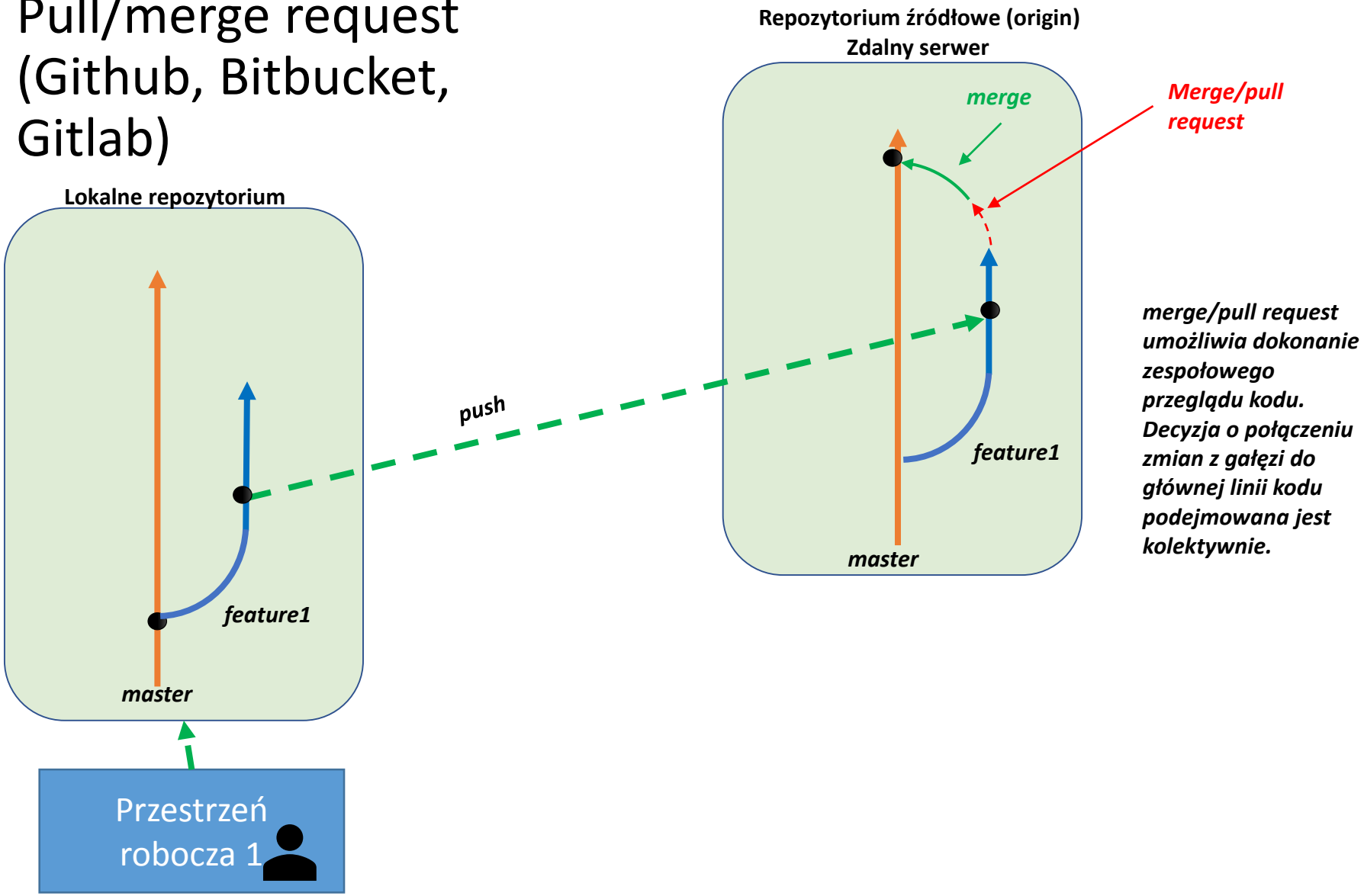
Rozproszona kontrola wersji

- synchronizacja



Łączenie zmian i przeglądy kodu

- Pull/merge request (Github, Bitbucket, Gitlab)

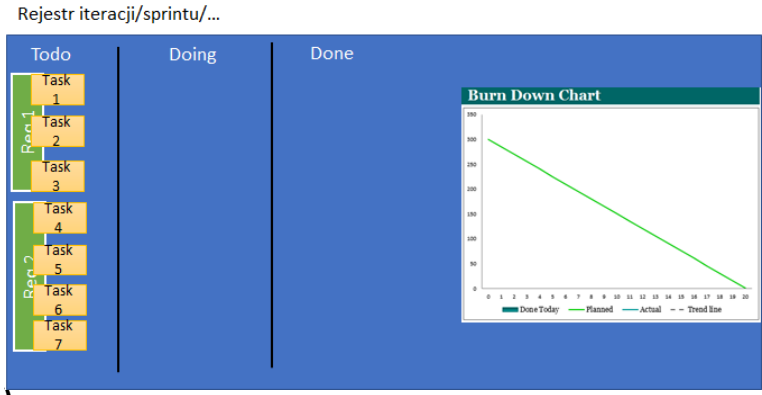


merge/pull request umożliwia dokonanie zespołowego przeglądu kodu. Decyzja o połączeniu zmian z gałęzi do głównej linii kodu podejmowana jest kolektywnie.

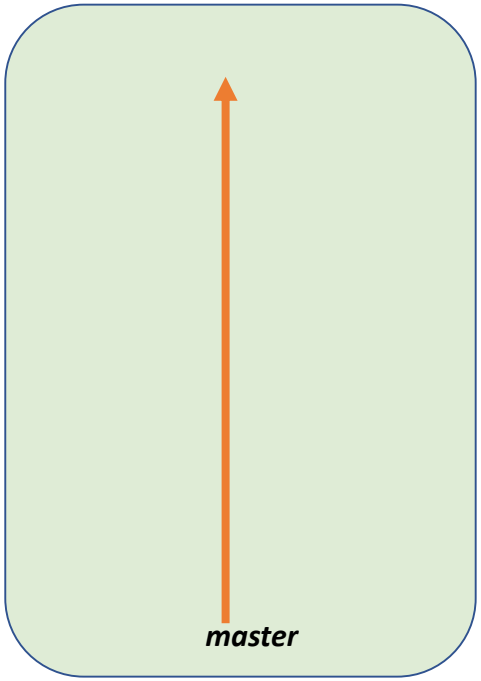
Realizacja przyrostowa a model pracy z repozytorium

Feature branch

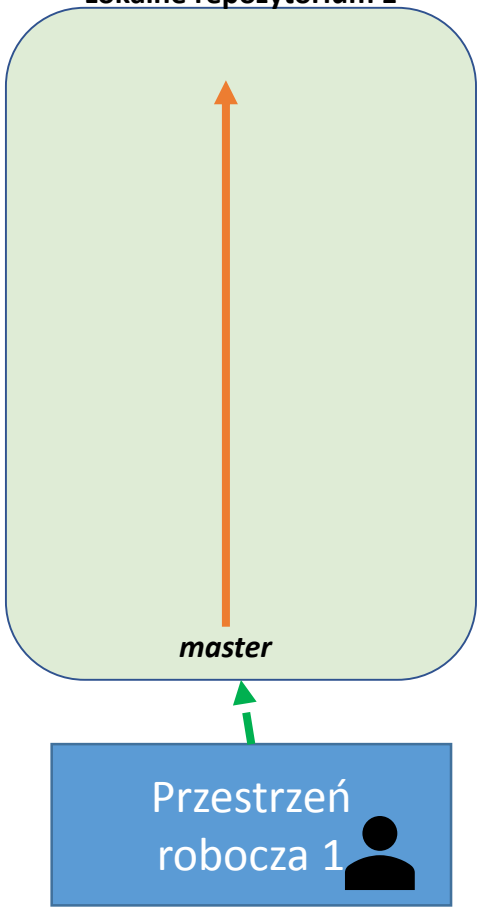
Feature branch



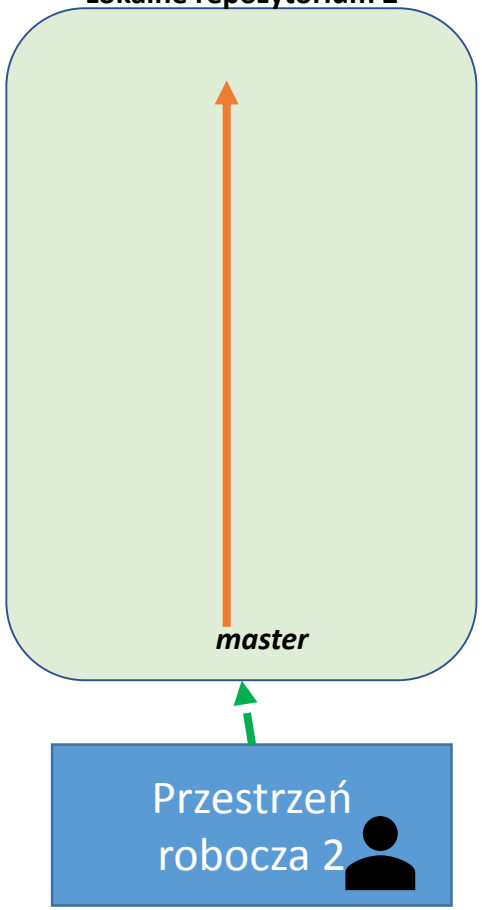
Repozytorium źródłowe (origin)
Zdalny serwer



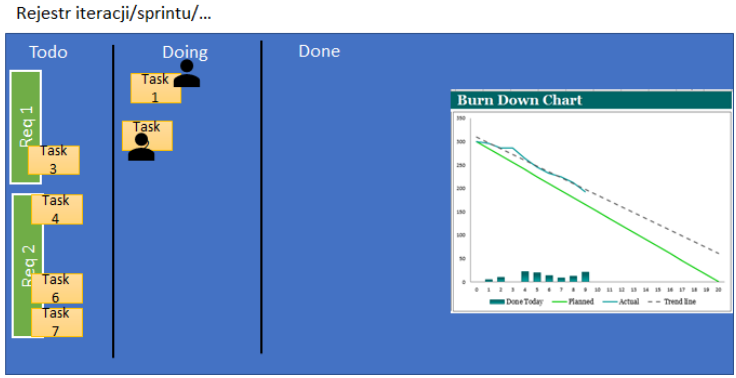
Lokalne repozytorium 1



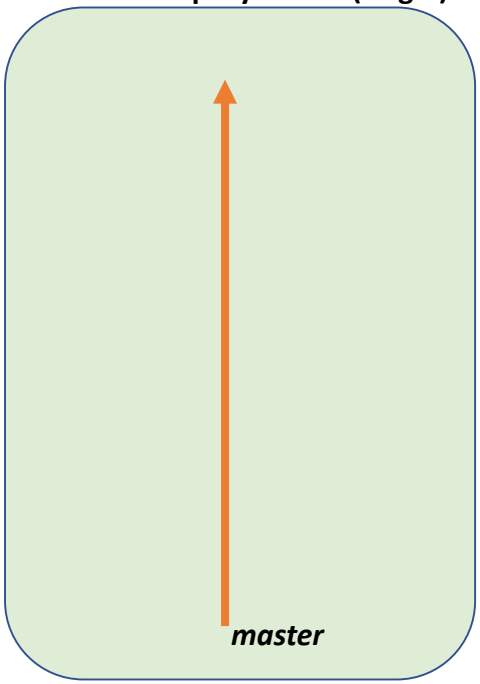
Lokalne repozytorium 2



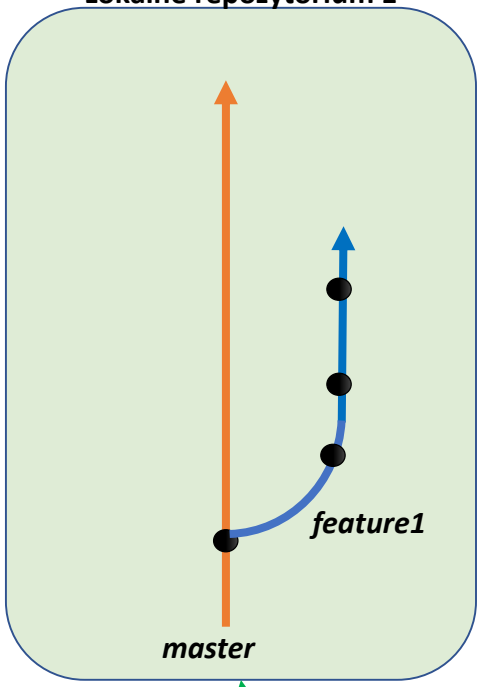
Feature branch – praca nad zadaniem

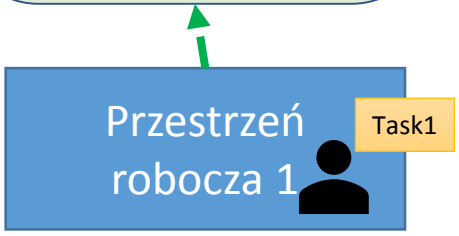


Centralne repozytorium (origin)

