Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

3D UML

Dokumentácia k inžinierskemu dielu

Vedúci tímu: Ing. Ivan Polášek, PhD.
Členovia tímu: Bc. Adam Kulíšek, Bc. Matej Jenis, Bc. Rami Mtier, Bc. Tomas Hnojčík, Bc. Patrik Kolek, Bc. Boris Buček, Bc. Miroslav Šafárik
Akademický rok: 2015/2016

Obsah

1 Podiel na vytváraní dokumentu

Názov kapitoly	Autor
Úvod	Bc. Patrik Kolek, Bc. Adam Kulíšek, Bc. Miroslav Šafárik
Moduly systému	všetci členovia tímu
Testovanie	Bc. Boris Buček, Bc. Miroslav Šafárik
Výskumné výsledky	Bc. Rami Mtier
Príručky - inštalačná príručka, používateľská príručka	všetci členovia tímu
Technická dokumentácia	všetci členovia tímu

Tabuľka 1: Autori jednotlivých kapitol dokumentu

2 Úvod

Tento dokument bol vytvorený za účelom zdokumentovania výstupu tímového projektu č. 13 s názvom Triskaidekaphobiacs v akademickom roku 2015/2016 na predmete Tímový projekt. Dokument je rozdelený na niekoľko celkov. V úvode sa čitateľ oboznámi s cieľmi projektu, ktoré boli spolu s vlastníkom projektu stanovené v priebehu prvých týždňov zimného semestra, a získa základný prehľad o vyvíjanom systéme pomocou náčrtu jeho architektúry. Nasledujúca kapitola Moduly Systému je rozdelená na štyri celky (analýza, návrh, implementácia a testovanie), a obsahuje hlavne podrobnejšie opisy jednotlivých komponentov a diagramy, ktoré zobrazujú ich štruktúru a väzby. Skúsenosti, ktoré sme nadobudli pri práci s metamodelom, sú zhrnuté v kapitole Výskumné výsledky. V prílohe sa nachádza inštalačná príručka, ktorú sme v prvých týždňoch museli spracovať na to, aby si každý člen tímu individuálne mohol spustiť projekt na vlastnom stroji a v letnom semestri sme ju aktualizovali o doplnené funkcionality. Súčasťou inžinierskeho diela je aj technická dokumentácia a používateľská príručka - obe vo forme prílohy.

2.1 Globálne ciele projektu

V rámci zimného a letného semestra bola naším hlavným cieľom integrácia existujúcich riešení (prototyp sekvenčného diagramu s editačnými funkciami správ, prototyp kombinovaných fragmentov a prototyp diagramu aktivít) do navrhnutej architektúry MMVCC. Práve podľa tejto architektúry bol vytvorený prototyp pre diagram aktivít, ku ktorému sme počas dvoch semestrov pridali vo formáte nového balíku riešenie pre sekvenčný diagram. Projekt bol na počiatku v takom stave, že viacero jeho častí bolo implementovaných ako samostatné nezávislé prototypy, ktoré bolo potrebné osobitne nakonfigurovať vo vývojovom prostredí a spustiť ako osobitnú aplikáciu. Od stiahnutia zdrojových kódov aplikácie po ich kompilovanie do spustiteľnej formy mohlo nastať množstvo komplikácií, čím sa tento proces značne a zbytočne predĺžil. Pokiaľ by používateľ chcel spustiť iný prototyp projektu, musel by takýto proces vykonať pre každý z prototypov separátne. Práve tento problém nás motivoval pre vytvorenie jednotného riešenia, ktoré podporuje prácu s viacerými typmi diagramov v 3D prostredí. Spájaním prototypov diagramu aktivít, sekvenčného diagramu a kombinovaných fragmentov sme chceli vytvoriť prehľadný, jednoducho inštalovateľný systém, do ktorého je možné jednoducho zapracovať akúkoľvek ďalšiu funkcionalitu. Pre tento účel bolo v prvom rade potrebné implementovať `Metamodel` a `MetamodelController` pre jednotlivé diagramy tak, aby spadali do architektúry prototypu diagramu aktivít.

2.1.1 Produktový backlog - Big picture

- Zobrazenie sekvenčného diagramu v prostredí a architektúre aktivity diagramu
- Implementácia fragmentov do novej architektúry sekvenčného diagramu
- Implementácia editačných funkcií
- Refaktorizácia zdrojového kódu
- Vylepšenie insertu
- Lepšie komentáre
- Zrušenie nepotrebných namespaces

- Podrobná dokumentácia
    - Diagramy tried zachytávajúce štruktúru prototypu
- Webová stránka 3D LAB
- Eliminácia branchov na bitbuckete (z troch vznikne jeden)

2.1.2 Prehľad definovaných používateľských príbehov

V tejto časti "big picture" kapitoly sú definované jednotlivé používateľské príbehy. Kurzívou sú zvýraznené príbehy, ktoré neboli náplňou tohto tímového projektu (a teda nie sú v prototype zrealizované). Kurzívou neoznačené príbehy sú v súčasnom prototype zrealizované, a teda používateľovi, resp. vlastníkovi produktu dostupné. Pod interakciou sa v používateľskom príbehu rozumie správa. Príbeh číslo 8 bol zrealizovaný, avšak je potrebné ho ešte čiastočne doladiť (stredný bod správy sa pri posune dostane mierne napravo od textu správy).

Ako používateľ 3D UML

- **Základné funkcie editovania**

  1. Ako používateľ 3D UML prototypu potrebujem pri vstupe do systému vidieť základné menu vstupu do jednotlivých druhov diagramov, kde sa mi hneď po stlačení voľby „Class diagram", „Sequence diagram" alebo „Activity diagram" zobrazia ich editory.
  2. Pri vstupe do editora "Sequence diagram" chcem vidieť ponuku jednotlivých možných elementov na vloženie do diagramu (čiary života - lifelines a interakcie - interactions).
  3. Musím mať možnosť vkladať nové prvky (interakcie a čiary života).
  4. Potrebujem zvolené prvky aj vyraďovať.
- **Presuny**

  5. Chcem aj presúvať čiary života v rámci jednej vrstvy, pričom ťahajú so sebou aj interakcie.
  6. Potrebujem presúvať čiary života aj medzi vrstvami, pričom ťahajú so sebou aj interakcie.
  7. Chcem aj presúvať interakcie v rámci jednej vrstvy hore/dole.
  8. Potrebujem presúvať aj interakcie medzi vrstvami.
  9. Chcem aj presúvať interakcie v rámci jednej vrstvy zmenou zdrojovej čiary života a/alebo cieľovej čiary života.
  10. Potrebujem presúvať interakcie zmenou zdrojovej čiary života a/alebo cieľovej čiary života aj medzi vrstvami.
- **Fragmenty**

  11. Musím mať možnosť pridávať nielen objekty a interakcie, ale aj jednoduché fragmenty typu Loop a Opt.
  12. Potrebujem pridávať aj zložené fragmenty typu Alt a Par (ako počet operandov).
  13. Chcem vnárať fragmenty jeden do druhého.
  14. Chcem aby sa zložené fragmenty typu Alt a Par vedeli v animácii zobraziť za sebou v priestore, alebo pod sebou v rámci jednej vrstvy.
  15. Okrem pridania fragmentov ich potrebujem aj odoberať.
  16. Potrebujem ich aj presúvať hore dole po vrstve.
  17. Potrebujem ich aj rozširovať alebo zužovať vo vrstve.

Ako správca kódu:

18. Potrebujem premigrovať sekvenčný diagram do architektúry diagramu aktivít po minuloročnom tíme 04/2014, čo znamená MMVCC.
19. Páčilo by sa mi owrapovať pomocou adapter/strategy 3D Controller, View a možno aj 3D Model kvôli perspektívnej zmene.
20. Je potrebné aj rozbehať metamodel sekvenčného diagramu ako aj pridať správu fragmentov (Richard Belan).
21. Bolo by pekné refaktorovať kód, vylepšiť insert funkciu, opraviť Popup okno.

Ako správca dokumentácie k produktu:

22. Potrebujem podrobnú technickú dokumentáciu novej architektúry.
23. Potrebujem inštalačnú príručku.
24. Potrebujem podrobnú technickú dokumentáciu jednotlivých funkcií.

Ako správca 3D laboratória

25. Potrebujem 3D LAB webovú stránku.
26. Chcel by som mať možnosť aktualizovať obsah.

2.1.3 Ciele pre zimný semester

Hlavným cieľom zimného semestra bol návrh metamodelu pre sekvenčný diagram, pomocou ktorého budeme vykresľovať jednotlivé elementy v 3D priestore. Aby bolo možné vykresľovať tieto elementy, bola potrebná implementácia pre zobrazenie sekvenčného diagramu v rámci nižšie uvedenej architektúry (obrázok 1). Zobrazenie spočíva vo vykreslení scény, do ktorej bude možné vkladať elementy pre sekvenčný diagram (čiara života, správa a kombinované fragmenty). Tieto elementy sú k dispozícii v rámci menu, ktoré je súčasťou vykreslenia prázdnej scény.

Medzi ciele pre zimný semester patrilo aj rozšírenie prototypu pre diagram aktivít, do ktorého chceme pridať existujúce prototypy pre sekvenčné diagramy (editačné funkcie a fragmenty). Rozšírenie prototypu je potrebné pre vytvorenie nezávislých komponentov jednotlivých diagramov pre prácu v 3D priestore. Tieto komponenty by mali byť nezávislé od typu diagramu (aktivít, tried, sekvenčný diagram, atď.). Mali by sa starať len o vytváranie a vykresľovanie elementov v 3D priestore pre vybraný diagram.

2.1.4 Ciele pre letný semester

V rámci letného semestra bolo za hlavný cieľ stanovené dorobenie práce s editačnými funkciami pre sekvenčný diagram, rešpektujúc pritom navrhnutý metamodel zo zimného semestra.

Editačné funkcie budú realizované pre čiary života (angl. lifelines), správy (angl. messages) a kombinované fragmenty (angl. combined fragments). Pod editačnými funkciami sa rozumie nasledujúce:

- pre čiary života:
    - pridávanie nových čiar života na ľubovoľné miesto v scéne,
    - pohyb čiar života po scéne v rôznych smeroch,
    - mazanie čiar života,
    - mazanie čiar života spolu s obsahom (napr. so správami, ktoré sú k nim pripojené).

- pre správy:
    - pridávanie správ medzi čiary života na miesto vybrané používateľom,
    - posúvanie správy vo vertikálnom smere,
    - zmena cieľa/zdroja správy,
    - mazanie správy.
- pre kombinované fragmenty:
    - pridávanie kombinovaných fragmentov obalením vybraných správ,
    - škálovanie, t.j. zväčšovanie a zmenšovanie kombinovaných fragmentov,
    - pridávanie operandov do kombinovaných fragmentov a ich úprava,
    - mazanie operandov z kombinovaných fragmentov,
    - mazanie kombinovaných fragmentov.

Všetky tieto editačné funkcie sú realizované nad metamodelom, t.j. akékoľvek pohyby so správami, resp. čiarami života, akékoľvek operácia nad kombinovanými fragmentami výusťujú v aktualizáciu metamodelu do takého stavu, aký je reprezentovaný aktuálnou vizuálnou podobou scény.

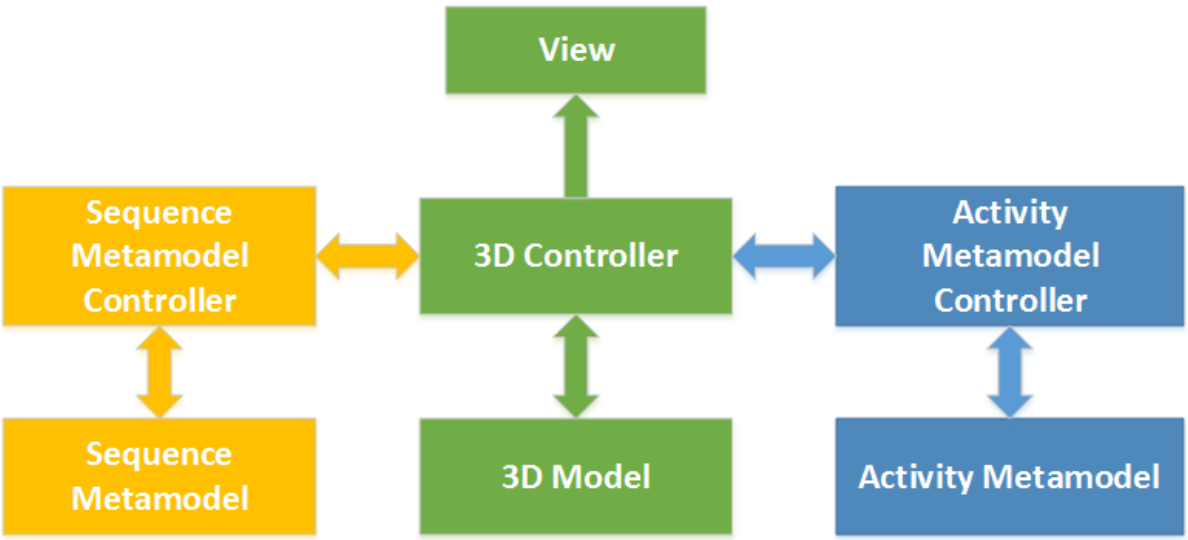
Ďalším čiastkovým cieľom pre letný semester bolo popri písaní nového kódu refaktorizovať starší, resp. konvencie porušujúci kód. V tomto smere sme sa snažili primárne o refaktorizáciu, ktorá vylepšuje kvalitu kódu z lexikálneho hľadiska.

2.2 Celkový pohľad na systém

Na obrázku 1 je znázornený jednoduchý náhľad do architektúry prototypu po integrácii prototypov sekvenčného diagramu (kombinované fragmenty a editačné funkcie) do prototypu diagramu aktivít. Obrázok je rozdelený do troch farebne označených častí.

Zelenou farbou sú znázornené komponenty, ktoré sú spoločné pre všetky typy diagramov. Ich komunikáciou a spoluprácou sa zabezpečuje vykreslenie jednotlivých diagramov.
V pravej časti obrázka sú komponenty, zobrazené modrou farbou, špecifické pre diagram aktivít. Návrh a implementácia týchto komponentov je výsledkom práce minuloročného tímového projektu. Podobným spôsobom boli navrhnuté komponenty pre sekvenčné diagram (žltá časť).

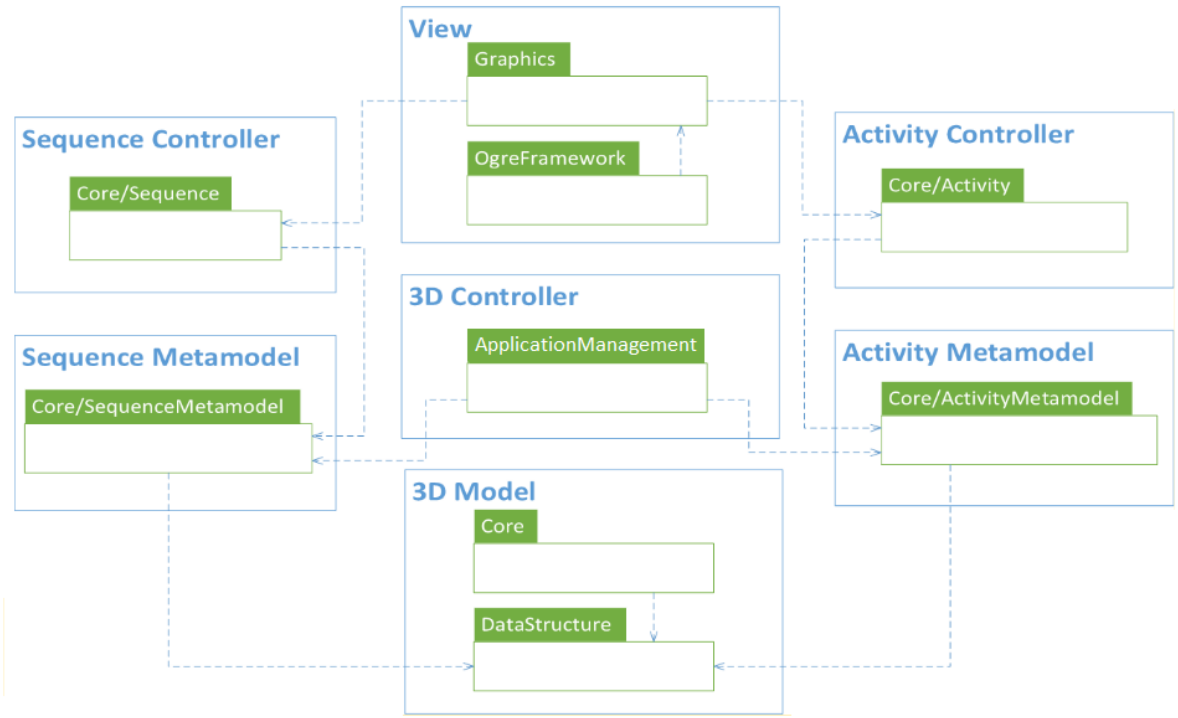
Jadrom našej práce a naším hlavným príspevkom je krabička reprezentujúca metamodel sekvenčného diagramu (Sequence Metamodel). Diagramaticky je metamodel pre sekvenčný diagram uvedený v kapitole Moduly systému v časti Návrh. Implementáciou metamodelu sa z "čiarok" a "šípok" vo vizuálnej podobe sekvenčného diagramu stala strojovo spracovateľná štruktúra, ktorá môže byť použitá v budúcnosti aj pre výskumné účely, napr. na odporúčanie návrhových vzorov alebo strojové vyhľadávanie zlých návrhových rozhodnutí a odporúčanie vzorov pre ich opravu (proces, ktorý sa zvykne označovať ako refaktorizácia do vzorov, angl. refactoring to patterns, pričom bližšie sa jej venuje napr. Joshua Kerievsky). Súčasťou celkového pohľadu na systém je sú aj triedne diagramy, pre ich komplexnosť sú však uvedené v technickej dokumentácii.



Obrázok 1: Architektúra prototypu po integrácii sekvenčného diagramu

3 Moduly Systému

Systém je zložený z niekoľkých balíkov, ktoré zoskupujú zdrojové kódy do logických celkov.



Obrázok 2: Balíková štruktúra zdrojového kódu s hlavnými komponentami

- **View:** úlohou `View` komponentu je vykresľovanie elementov diagramu a vrstiev v 3D priestore. Tento 3D priestor je definovaný scénou, v ktorej sa môže používateľ

pohybovať. Rovnako do tejto scény môže pridávať elementy, ktoré má k dispozícii v rámci menu pre konkrétny typ diagramu. Aby sme boli schopní tieto elementy vykresliť, musí mať každý element definovaný svoj vlastný algoritmus a informácie o vykreslení ako napríklad typ použitého písma. Vykresľovanie elementov sekvenčného diagramu je definované a implementované v triedach v balíku `Graphics`. Pre samotné grafické vykreslenie elementov do scény sa využíva knižnica Ogre, ktorej základné grafické operácie sú definované v balíku `OgreFramework`.

- **3D Controller:** stará sa o vytváranie elementov diagramu. Rovnako jeho úlohou je aj kontrolovať stav aplikácie, v ktorej sa nachádzame. Všetky triedy, ktoré sa starajú o riadenie a kontrolu stavu aplikácie, v ktorej sa nachádza, sú zahrnuté v balíku `ApplicationManagement`.
- **3D Model:** nakoľko pracujeme v 3D priestore, musíme poznať štruktúru dát elementov pre túto dimenziou. Všetky modely elemetov sú zachytené v balíkoch `Core` a `Datastructure`.
- **Activity/Sequence Controller:** ich úlohou je vytvárať a modifikovať model elementy v rámci UML diagramu (pridávanie, mazanie a aktualizácia elementov).
- **Activity/Sequence Metamodel:** určujú vzťahy medzi komponentami jednotlivých diagramov UML diagram.

3.1 Analýza

V tejto kapitole prezentujeme analýzu prototypu diagramu aktivít, ktorý bol vytvorený v rámci minuloročného tímového projektu, a tiež analýzu editačných funkcií pre elementy sekvenčného diagramu.

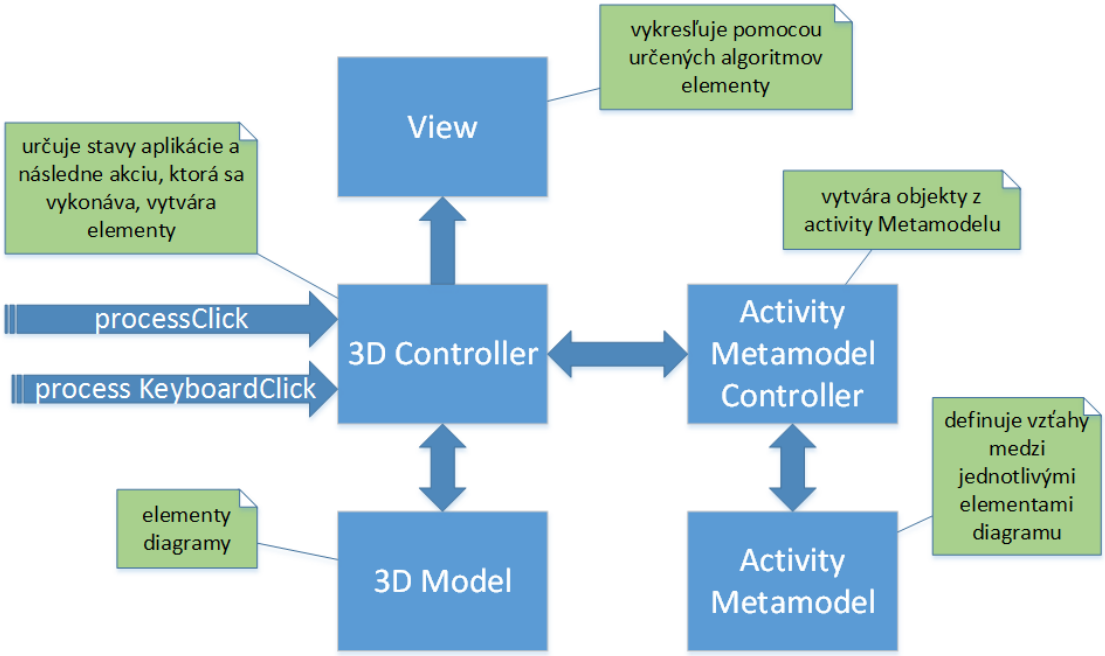
3.1.1 Prototyp diagramu aktivít

V tejto časti je opísaný existujúci prototyp diagramu aktivít. Tento prototyp je funkčný program písaný v jazyku C++ s použitou grafickou knižnicou Ogre[1]. Použitý architektonický štýl je modifikovaný MVC.

3.1.1.1 Opis architektonického štýlu

Modifikácia MVC architektonického štýlu spočíva v pridaní jedného extra ovládača (angl. controller) a jedného modelu. Pridaný model nazývame metamodel a slúži na určenie vzťahov medzi jednotlivými elementami. Údaje z metamodelu môžu byť využité napríklad na automatické generovanie diagramov z prototypu do nástroja Enterprise Architect a naopak. Pridaný ovládač je Activity Metamodel Controller, ktorý vytvára a modifikuje metamodel elementy v diagrame aktivít. Druhým modelom, s ktorým pracujeme je 3D Model, ktorý definuje údaje v elemente, informácie o umiestnení elementov do scény a ich grafickú reprezentáciu. Obrázok zobrazuje prepojenie popísaných elementov vzniknutého architektonického štýlu MMVCC.

