

# Systemy kontroli wersji



## Wprowadzenie

Aleksander Lamża  
ZKSB · Instytut Informatyki  
Uniwersytet Śląski w Katowicach

[aleksander.lamza@us.edu.pl](mailto:aleksander.lamza@us.edu.pl)

- W czym tkwi problem?
- Jak sobie radzić?
- Wspólna praca
- Metoda kopiuj – modyfikuj – scalaj
- Trochę terminologii

To jest **Zuzia**



# Co robi Zuzia?

Zuzia **tworzy** pliki



Zuzia **modyfikuje** pliki

```
// Source code for Famaia Icon Set
// Author: "Liban Incubus Janydmail.com"
// Licensed under GPL

#include <stdint.h>
#include <math.h>
#include <string.h>

#define N_PIXELS 32768
#define N_COLORS 256

typedef struct _MhFile MhFile;
typedef struct _MhFileClass MhFileClass;

typedef enum {
    MhFile_Flags = 0,
    MhFile_Flags2 = 1
} MhFileFlags;

MhFileClass
```

Zuzia **zapisuje** zmiany



```
// Source code for Famaia Icon Set
// Author: "Liban Incubus Janydmail.com"
// Licensed under GPL

#include <stdint.h>
#include <math.h>
#include <string.h>

#define N_PIXELS 32768
#define N_COLORS 256

typedef struct _MhFile MhFile;
typedef struct _MhFileClass MhFileClass;

typedef enum {
    MhFile_Flags = 0,
    MhFile_Flags2 = 1
} MhFileFlags;

MhFileClass
```

# Co robi Zuzia?



## Zuzia **tworzy** pliki



## Zuzia **modyfikuje** pliki

```
// Source code for Faenza Icon Set
// Author: Luca Mathieu (luca.mathieu@gmail.com)
// Licensed under GPL

#include <gtk/gtk.h>
#include <math.h>
#include <string.h>

#define M_PI 3.14159265358979323846
#define M_SQRT3 1.732050807568877293527446371
#define M_SQRT30 5.477225575051661434273616774

typedef struct _MyStyle MyStyle;
typedef struct _MyStyleClass MyStyleClass;

typedef enum {
    CAIRO_STROKE = 1,
    CAIRO_FILL = 2
} DrawingOperation;
```

## Zuzia zapisuje zmiany



Chciałabym mieć jakąkolwiek  
kontrolę nad zmianami  
wprowadzanymi w moim pliku!

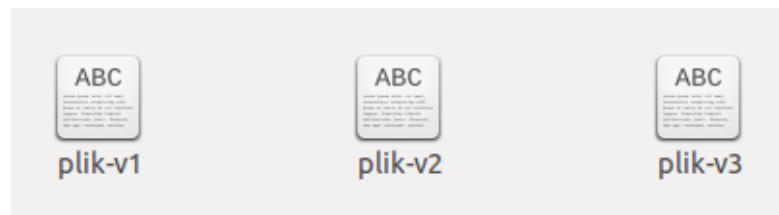


## A po co?

- Żeby mieć dostęp do poprzednich wersji (możliwość powrotu).
- Żeby znać historię zmian (kiedy i jak był modyfikowany plik).
- Żeby móc porównać różne wersje tego samego pliku.

# Domowe sposoby kontroli wersji

Możemy zapisywać kolejne wersje pod zmienioną nazwą:



Jeżeli mamy więcej plików, tę samą metodę można zastosować do katalogów. Zamiast numeru wersji możemy dopisywać datę lub opis zmian.

*...i co?*

**Choćbyśmy się nie wiem jak starali, i tak skończymy źle...**

# Wady domowych sposobów kontroli wersji



Hm... Na którym pliku  
ostatnio pracowałam?  
Chyba oszaleję!

- Dużo żmudnej ręcznej pracy – zmiana nazw plików.
- Łatwo o pomyłkę (trudno o brak pomyłki).
- Utrudnione obserwowanie zmian wprowadzanych w plikach.
- CHAOS!