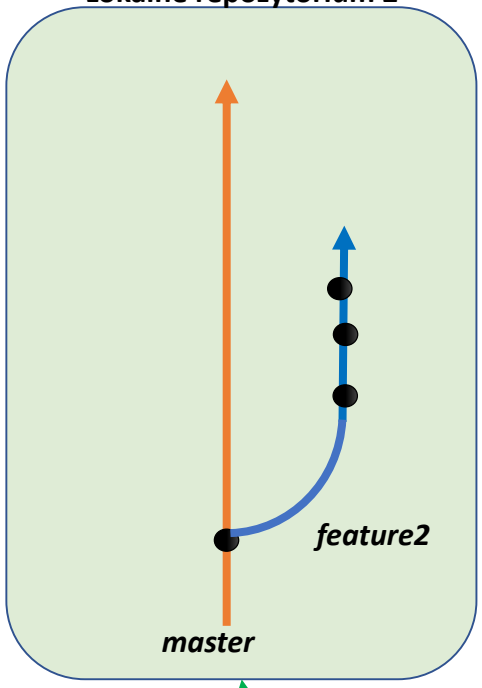


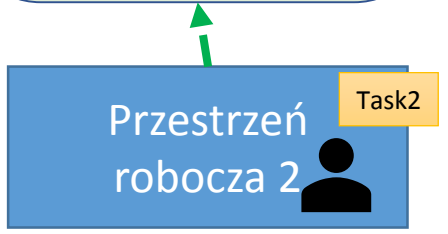
Lokalne repozytorium 1



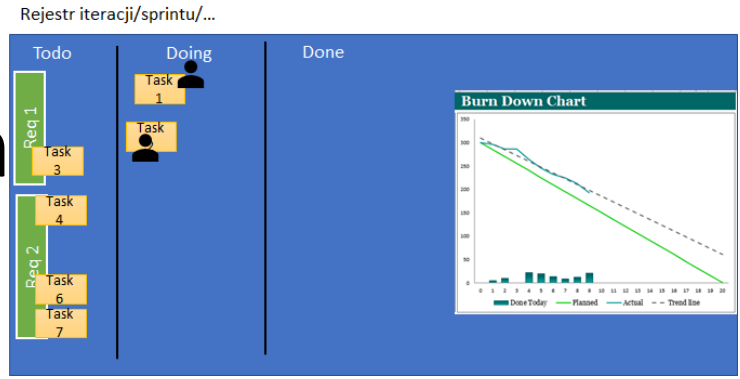


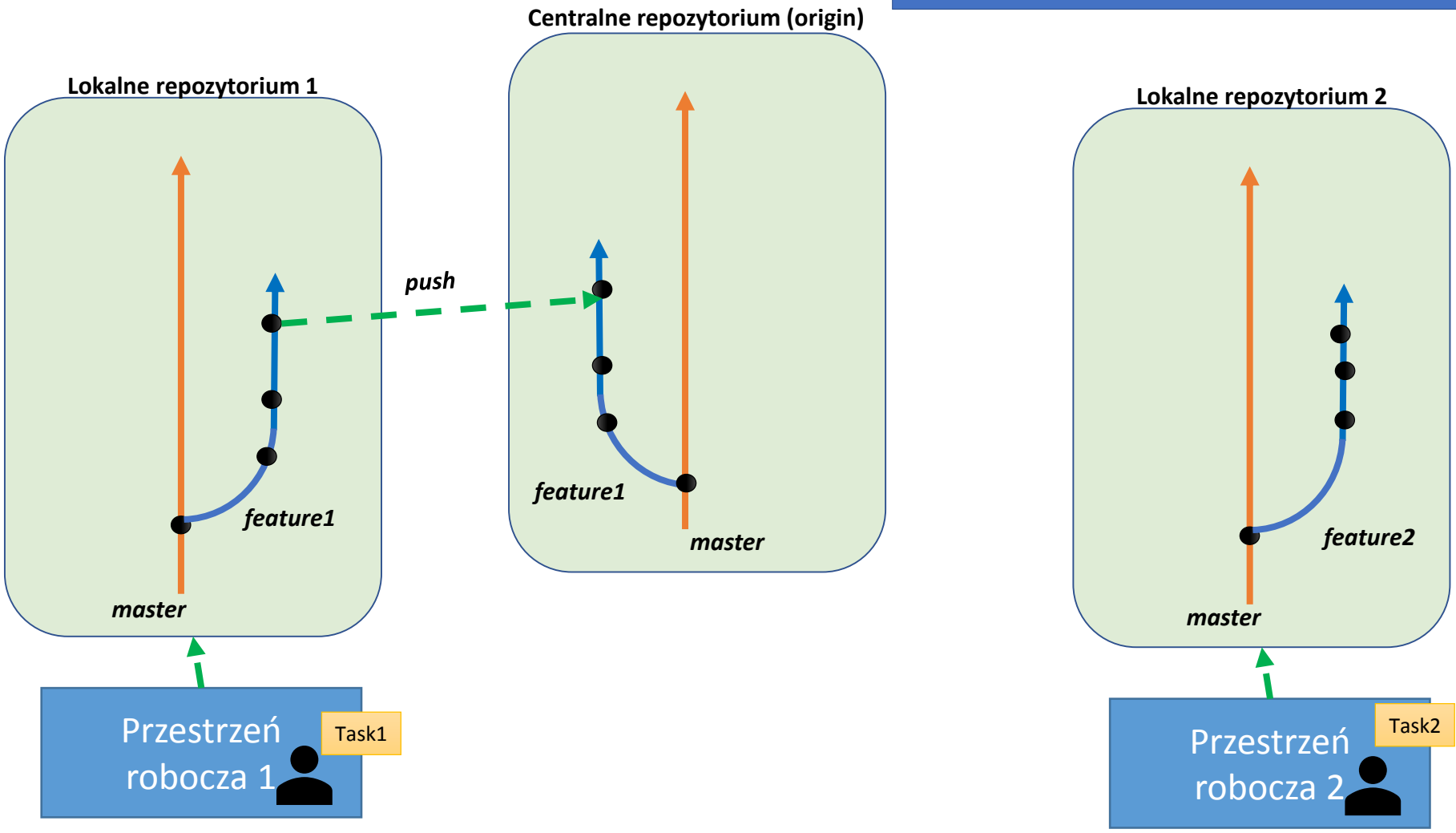
Lokalne repozytorium 2





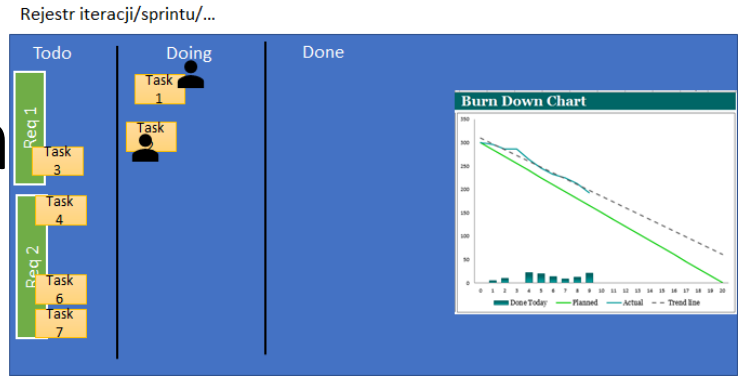
Feature branch – kończenie pracy nad zadaniem

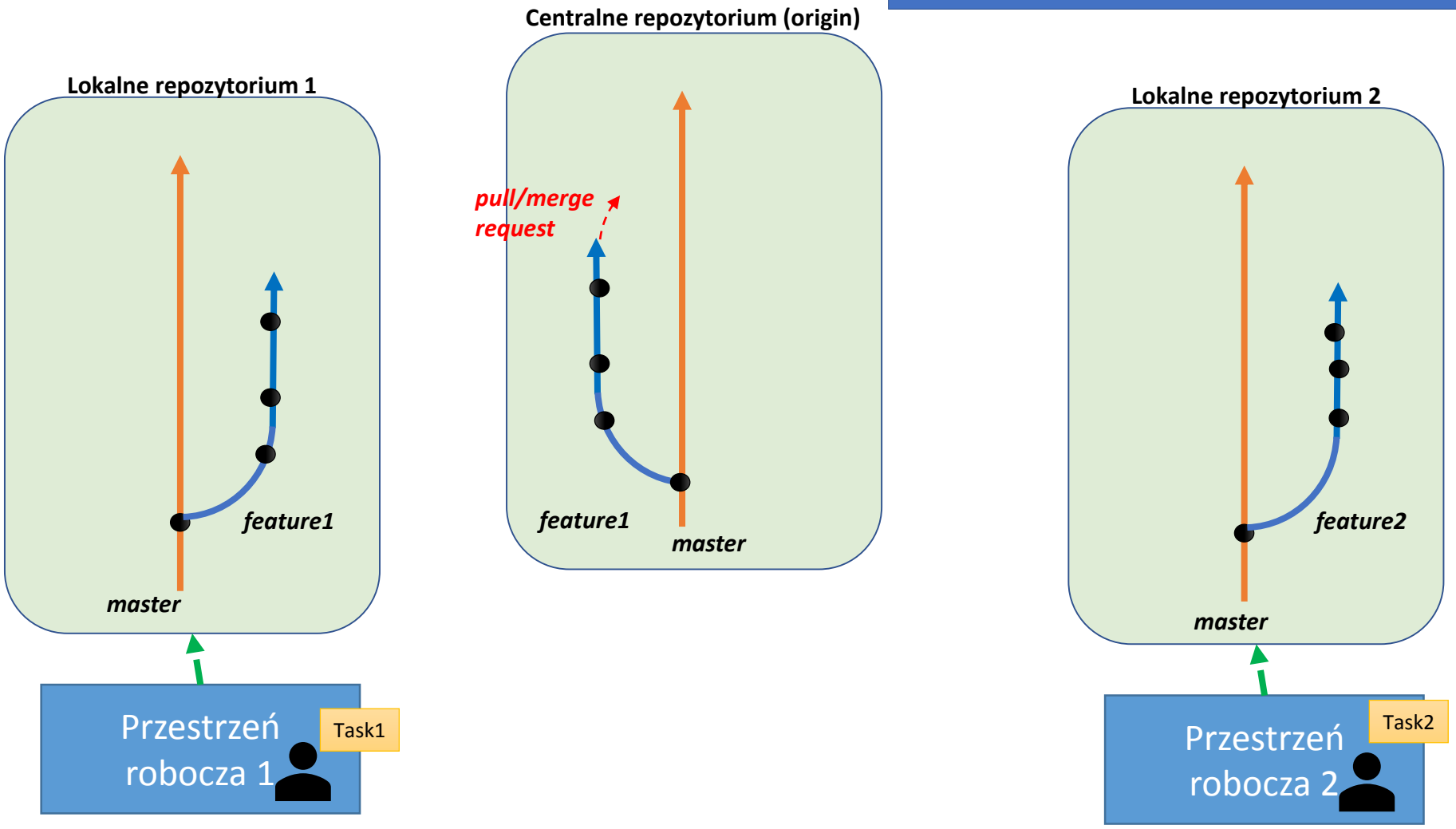




Feature branch – kończenie pracy nad zadaniem

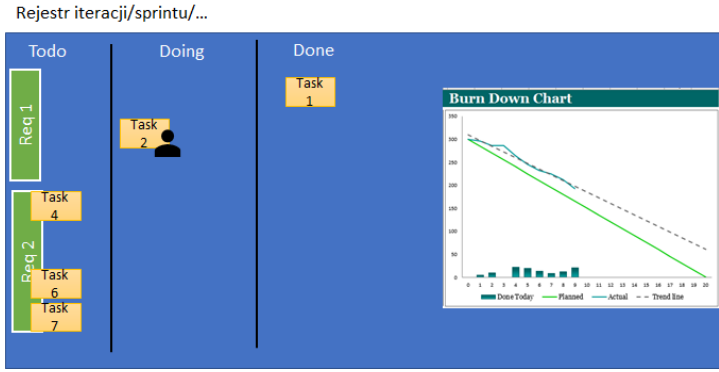
Faza przeglądu kodu





Feature branch – kończenie pracy nad zadaniem

Akceptacja



Centralne repozytorium (origin)

