TODO:

1. Ogólny opis interfejsu, założenia projektowe, schemat działania (myślę że wystarczy to co Błazej mówił na prezentacji + grafiki z prezentacji ładnie opisane)
2. Budowa aplikacji, moduły wymagane, moduły opcjonalne +6./
3. Opis modułów
4. Opis zdarzeń
5. Konfiguracja (cxse)
6. Jak podmienić moduł?

# Moduły

Placeholder na spis treści

## Template

### OPis

aa

### Model danych

aa

### Sposób działania modułu

aa

### Zdarzenia

**Zdarzenia obsługiwane przez moduł**

|  |  |
| --- | --- |
| **Zdarzenie** | **Opis działania** |
|  |  |

**Zdarzenia wysyłane przez moduł ”**

|  |  |
| --- | --- |
| **Zdarzenie** | **Opis działania** |
|  |  |

### Ustawienia

aa

|  |  |
| --- | --- |
| **Ustawienie** | **Opis** |
|  |  |

## Arrow

### OPis

Zadaniem modułu “arrow” jest rysowanie strzałek łączących wierzchołki w grafie. Strzałki rysowane są poprzez kliknięcie myszką od wyjścia wierzchołka do wejścia wierzchołka w grafie. Ponadto moduł reaguje na zaznaczenie strzałki poprzez zmianę jej koloru

### Model danych

Strzałka dołączana jest do wejść i wyjść wierzchołków, które łączy, a więc nie posiada własnej reprezentacji danych.

### Sposób działania modułu

Moduł obsługuje dwa tryby działania: ControlFlow oraz DataFlow.

**ControlFlow:**

Moduł wyszukuje najkrótsze możliwe połączenie pomiędzy konkretnymi wierzchołkami, pomiędzy którymi narysowana ma zostać strzałka.

**DataFlow:**

Rysuje strzałkę pomiędzy wyjściami i wejściami konkretnych wierzchołków

## Atomic

### OPis

### Model danych

aa

### Sposób działania modułu

aa

### Zdarzenia

**Zdarzenia obsługiwane przez moduł**

|  |  |
| --- | --- |
| **Zdarzenie** | **Opis działania** |
|  |  |

**Zdarzenia wysyłane przez moduł ”**

|  |  |
| --- | --- |
| **Zdarzenie** | **Opis działania** |
|  |  |

### Ustawienia

aa

|  |  |
| --- | --- |
| **Ustawienie** | **Opis** |
|  |  |

## History

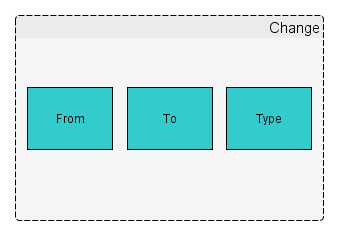
### OPis

Zadaniem modułu historii jest umożliwienie zapamiętania zmian dokonanych przez użytkownika, w celu ewentualnych ich cofnięcia bądź powtórzenia. Po zmianie moduł zapisuje zmienione dane w tablicy. Cofnięcie zmiany lub powtórzenie czynności możliwe jest za pomocą skrótów klawiszowych bądź też poprzez wybór odpowiedniej opcji z menu głównego.

Moduł jest powiązany z innymi modułami za pomocą zdarzeń, które są obsługiwane przez moduł historii.

### Model danych

Dane są przechowywane w tablicy i mają następującą postać:



From: Obiekt, w którym przechowywane są dane historyczne, to znaczy takie które są zachowane przed dokonaniem zmiany przez użytkownika.

To: Obiekt, w którym przechowywane są dane aktualne, to znaczy takie, które są w danym momencie wyświetlane, po zmianach dokonanych przez użytkownika.