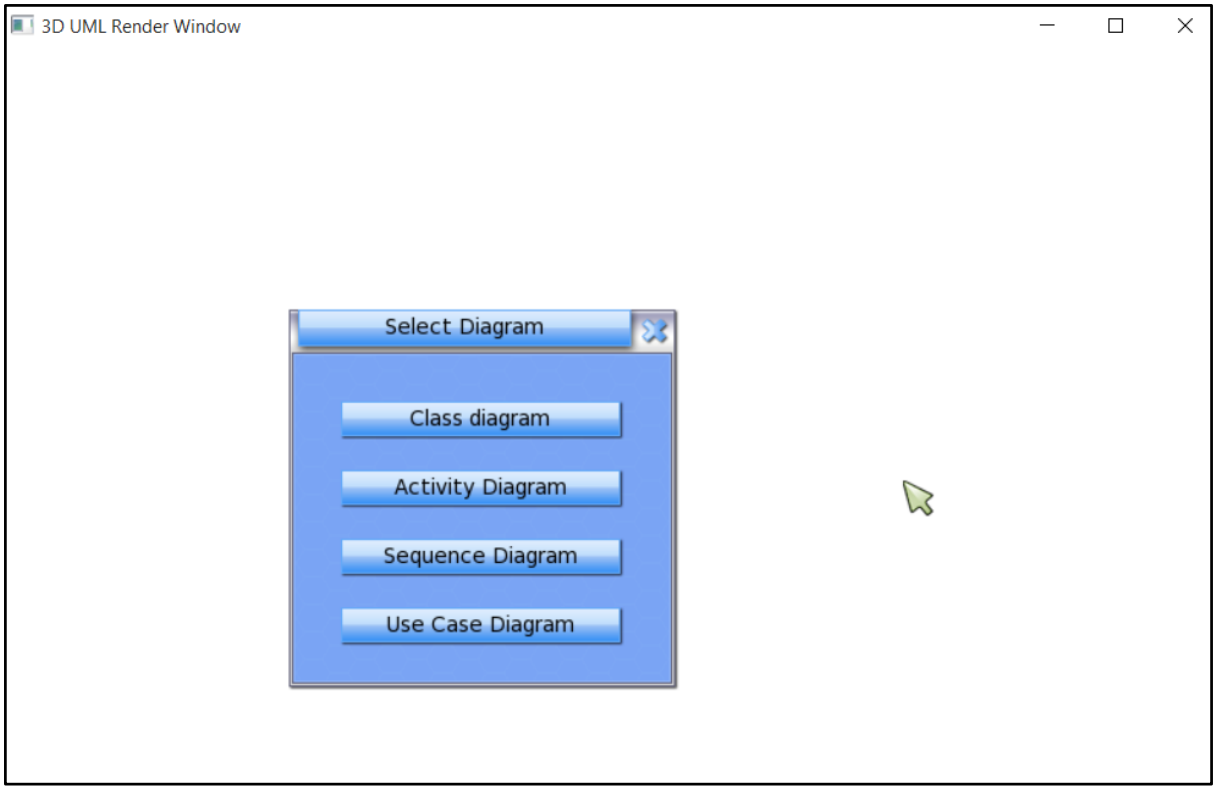
[1] http://www.ogre3d.org/



Obrázok 3: Architektúra prototypu diagramu aktivít

3.1.1.2 Komponent 3D controller

Aplikácia sa môže nachádzať v rôznych stavoch, v ktorých sú . Všetky stavy sú uložené v triede `ApplicationStateManager`, ktorá používa architektonický vzor `Singleton`. Pri spustení programu sa vyvolá v triede `Main` metóda `CreateScene`, ktorá určí aplikácií stav `SelectDiagramContext`. Tento stav vytvorí úvodnú obrazovku, z ktorej je možné vyberať typ vytváraného diagramu (obrázok 4). Po zvolení vhodného diagramu sa vykreslí prázdna scéna, v ktorej je možné vytvárať 3D UML diagram. V prototype z minuloročného tímového projektu je funkčný len diagram aktivít. Jednotlivé stavy sú definované v balíku `ApplicationStates`.

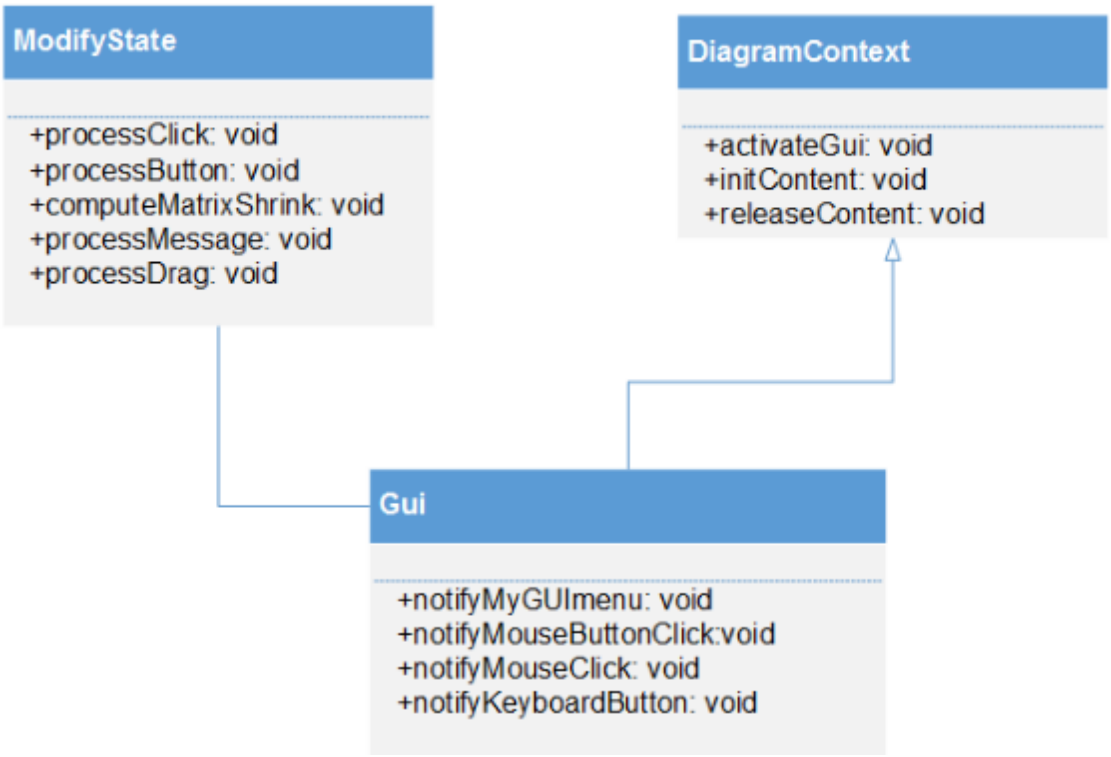


Obrázok 4: Používateľské menu

Pri interakcií používateľa s aplikáciu sa nachádza v editačnom stave. V editačnom móde je možné vytvárať potrebné diagramy. Tento proces sa v prototype deje pomocou triedy `ModifyState`. Táto trieda inicializuje akciu pomocou metód `procesClick` (kliknutie tlačidlom na myši) a `procesButton` (kliknutie klávesou). V metódach nájde vytvorený objekt typu `DiagramContext`, ktorý v prípade diagramu aktivít je trieda `Gui`. Následne vyvoláva metódy podľa typu akcie

- **`notifyMyGuiMenu()`** - reaguje pri kliknutí na Menu v aktivity diagrame
- **`notifyKeyboardClick()`** - reaguje pri kliknutí na tlačidlo v klávesnici
- **`notifyMouseButtonClick()`** - reaguje pri kliknutí na tlačidlo
- **`notifyMouseClick()`** - reaguje pri kliknutí na vykreslený priestor

Aplikácia si v globálnych premenných pamätá posledné kliknuté tlačidlo, súradnice bodu na vykreslenie a typ elementu, s ktorým sa pracuje. Vyššie opísaná štruktúra v aplikácií sa nachádza na obrázku 5.



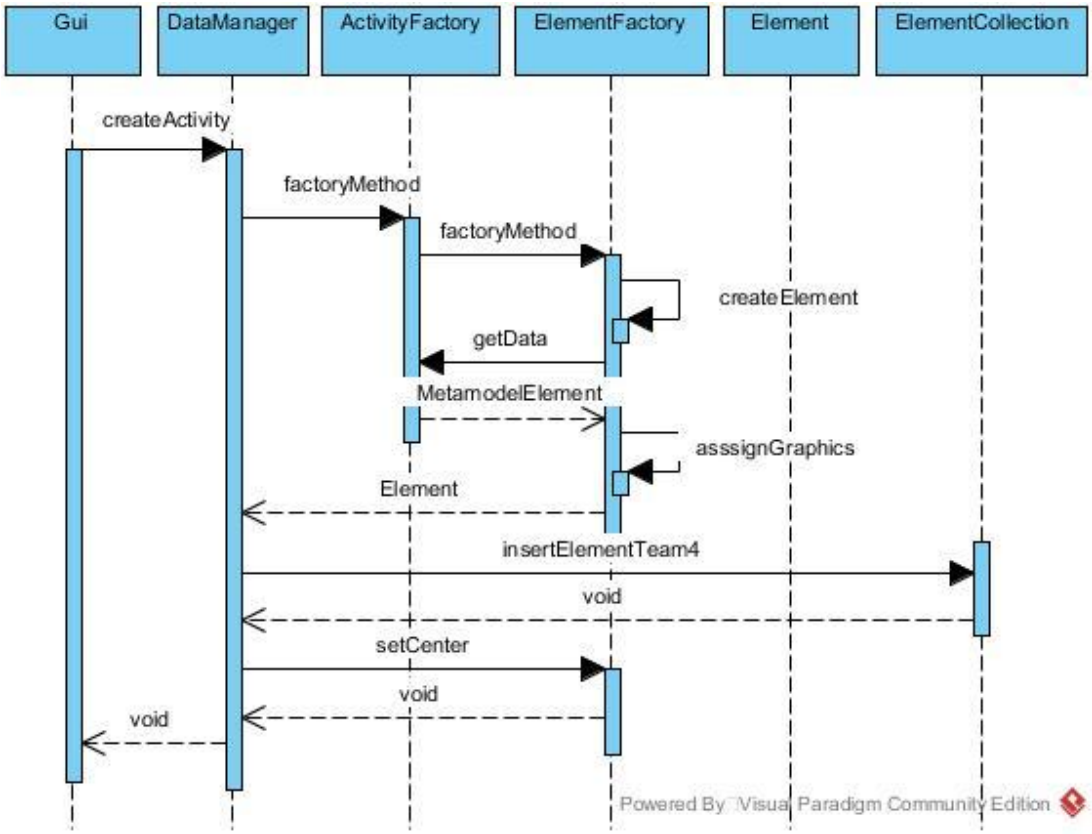
Obrázok 5: Štruktúra aplikácií

3.1.1.3 Vkladanie elementu

Elementy analyzovaného prototypu sa ukladajú do pamäte vo forme dvojíc (`Element`, `MetamodelElement`):

- **`Element`** - triedy v štruktúre `Core/activity`, ktoré dedia od triedy Element
- **`MetamodelElement`** - elementy v štruktúre `Core/Metamodel`, ktoré dedia od triedy `Metamodel`

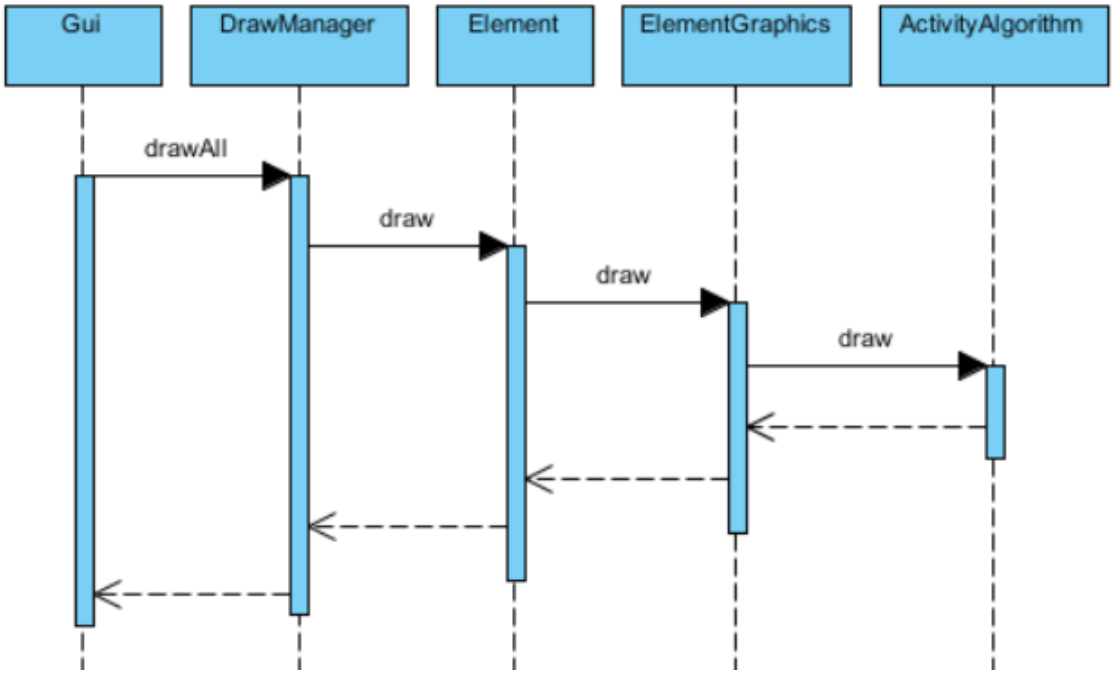
Dvojice objektov sa ukladajú do singleton štruktúry triedy `ElementCollection`. Triedy dediace od triedy `ElementFactory` vytvárajú elementy diagramu aktivít. Všetky komponety sa nachádzajú v statickej štruktúre obsiahnutej v triede `ElementCollection`. Zjednodušený proces ukladania je zobrazený na obrázku 6 vo forme sekvenčného diagramu.



Obrázok 6: Vkladanie aktivity do Model komponentu

3.1.1.4 Vykresľovanie jednotlivých komponetov

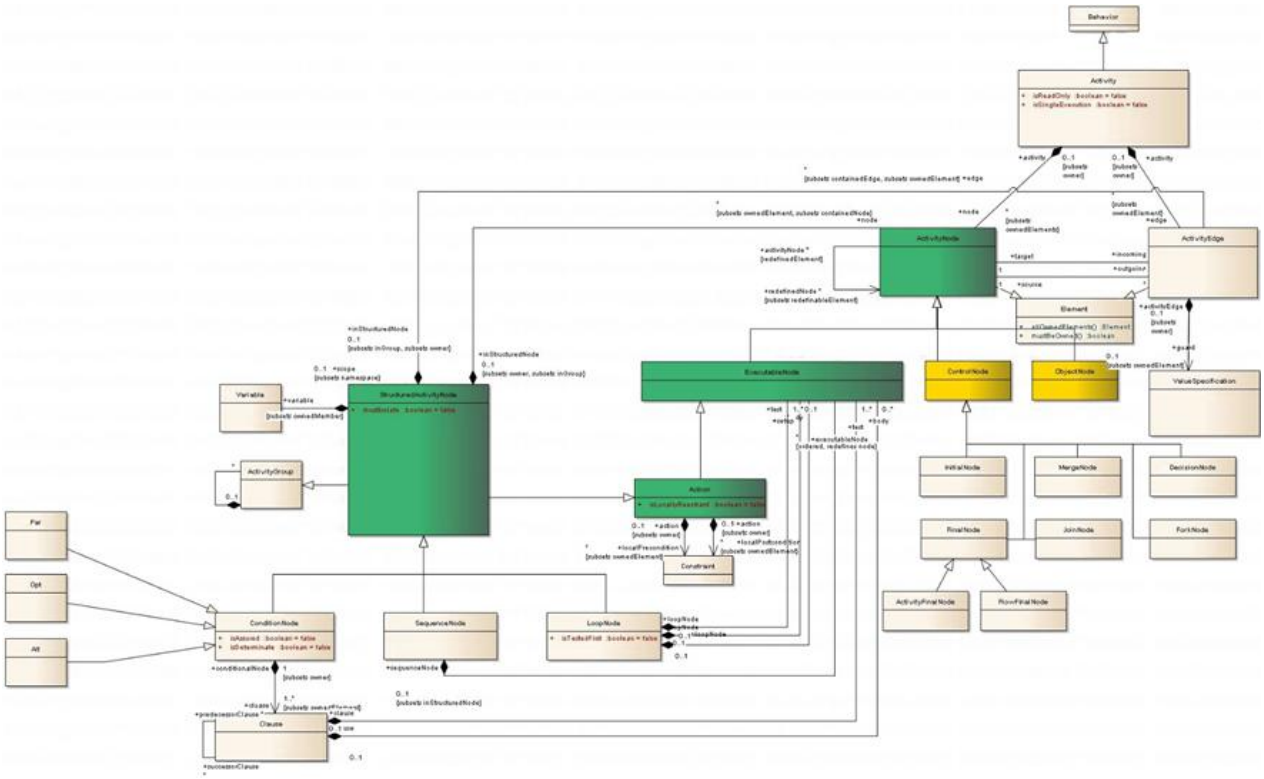
Po vložení komponentov sa v objekte typu `DrawManager` vyvolá metóda, ktorá vykreslí všetky objekty uchovávané v štruktúre `ElementCollection`. Pre každý pár (`Element`, `MetamodelElement`) sa vyvolá metóda `draw` v objekte typu `Element`. Každý `Element` má agregovaný `ElementGraphics`. Ten obsahuje kresliaci algoritmus. Proces je zobrazený na obrázku 7. Nevýhodou zostáva vykresľovanie všetkých objektov, čo nie je veľmi efektívne.



Obrázok 7: Sekvenčný diagram algoritmu vykresľovania

3.1.1.5 Metamodel diagramu aktivít

Metamodel je štandardne definovaný pre diagram aktivít vo forme definovanej na obrázku 8.



Obrázok 8: Diagram metamodelu diagramu aktivít

3.1.2 Prototyp sekvenčného diagramu

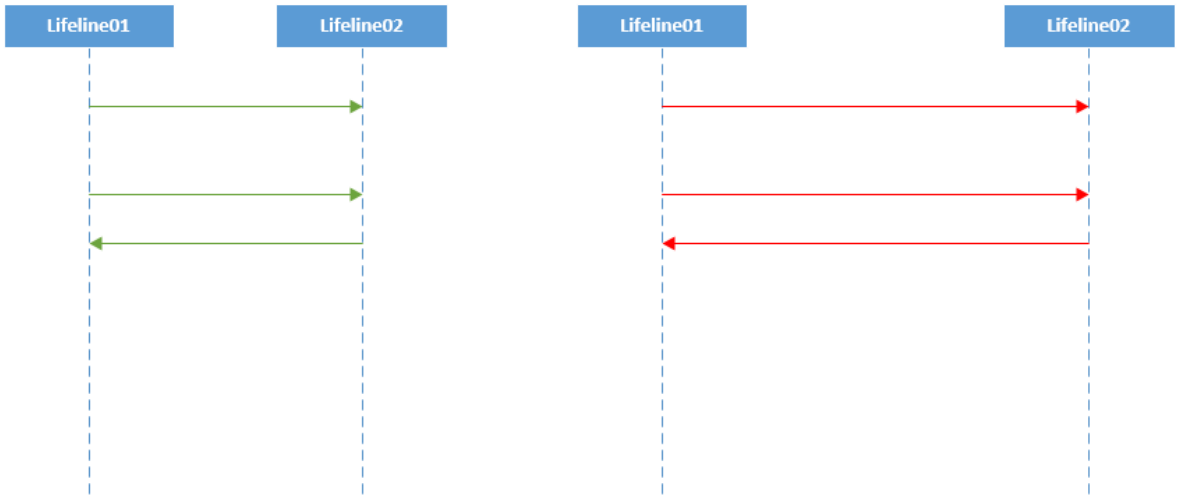
Architektúra sekvenčného diagramu je veľmi podobná diagramu aktivít. Na rozdiel od diagramu aktivít používa klasickú architektúru MVC. Sekvenčný diagram obsahuje jeden ovládač, ktorý pracuje s model komponentom definovaným v `Core` ako triedy dediace od triedy `Element`. Informácia o štruktúre diagramu sa na rozdiel od diagramu aktivít neuchováva v metamodeli ale v elementoch jednotlivých tried modelu. Triedy dediace od `ElementFactory` vytvárajú elementy z jednotného modelu. Analyzované procesy ako vykresľovanie prvkov, prechádzanie stavov sú identické s diagramom aktivít. Veľkým rozdielom je aj forma ukladania elementov, ktorá nevyužíva dvojicu (element `model-u` a element `metamodel-u`), ale pomocou dedenia metamodelových elementov od modelových elementov s informáciami vytvára ľahko ukladajúcu štruktúru.

3.1.3 Editačné funkcie pre čiary života

3.1.3.1 Horizontálny pohyb čiary života

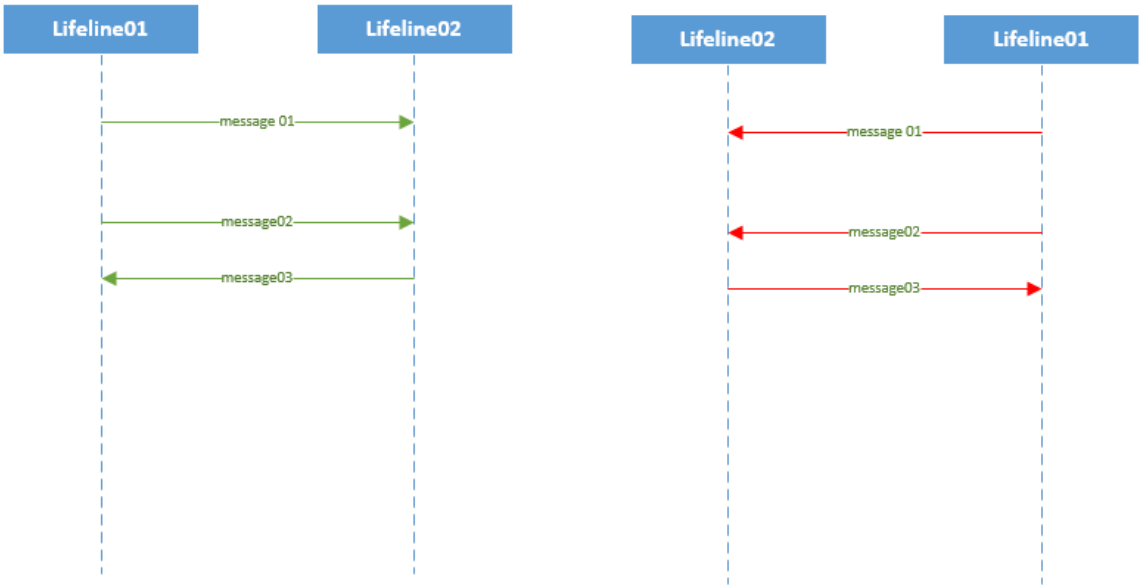
Používateľ môže pohybovať čiarami života v dvoch smeroch – vľavo a vpravo. V takomto prípade sa presúva iba označené okno čiary života. Rovnako sa musia tomuto presunu prispôsobiť správy, ktorých stav môže byť nasledovný:

1. Zväčšenie/zmenšenie dĺžky zobrazenej správy - prispôsobenie veľkosti správ podľa pozície, o akú používateľ posunul okno čiary života.



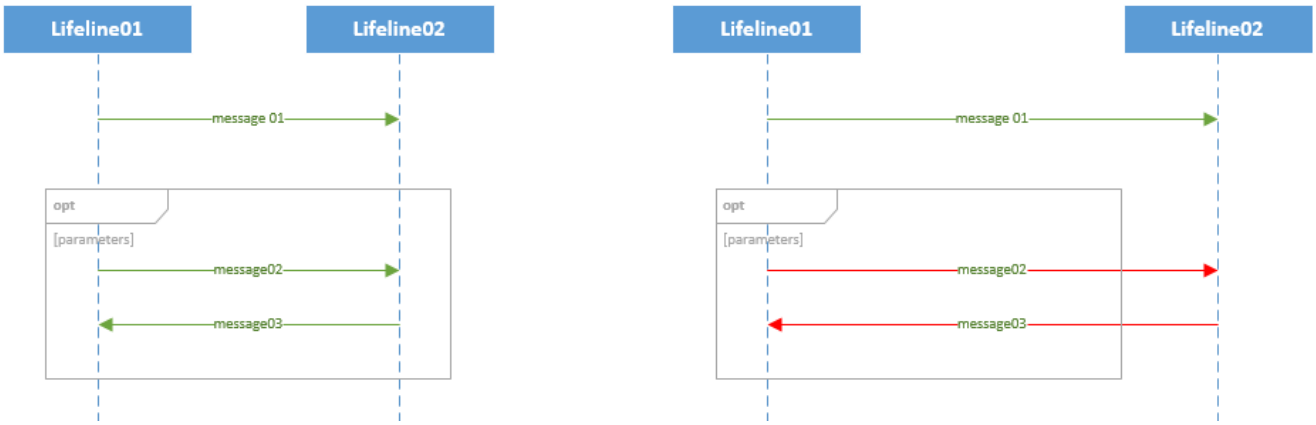
Obrázok 9: Zväčšenie/zmenšenie dĺžky zobrazovanej správy

2. Zmena orientácie správy – v prípade, že sa vymenia pozície dvoch čiar života, musia sa tejto výmene prispôsobiť a prekresliť aj všetky správy , ktoré obsahuje označená čiara života



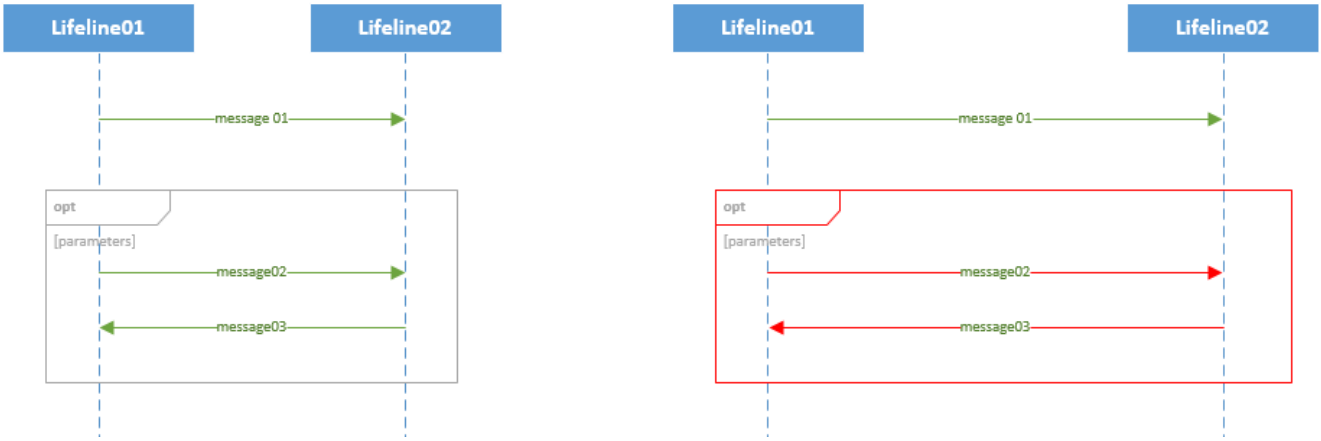
Obrázok 10: Zmena orientácie správy

3. Čiara života obsahuje fragment – v prípade posunu čiary života, sa okno fragmentu:
    a. Nemení, mení sa len dĺžka správ, ktoré môžu vyjsť aj mimo okno fragmentu



Obrázok 11: Posun čiary života bez zmeny veľkosti fragmentu

    b. Mení spolu so správami, ktoré obsahuje.