Zmiany realizujące „task 1” połączone z główną linią rozwojową

Lokalne repozytorium 1

feature1

master

Przestrzeń robocza 1

merge

feature1

master

Lokalne repozytorium 2

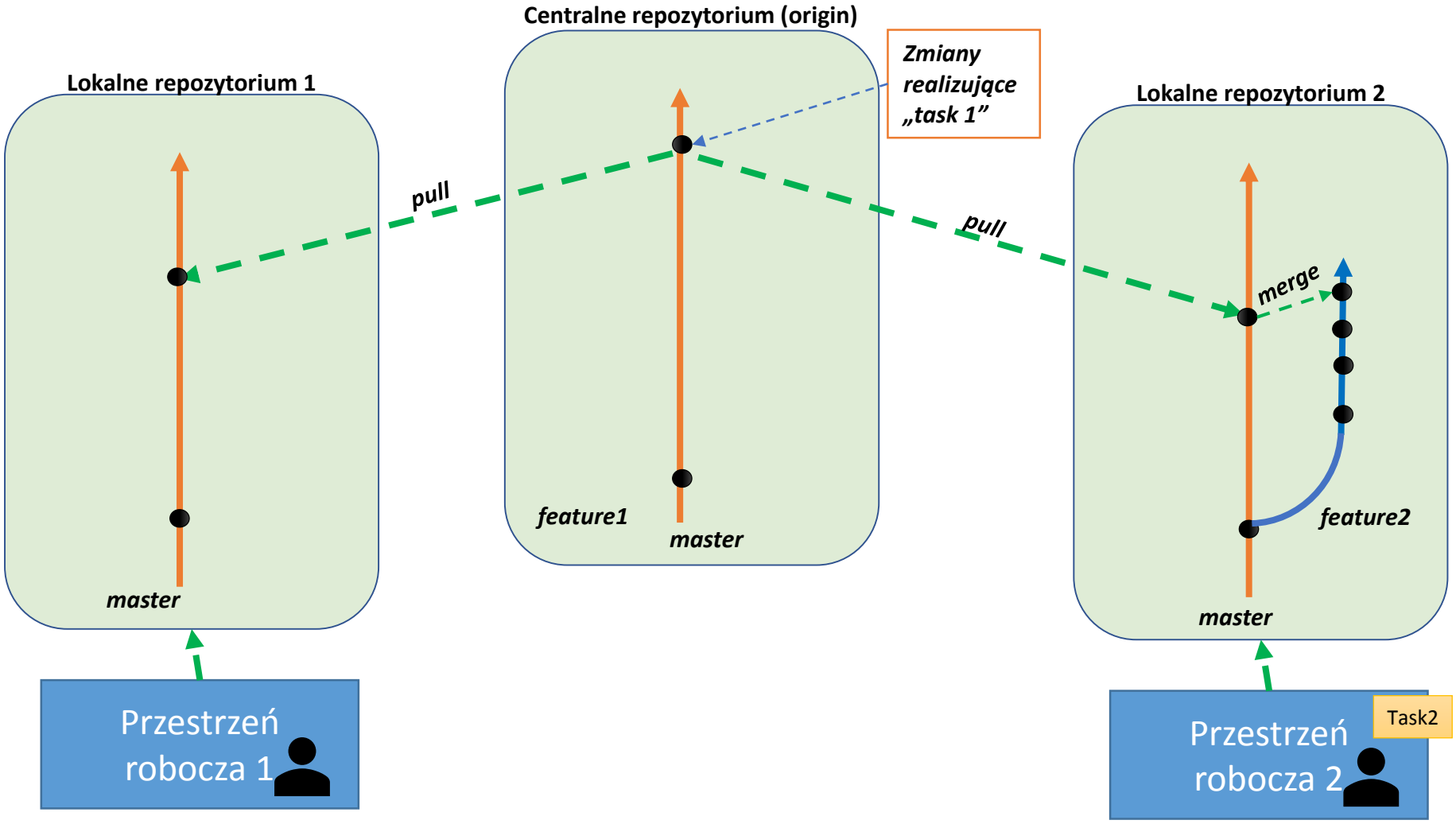
feature2

master

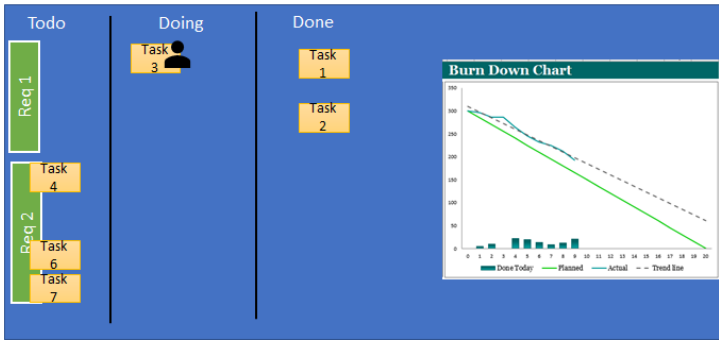
Przestrzeń robocza 2

Task2

Aktualizacja lokalnych repozytoriów



...



Centralne repozytorium (origin)

Zmiany realizujące „task 2”

merge

pull/merge request

Zmiany realizujące „task 1”

feature2

master

Lokalne repozytorium 1

pull

merge

feature3

master

Przestrzeń robocza 1

Task3

push

Lokalne repozytorium 2

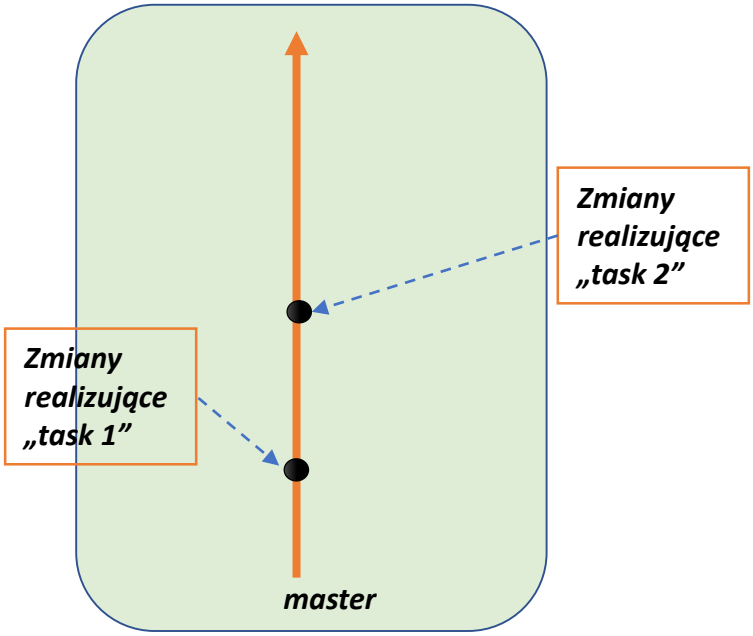
feature2

master

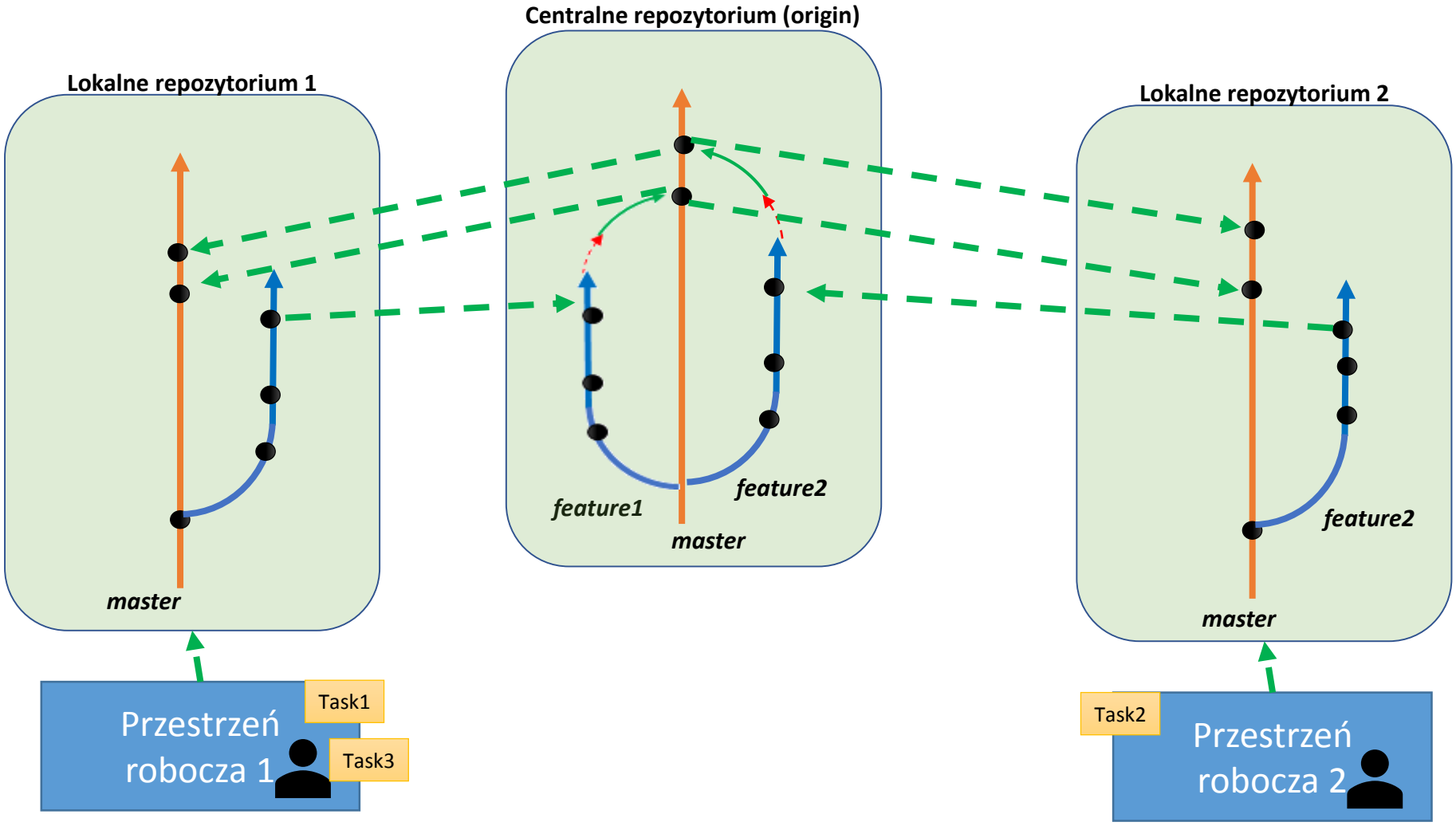
Przestrzeń robocza 2

Zmiany w trakcie iteracji

- W realizacji przyrostowej wymagania wybrane do przyrostu są **zamrażane**.
  - W trakcie prac zmiany realizujące zadania są łączone w sposób ciągły.
    - Pojedyncze zadania nie realizują kompletnego wymagania.
  - Modyfikacja/usunięcie wymagania jest teoretycznie możliwa
    - Ale praktycznie bardzo trudna
  - Dodanie wymagania jest możliwe
    - Ale wpływa na harmonogram
- Z punktu widzenia procesu nie jest to pożądane