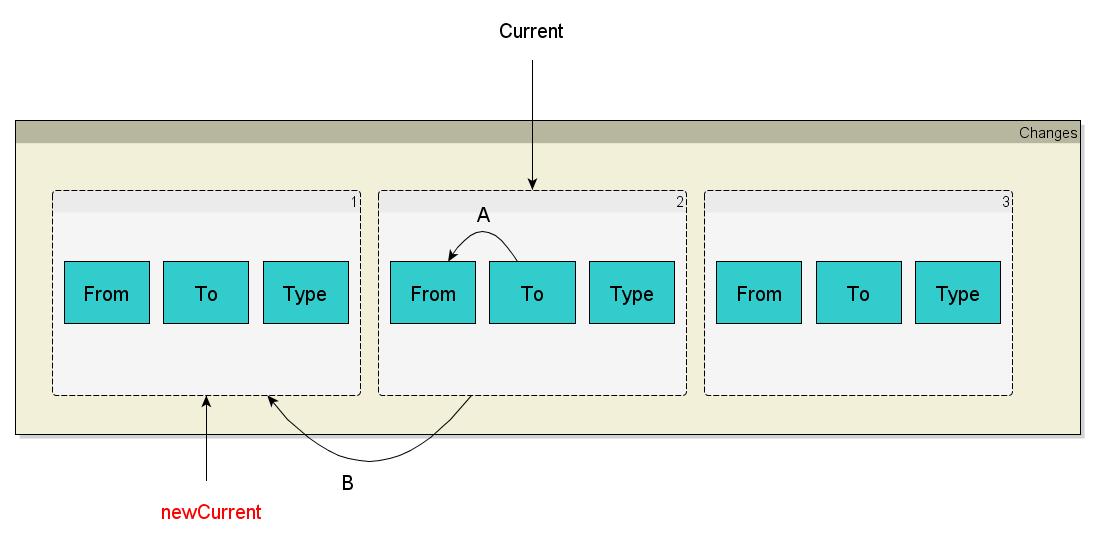
Type: Typ zmiany zapisany w postaci tekstowej pozwalający na rozróżnienie ostatniej dokonanej zmiany.

Tablica zmian zawierać może w sobie wiele powyższych obiektów. Ich liczba ustalana jest w osobnym pliku ustawień. Do wskazania na ostatnią dokonaną zmianę służy wskaźnik zdefiniowany w module.