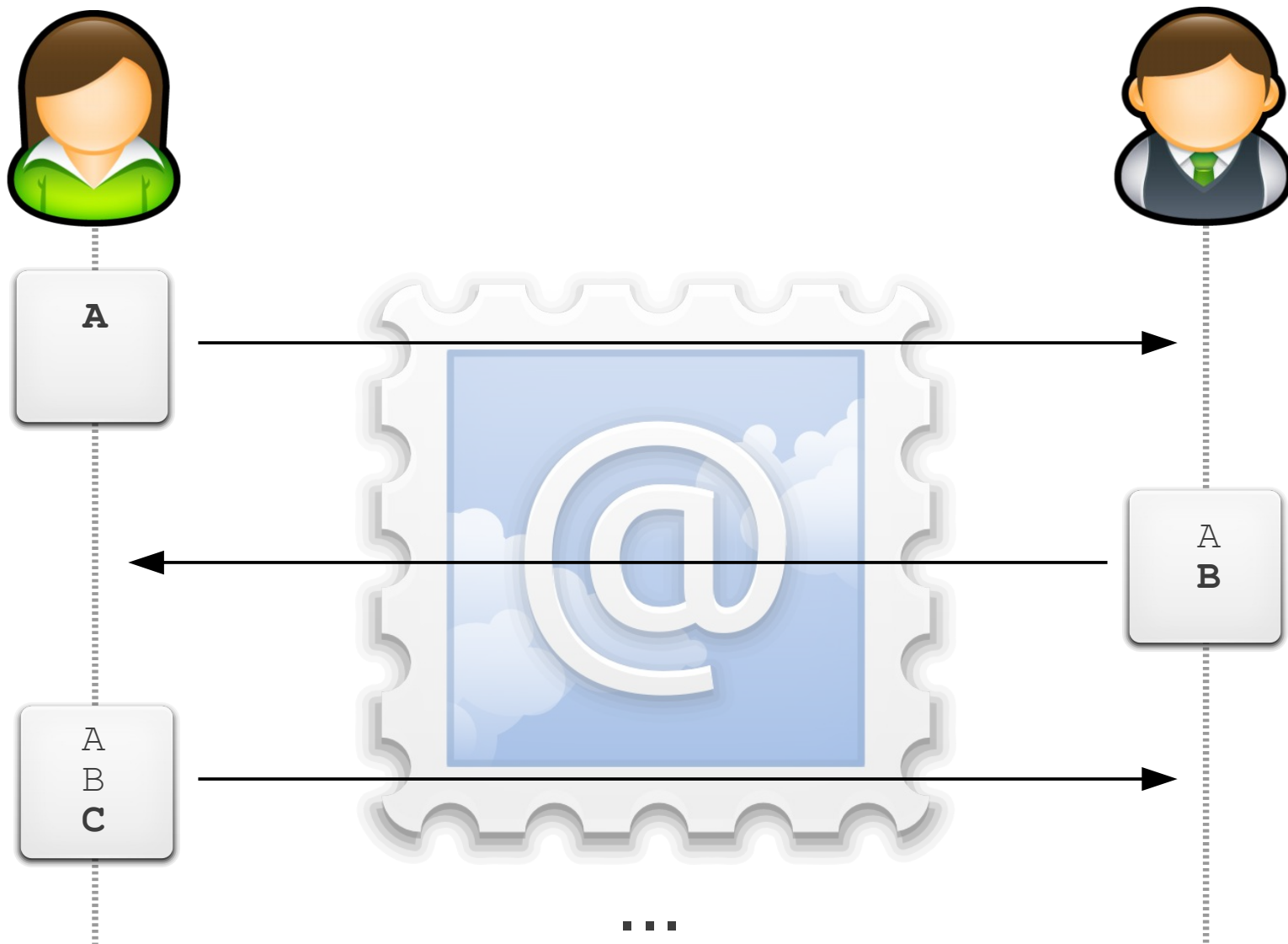
# Na tym nie koniec problemów

A co jeśli Zuzia zapozna miłego Heńka, z którym zechce wspólnie pracować nad plikiem?



# Razem, ale osobno

Niestety (dla Heńka) Zuzia stwierdziła, że popracują co prawda **razem**, ale **osobno**.  
Będą sobie wysyłać plik e-mailem.