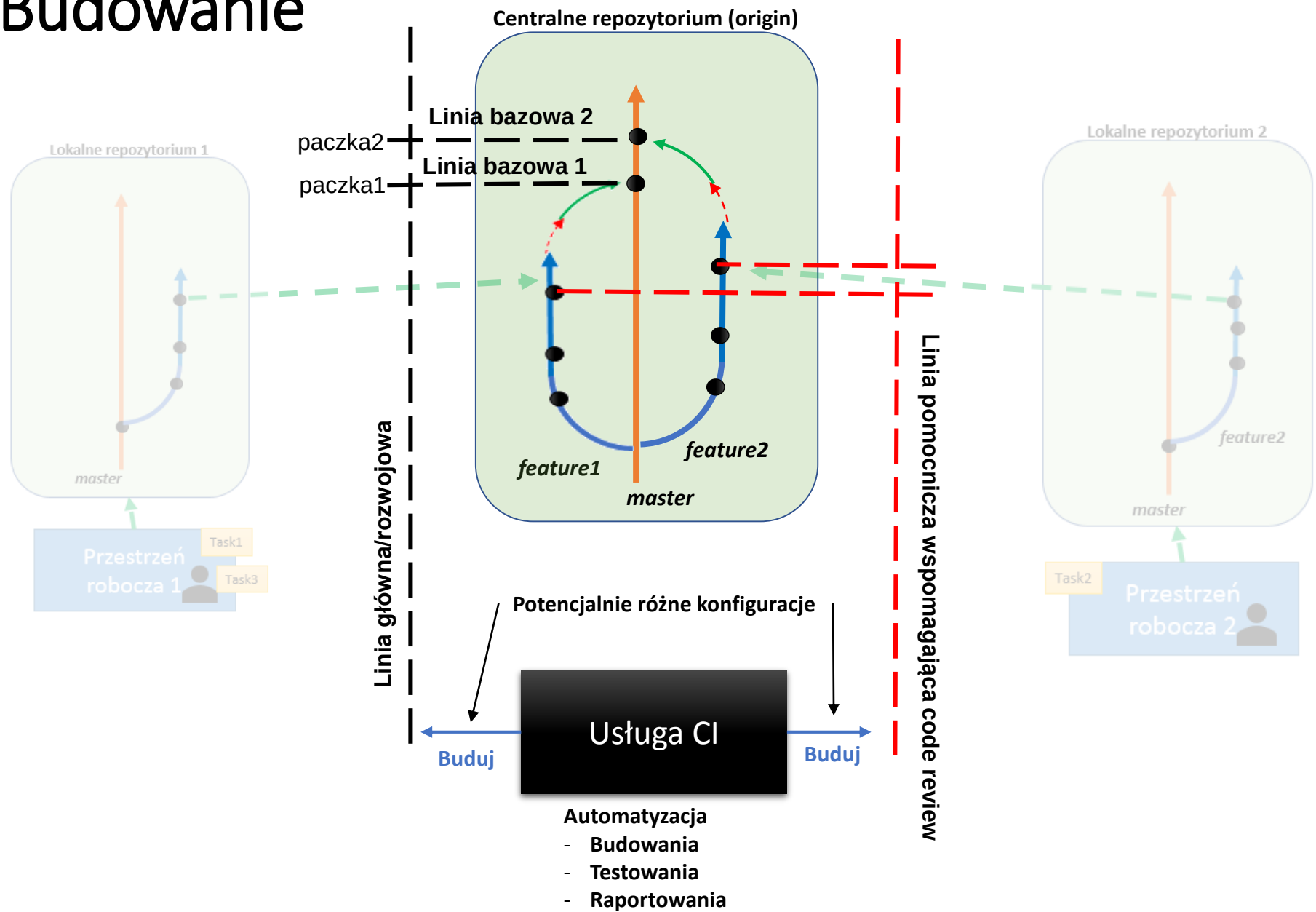


Ciągła integracja (CI)
Częste łączenie zmian



Ciągła integracja

Budowanie



Automatyzacja

- Budowania
- Testowania
- Raportowania

Elementy platformy budowania

System służący rozwojowi oprogramowania, w którego skład wchodzą narzędzia takie jak kompilatory, edytory kodu źródłowego, itd.

Programiści pobierają kod źródłowy z repozytorium wersji do prywatnej przestrzeni roboczej przed dokonaniem zmian w systemie.

Usługa budowania (ang. build server, serwer CI), wykorzystywana do budowania ostatecznej, wykonywalnej wersji systemu.

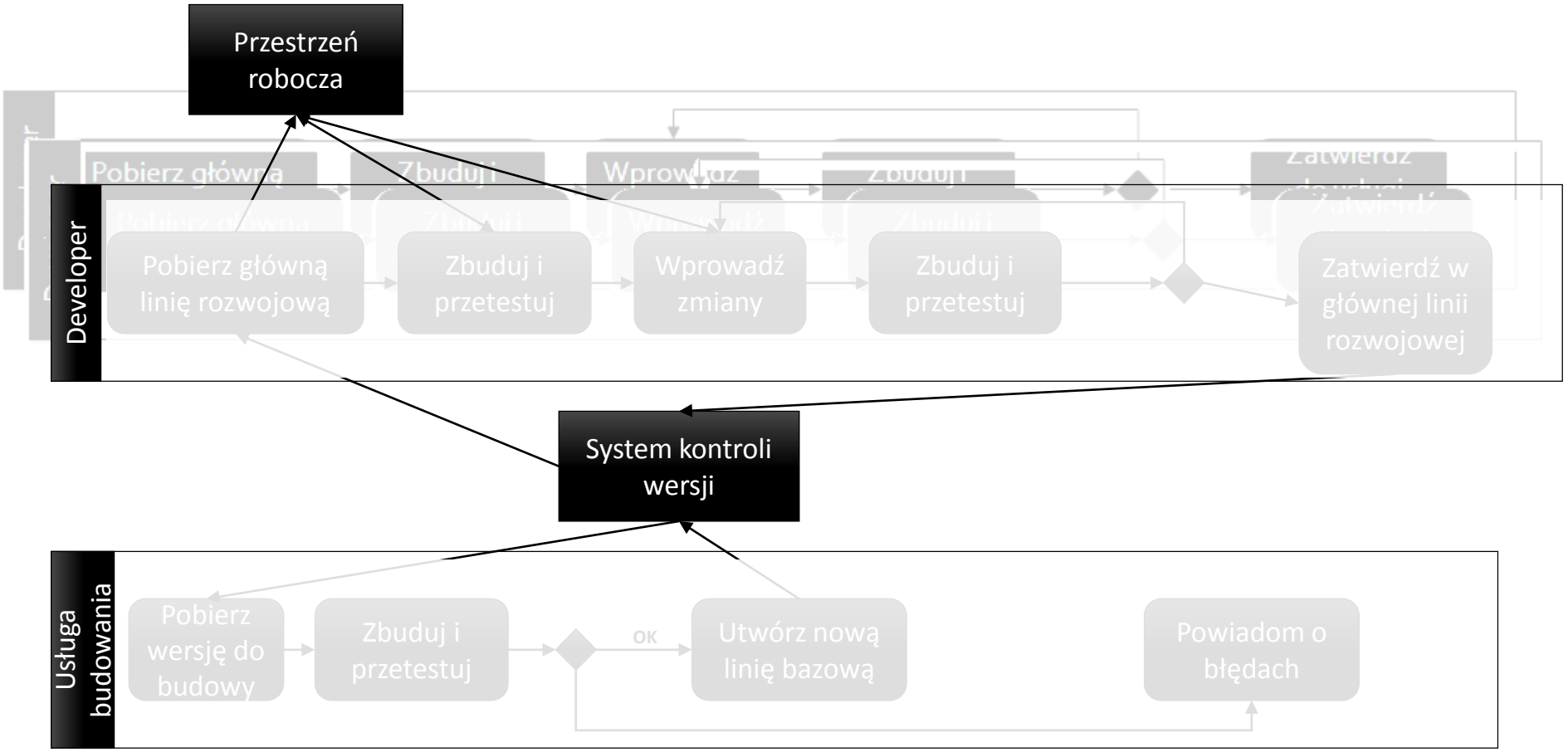
Programiści pobierają kod źródłowy z repozytorium wersji przed uruchomieniem procesu budowy. Budowa systemu może wymagać zewnętrznych bibliotek, które nie są przechowywane w systemie zarządzania wersjami.

Środowisko docelowe, reprezentujące platformę na które system będzie wykonywany.

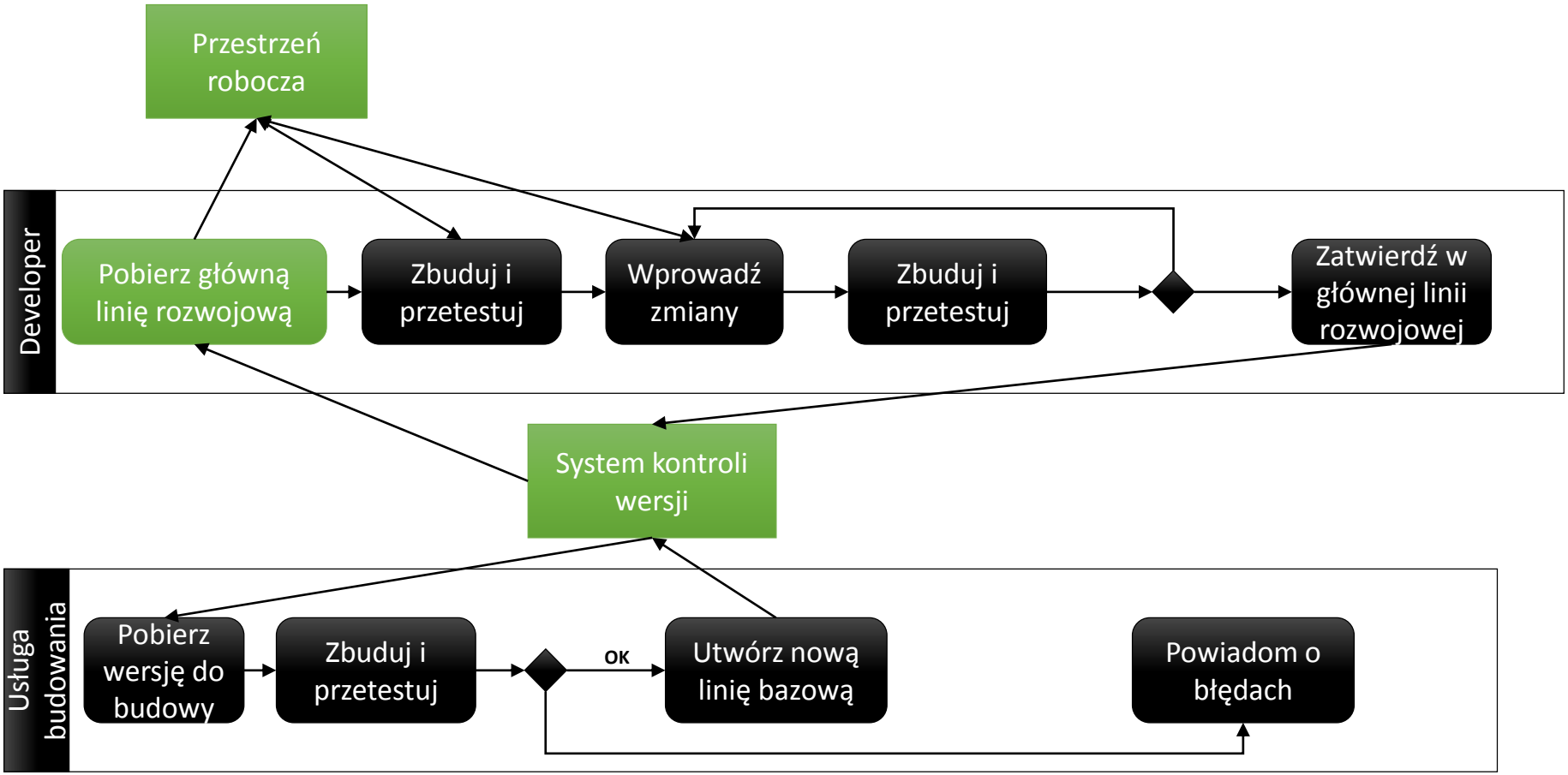
Funkcjonalności systemu budującego

- Generowanie skryptów budujących
- Integracja z systemem zarządzania wersjami
- Minimalizacja potrzeby rekompilacji komponentów
- Tworzenie wykonywalnego systemu
- Automatyzacja testów
- Raportowanie
- Generacja dokumentacji