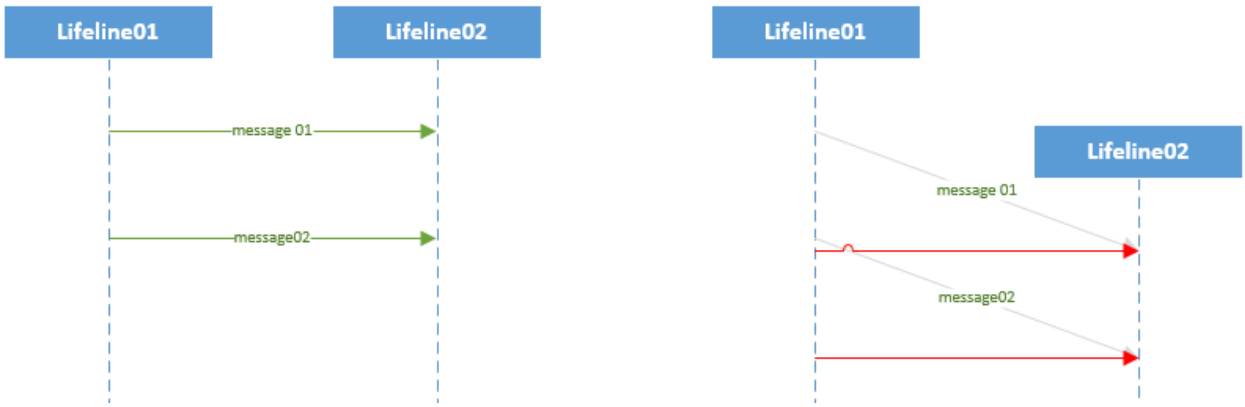


Obrázok 12: Posun čiary života so zmenou veľkosti fragmentu

3.1.3.2 Vertikálny pohyb čiary života

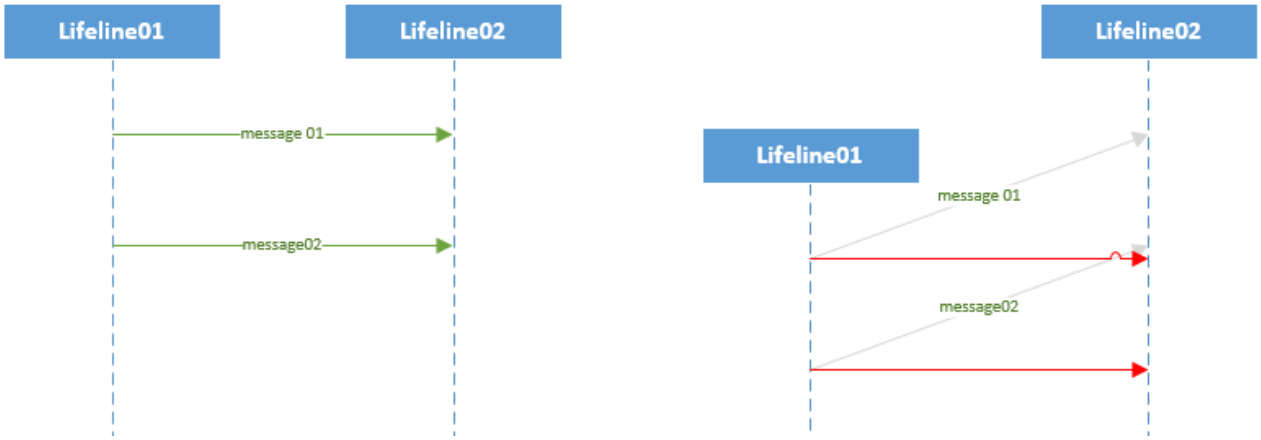
Ďalšou možnosťou je pohybovanie s vybranou čiarou života vo vertikálnom smere (hore/dole). V takomto prípade sa musia tejto vertikálnej zmene prispôsobiť aj správy, ktoré sú prepojené s označenou čiarou života. Môžu nastať 2 prípady:

1. Posunutá čiara života obsahuje bod konca správy (`MessageEnd`). V takomto prípade sa musí zmeniť pozícia zdroja správy (pozícia `MessageOccuranceSpecification`) na základe pozície bodu konca správy.



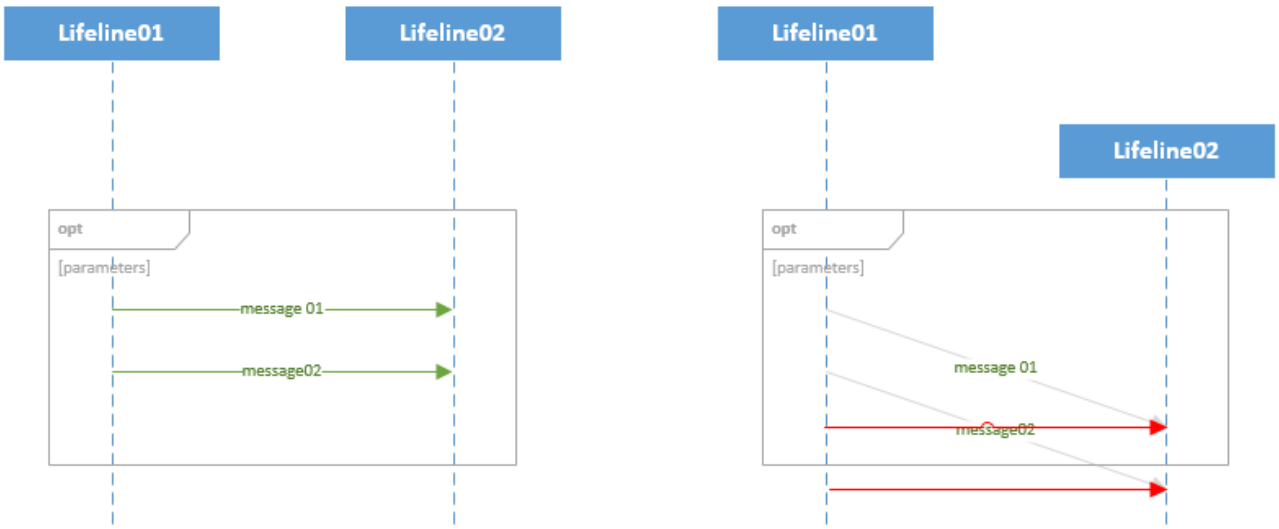
Obrázok 13: Vertikálny pohyb čiary života smerom nadol

2. Posunutá čiara života obsahuje začiatok správy (`MessageOccuranceSpecification`). Na základe tohto bodu sa musí zmeniť pozícia bodu konca správy (`MessageEnd`).



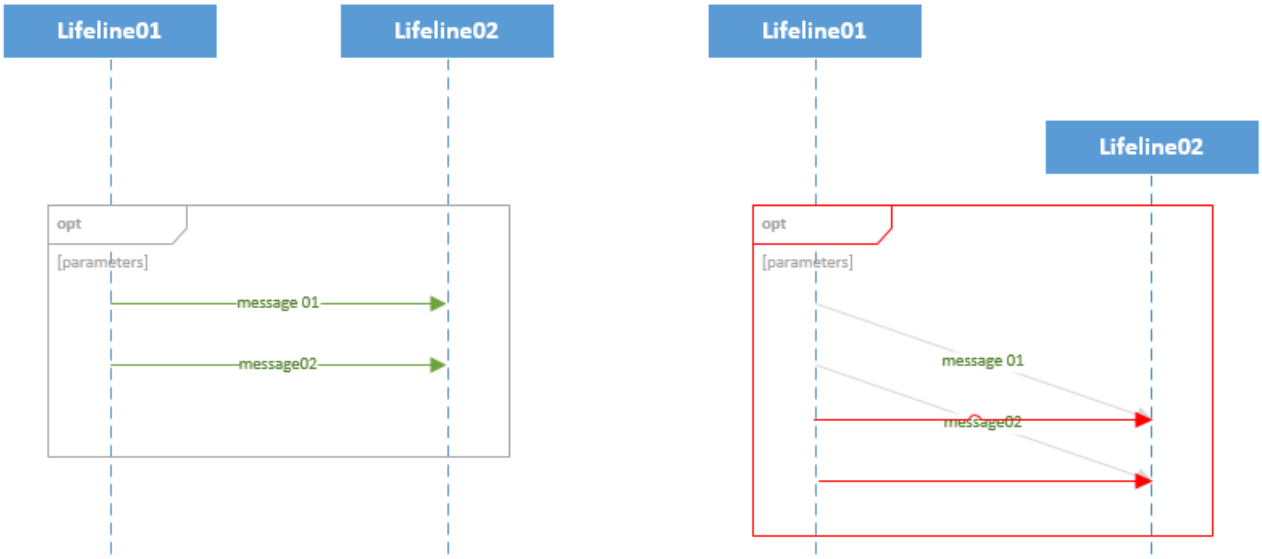
Obrázok 14: Vertikálny pohyb čiary života smerom nahor

3. Čiara života obsahuje fragment – v prípade posunu čiary života, sa okno fragmentu:
    a. Nemení, mení sa len pozícia bodov správ, ktoré môžu vyjsť aj mimo okno kombinovaného fragmentu.



Obrázok 15: Pohyb čiary života bez pohybu kombinovaného fragmentu

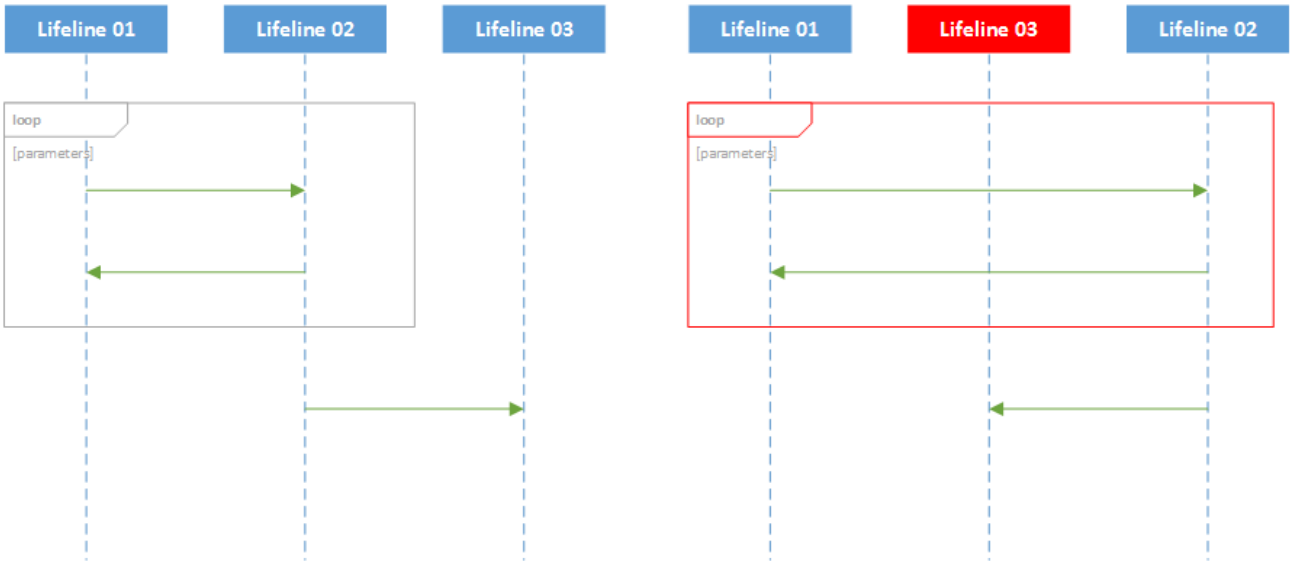
    b. Mení spolu s bodmi pozície správ.



Obrázok 16: Pohyb čiary života s pohybom kombinovaného fragmentu

3.1.3.3 Vloženie čiary života do kombinovaného fragmentu

Používateľ môže pri posune čiary života naraziť na kombinovaný fragment, do ktorého môže vložiť posúvanú čiaru života. V takomto prípade kombinovaný fragment obsahuje čiaru života spolu so správami, ktoré obsahuje.

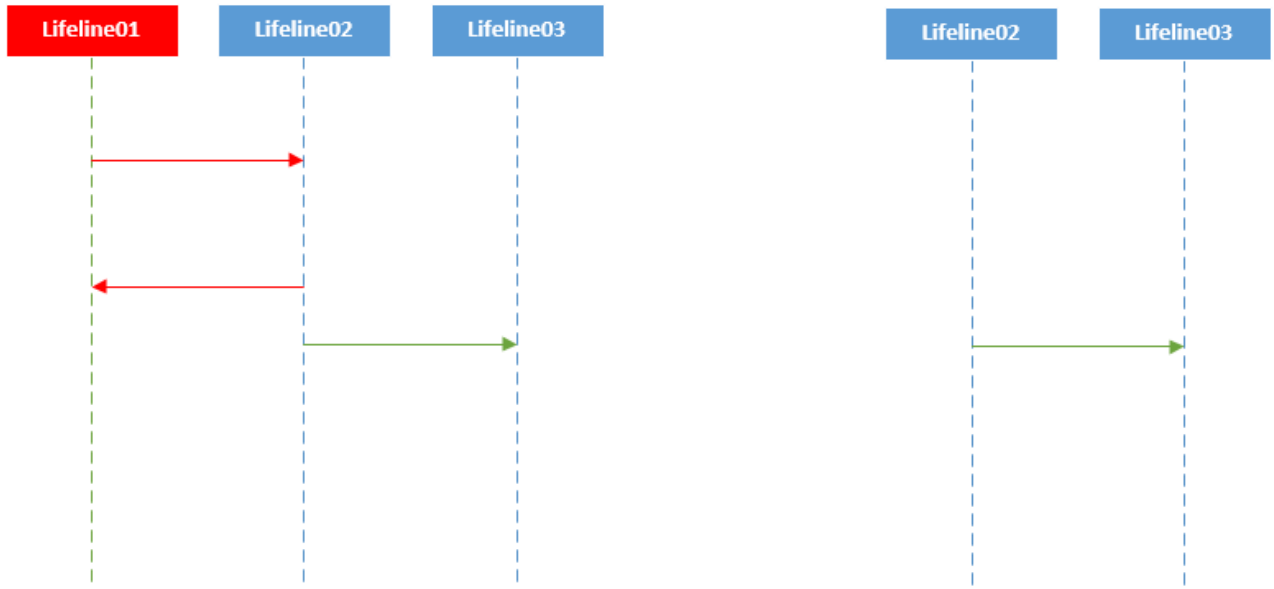


Obrázok 17: Vloženie čiary života do kombinovaného fragmentu

3.1.3.4 Vymazanie čiary života

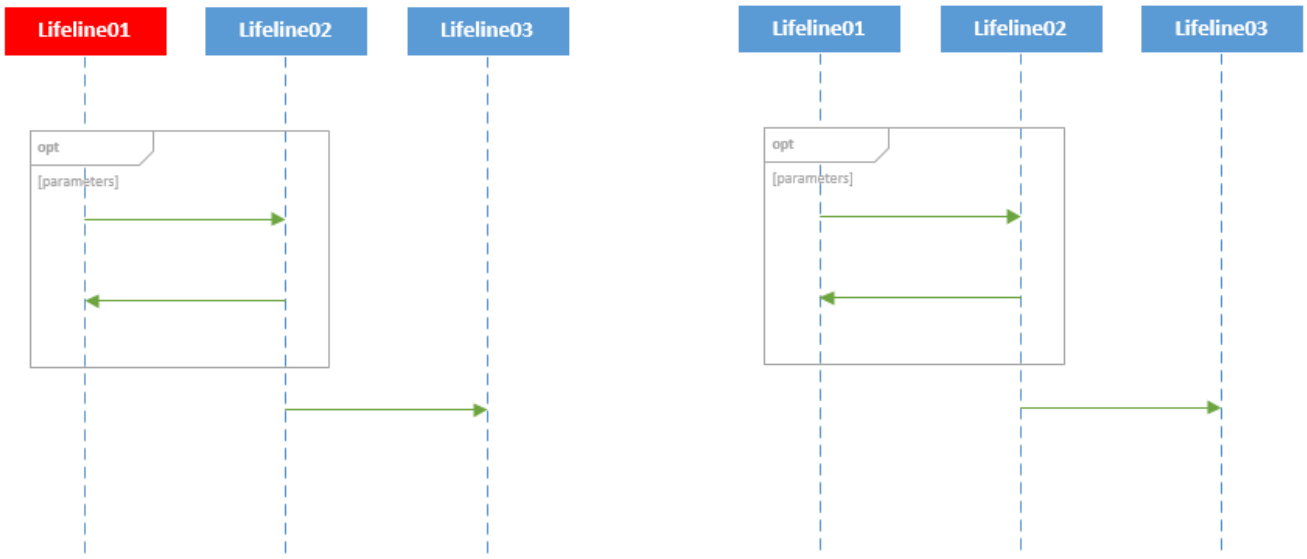
V prípade vymazania vybraných čiar života zo scény je potrebné uvažovať o 2 prípadoch:

1. Čiara života obsahuje iba správy, ktoré nie sú súčasťou žiadneho kombinovaného fragmentu – výsledkom je vymazanie čiary života zo scény spolu so všetkými správami, ktoré sú s touto čiarou života prepojené.



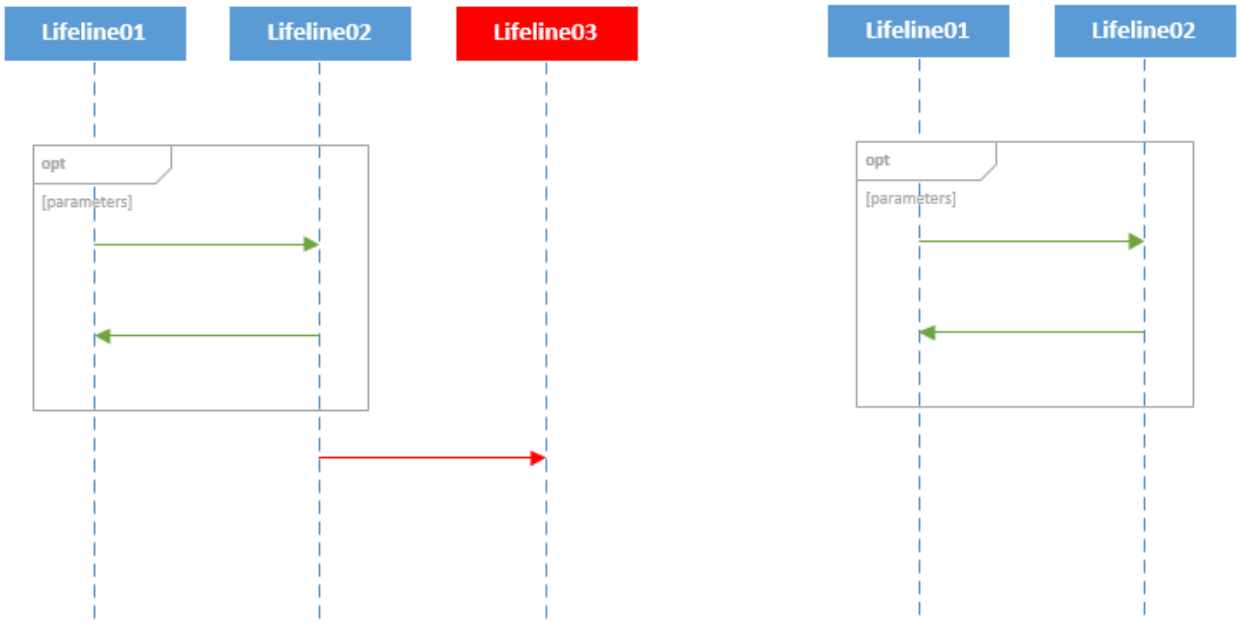
Obrázok 18: Zmazanie čiary života so zmazaním na ňu naviazaných správ

2. Čiara života obsahuje správy, ktoré sú súčasťou kombinovaného fragmentu. Tento fragment predstavuje ďalšiu vrstvu, ktorá obsahuje vnorené elementy (správy, operandy a pod.). Preto v takomto prípade nie je možné ihneď zmazať čiaru života. Najskôr je potrebné zmazať kombinovaný fragment a následne po tejto úspešnej akcii môžeme zmazať vybranú čiaru života.



Obrázok 19: Nezmazanie čiary života, ktorá zasahuje do kombinovaného fragmentu

3. V prípade výskytu kombinovaného fragmentu v interakcii môže používateľ vymazať len čiaru života, ktorá nie je s týmto fragmentom prepojená.