### Sposób działania modułu

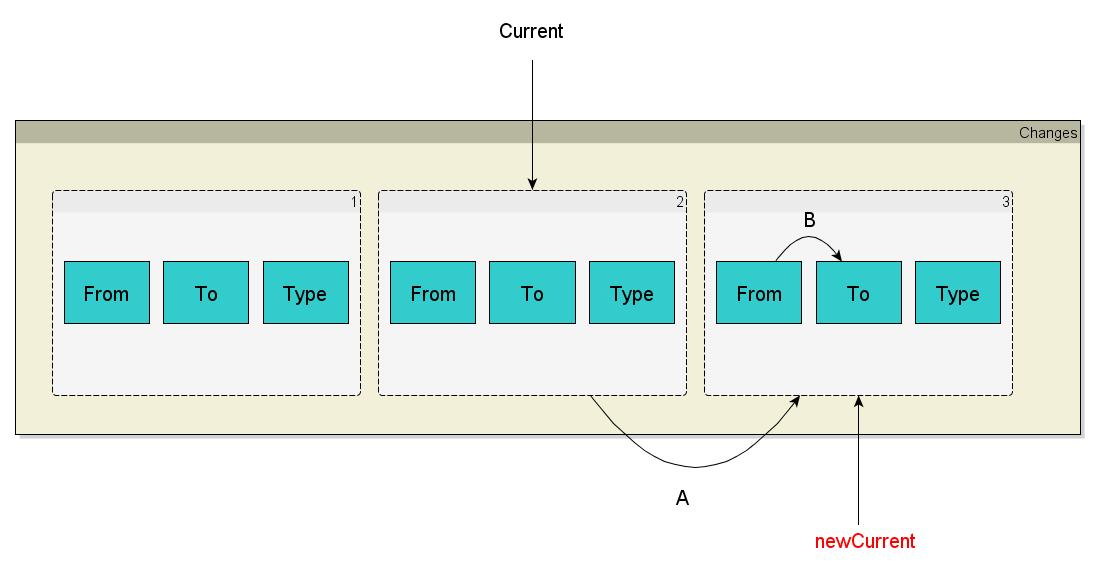
**Cofnięcie zmiany:**

Wskaźnik “Current” ustawiony jest na konkretnej pozycji. Przy cofnięciu zmiany zmieniają się dane, które są aktualnie wyświetlane. Dane, które należy podmienić determinowane są przez typ “Type”. Zmianę aktualnych danych reprezentuje na rysuku strzałka A. Następnie wskaźnik przesuwany jest na poprzednią pozycję w tablicy: strzałka B.



**Powtórzenie zmiany:**

Wskaźnik “Current” ustawiony jest na konkretnej pozycji. Przy powtórzeniu zmiany najpierw zmienia się położenie wskaźnika, co reprezentowane jest na rysunku przez strzałkę A, a następnie podmieniane są wyświetlane dane. Dane, które należy podmienić determinowane są przez typ “Type”. Zmianę aktualnych danych reprezentuje na rysuku strzałka B.

**Cofnięcie wszystkich zmian:**

Polega ono na przejściu z aktualnej zmiany wskazywanej przez wskaźnik do początku tablicy, co odbywa się na takiej samej zasadzie jak dla jednej zmiany, z tym że podmiana danych wykonywana jest w pętli aż osiągnięty zostanie początek tablicy.

**Powtórzenie wszystkich zmian:**  
  
Polega ono na przejściu z aktualnej zmiany wskazywanej przez wskaźnik na koniec tablicy zmian, co odbywa się na takiej samej zasadzie jak dla pojedynczej zmiany, z tym że podmiana danych dokonywana jest w pętli, aż osiągnięty zostanie koniec tablicy.

### Zdarzenia

**Zdarzenia obsługiwane przez moduł “history”:**

|  |  |
| --- | --- |
| **Zdarzenie** | **Opis działania** |
| historyChanged | Dodaje obiekt zmiany do tablicy zmian |
| shortcutBack | Cofa zmianę |
| shortcutForward | Powtarza zmianę |
| shortcutBackAll | Cofa wszystkie zmiany |
| shortcutForwardAll | Powtarza wszystkie zmiany |
| historyDeleteBlankNode | Usuwa wierzchołek jednocześnie przekazując go do modułu “blankNode” |

**Zdarzenia wysyłane przez moduł “history”**

|  |  |
| --- | --- |
| **Zdarzenie** | **Opis działania** |
| refreshView | Odświeża aktualnie wyświetlany widok |
| replaceSsdl | Podmienia wyświetlany graf. |

### Ustawienia

Ustawienia przechowywane są w oddzielnym folderze ustawień. Plik ustawień modułu *historySettings*:

|  |  |
| --- | --- |
| **Ustawienie** | **Opis** |
| historySize | Maksymalny rozmiar , jaki osiągnąć może tablica zawierająca zmiany w module historii. |

# Zdarzenia

### Konwencja

Do opisu przepływu danych między zdarzeniami przyjęto następującą konwencję:



Module1: Elipsa symbolizująca moduł wywołujący zdarzenie

InvokeEvent: Strzałka wychodząca z elipsy, wchodząca do prostokąta symbolizuje wywołanie zdarzenia przez moduł

Event: Prostokąt symbolizujący zdarzenie. Etykieta jest nazwą zdarzenia

ListenEvent Strzałka wychodząca z prostokąta, wchodząca do elipsy symbolizuje reakcję modułu na zdarzenie, na które moduł ten nasłuchuje.

Module2: Elipsa symbolizująca moduł obsługujący zdarzenie, na które nasłuchuje.