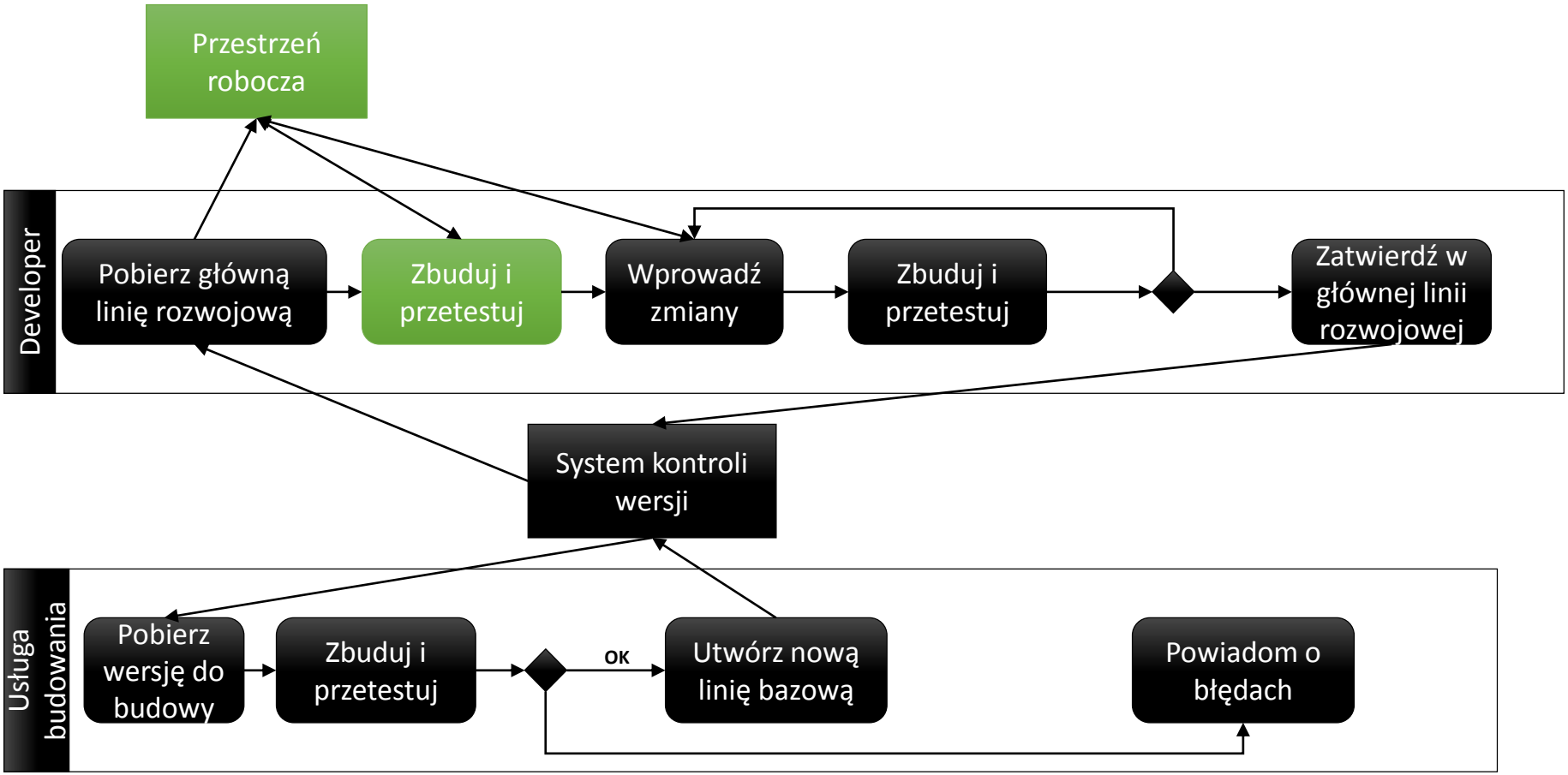
Ciągła integracja (ang. continuous integration)



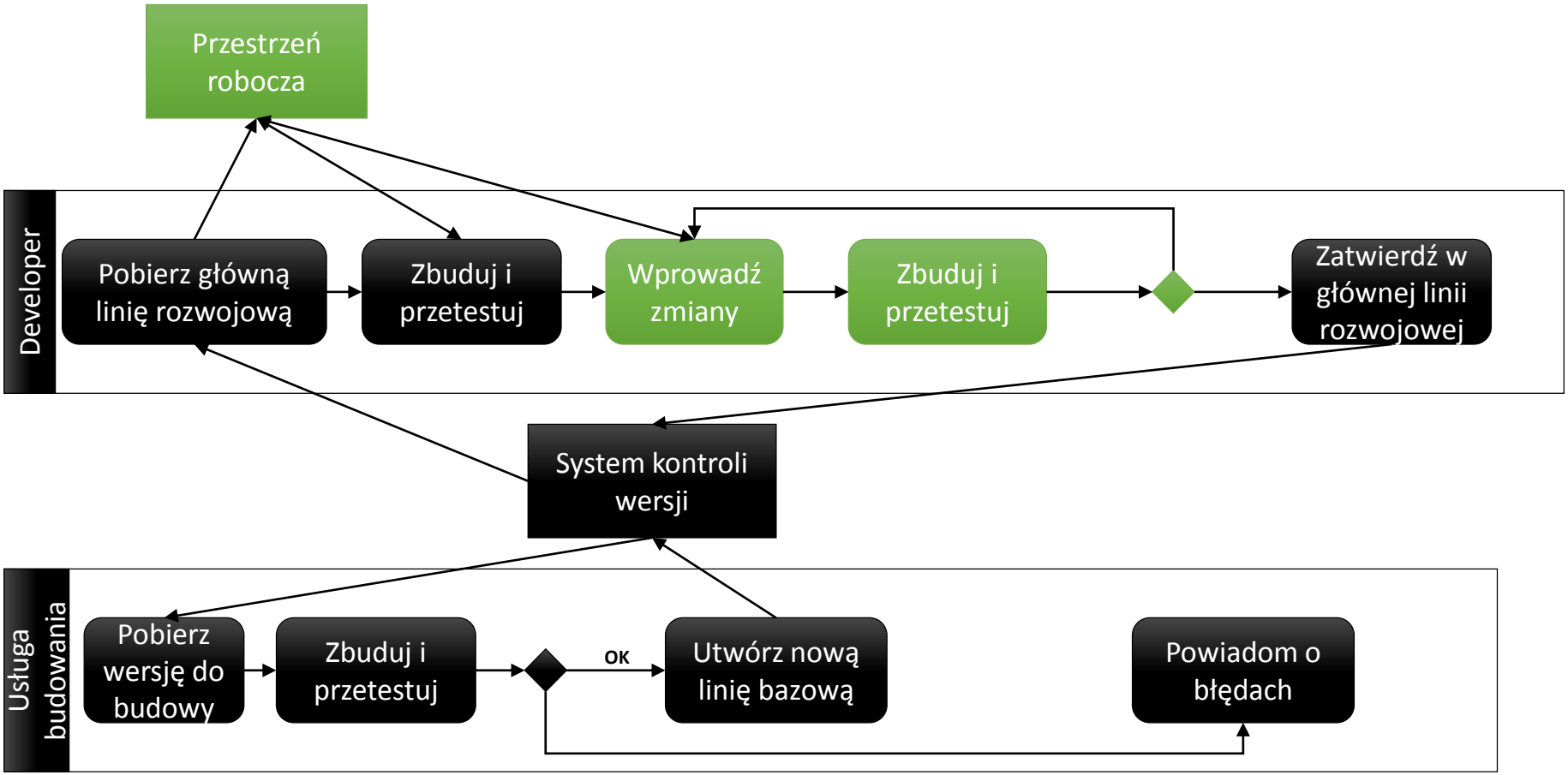
Ciągła integracja (1)



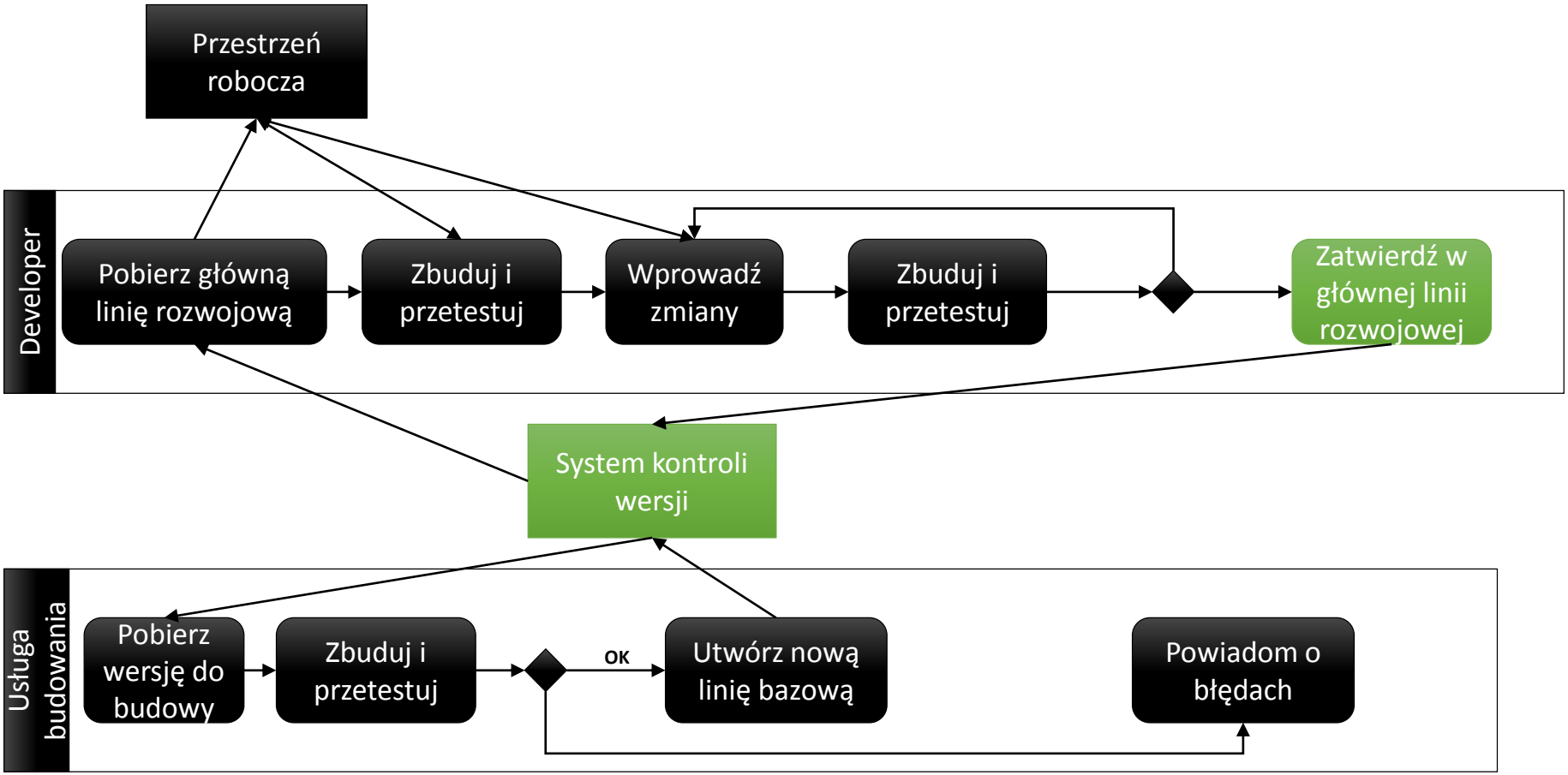
Ciągła integracja (2)



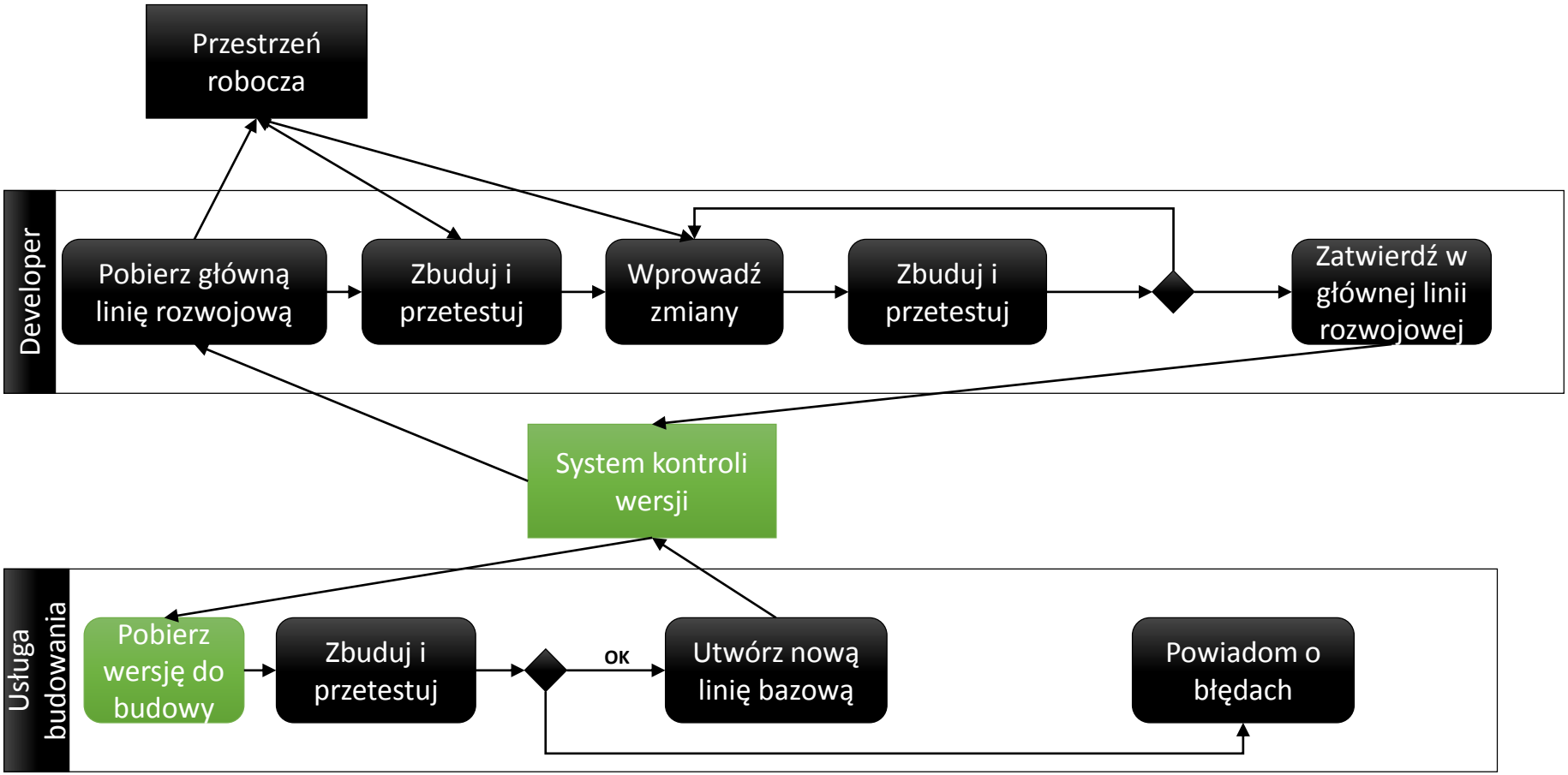
Ciągła integracja (3)