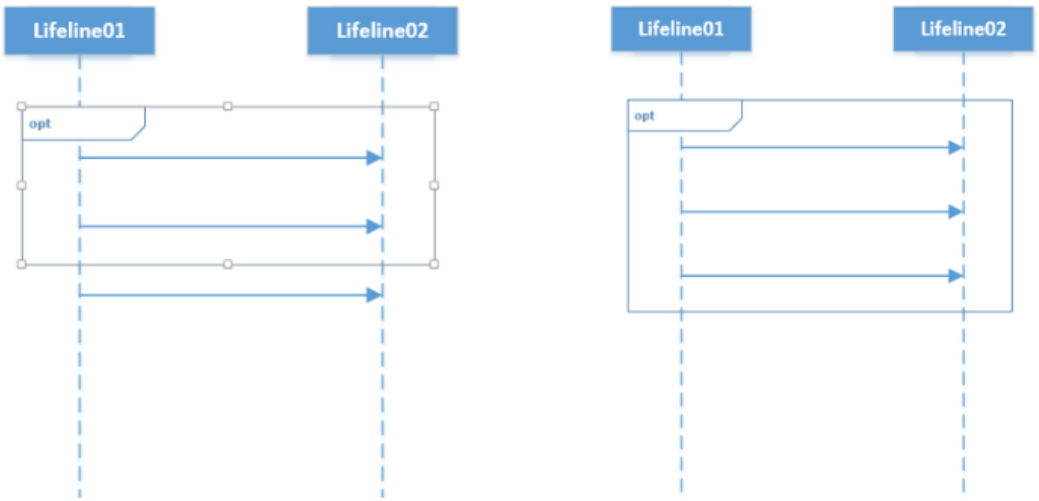
Obrázok 20: Zmazanie čiary života so správou mimo kombinovaného fragmentu

3.1.4 Editačné funkcie pre kombinované fragmenty

3.1.4.1 Zmena veľkosti kombinovaného fragmentu

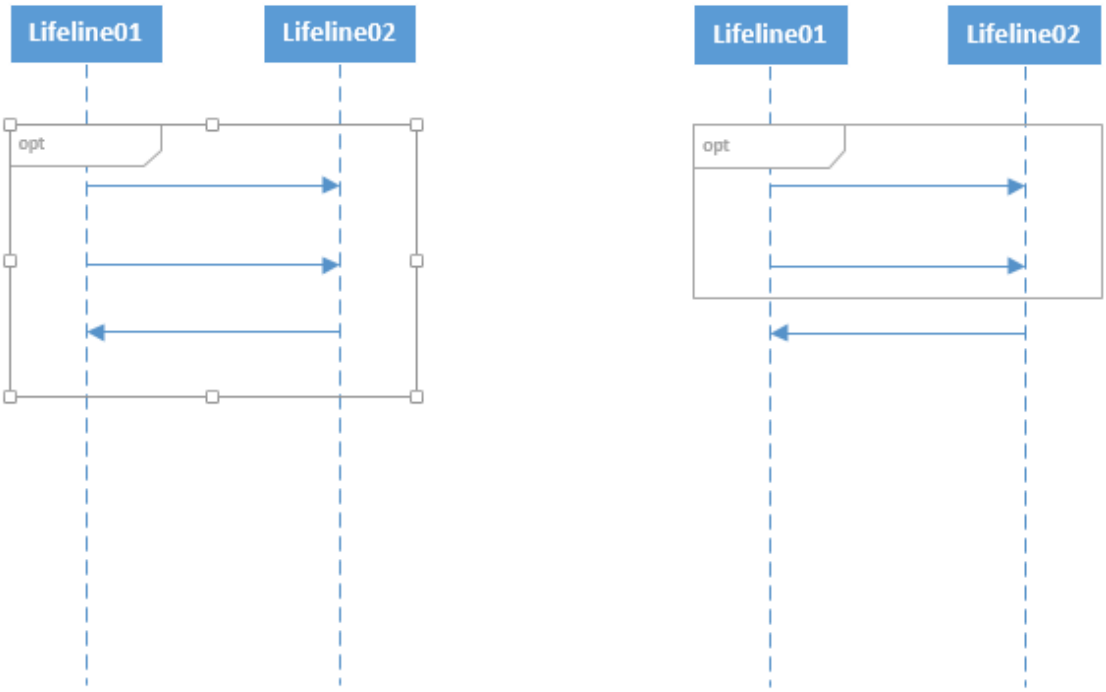
Používateľ môže upravovať veľkosť fragmentov kliknutím a ťahaním za jeden z rohov označeného fragmentu. Pri pustení tlačidla sa upraví celý metamodel podľa nových elementov, ktoré fragment po zmene veľkosti zahŕňa. Jediné, čo používateľ na obrazovke uvidí, je, že fragment nové elementy obalí. Používateľ sa môže rozhodnúť upravovať veľkosť fragmentu v rôznych prípadoch:

1. Zväčšenie označeného fragmentu s obsiahnutím existujúcej správy, resp. správ.



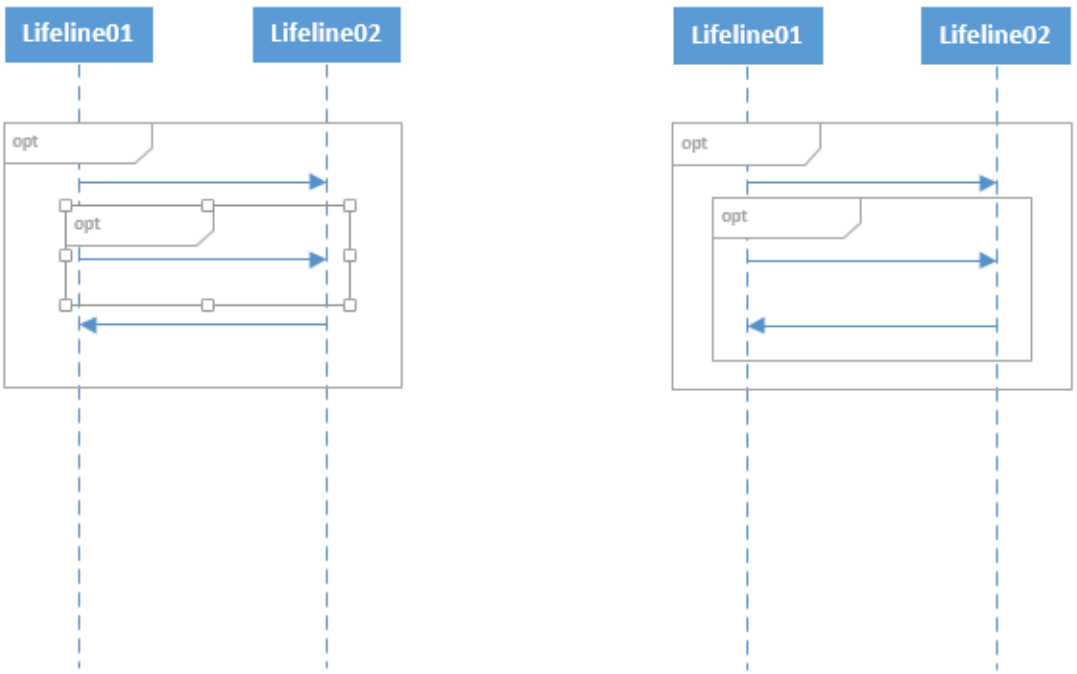
Obrázok 21: Zväčšenie kombinovaného fragmentu s obsiahnutím správy

2. Zmenšenie označeného fragmentu – správu alebo správy, ktoré boli súčasťou tohto fragmentu, už nebude tento fragment obsahovať.



Obrázok 22: Zmenšenie kombinovaného fragmentu s odobratím správ z neho

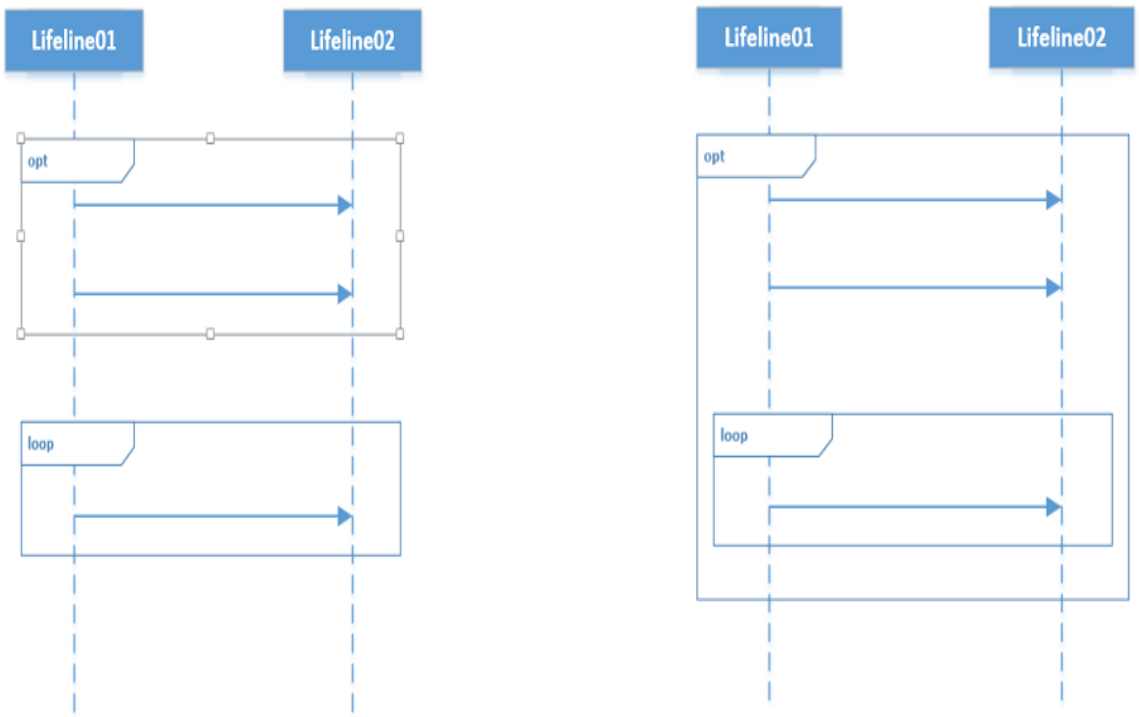
3. Zväčšenie vnoreného fragmentu – vnorený fragment obsiahne správu alebo správy v rodičovskom fragmente.



Obrázok 23: Zväčšenie vnoreného kombinovaného fragmentu

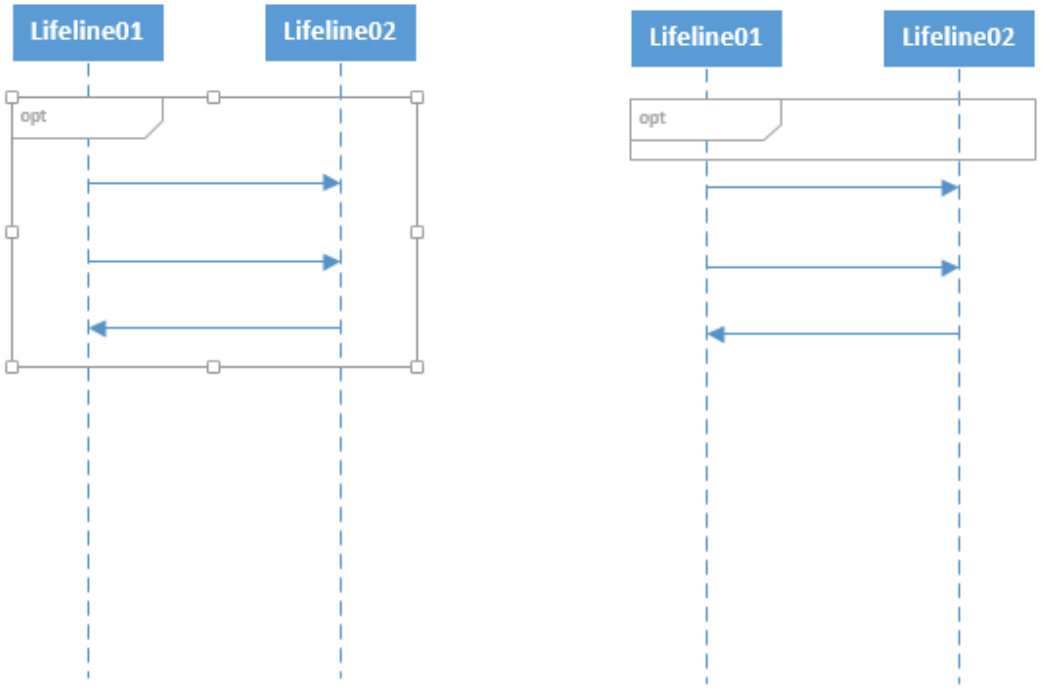
4. Zväčšenie vnoreného fragmentu – vnorený fragment nemôže byť zväčšený za hranice rodičovského fragmentu.

5. Obsiahnutie fragmentu iným fragmentom – fragment bude považovaný za obsiahnutý len vtedy, ak sa jeho hlavička bude nachádzať v označenom fragmente



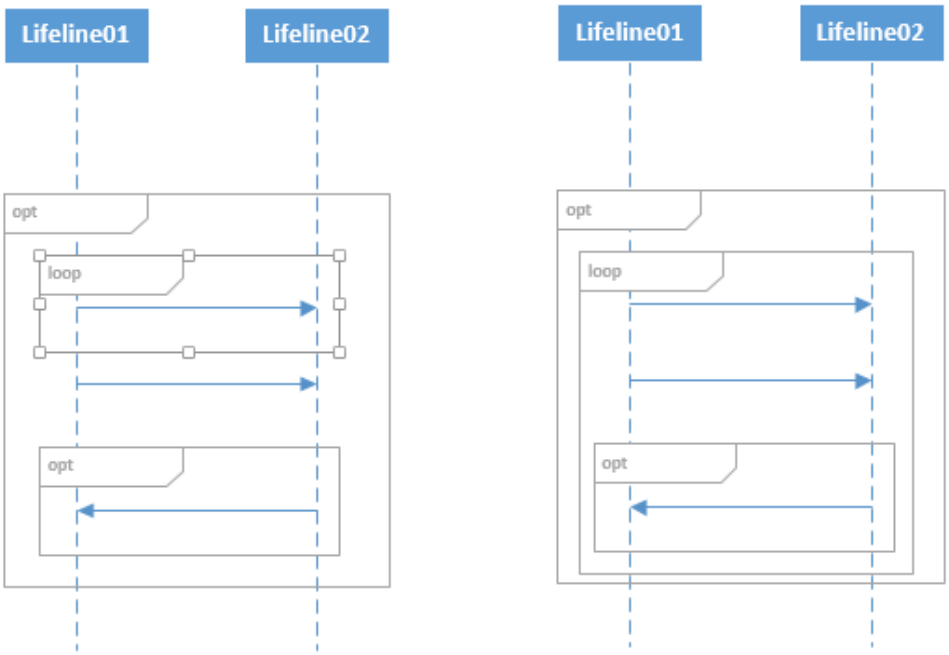
Obrázok 24: Obsiahnutie kombinovaného fragmentu iným fragmentom

6. Zmenšenie fragmentu tak, že vznikne prázdny fragment – takáto situácia je prípustná.



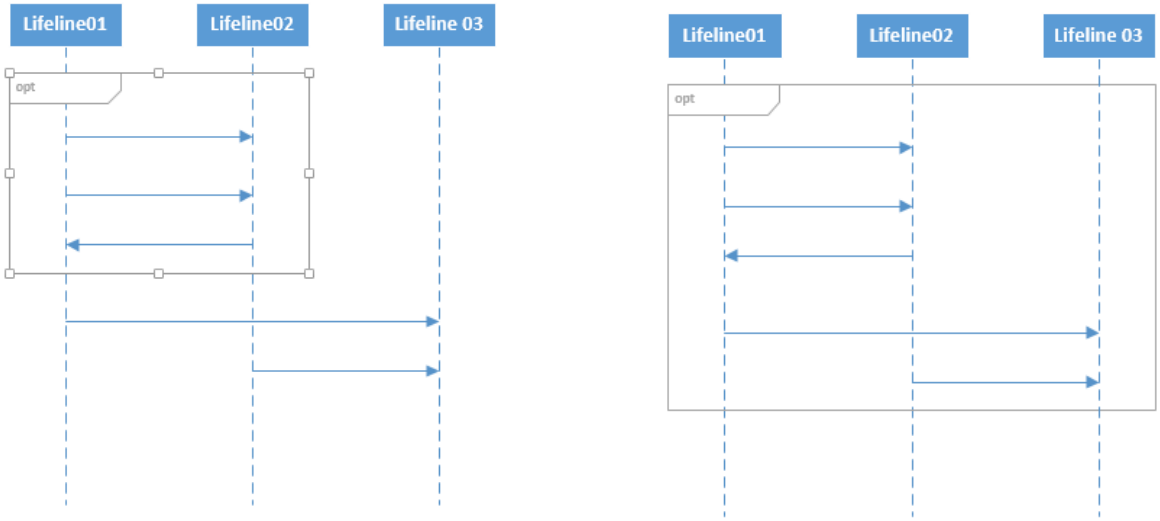
Obrázok 25: Zmenšenie fragmentu tak, že vznikne prázdny fragment

7. Zväčšenie vnoreného fragmentu tak, že obalí iný vnorený fragment.



Obrázok 26: Zväčšenie vnoreného fragmentu s obalením iného fragmentu

8. Prípady 1-7 musia byť zovšeobecnené aj pre prípady viacerých čiar života (nech je ich počet ľubovoľný – *n*). Ilustrujeme zväčšenie fragmentu pre prípad *n = 3* s obalením správy, ktorá prechádza od prvej čiary života k tretej, a rovnako obalenie správy, ktorá prechádza od druhej čiary života k tretej.

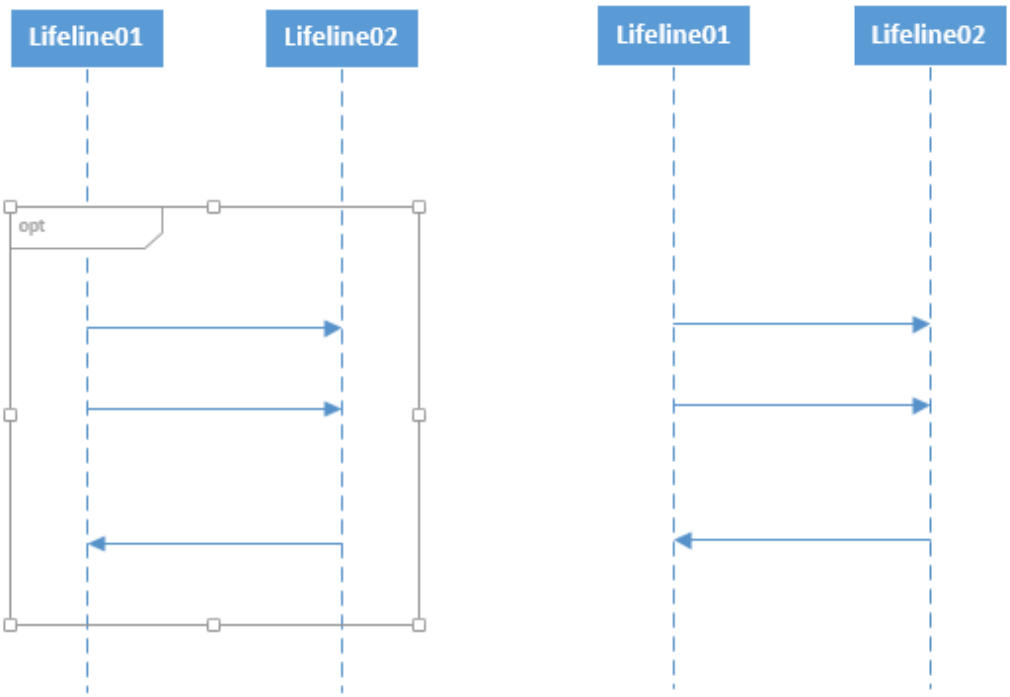


Obrázok 27: Zväčšenie fragmentu pre viac čiar života

3.1.4.2 Zmazanie fragmentu

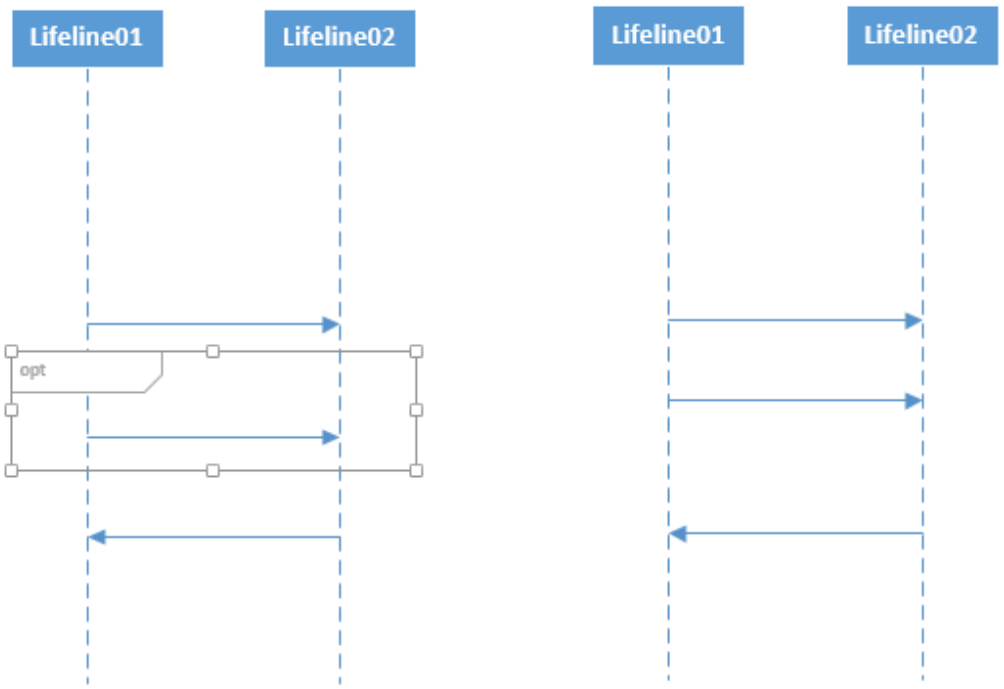
V prípade, že používateľ nie je spokojný so svojím návrhom, má možnosť fragmenty zmazať. Aby fragment zmazal, potrebuje najprv daný fragment označiť a následne kliknúť pravým tlačidlom myši, aby sa mu objavilo kontextové menu, ktoré obsahuje možnosť odstrániť označený element. Obsah fragmentu sa pri zmazaní zachováva.

1. Odstránenie fragmentu, ktorý nad sebou a ani pod sebou neobsahuje správy – správy, ktoré tento fragment obsahuje, sa stanú súčasťou rodiča.



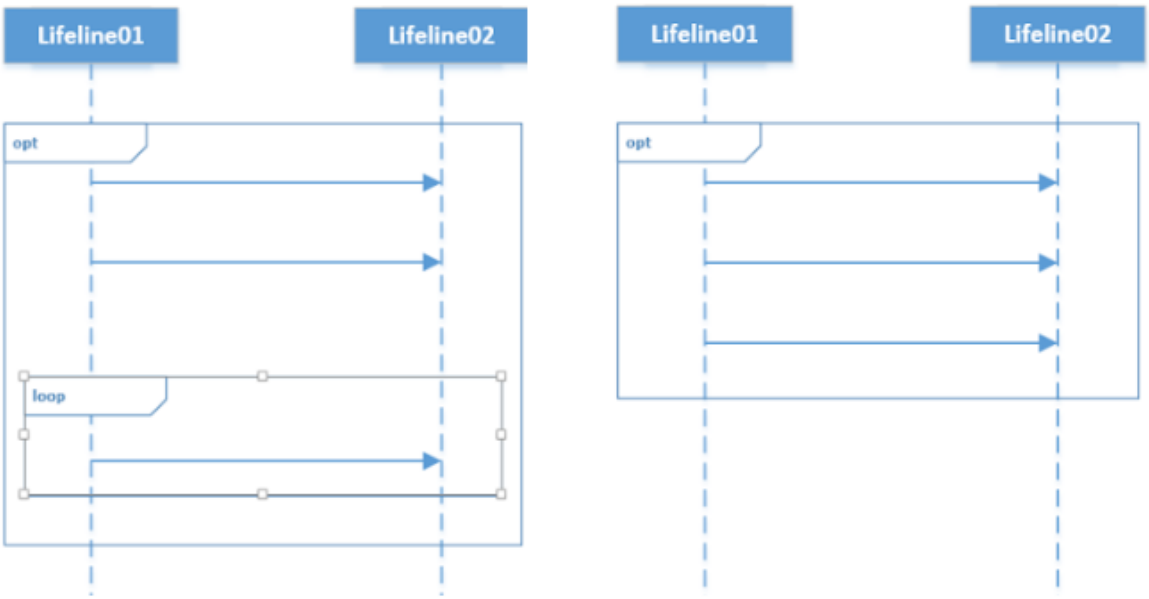
Obrázok 28: Zmazanie kombinovaného fragmentu bez správ nad ním alebo pod ním

2. Odstránenie fragmentu, ktorý pod sebou alebo nad sebou obsahuje správy – správy, ktoré tento fragment obsahuje, sa stanú súčasťou rodičovského fragmentu, pričom v metamodeli dôjde k zlúčeniu jednotlivých dvoch (ak obsahuje správy len nad sebou, alebo len pod sebou) alebo troch operandov (správy nad sebou aj pod sebou) do jedného interakčného operandu. Zlučovanie operandov sa deje nad rodičom mazaného kombinovaného fragmentu.



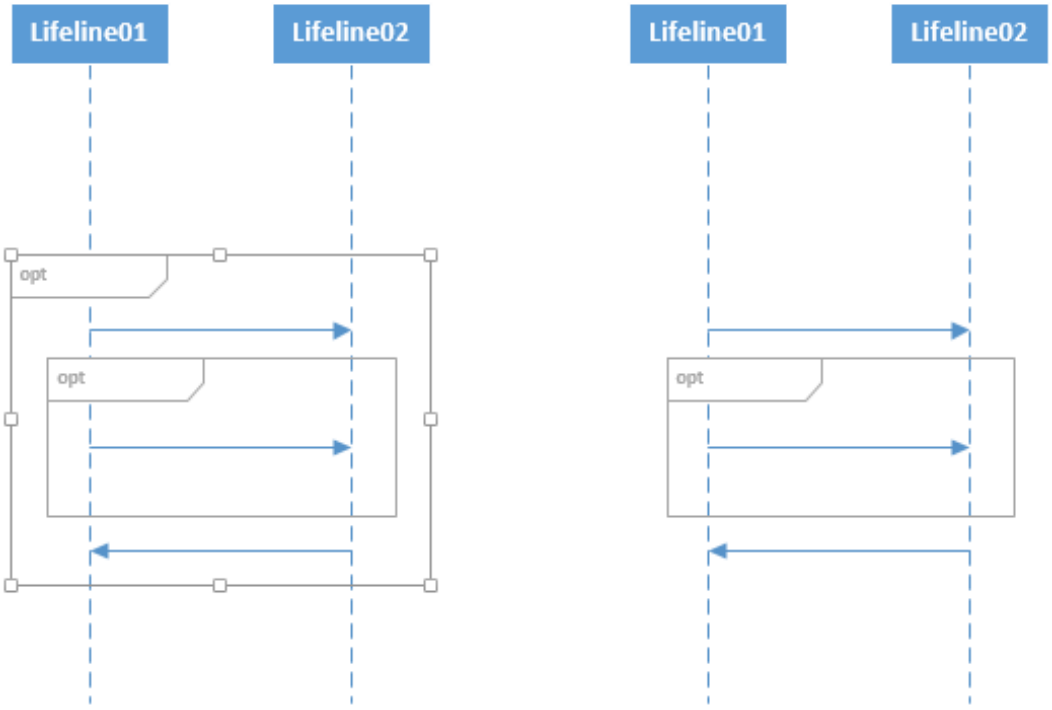
Obrázok 29: Zmazanie kombinovaného fragmentu so správami nad ním alebo pod ním

3. Odstránenie vnoreného fragmentu – správa obsiahnutá týmto fragmentom sa presunie do rodičovského fragmentu.



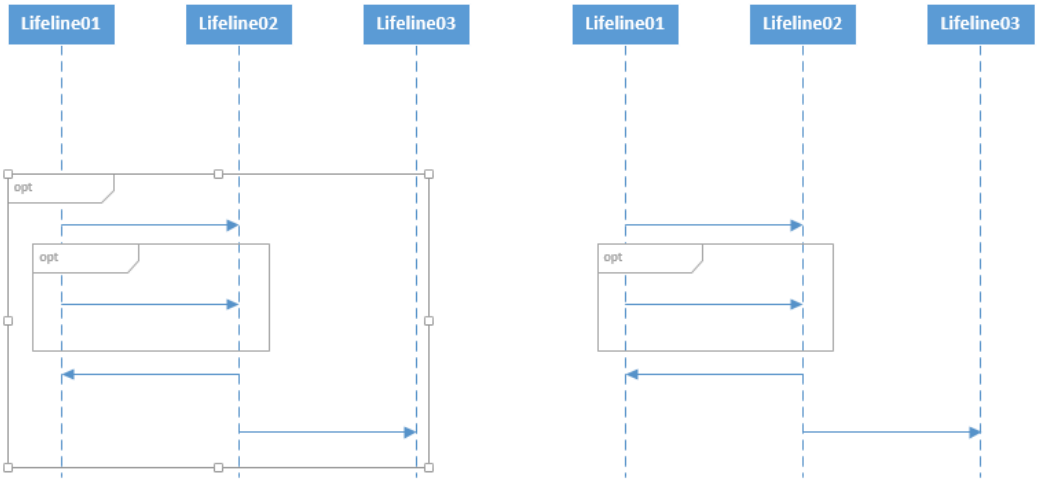
Obrázok 30: Odstránenie vnoreného fragmentu

4. Odstránenie vonkajšieho fragmentu v prípade vnorených fragmentov – celý obsah mazaného vonkajšieho fragmentu (t.j. s potenciálnymi ďalší fragmentmi a správami) sa presunie do rodiča mazaného vonkajšieho fragmentu.