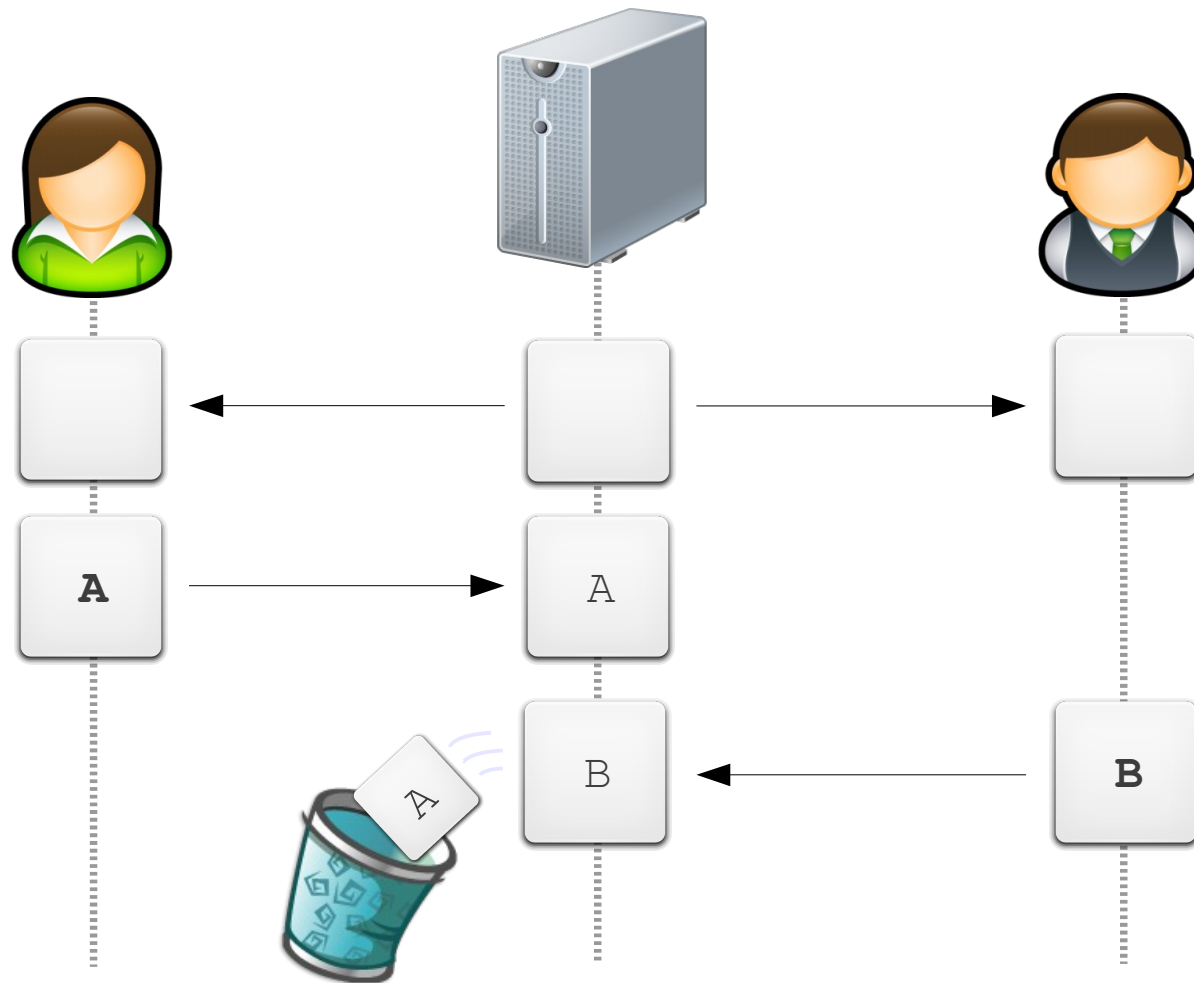
# Na tym nie koniec problemów

Problem w tym, że kiedy Zuzia pracuje nad plikiem, Heniek się nudzi.  
Musi czekać, aż Zuzia prześle mu zmodyfikowaną wersję.



A może umieścilibyśmy nasz plik  
na jakimś serwerze?

# Pliki ładują na serwerze



## **AWARIA!**

Heniek wrzucił na serwer swoją wersję pliku i **nadpisał** wersję Zuzi.