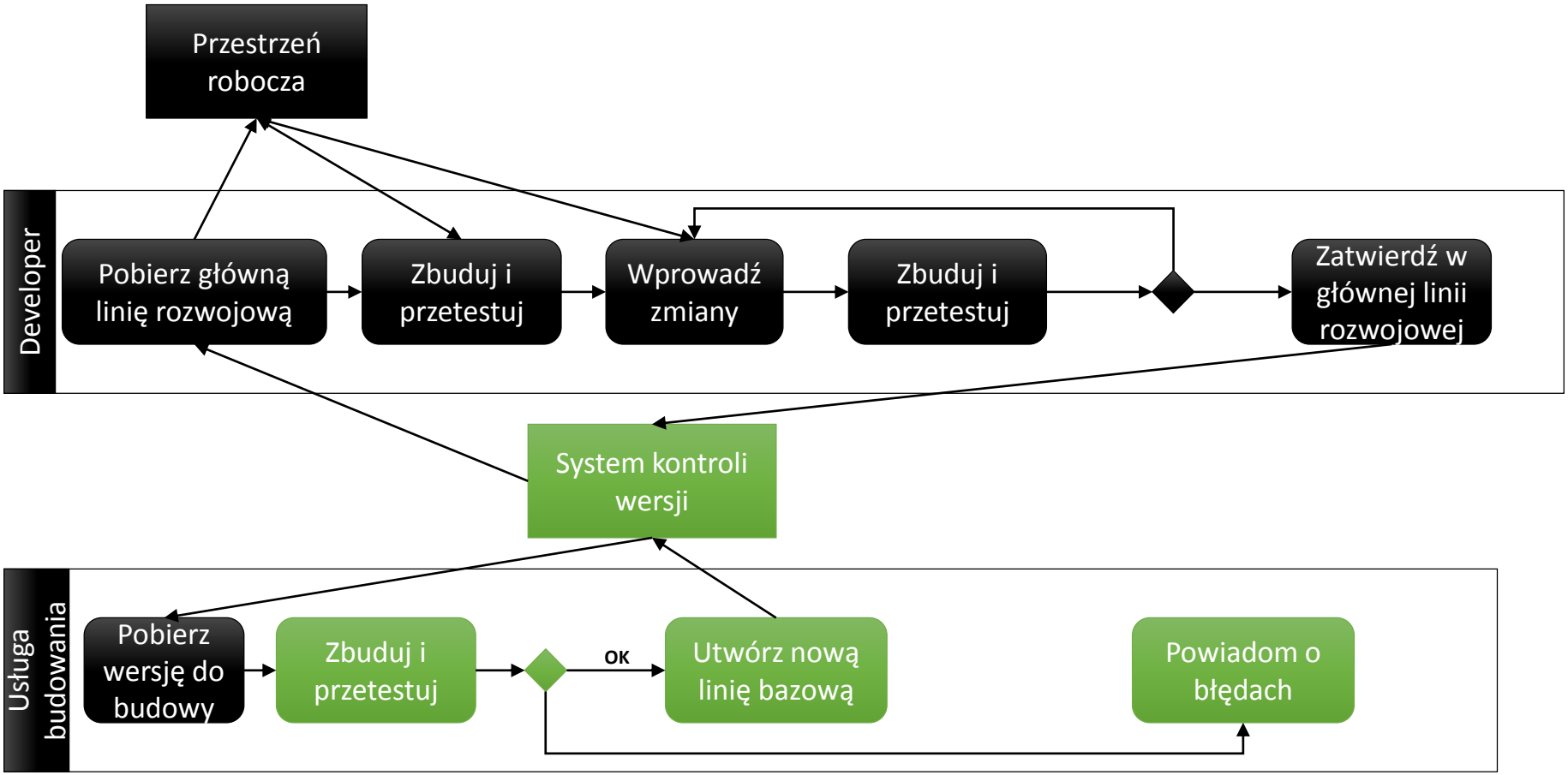
Ciągła integracja (4)



Ciągła integracja (5)



Ciągła integracja (6)



Ciągła integracja – automatyzacja procesu rozwoju

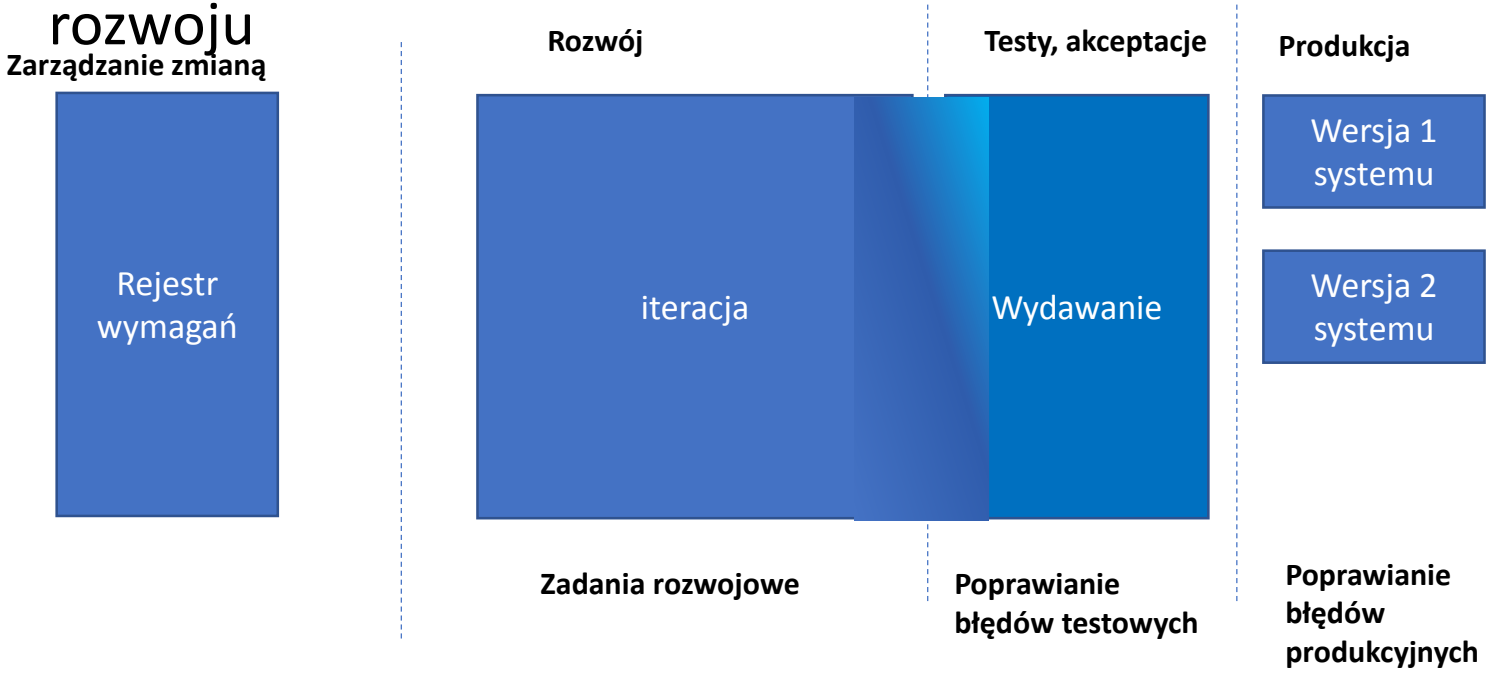
- **Ograniczenia**
  - Zespoły odpowiedzialne za przygotowanie środowiska uruchomieniowego czekają na poprawki w dokumentacji
  - Testerzy czekają na „dobre buildy”
  - Zespół rozwijający system otrzymuje raporty z błędami tydzień po tym jak rozpoczął pracę nad nowymi funkcjonalnościami
    - (długa pętla zwrotna pomiędzy zespołami: (**development i operational**)
  - Brak zapewnienia, że architektura aplikacji pozwala spełnić wymagania niefunkcjonalne

Dostarczanie przyrostowe a model pracy z repozytorium

Git flow

Dostarczanie przyrostowe

- Model pracy z repozytorium dla dostarczania przyrostowego jest rozszerzeniem (uogólnieniem) modelu z realizacji przyrostowej.
  - Przyrost staje się wydaniem, wydanie może stać się wersją produkcyjną
  - Wydanie może podlegać zmianom równoległym do rozwoju