Obrázok 31: Odstránenie vonkajšieho fragmentu v prípade vnorených fragmentov

5. Prípady 1-4 musia byť zovšeobecnené aj pre prípady viacerých čiar života (nech je ich počet ľubovoľný – *n*). Ilustrujeme zmazanie vonkajšieho fragmentu pre prípad *n = 3*.

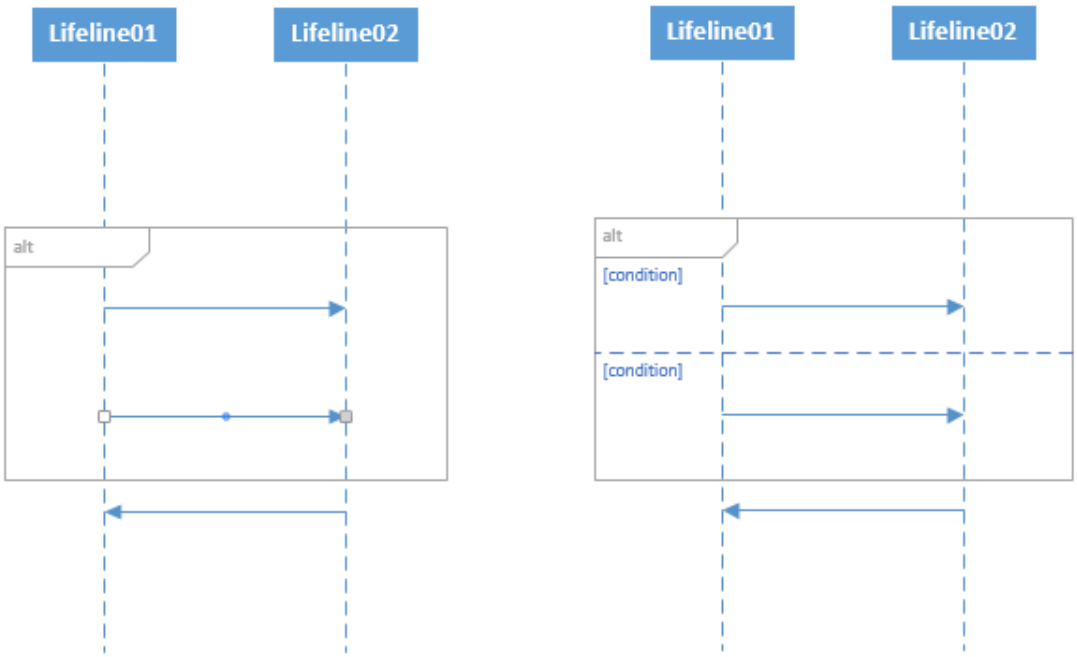


Obrázok 32: Zmazanie fragmentu pre viacero čiar života

3.1.4.3 Pridávanie operandov do kombinovanému fragmentu a ich úprava

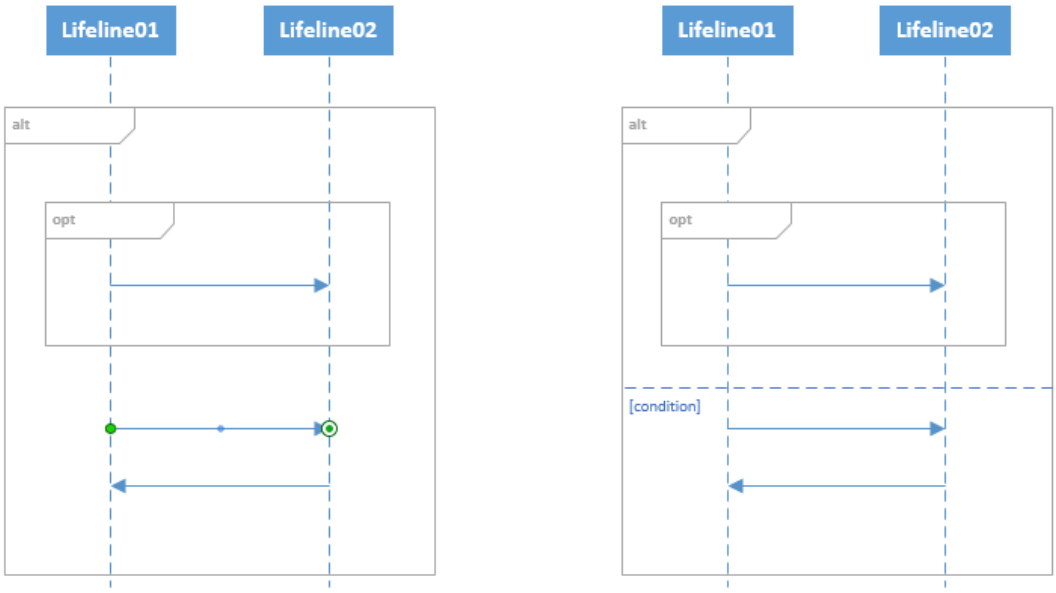
Používateľ môže pridať operandy do kombinovaného (napr. alt) fragmentu. Operandy pridá používateľ tak, že si vyberie správu, nad ktorú chce daný operand pridať, a zvolí príslušné tlačidlo pre pridanie operandu v menu. Alternatívne môže zvoliť kombinovaný fragment namiesto správy (pozri prípad č. 3). Následne klikne do plochy nad zvolenú správu / fragment, zadá podmienku a svoju voľbu potvrdí.

1. Pridanie operandu v jednoduchom fragmente.



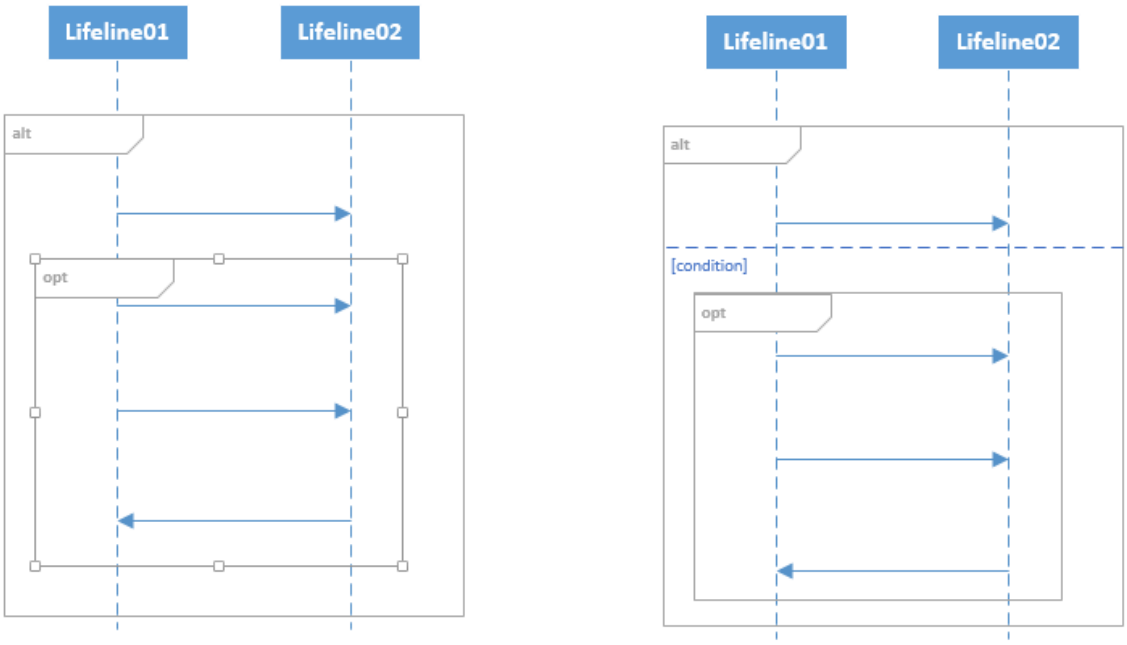
Obrázok 33: Pridanie operandu v jednoduchom fragmente

2. Pridanie operandu do fragmentu, ktorý obsahuje vnorený fragment. Vnorený fragment nemá byť súčasťou nového operandu.



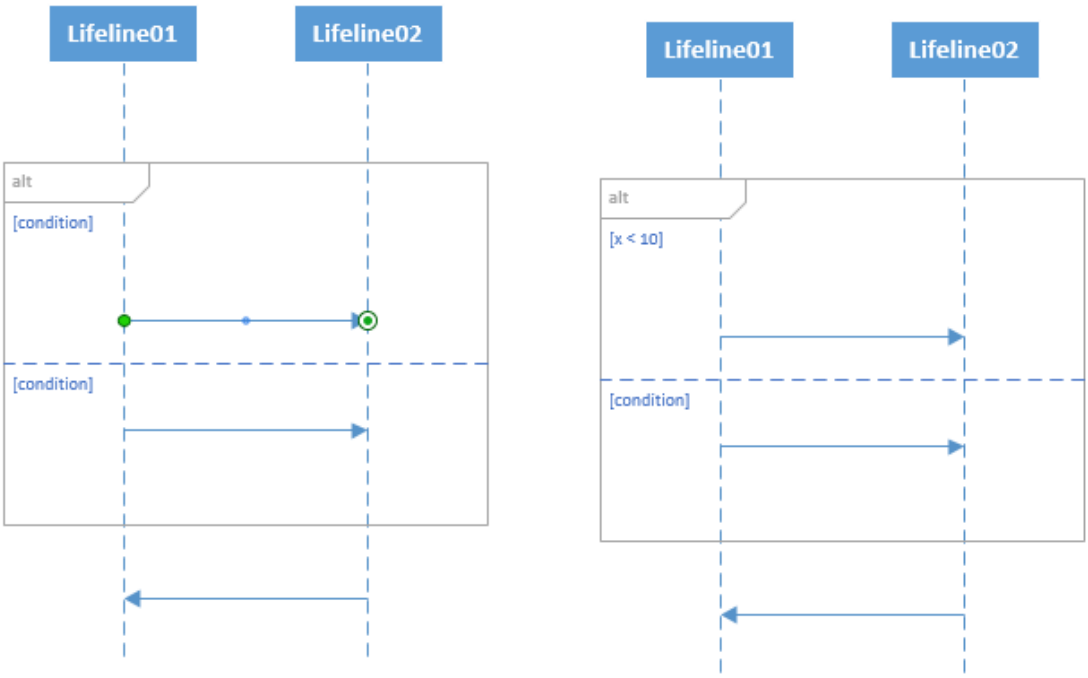
Obrázok 34: Pridanie operandu vo fragmente s vnoreným fragmentom pod vnorený fragment

3. Pridanie operandu do fragmentu, ktorý obsahuje vnorený fragment. Vnorený kombinovaný fragment má byť súčasťou nového operandu (ako prvý objekt v ňom). V tomto prípade používateľ namiesto správy zvolí príslušný kombinovaný fragment a klikne nad neho.



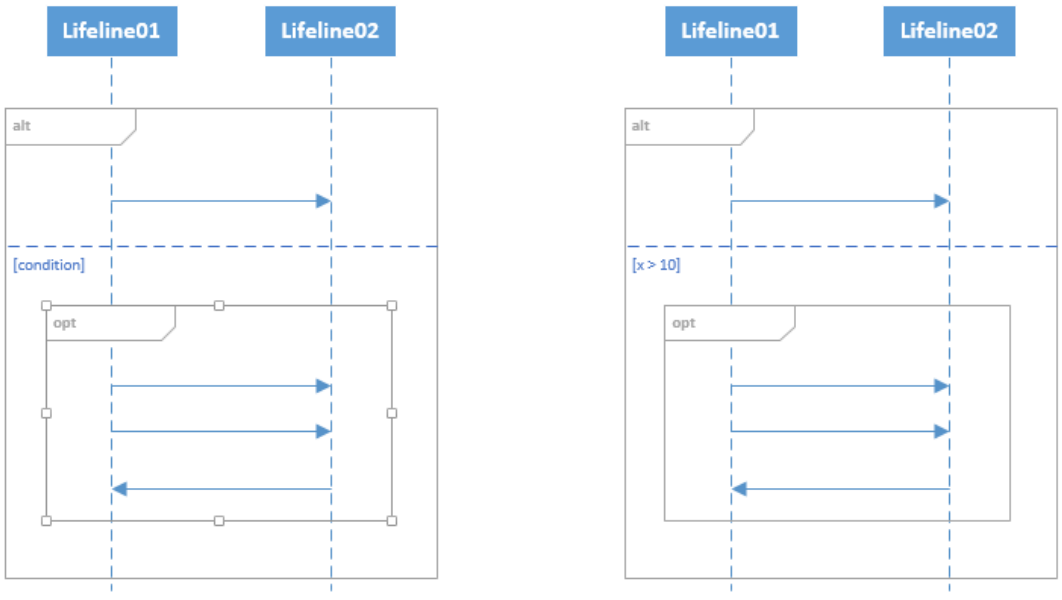
Obrázok 35: Pridanie operandu vo fragmente s vnoreným fragment nad vnorený fragment

4. Pridanie operandu do kombinovaného fragmentu so zvolenou prvou správou v danom operande – umožní zmeniť podmienku daného operandu.



Obrázok 36: Úprava podmienky pre prvý operand kombinovaného fragmentu

5. Pridanie operandu do fragmentu, ktorý obsahuje vnorený fragment. Vnorený fragment má byť súčasťou nového operandu (ako prvý objekt v ňom), vnorený fragment už ale je prvým objektom v danom operande – takáto akcia umožní používateľovi môže zmeniť podmienku v tomto operande (daný vnorený fragment musí byť pre tieto účely označený, t.j. treba vykonať kliknutia ako v prípade 3).

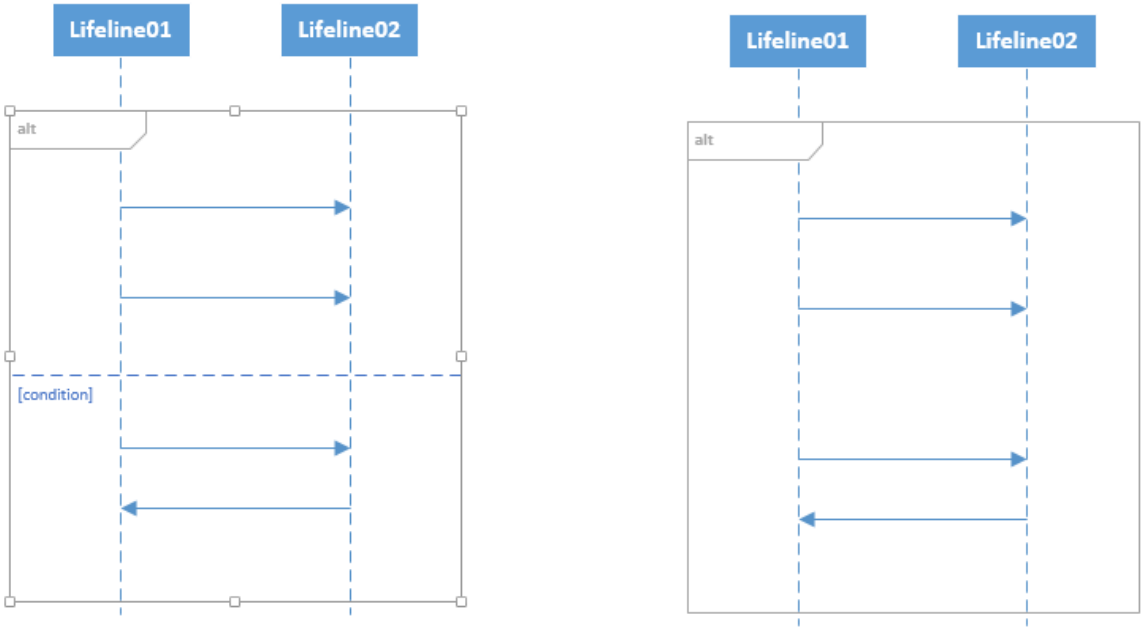


Obrázok 37: Úprava podmienky operandu, ktorý obsahuje len kombinovaný fragment

3.1.4.4 Mazanie operandov kombinovaných fragmentov

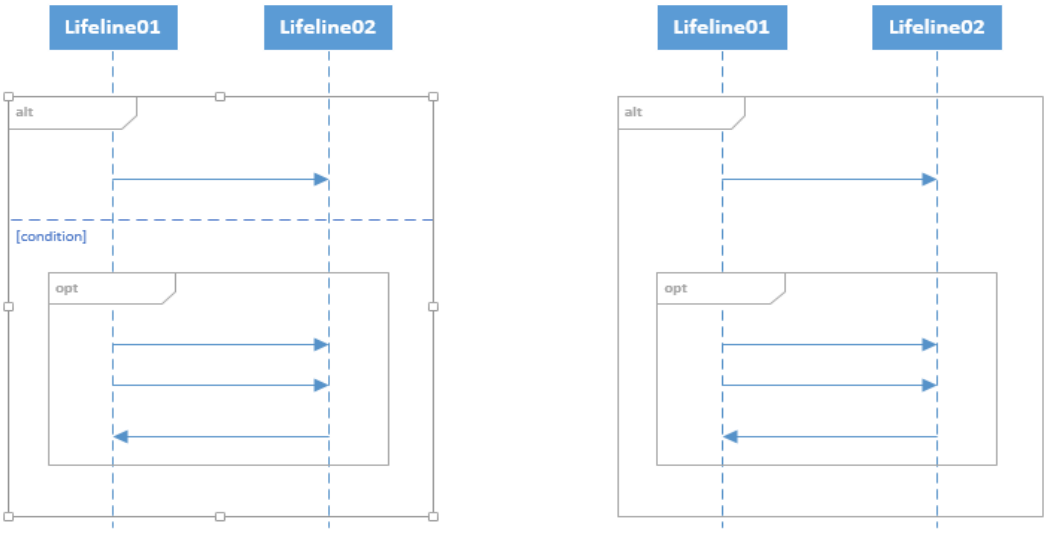
Používateľ môže zmazať vytvorené operandy v kombinovaných fragmentov. Pre tieto účely označí kombinovaný fragment, ktorého operandy chce zmazať, a vyberie príslušné tlačidlo v menu. Následne napíše číslo operandu, ktorý chce vymazať (indexovanie od 1). Z metamodelového hľadiska zmazanie operandu nevymaže obsah tohto operandu, ale formálne len zlúči mazaný operand s inými existujúcimi operandami, t.j. vizuálne zanikne deliaca čiara.

1. Zmazanie operandu v jednoduchom kombinovanom fragmente.



Obrázok 38: Zmazenie operandu v jednoduchom kombinovanom fragmente

2. Zmazanie operandu vo fragmente, ktorý obsahuje vnorený fragment.



Obrázok 39: Zmazanie operandu vo vonkajšom kombinovanom fragmente v prípade vnorených fragmentov

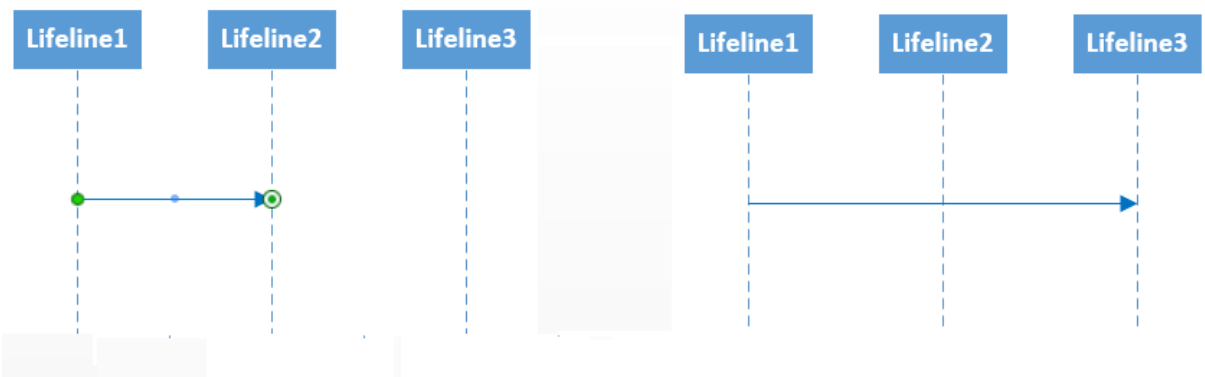
3. Zmazanie operandu kombinovaného fragmentu, pričom mazaný operand je jediným operandom tohto fragmentu – zakázaný prípad.

3.1.5 Editačné funkcie pre správy

3.1.5.1 Zmena cieľovej čiary života správy

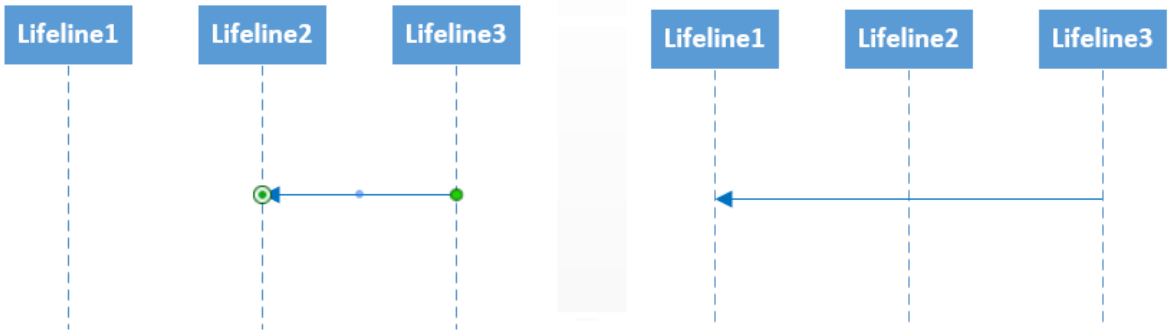
Používateľ má možnosť po označení správy zmeniť jej cieľovú destináciu - čiaru života. Správu je potrebné chytiť za stredový bod správy (v strede čiary reprezentujúcej správu) a následne potiahnuť či už doľava alebo doprava smerom k čiare života, ktorá má reprezentovať novú cieľovú destináciu správy v sekvenčnom diagrame. Posun kurzora sa musí vykonávať v rámci chytenia správy, a teda po kliknutí, resp. označení správy je nutné nepustiť ľavé tlačidlo na myši, až pokiaľ nie sme spokojní s výslednou zmenou.