## engineloaded

### OPis

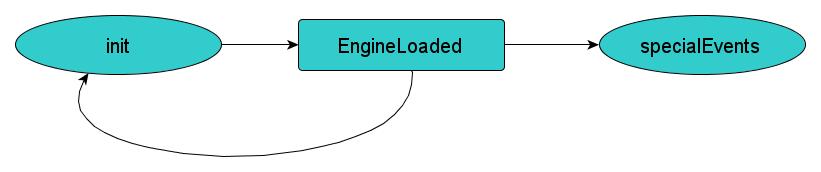
EngineLoaded jest to zdarzenie, które wywoływane jest przy inicjalizacji interfejsu. Ładuje SSDL.

### Powiązane moduły

**Zdarzenia powiązane z modułem**

|  |  |
| --- | --- |
| **Moduł** | **Rola** |
| Init | Wywoływacz, Słuchacz |
| specialEvents | Słuchacz |

### graf przepływu



## formsubmitted

### OPis

Zdarzenie wywoływane jest w momencie kliknięcia przycisku zapisz znajdującego się pod formularzem. Przekazuje dane z formularza zmodyfikowanego przez użytkownika do wierzchołka i zamyka formularz. Następuje również wyczyszczenie danych w celu uniknięcia ich nadpisania przy następnym wykorzystaniu formularza.

### Powiązane moduły

**Zdarzenia obsługiwane przez moduł**

|  |  |
| --- | --- |
| **Moduł** | **Rola** |
| formGenerator | Wywoływacz, Słuchacz |

### graf przepływu

## formsubmitted.jpg

## openform

### OPis

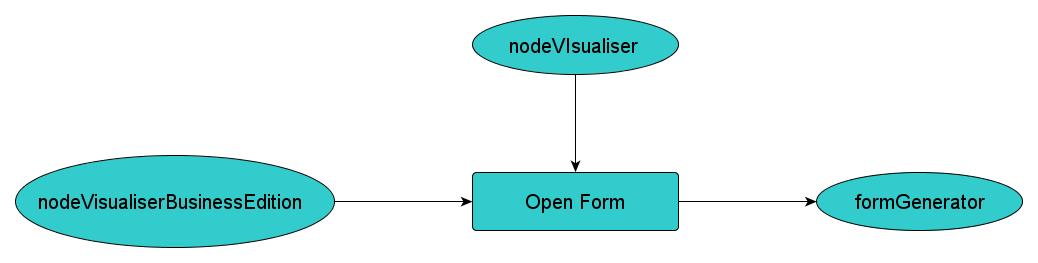
Zdarzenie reaguje na akcję otwarcia formularza. Akcja ta może być wywołana przez użytkownika za pośrednictwem modułów obsługujących wizualizację wierzchołków.

### Powiązane moduły

**Zdarzenia obsługiwane przez moduł**

|  |  |
| --- | --- |
| **Moduł** | **Rola** |
| nodeVisualiserBusinessEdition | Wywoływacz |
| nodeVisualiser | Wywoływacz |
| formGenerator | Słuchacz |

### graf przepływu



## SSDLloaded

### OPis

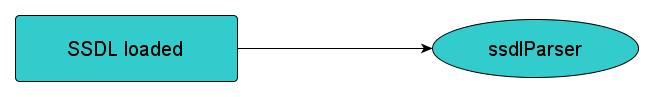
Zdarzenie wywoływane automatycznie przodczas zdarzenia EngineLoaded. Informuje, że ssdl został załadowany.

### Powiązane moduły

**Zdarzenia obsługiwane przez moduł**

|  |  |
| --- | --- |
| **Moduł** | **Rola** |
| ssdlParser | Słuchacz |

### graf przepływu



## ssdlparsed

### OPis

Zdarzenie informujące o fakcie przeparsowania SSDL

### Powiązane moduły

**Zdarzenia obsługiwane przez moduł**

|  |  |
| --- | --- |
| **Moduł** | **Rola** |
| ssdlParser | Wywoływacz |
| Init | Wywoływacz |
| Validator | Słuchacz |
| Atomic | Słuchacz |

### graf przepływu