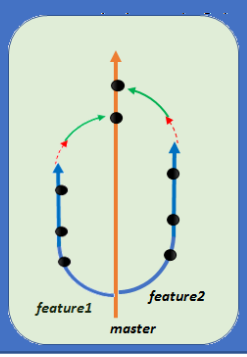
Dostarczanie przyrostowe

Zarządzanie zmianą

Rejestr wymagań

Rozwój

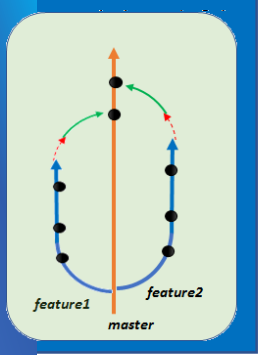
Iteracja



Zadania rozwojowe

Testy, akceptacje

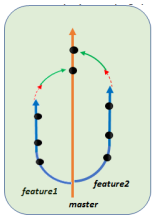
Wydawanie



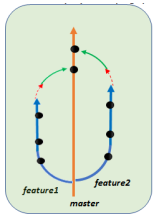
Poprawianie błędów testowych

Produkcja

Wersja 1 systemu



Wersja 2 systemu

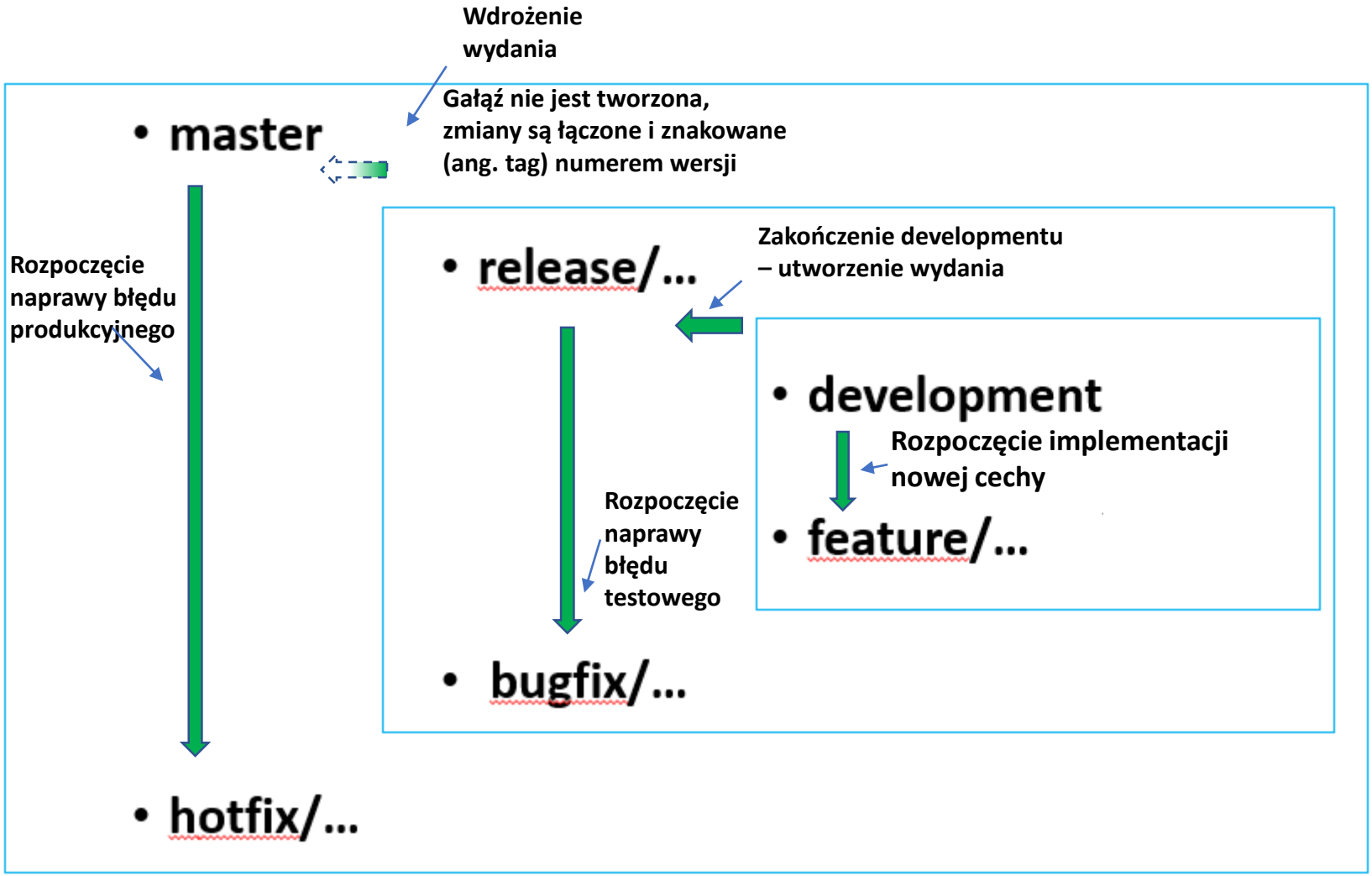


Poprawianie błędów produkcyjnych (błędy krytyczne)

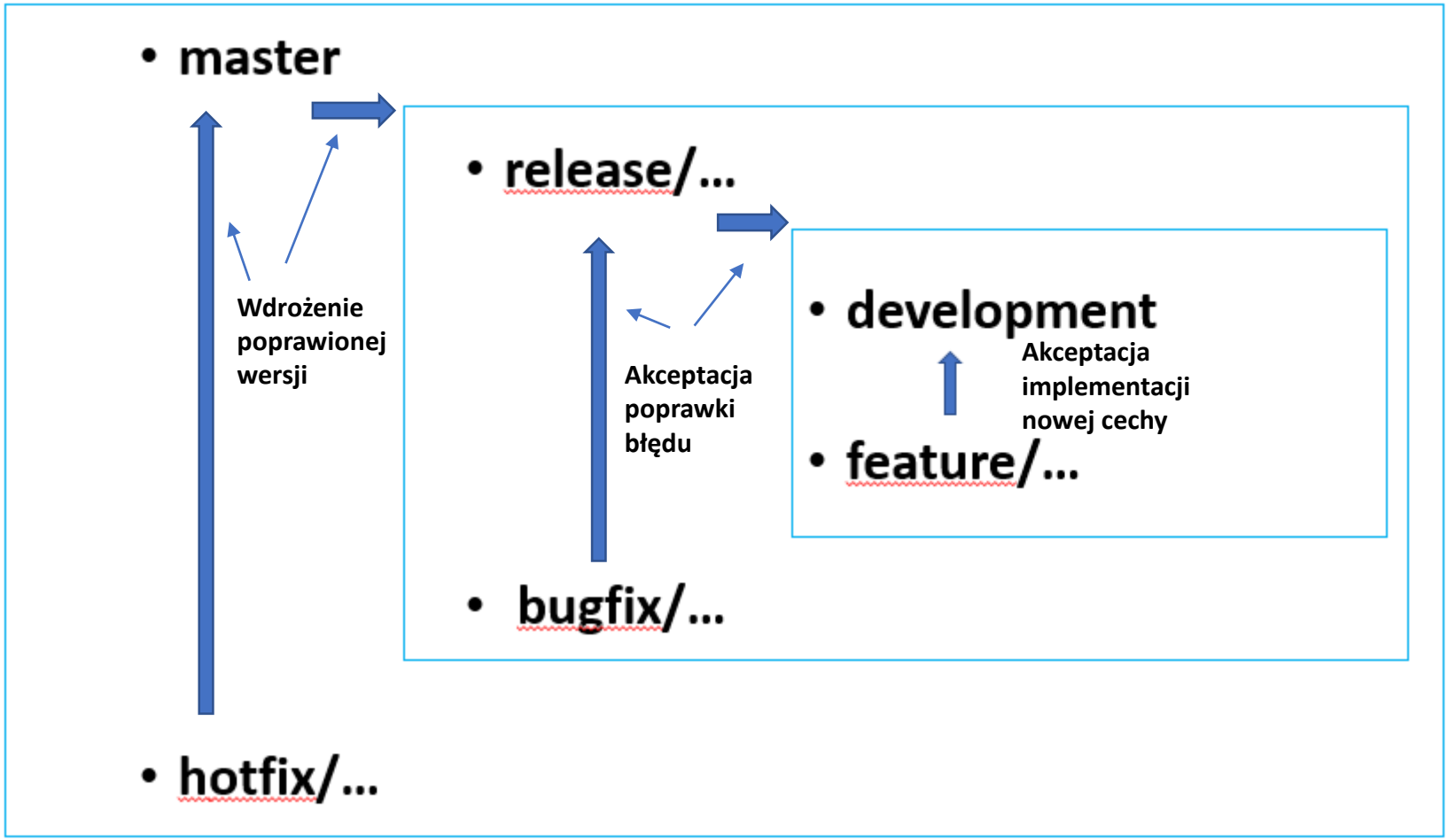
Git flow – model gałęzi

master	release/…	development	feature/…	bugfix/…	hotfix/…
•gałąź produkcyjna zawierająca wersje produkcyjne systemu	•gałęzie z kodem wydania (potencjalne zmiany dotyczą błędów testowych)	•gałąź rozwojowa (przyrostu) wyznaczająca główną linię rozwojową projektu	•gałęzie związane z rozwojem cech systemu – zadania w ramach iteracji	•gałęzie wprowadzające poprawki do błędów pojawiających siew trakcie testów	•gałęzie wprowadzające poprawki błędów krytycznych do wersji produkcyjnych