1. Zmena cieľovej čiary života správy nachádzajúcej sa vpravo od pôvodnej.



Obrázok 40: Rozšírenie čiary života smerom doprava

2. Zmena cieľovej čiary života nachádzajúcej sa vľavo od pôvodnej.



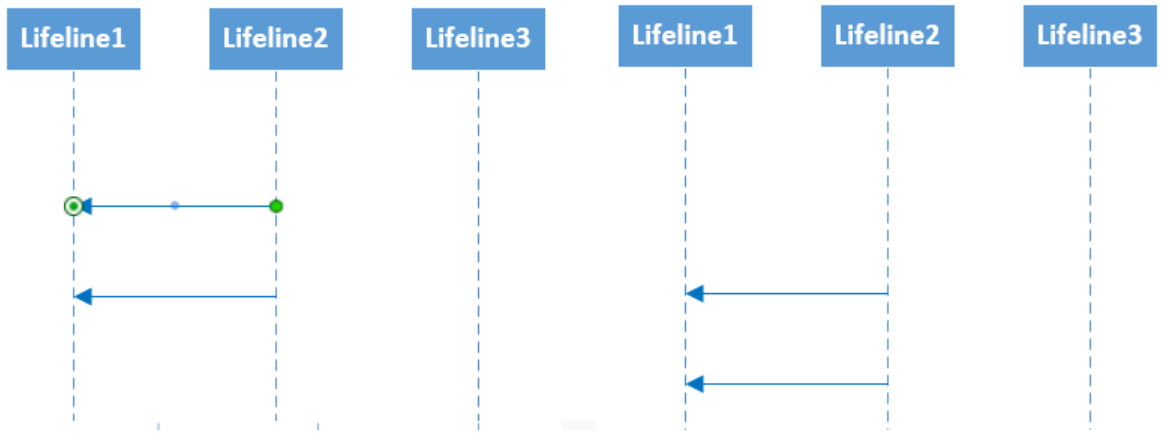
Obrázok 41: Rozšírenie čiary života smerom doľava

3.1.5.2 Zmena pozície správy v rámci vertikálneho pohybu

Správu možno v rámci kontextu priestoru ohraničeného čiarami života, ku ktorým správa patrí, posúvať vo vertikálnom smere. V prvom kroku je správu potrebné chytiť a posúvať kurzor podľa potreby hore alebo dole od jej pôvodnej pozície. Grafický posun správy, ale aj posun správy v rámci metamodelu sa vykonáva interaktívne a v reálnom čase, takže medzivýsledky posunu je

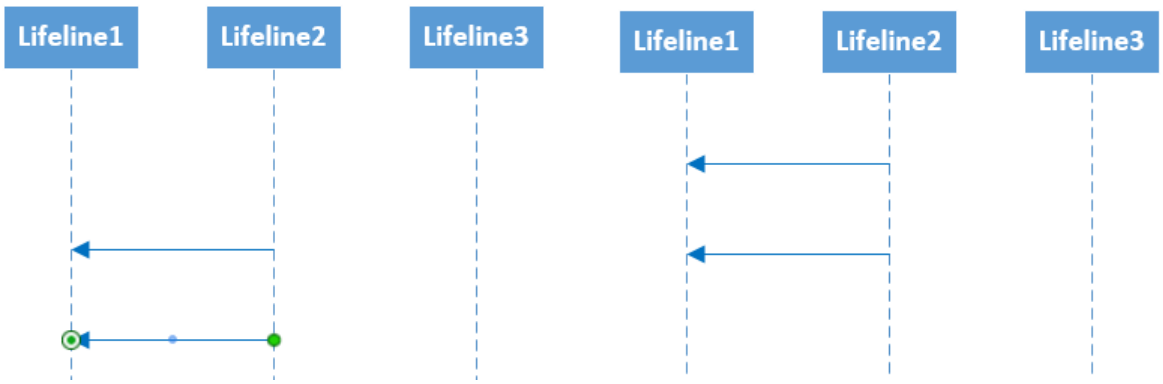
možno okamžite vidieť. V prípade spokojnosti s novou pozíciou správy je potrebné pre jej uloženie pustiť ľavé tlačidlo na myši.

1. Presun správy medzi viacerými správami.
    a. Správa sa nachádza nad inou správou.



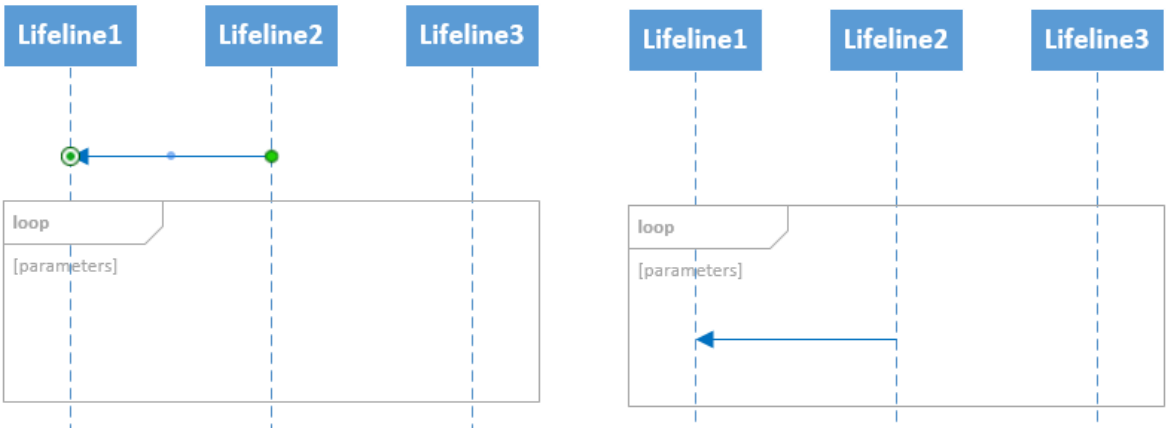
Obrázok 42: Presunutie správy pod inú správu

    b. Správa sa nachádza pod inou správou



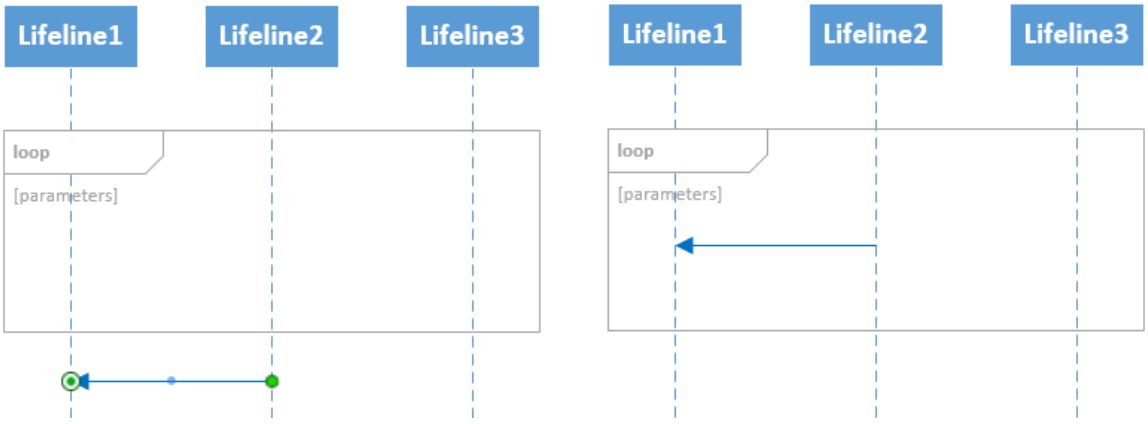
Obrázok 43: Presunutie správy nad inú správu

2. Presun správy do kombinovaného fragmetu.
    a. Správa sa nachádza nad kombinovaným fragmentom.



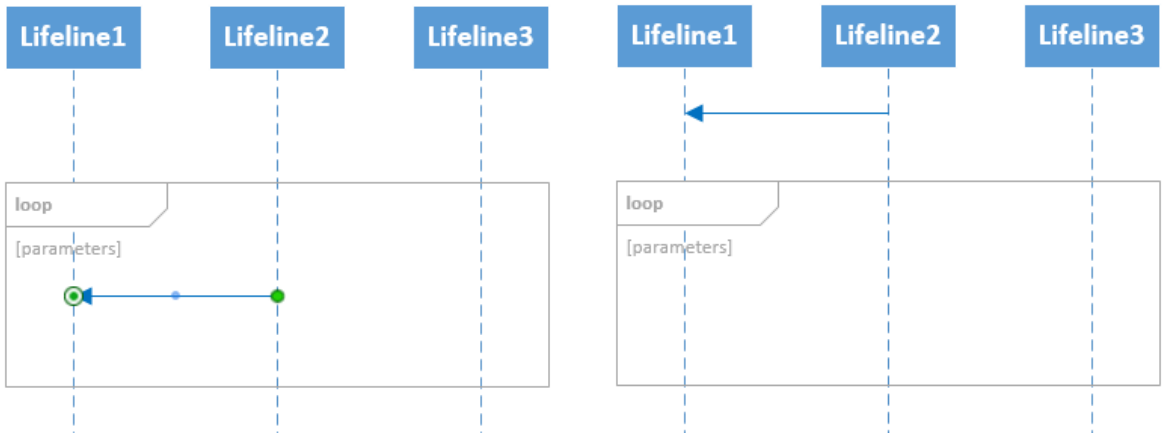
Obrázok 44: Presun správy do kombinovaného fragmentu z pôvodnej pozície nad ním

b. Správa sa nachádza pod kombinovaným fragmentom.



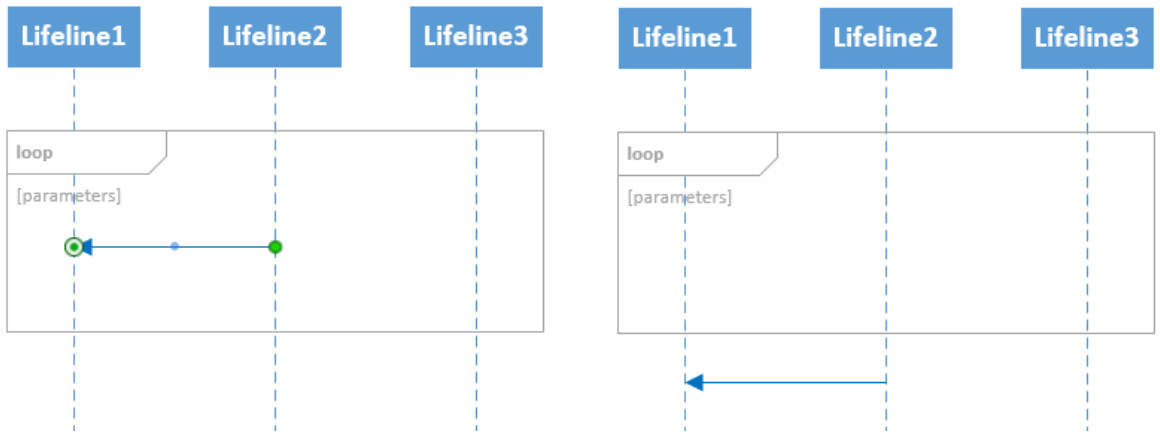
Obrázok 45: Presun správy do kombinovaného fragmentu z pôvodnej pozície pod ním

3. Presun správy von z kombinovaného fragmetnu.
   a. Správa sa presúva nad kombinovaný fragment.



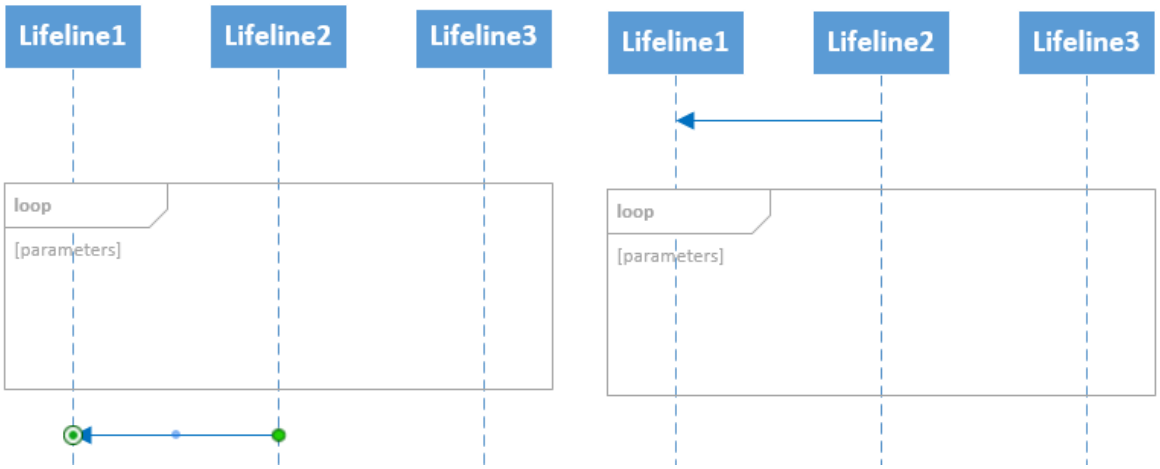
Obrázok 46: Presun správy von z kombinovaného fragmentu nad neho

   b. Správa sa presúva pod kombinovaný fragment.



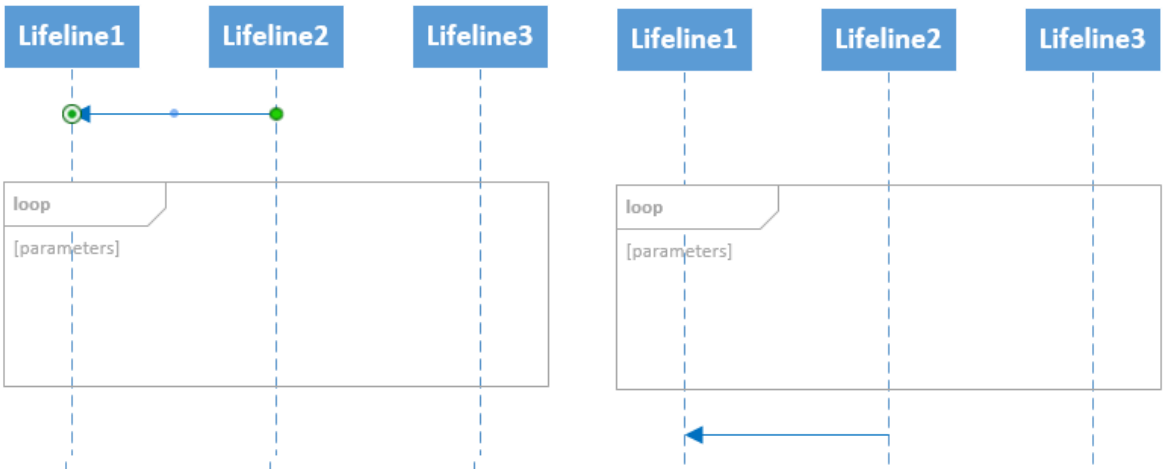
Obrázok 47: Presun správy von z kombinovaného fragmentu pod neho

4. Presun správy nad kombinovaný fragment.



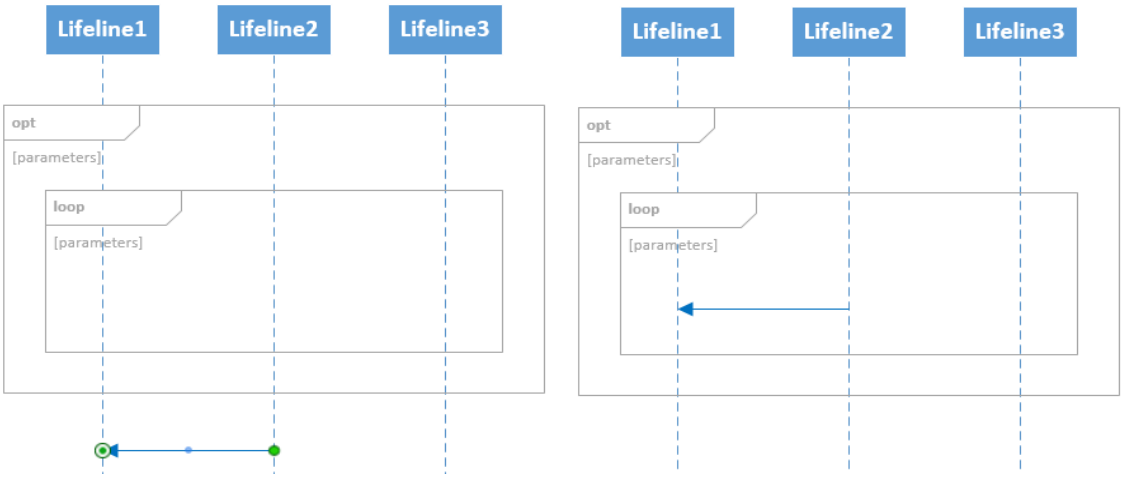
Obrázok 48: Presun správy pod kombinovaným fragmentom nad neho

5. Presun správy pod kombinovaný fragment.



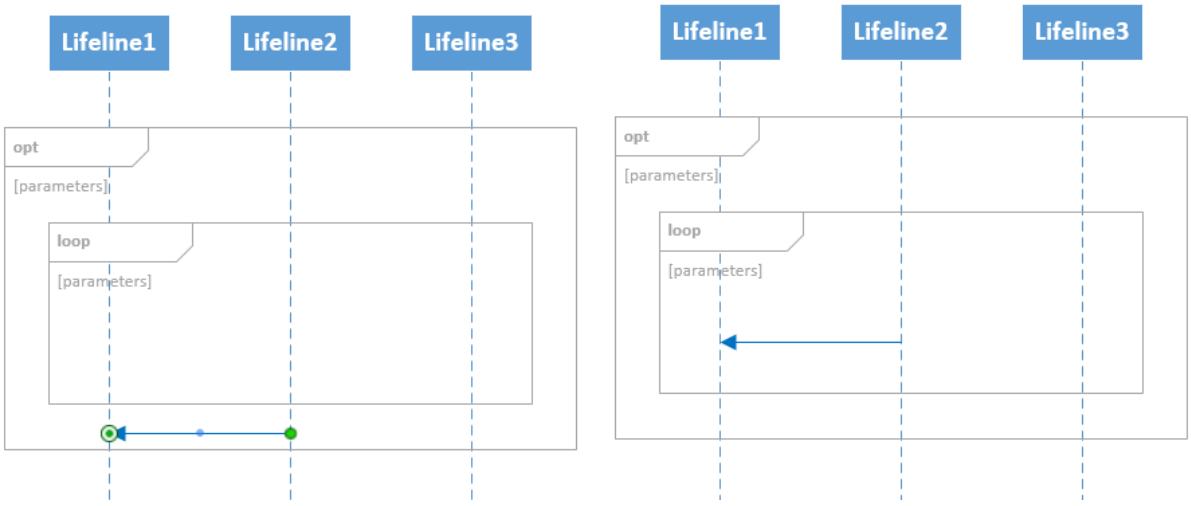
Obrázok 49: Presun správy nad kombinovaným fragmentom pod neho

6. Presun správy do vnoreného kombinovaného fragmentu.
    a. Správa sa nachádza mimo rodičovského fragmentu.



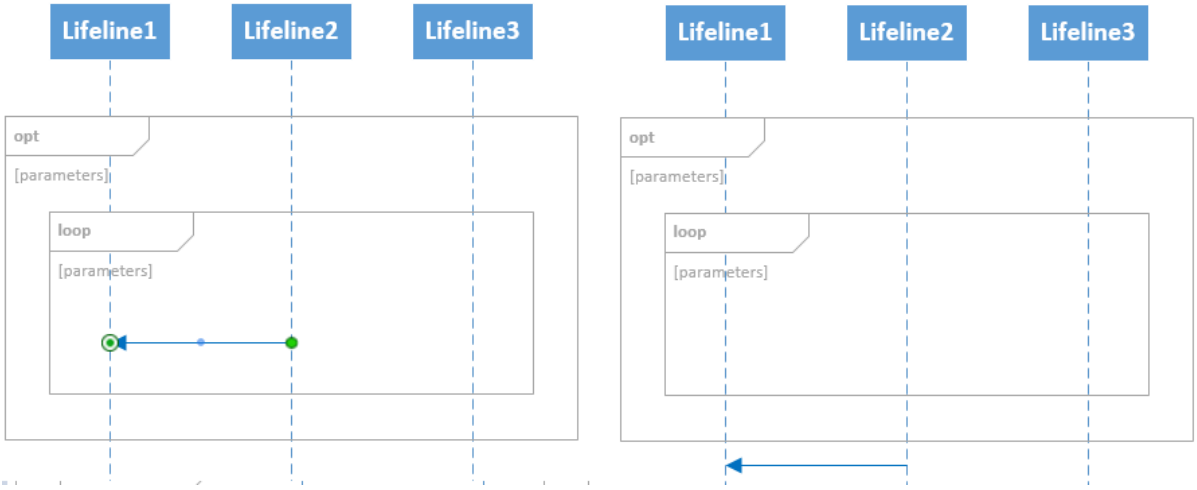
Obrázok 50: Presun správy do vnoreného kombinovaného fragmentu

b. Správa sa nachádza v rodičovskom fragmente.



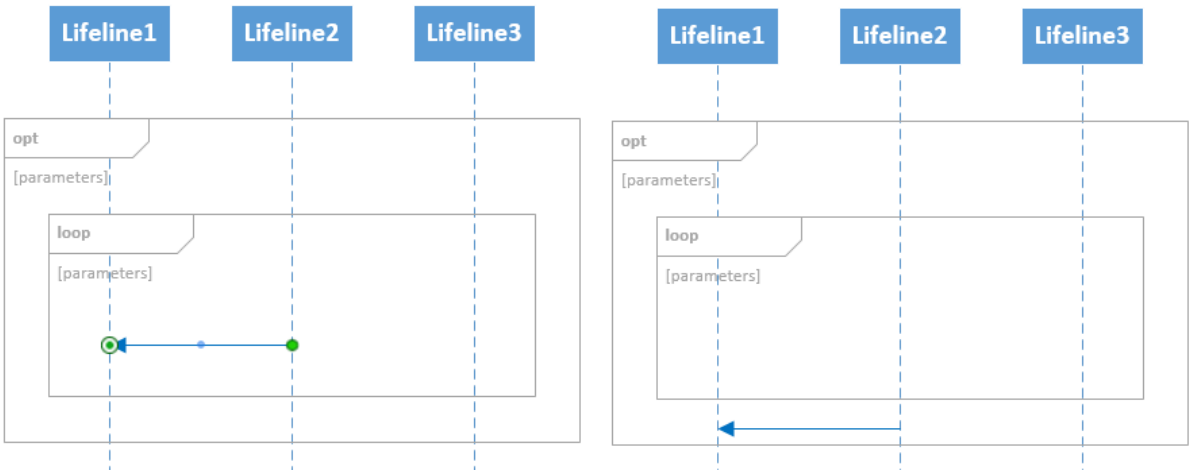
Obrázok 51: Presun správy z rodičovského fragmentu do vnoreného fragmentu

7. Presun správy z vnoreného kombinovaného fragmentu.
   a. Správa sa presúva mimo rodičovského fragmentu.



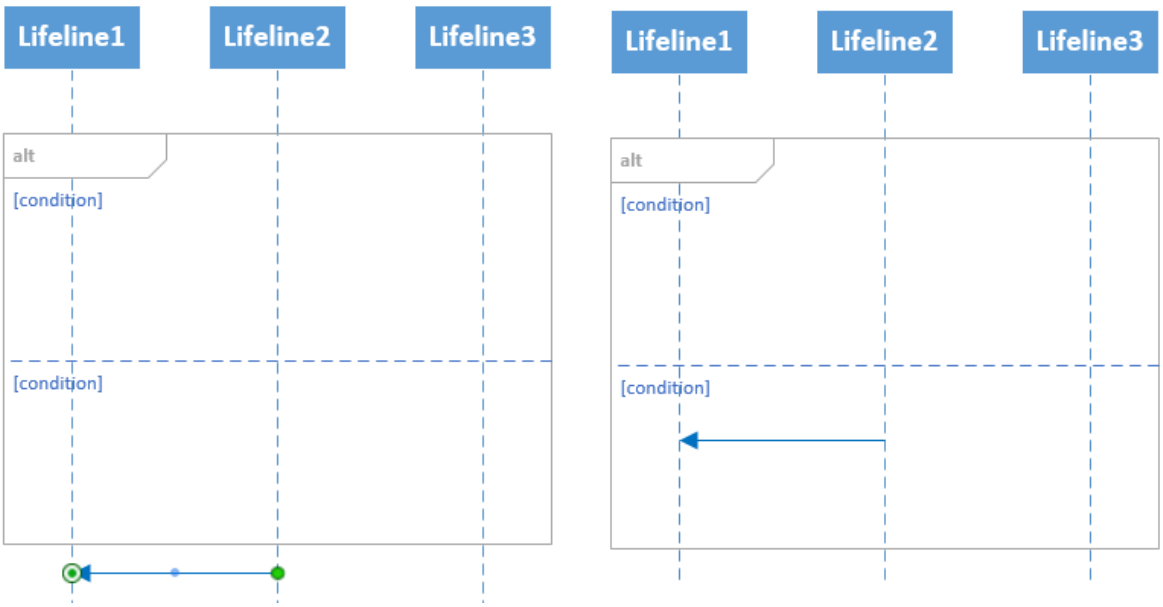
Obrázok 52: Presun správy z vnoreného fragmentu mimo strom fragmentov

   b. Správa sa presúva do rodičovského fragmentu.