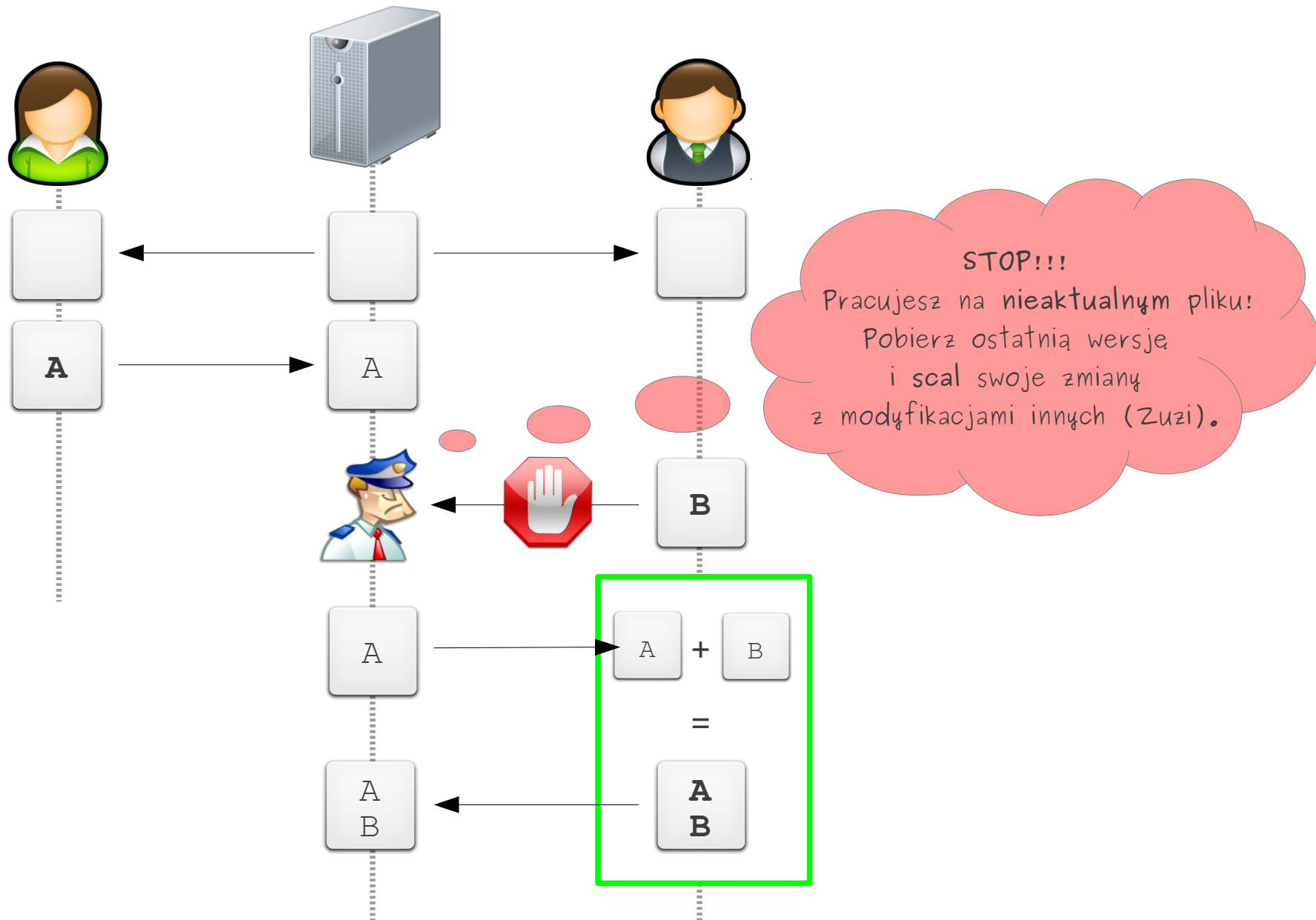
# Heniek przez jakiś czas powinien unikać Zuzi...



Przez tego #&\$@& Heńka cała moja praca  
poszła na marne!

Przydałby się jakiś strażnik,  
który zapobiegałby takim sytuacjom.