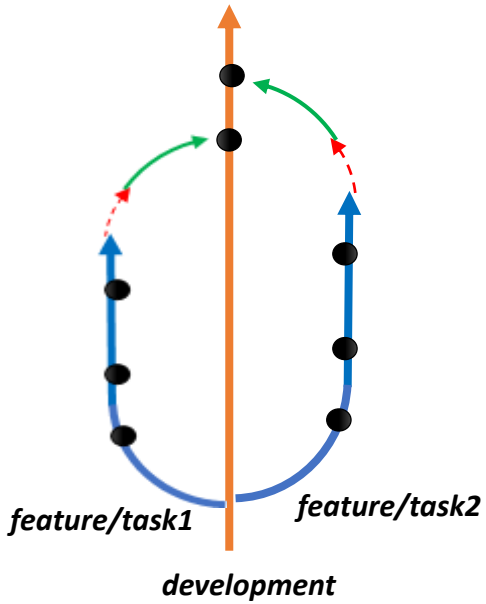
Git flow – model tworzenia gałęzi

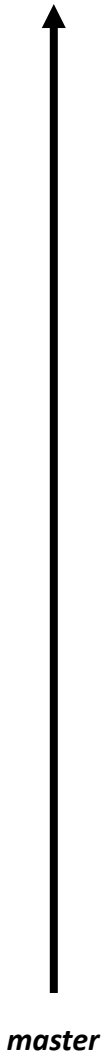


Git flow – model łączenia zmian



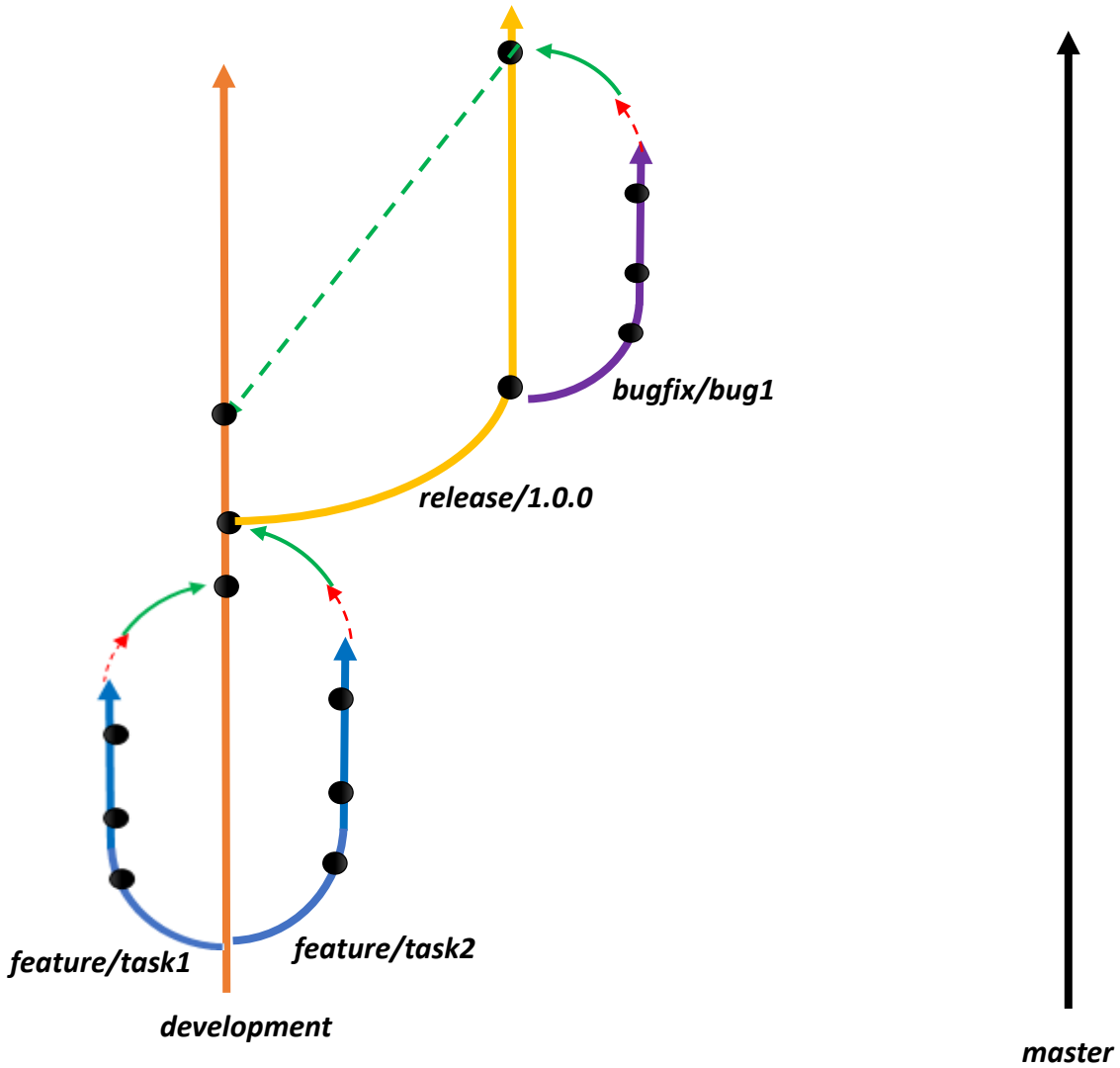
Development -> feature/…





development -> feature/…

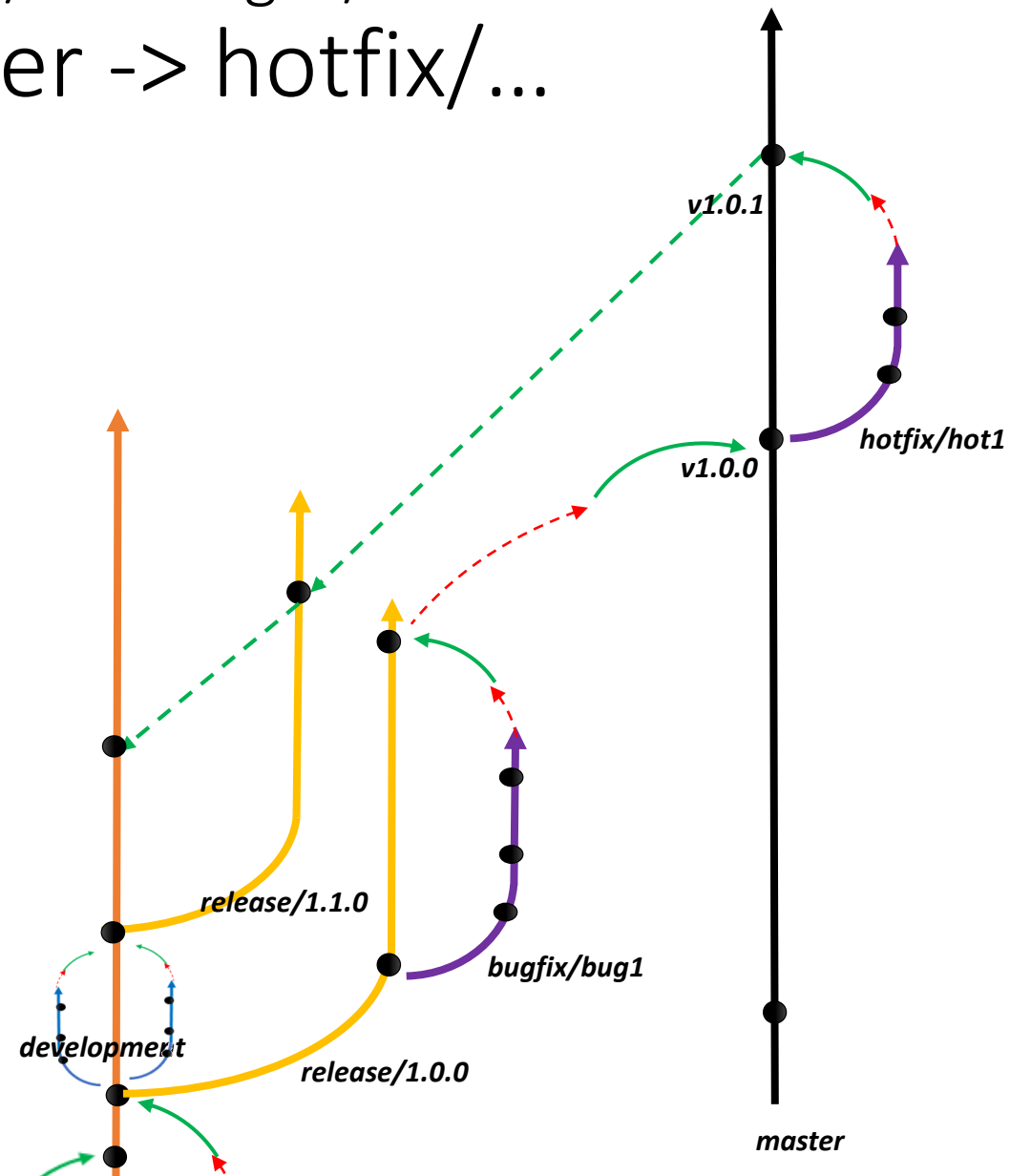
release/… -> bugfix/…



development -> feature/…

release/… -> bugfix/…

master -> hotfix/…



Wersjonowanie semantyczne

- http://semver.org/
- X.Y.Z
  - X – major, Y – minor, Z – patch
- Jakakolwiek zmiana w opublikowanym, w danej wersji oprogramowaniu wymaga zmiany wersji.
- X = zero – inicjacja projektu – wszystko może ulec zmianie
- X++ - wprowadzone niekompatybilne zmiany
- Y++ - wprowadzone zmiany kompatybilne wstecz
- Z++ - wprowadzono poprawkę (bez zmian w funkcjonalności)
- Możliwe jest dołączanie identyfikatorów
  - 1.0.0-alpha, 1.0.0-beta+exp.sha.5114f85

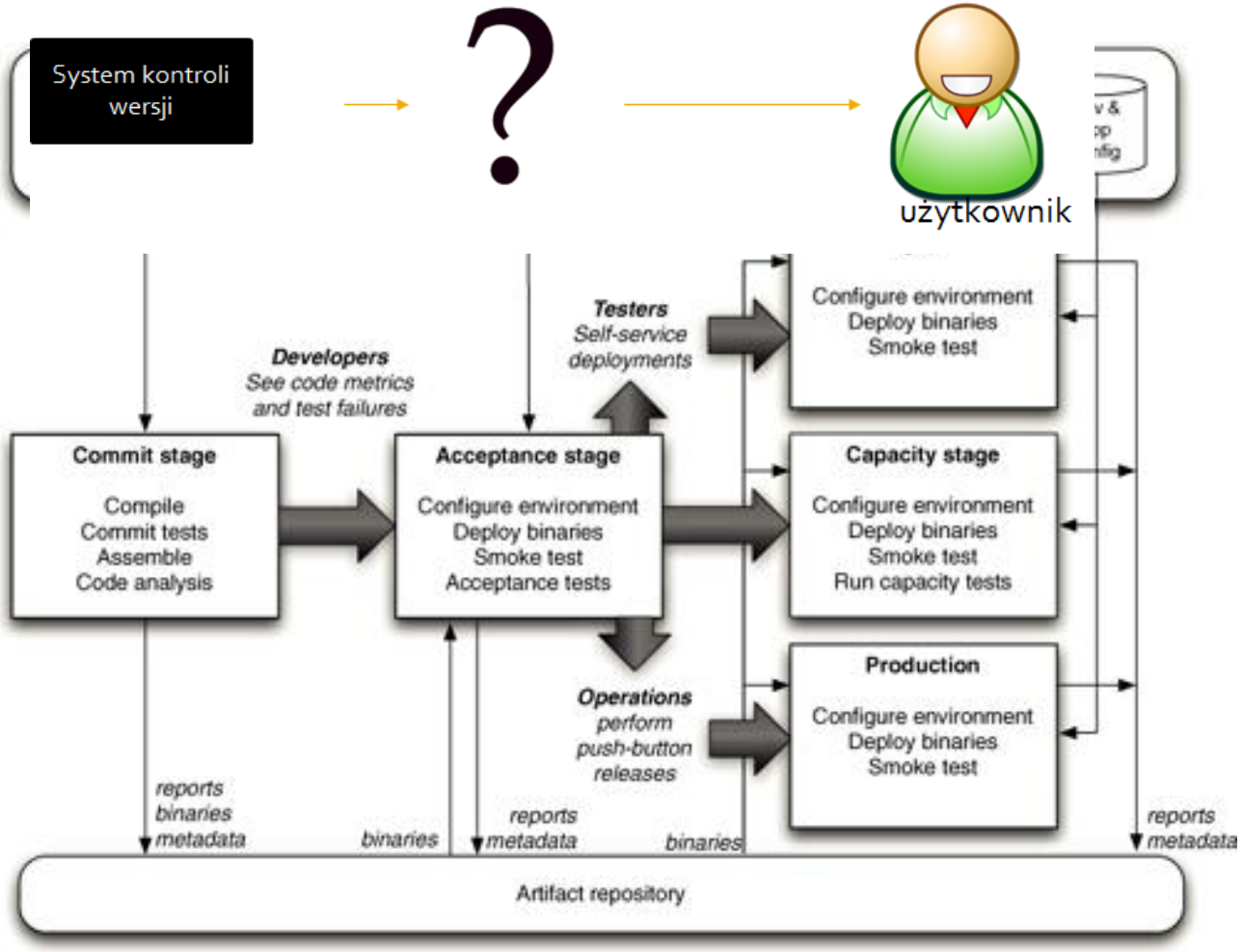
Ciągłe wdrażanie (ang. continuous deployment)

- Ciągła integracja – automatyzacja procesu rozwoju
- Ciągłe wdrażanie - automatyzacja całości procesu inżynierii oprogramowania
  - W dowolnym momencie wdrożenie wybranej wersji zbudowanego oprogramowania (za pomocą „przycisku”)
    - Automatyczne umieszczenie w środowisku
      - testowym,
      - zapewniania jakości,
      - produkcyjnym,
      - …

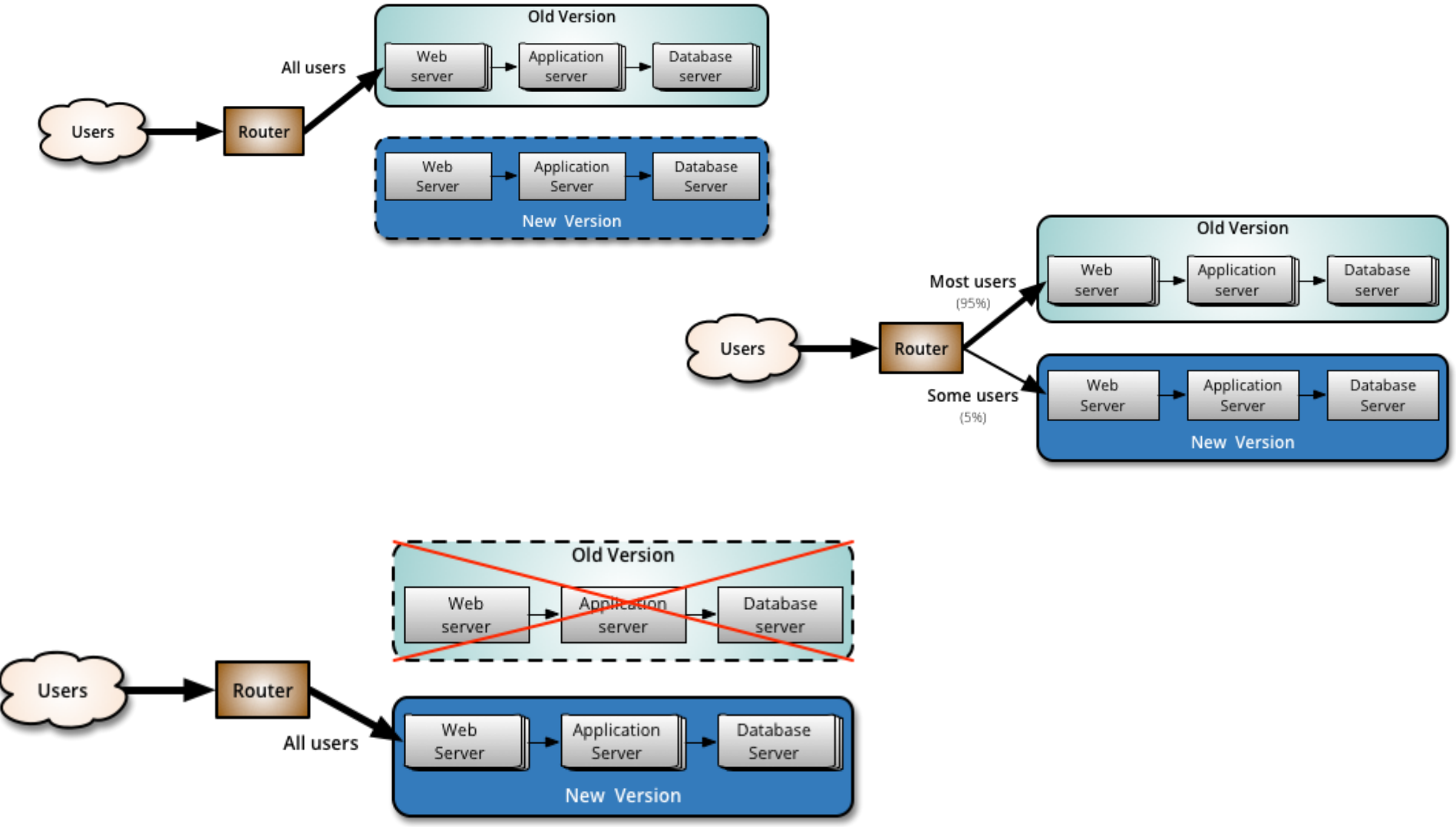
Ciągłe dostarczanie (ang. continuous delivery)

- Mechanizmy pozwalające na automatyczne dostarczenie wersji oprogramowania do klienta jeżeli tylko przejdzie ono wszystkie etapy zatwierdzania jakości.
  - Potencjalnie dowolna wersja systemu może być dostarczona klientowi
  - Promuje całkowitą automatyzację -> rurociąg dostarczający oprogramowanie (ang. deployment pipeline)

Rurociągi dostarczania



Canary release



http://martinfowler.com/bliki/CanaryRelease.html