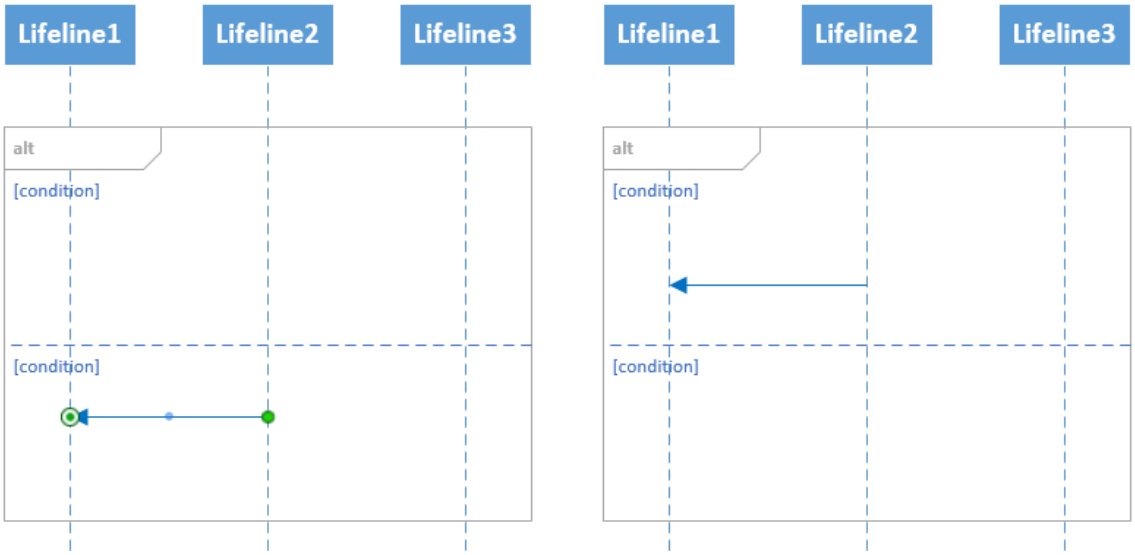
Obrázok 53: Presun správy do rodičovského fragmentu z vnoreného fragmentu

8. Presun správy do operandu kombinovaného fragmentu.
    a. Správa sa nachádza mimo kombinovaného fragmentu.



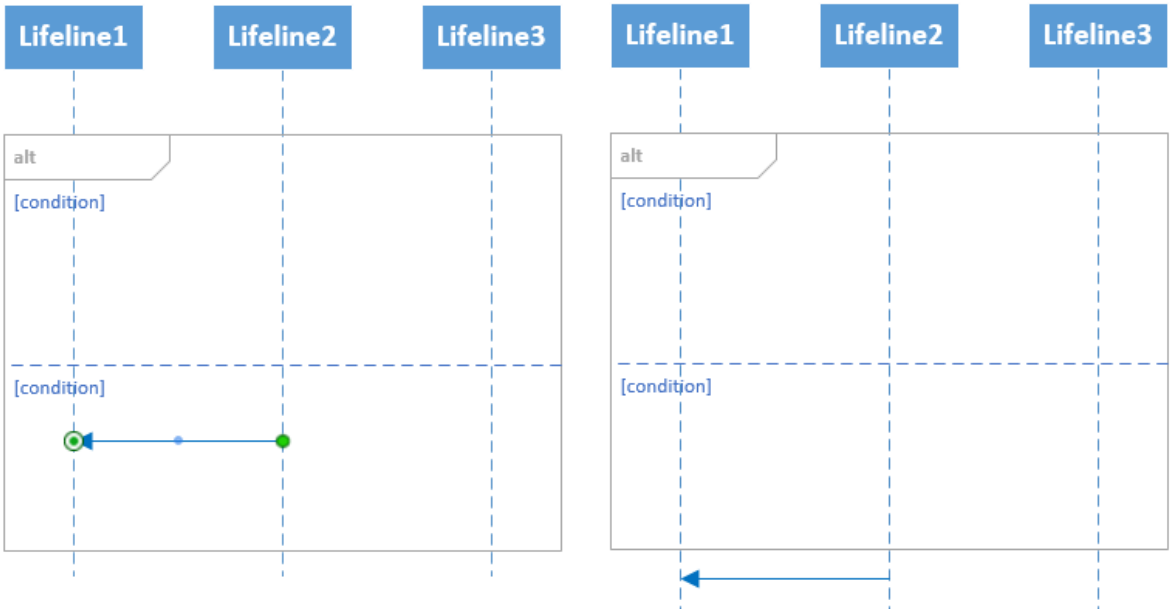
Obrázok 54: Presun správy do operandu kombinovaného fragmentu

    b. Správa sa nachádza v inom operande kombinovaného fragmentu.



Obrázok 55: Presun správy medzi operandami kombinovaného fragmentu

9. Presun správy von z operandu kombinovaného fragmentu.
    a. Správa sa presúva von z kombinovaného fragmentu.

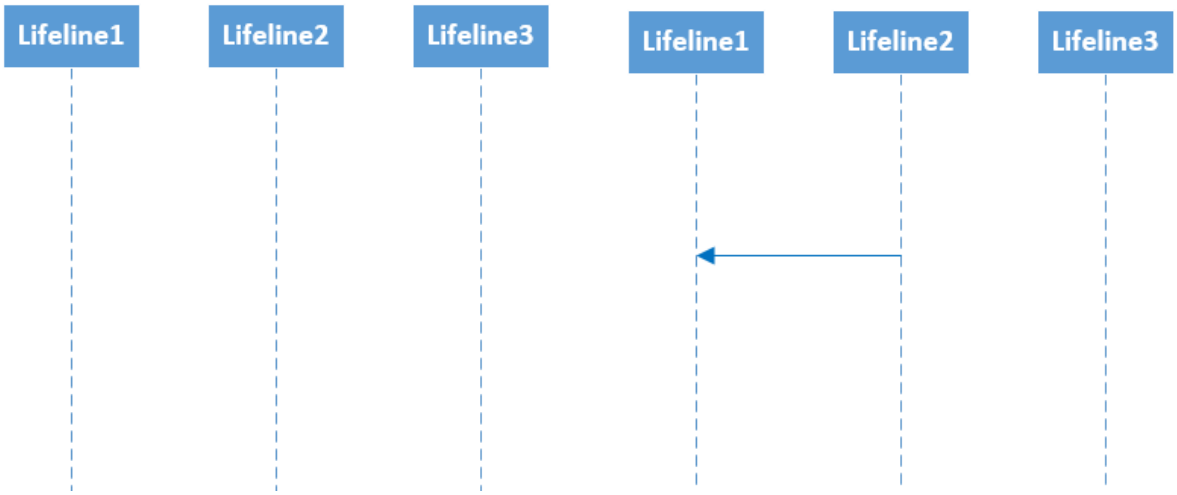


Obrázok 56: Presun správy z operandu kombinovaného fragmentu mimo fragment

3.1.5.3 Pridanie novej správy

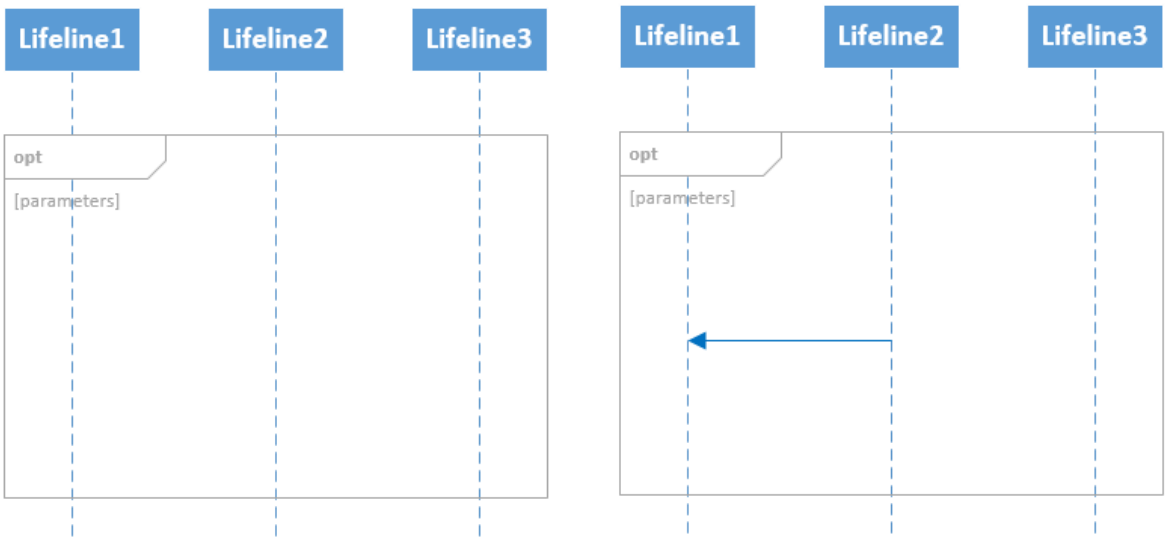
Používateľ môže pridať novú správu kdekoľvek v rámci priestoru ohraničeného dvoma označenými čiarami života kliknutím na plochu vrstvy. Správa sa graficky vloží na používateľom vybrané miesto a taktiež sa vloží na správne miesto do metamodelu diagramu. Správu je možné vkladať do akéhokoľvek korektného fragmentu interakcií, či už kombinovaného fragmentu, operandu alebo množiny za sebou idúcich správ.

1. Pridanie správy do čistého fragmentu interakcií.



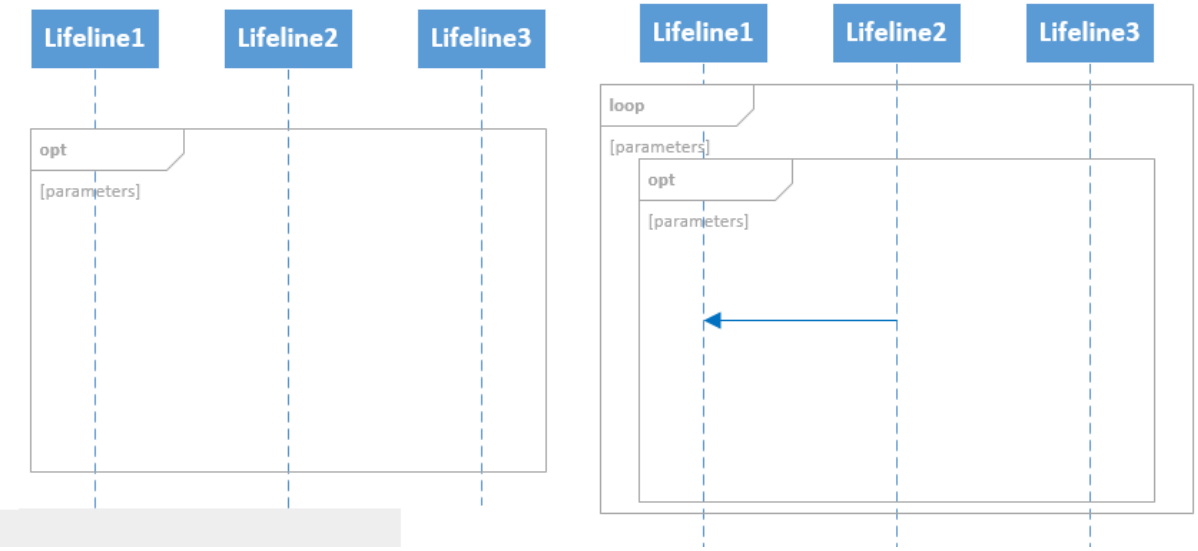
Obrázok 57: Pridanie správy do fragmentu interakcií

2. Pridanie správy do kombinovaného fragmentu.



Obrázok 58: Pridanie správy do kombinovaného fragmentu

3. Pridanie správy do vnoreného kombinovaného fragmentu.

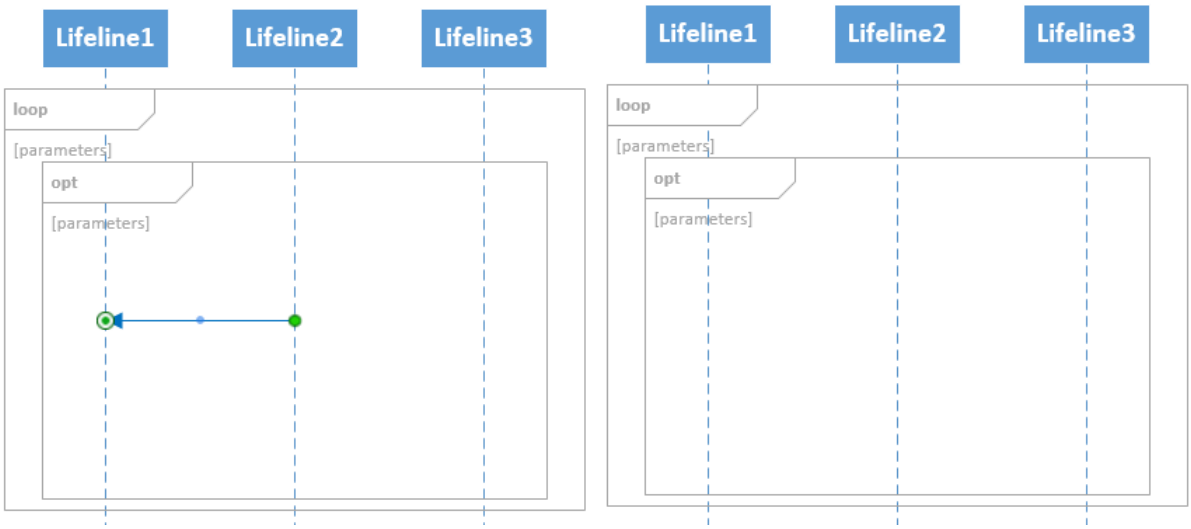


Obrázok 59: Pridanie správy do vnoreného kombinovaného fragmentu

3.1.5.4 Vymazanie správy

Používateľ môže akúkoľvek správu z diagramu vymazať. Správu najprv označí kliknutím na stred čiary správy a následne klikne pravým tlačidlom myši a zvolí možnosť "delete". Správa sa odstráni aj z metamodelu, aj z grafického zobrazenia diagramu.

1. Vymazanie správy nachádzajúcej sa na rôznom mieste v rámci sekvenčného diagramu.



Obrázok 60: Zmazanie správy z kombinovaného fragmentu

3.2 Návrh

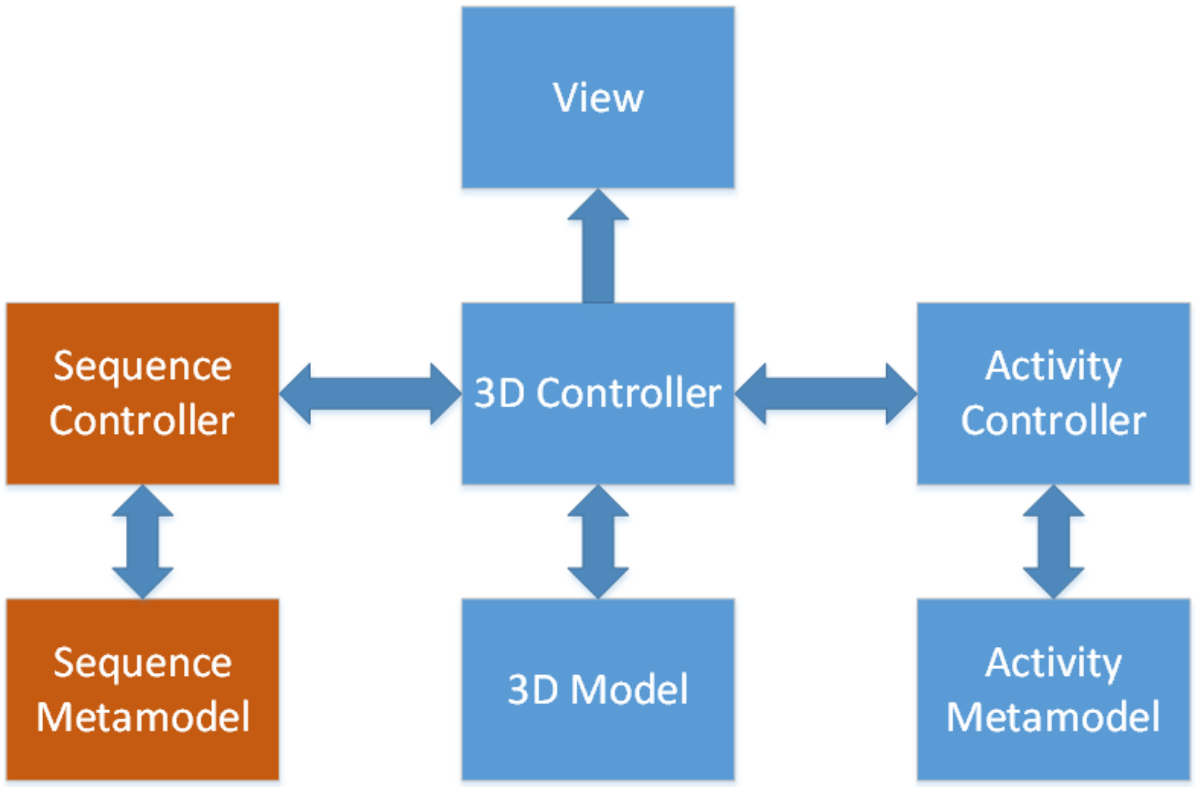
Pri analyzovaní diagramov bola identifikovaná základná funkcionalita aplikačných prototypov. Nasledoval proces spájania prototypov. Prvotným princípom bolo vytvorenie spoločnej aplikácie, ktorá by obsahovala funkcionalitu všetkých troch existujúcich prototypov, keďže doteraz fungovali ako samostatné aplikácie.

Využili sme existujúcu implementáciu návrhu novej (MMVCC) architektúry diagramu aktivít a snažili sa navrhnúť najefektívnejšiu cestu integrovania sekvenčného diagramu do tejto architektúry.

Keďže pre tvorbu pôvodného sekvenčného diagramu bola použitá architektúra MVC, bolo potrebné túto architektúru zmeniť. S pomocou vedúceho projektu sme navrhli migráciu sekvenčného diagramu do novej architektúry podľa obrázku 61.

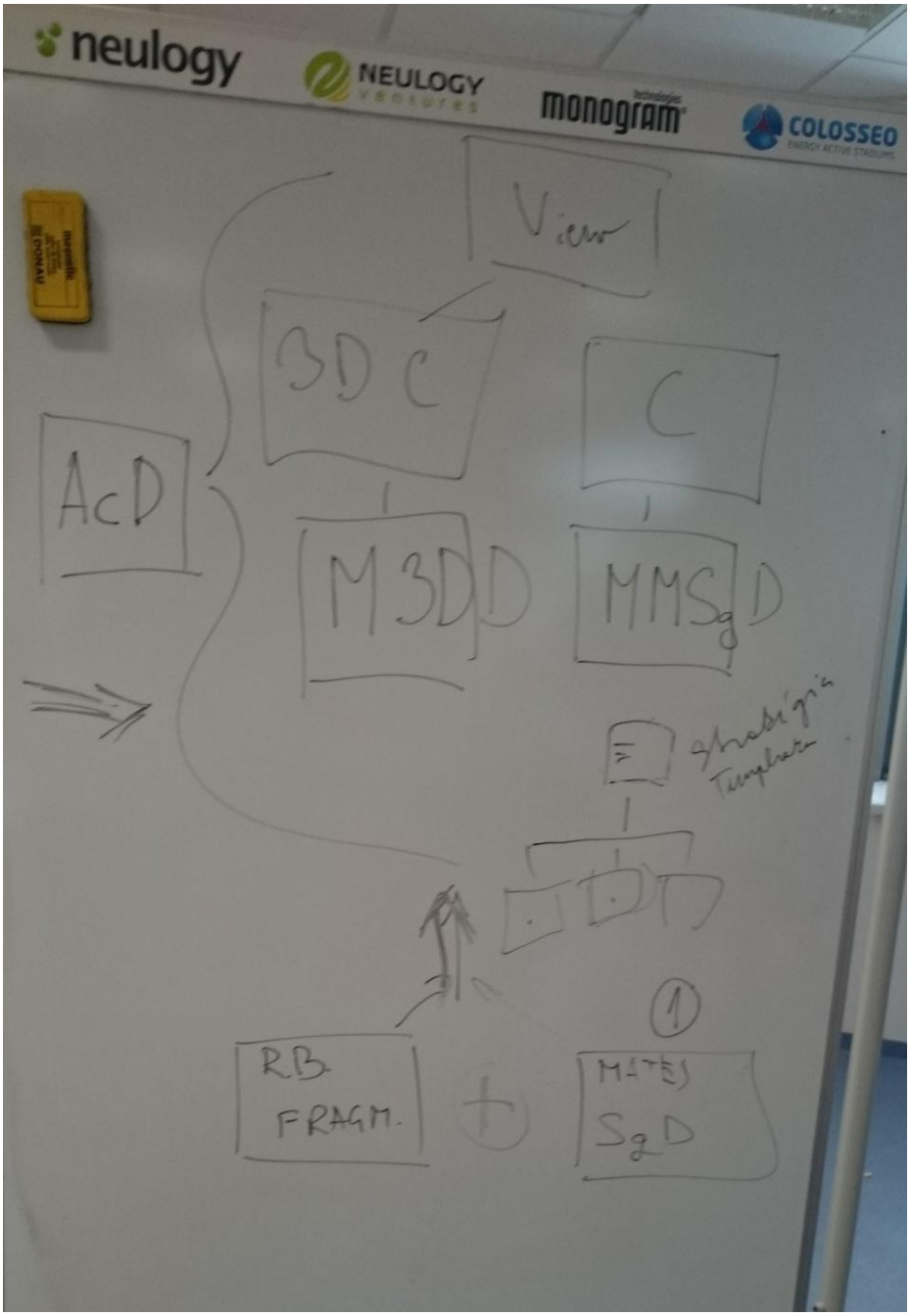
Na obrázku je vidieť architektúra diagramu aktivít spolu s pridanými komponentami pre sekvenčný diagram. Pozostáva z jedného `view` komponentu, ktorý slúži na zobrazovanie jednotlivých elementov na obrazovku, `3D controller`-a slúžiaceho na uchovávanie stavov v aplikácií a vytváranie elementov diagramu. V `3D model`-i sa nachádzajú všetky prvky, ktoré popisujú, ako by mali vyzerať jednotlivé elementy ako vrstvy. V štandardnom `controller-i` sa vytvárajú metamodel objekty definované v `SequenceMetamodel`. Tie určujú vzťahy medzi komponentami sekvenčného diagramu.

V našom návrhu implementácie vytvárame do existujúcej architektúry prototypu diagramu aktivít, prídavné moduly `SequenceController` a `SequenceModel`. Pre tieto komponenty budeme používať vytvorené triedy v prototypoch sekvenčných diagramov. V prípade nájdenia nedostatkov v architektúre odkonzultujeme s produktovým vlastníkom refaktorizáciu.



Obrázok 61: Cieľová architektúra výsledného prototypu

Obrázok nižšie, ktorý sme vytvorili počas brainstormingu na stretnutí, uvádza náhľad na štandardné pripojenie ovládača (na obrázku skratka C) a metamodel komponentu (na obrázku skratka MMSgD) k už existujúcemu diagramu aktivít.