# Co ma zrobić Heniek, żeby następnym razem nie podpaść Zuzi?



# Uff... Nareszcie wygląda to lepiej

To co przed chwilą widzieliście, to metoda

**kopiuj – modyfikuj – scalaj**

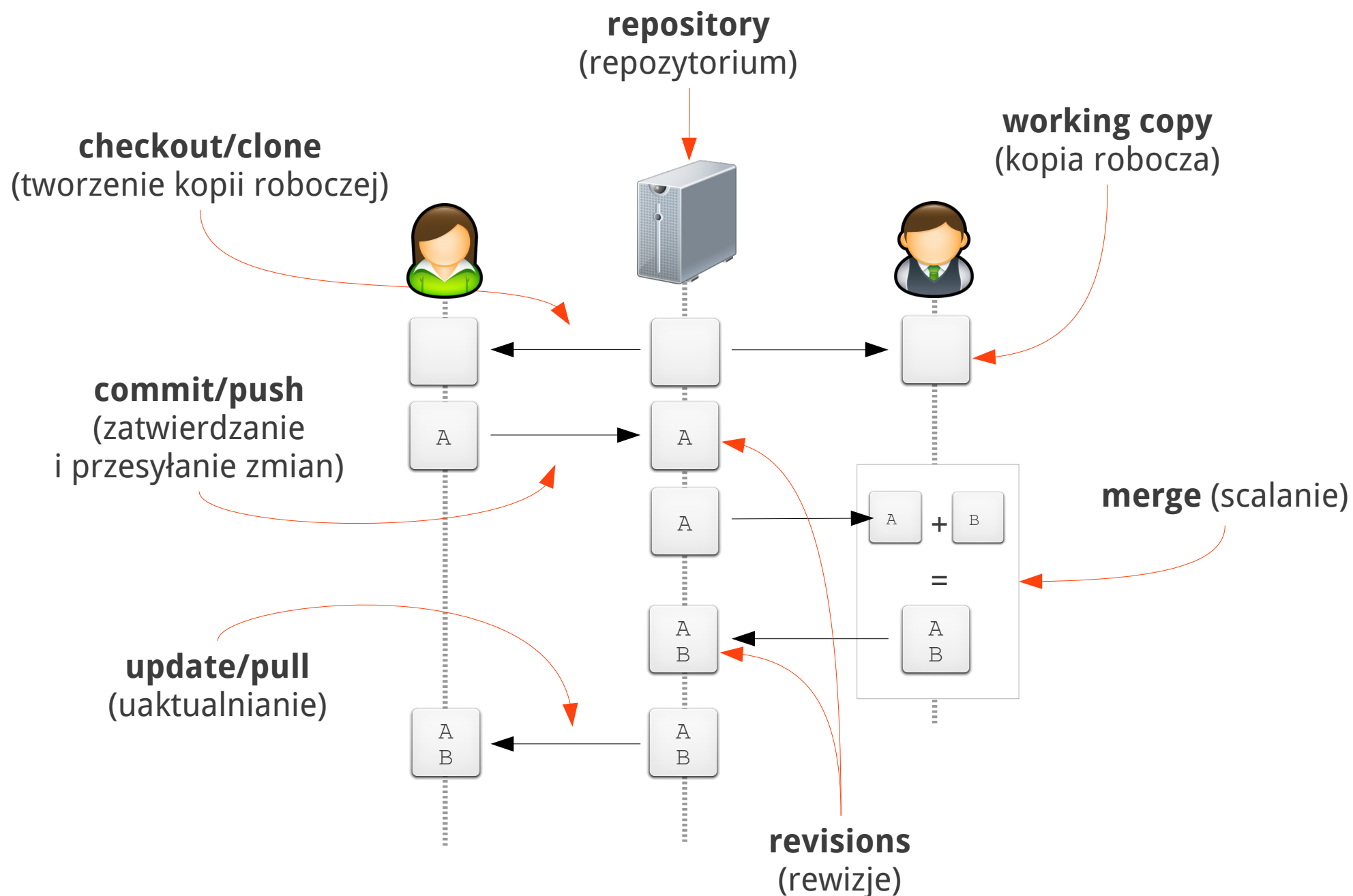
*(copy – modify – merge)*

Ta metoda jest stosowana w wielu systemach kontroli wersji, na przykład w **Subversion (SVN)**.

Jest mu poświęcony osobny wykład.



# Terminologia stosowana w systemach kontroli wersji





Co dalej?