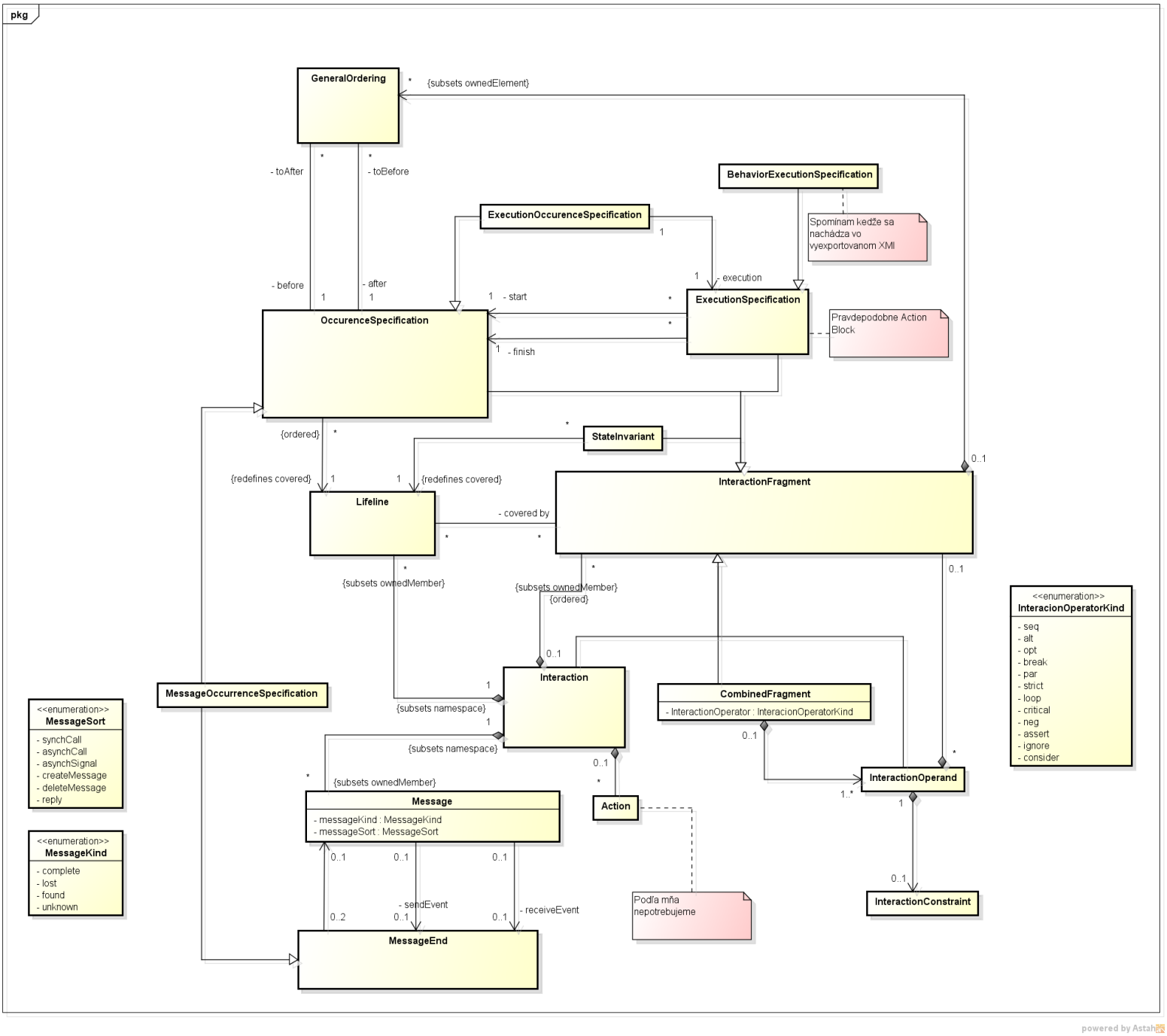


Obrázok 62: Náčrt architektúry prototypu diagramu aktivít a spôsobu naviazania zvyšných prototypov tak, ako sme ho vytvorili spolu s vlastníkom produktu

3.2.1 Návrh pridávaného Metamodelu sekvenčného diagramu

Ako je spomenuté vyššie, metamodel určuje základnú štruktúru prepojení elementov v prototype. Je odvodený od metamodelu sekvenčného diagramu v UML Superstructure[2]. Slúži na ukladanie údajov zo scény, aby sme k nim vedeli správne pristupovať a korektne s nimi pracovať. Neskôr môže slúžiť taktiež na import a export údajov z alebo do prototypu. Jeho štruktúru zobrazuje obrázok 63**.**

[2] http://www.omg.org/spec/UML/2.4.1/

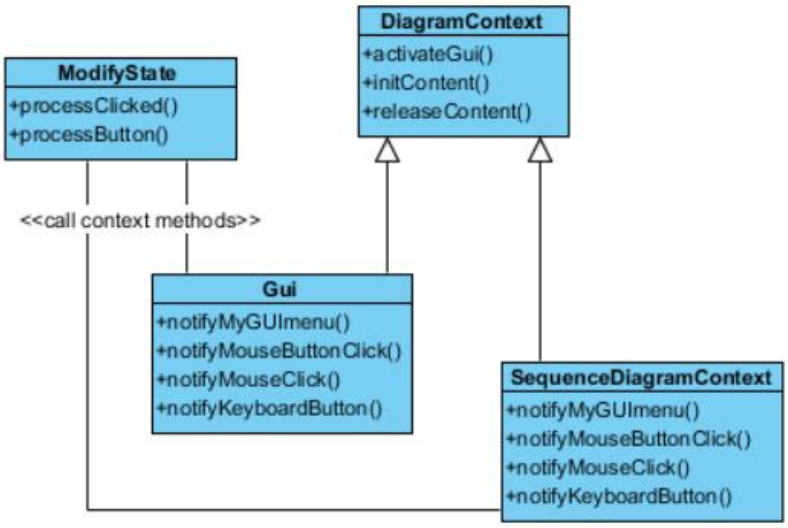


Obrázok 63: Metamodel sekvenčného diagramu

3.2.2 Rozšírenie časti 3D controller

V tomto komponente sú definované stavy aplikácie, manažéri a štruktúra udržiavajúca modelové elementy. Stavy určujú, aké akcie je možné vykonávať v programe. Napríklad `ModifyState` určuje stav, kde je možné vkladať elementy do diagramu. Manažéri zodpovedajú v aplikácii za akcie vkladania (`DataManager`), vykresľovania (`DrawManager`). Poslednou dôležitou

časťou je štruktúra zabezpečujúca vkladanie elementov do zoznamu elementov (`ElementCollection`).
Rozšírenie časti `3D controller-a` bude pozostávať z rozšírení tried `DataManager`, `DrawManager`, `ElementCollection`, `DiagramContext` pomocou dedenia na triedy `SequenceDataManager`, `SequenceDrawManager`, `SequenceElementCollection`, `SequenceDiagramContext`. Takéto rozšírenie bolo určené z dôvodu, aby sme neovplyvňovali diagram aktivít.
Všetky tieto rozšírenia sa dejú pomocou dedenia. Ako príklad je uvedené rozšírenie komponentu `DiagramContext` (obrázok 64). Pre túto akciu bolo potrebné vytvoriť novú triedu `SequenceDiagramContext`. Jej názov určuje, že sa jedná o kontext sekvenčného diagramu. Na rozdiel od časti diagramu aktivít v balíku `Gui` sa nevytvárajú žiadne globálne premenné. Všetky sú definované ako premenné v triede `SequenceDiagramContext`. V prípade práce so sekvenčným UML diagramom sa volajú metódy `SequenceDiagramContext`-u.



Obrázok 64: Návrh časti rozšírenia modulu `3D controller`

3.2.3 Rozšírenie komponentu 3D model

V časti model je potrebné pridať elementy sekvenčného diagramu. Kedže metamodel určuje vzťahy medzi komponentami sekvenčného diagramu, model bude určovať vlastnosti jednotlivých prvkov. Prepojenie medzi modelom a metamodelom bude realizované pomocou vzájomnej agregácie. Tieto triedy budú definovať údaje pre jednotlivé komponenty diagramu. Triedy (napr. `Lifeline`, `SequenceMessage`) budú dediť od triedy `SequenceElement` (obrázok 13), ktorá bude dediť od triedy `Element`. Pre veľkú zložitosť na diagrame sú uvedené len príklady tried pre pochopenie princípu dopĺňanej štruktúry.

3.2.4 Návrh pridávanej Metamodel-Controller jednotky

Pridávanú kontrolnú jednotku (controller) pre sekvenčný diagram budú tvoriť triedy, ktoré vytvárajú metamodel elementy sekvenčného diagramu a riadia operácie s metamodelom. Taktiež prideľujú `metamodel` elementom vykresľovacie algoritmy a grafické informácie. Triedy sú definované v štruktúre `Core/Sequence`. Prídavný `controller` sa nachádza v štruktúre `Core/Sequence`.

3.2.5 Modifikácia 3D View komponetu

Súčasťou `View` komponentu sú triedy dediace od triedy `DrawingAlgorithm`. Určujú spôsob vykreslenia tvaru elementu. V nich sú agregované informácie ohľadom grafického spôsobu vykreslenia (napr. použitie typu písma, šírka textu) vo forme tried dediacich od triedy `ElementGraphics`. V prototype je potrebné postupne pridávať pre každý element vlastné triedy pre vykreslenie implementovaných elementov. Tie sa budú vkladať do štruktúry `Graphics/sequence`. Na obrázku 65 je uvedený príklad, kde je možné vidieť štruktúru jednotlivých vykresľovacích tried a grafických informácií. Kvôli veľkej zložitosti je diagram tried len ukážkový a neobsahuje všetky triedy zahrnuté v štruktúrach.

3.2.6 Metamodel a kombinované fragmenty

V tejto časti podkapitoly popisujeme čitateľovi zmeny, ktoré je potrebné vykonať na realizáciu príslušných editačných funkcií na metamodelovej úrovni pri kombinovaných fragmentoch. Konkrétne sú tu opísané zmeny na metamodeli pri vykonaní nasledujúcich akcií:

- pridanie kombinované fragmentu a vnoreného kombinovaného fragmentu,
- zmazanie kombinovaného fragmentu,
- škálovanie kombinovaného fragmentu,
- pridanie operandu do kombinovaného fragmentu
- zmazanie operandu z kombinovaného fragmentu

3.2.6.1 Pridanie kombinovaného fragmentu

V rámci sekvenčného diagramu je možné pridávať do neho kombinované fragmenty. Kombinované fragmenty sú buď jednoduché, alebo vnorené. Pod vnoreným kombinovaným fragmentom sa chápe fragment, ktorý je obsiahnutý v nejakom inom kombinovanom fragmente.

Pridanie jednoduchého kombinovaného fragmentu do diagramu treba odlíšiť v rôznych prípadoch. Nie je však potrebné rozlišovať medzi pridaním fragmentu do vrstvy a do iného kombinovaného fragmentu, keďže vrstva je reprezentovaná triedou `CombinedFragment`. Prípady, ktoré môžu nastať, sú nasledujúce:

1. kombinovaný fragment obalí všetky správy, ktoré sú medzi dvomi čiarami života, t.j. všetky správy daného interakčného fragmentu. V takomto prípade nie je potrebné deliť operand, takže kombinovaný fragment len nahradí pôvodný interakčný fragment.

2. kombinovaný fragment obalí niekoľko dolných alebo niekoľko horných správ daného interakčného fragmentu. V tomto prípade sa pôvodný jeden operand interakčného fragmentu rozdelí nasledujúcim spôsobom:
    a. správy mimo vytváraného kombinovaného fragmentu budú súčasťou operandu, ktorý bude obsahovať interakčný fragment,
    b. správy, ktoré má kombinovaný fragment obsiahnuť, sa stanú súčasťou novovytvoreného operandu, ktorý bude obsahovať kombinovaný fragment.
3. kombinovaný fragment obalí niekoľko správ, pričom nad a aj pod kombinovaným fragmentom zostanú správy, ktoré nebudú jeho súčasťou. V tomto prípade sa pôvodný jeden interakčný operand rozseká na 3 časti nasledujúcim spôsobom:
    a. správy, ktoré sa budú nachádzať nad kombinovaným fragmentom, budú súčasťou jedného interakčného operandu, ktorý bude v sebe obsahovať interakčný fragment,
    b. správy, ktoré sa stanú súčasťou kombinovaného fragmentu, budú vložené do novovytvoreného interakčného operandu, ktorý v sebe bude obsahovať kombinovaný fragment,
    c. správy, ktoré sa budú nachádzať pod fragmentom, sa stanú súčasťou nového interakčného operandu, ktorý bude v sebe obsahovať interakčný fragment.

Vnorený kombinovaný fragment sa vkladá označením správ, ktoré sa nachádzajú v nejakom existujúcom kombinovanom fragmente. V kontexte metamodelových zmien dôjde k takému istému rozsekaniu pôvodných operandov, ako bolo popísané v prípade vloženia jednoduchého kombinovaného fragmentu. Ešte raz podotýkame, že pre ekvivalentné chápanie jednoduchých a vnorených kombinovaných fragmentov je potrebné, aby vrstva bola definovaná ako `CombinedFragment`.

Načrtnime ešte prípad, kedy vkladaný kombinovaný fragment môže obaliť správy, nad ktorými sa nachádza iný kombinovaný fragment. Tento je prirodzene súčasťou nejakého interakčného operandu a správy pod ním (ktoré novovytváraný fragment obalí) sú súčasťou iného interakčného operandu. V závislosti od toho, či má fragment obaliť všetky správy v tomto operande, alebo len časť z nich, sa príslušný operand rozdelí (v prípade obalenia všetkých správ nie je potrebné delenie operandu, avšak v prípade obalenia len časti správ je potrebné rozseknúť pôvodný interakčný operand takým spôsobom, ako je opísané v bodoch 3a), 3b) a 3c).

Obalenie správ v danom interakčnom operande kombinovaným fragmentom teda vyvolá rozseknutie tohto operandu na jednu, dve alebo tri časti tak, ako sme to opísali vyššie.

V tomto momente ešte netreba zabudnúť na správanie `GeneralOrdering-ov`. V prípade, že sa správy nachádzajú len v jednom operande, ich príslušné GeneralOrderingy sú prirodzene pospájané. V momente, keď sa daný operand so správami rozsekne na viacero častí, rozseknú sa aj príslušné `GeneralOrdering-y`, a to tak, že posledná správa v danom interakčnom operande bude mať nasledujúci `GeneralOrdering` nastavený na *NULL*.

3.2.6.2 Zmazanie kombinovaného fragmentu

Zmazanie kombinovaného fragmentu môžeme z metamodelového hľadiska uvažovať na dvoch úrovniach:

- deštruktívne zmazanie, ktoré spôsobí kompletné vymazanie nielen fragmentu, ale aj celého jeho obsahu, t.j. všetkých iných fragmentov či správ v ňom,

- nedeštruktívne zmazanie, ktoré zachová obsah fragmentu a presunie tento obsah v metamodeli o úroveň vyššie.

Načrtnime rekurzívnu štruktúru kombinovaných fragmentov v metamodeli:

- kombinovaný fragment je garantovane súčasťou nejakého interakčného operandu,
- kombinovaný fragment obsahuje (min. jeden) interakčný operand, ktorý obsahuje buď interakčný fragment, alebo ďalší kombinovaný fragment.

Deštruktívne zmazanie by teda muselo odstrániť celý kombinovaný fragment (rekurzívne). Následne by muselo nastať zlúčenie jednotlivých operandov. Vzhľadom na to, že správy je možné mazať individuálne, navrhujeme pre mazanie použiť nedeštruktívny prístup.

Pri nedeštruktívnom prístupe je z metamodelového hľadiska potrebné zabezpečiť, aby sa interakčný operand, ktorý daný kombinovaný fragment obsahuje, presunul rekurzívne o práve jednu úroveň vyššie. Pre ilustráciu uvažujme nasledujúci príklad.

Predpokladajme, že máme na vrstve tri interakčné operandy, z toho jeden (bez ujmy na všeobecnosti nech je to stredný) obsahuje kombinovaný fragment. Kombinovaný fragment v sebe obsahuje garantovane ďalší interakčný operand. Zmazanie tohto vnoreného kombinovaného fragmentu musí presunúť jeho interakčný operand o úroveň vyššie, t.j. na úroveň vrstvy. Týmto sa zabezpečí aj presunutie celého obsahu na vrstvu (a to aj v prípade, že kombinovaný fragment obsahuje ďalšie vnorené kombinované fragmenty; v tomto prípade sa zníži hĺbka vnorenia o jednotku).

Presunutie interakčného operandu rekurzívne o úroveň vyššie však samo o sebe nemusí stačiť. V príklade opísanom vyššie by nastal problém v prípade, že by vnorený kombinovaný fragment v sebe neobsahoval ďalší vnorený kombinovaný fragment, ale len správy. V tomto prípade by sme vo výsledku mali tri interakčné operandy na vrstve, z ktorých každý by obsahoval len interakčný fragment. V tomto momente by malo byť zrejmé, že tieto tri interakčné operandy možno zlúčiť do jedného operandu a príslušné `GeneralOrdering-y` možno pospájať dokopy.

Z tohto dôvodu navrhujeme, aby po zmazaní kombinovaného fragmentu, t.j. presunutí jeho interakčného operandu rekurzívne o úroveň vyššie, došlo zároveň aj ku kontrole, či nevznikli také interakčné operandy, ktoré je možné spojiť do jedného. Do jedného možno spojiť každé dva také interakčné operandy, ktoré sa nachádzajú v metamodeli "pod sebou", aspoň jeden z nich neobsahuje podmienku (`constraint` je `NULL`) a obsahujú interakčné fragmenty. V prípade, že neexistuje viacero interakčných operandov, ale len jeden operand, nie je potrebné zlučovací mechanizmus aktivovať. Takýto prípad nastane napr. vtedy, keď vrstva obsahuje len jeden kombinovaný fragment, ktorý je zároveň jediným interakčným operandom tejto vrstvy, a keď tento kombinovaný fragment zmažeme.

Špeciálny prípad mazania nastáva v prípade, že mazaný kombinovaný fragment obsahuje viac ako jeden operand. Prenášať tieto interakčné operandy na vonkajšie kombinované fragmenty (alebo nebodaj na vrstvu, čo by nebolo správne ani z metamodelového hľadiska) nepovažujeme za rozumné. Preto navrhujeme, aby tieto operandy boli najprv vymazané, a to tak, že ich podmienka bude nastavená na `NULL`. O zlúčenie operandov sa následne postará zlučovací mechanizmus uvedený vyššie. Obsah operandov sa takto prirodzene zachová.

3.2.6.3 Pridanie operandu do kombinovaného fragmentu

Kombinovaný fragment môže obsahovať viac ako jeden operand. Každý operand kombinovaného fragmentu okrem toho môže obsahovať podmienku, napr. v prípade LOOP fragmentu je táto podmienka zvyčajne ukončovacou podmienkou cyklu. Pridanie operandu do kombinovaného fragmentu navrhujeme riešiť nasledujúci dvomi alternatívami:

1. v prípade, že je potreba pridať operand do kombinovaného fragmentu, ktorý už obsahuje správy, navrhujeme označiť niektorú zo správ. V prípade, že bude zvolená prvá správa, nedôjde k pridaniu operandu, ale len bude možnosť zmeniť podmienku prvého operandu kombinovaného fragmentu. V prípade, že používateľ zvolí druhú, tretiu, …, až poslednú správu, vyvolá pridanie nového operandu kliknutím nad ňu so zvolenou operáciou pre pridanie operandu,
2. v prípade, že je potreba pridať operand do kombinovaného fragmentu nad nejaký iný, vnorený kombinovaný fragment, navrhujeme pridanie operandu realizovať tak, že bude označený príslušný vnorený kombinovaný fragment a kliknutím nad neho so zvolenou operáciou pre pridanie operandu dôjde k samotnému pridaniu operandu.

Z metamodelového hľadiska znamená pridanie nového operandu do kombinovaného fragmentu vytvorenie nového interakčného operandu v tomto fragmente spolu so zadaním podmienky (angl. *constraint*) pre tento operand. Pre zvolenú správu, resp. kombinovaný fragment sa pôvodný interakčný operand kombinovaného fragmentu rozsekne na dve časti takýmto spôsobom:

- zvolená správa, resp. kombinovaný fragment určuje miesto rozseknutia daného interakčného operandu na dve časti,
- označený kombinovaný fragment bude pridaný alebo označená správa bude pridaná do novovytvoreného interakčného operandu (druhá časť), pričom predchádzajúci `GeneralOrdering` (v prípade správy) bude nastavený na `NULL`,
- správa nachádzajúca sa alebo kombinovaný fragment nachádzajúci sa nad zvolenou správou bude zaradená, resp. zaradený do pôvodného interakčného operandu (prvá časť), pričom nasledujúci `GeneralOrdering` (v prípade správy) bude nastavený na `NULL`.

3.2.6.4 Zmazanie operandu z kombinovaného fragmentu

Operandy kombinovaného fragmentu možno mazať. Ako prvé navrhujeme, aby pred zmazaním operandu bolo nutné označiť ten kombinovaný fragment, ktorý sa bude mazať. Následne sa zvolí operácia mazania operandu. Samotné zmazanie operandu navrhujeme z metamodelového hľadiska riešiť nasledujúco:

- pre mazaný operand je potrebné zmazať podmienku,
- po zmazaní podmienky navrhujeme aplikovať mechanizmus ako v prípade zmazania kombinovaného fragmentu, ktorý zabezpečí zlúčenie každých dvoch interakčných operandov, ktoré obsahujú interakčný fragment a aspoň jeden z nich neobsahuje podmienku, do jedného operandu.

Modelovo si túto situáciu môžme predstaviť pre kombinovaný fragment s dvomi operandami, pričom chceme z tohto kombinovaného fragmentu zmazať druhý operand. Formálne najprv v metamodeli zanikne podmienka druhého operandu, následne je príslušný operand

zlúčený s prvým interakčným operandom, čo vyústi vo vytvorenie len jedného interakčného operandu. Analogicky bude situácia fungovať aj pri mazaní kombinovaného fragmentu s viacerými operandami. V jednom kroku je možné zmazať len jeden operand kombinovaného fragmentu.

Pri implementácii treba dať pozor na to, aby sa prvý operand fragmentu nedal zmazať, keďže fragment musí vždy aspoň jeden operand obsahovať. Možno však upravovať podmienku prvého operandu.

3.2.6.5 Škálovanie kombinovaného fragmentu

Škálovanie je z hľadiska metamodelových zmien najkomplexnejšie. Komplexnejšie je už len hýbanie fragmentu (horizontálne, vertikálne a nakoniec horizontálno-vertikálne), ktoré sme v rámci tohto projektu vzhľadom na tento fakt nerealizovali.

Vzhľadom na komplexnosť škálovania kombinovaného fragmentu uvedieme metamodelové zmeny na dvoch úrovniach: najprv škálovanie, ktoré zahŕňa prácu so správami, a následne škálovanie, ktoré zahŕňa prácu s inými kombinovanými fragmentami.

Z metamodelového hľadiska môže byť kombinovaný fragment zhora aj zdola obklopený správami, t.j. inými interakčnými operandami. Zväčšenie kombinovaného fragmentu musí zabezpečiť pridanie vybraných správ do neho. Formálne sa teda musia presunúť správy z interakčného operandu pod fragmentom alebo nad fragmentom do interakčného operandu, ktorý reprezentuje kombinovaný fragment. Rovnako sa musia tieto správy napojiť na `GeneralOrdering` správ vo fragmente (v prípade pridávania správ pod kombinovaným fragmentom sa musia napojiť na nasledujúci `GeneralOrdering` poslednej správy, naopak v prípade prídavania správ nad kombinovaným fragmentom sa musia napojiť na predchádzajúci `GeneralOrdering` prvej správy). Treba si uvedomiť, že zväčšenie fragmentu môže potenciálne úplne odstrániť správy z interakčného fragmentu pod ním, resp. nad ním: príslušný operand reprezentujúci daný interakčný fragment teda formálne zaniká a počet operandov vrstvy, resp. kombinovaného fragmentu sa zníži o jednotku.

Zmenšovanie kombinovaného fragmentu reprezentuje inverznú operáciu. V prípade zmenšenia kombinovaného fragmentu je potrebné zabezpečiť, aby sa príslušné správy, ktoré už nebudú pokryté kombinovaným fragmentom, presunuli do interakčného operandu nad fragmentom, resp. pod fragmentom, a adekvátne sa prepojili GeneralOrderingy. Tak ako pri zväčšovaní fragmentu, tak aj pri zmenšovaní je potrebné myslieť na to, že interakčný operand nad fragmentom, resp. pod fragmentom nemusí existovať, a je teda najprv ho potrebné vytvoriť (počet operandov vrstvy, resp. kombinovaného fragmentu v tomto prípade narastie o jednotku).

Zväčšenie kombinovaného fragmentu tak, aby obklopil aj iný kombinovaný fragment, je z metamodelového hľadiska potrebné tiež vyriešiť. Formálne sa iný kombinovaný fragment má vnoriť do zväčšovaného kombinovaného fragmentu. Iný kombinovaný fragment môže nad sebou a aj pod sebou obsahovať správy, t.j. interakčné operandy. Interakčný operand, ktorý reprezentuje vnáraný kombinovaný fragment, sa formálne musí stať interakčným operandom zväčšovaného kombinovaného fragmentu. Ak vnáraný kombinovaný fragment obsahuje nad sebou správy, je potrebné pri zväčšovanom kombinovanom fragmente overiť dve situácie:

- zväčšovaný fragment obsahuje ako posledný element správu správu - v takomto prípade je možné napojiť správy nad vnáraným fragmentom cez `GeneralOrderingy`,

- zväčšovaný fragment neobsahuje ako posledný element správu (môže obsahovať napr. iný vnorený kombinovaný fragment) - v takomto je pre správy nad vnáraným fragmentom potrebné vytvoriť nový operand.

Správy pod vnáraným kombinovaný fragmentom sú pridané vo forme ďalšieho interakčného operandu. Popísané kroky fungujú pre škálovanie smerom nadol. Pre škálovanie smerom nahor sa inverzne otáča logika (t.j. uvažuje sa, či zväčšovaný fragment obsahuje ako prvý element správu).

Zmenšovanie kombinovaného fragmentu tak, aby sa v ňom obsiahnutý vnorený fragment vynoril, funguje inverzne a analogicky k zväčšovaniu fragmentu.

3.3 Implementácia

Vývoj cieľového produktu je vytváraný metódou modifikovanej agilnej metodiky SCRUM. Používa sa agilný vývoj a vytvárajú sa funkčné prototypy, ktoré spĺňajú jednotlivé položky z produktového backlogu.

Naším hlavným cieľom na zimný semester bolo vytvoriť prototyp sekvenčného diagramu s dodržaním architektúry vytvorenej minuloročným tímom a základnou funkcionalitou vkladania elementov do diagramu. Do prototypu sme ako prvé natiahli metamodel sekvenčného diagramu, ktorý je miernou úpravou metamodelu definovaného v UML Superstructure. Jeho štruktúra je zobrazená na obrázku 11. Pre metamodel bol priradený samostatný balíček v prototype, podobne ako tomu je aj v metamodeli diagramu aktivít. Na základe definovaných prepojení medzi triedami metamodelu v diagrame sme vedeli definovať vzťahy agregácie či dedenia aj implementačne. Implementáciou metamodelu sme si vytvorili podmienky pre začiatok grafickej a logickej implementácie jednotlivých scenárov.
Ako sme spomenuli, pre nás dôležitým používateľským príbehom je vloženie elementu do scény. Tento proces bol však vzhľadom na rozsiahlosť prototypu príliš komplexný a preto sme ho rozdelili na niekoľko podúloh, ktorých priebeh a implementácia sú opísané nižšie.

3.3.1 Algotimus vykreslenia grafickej plochy s nefunkčným menu

Logicky prvým krokom, ktorý používateľ po spustení aplikácie vykoná, je zvolenie typu diagramu, s ktorým chce pracovať. Naši predchodcovia vytvorili prostredie, v ktorom je možnosť výberu sekvenčného diagramu priamo poskytnutá, no nefunkčná. Na tento fakt sme nadviazali a tlačidlo sekvenčného diagramu sme oživili. Prvým krokom bolo vytvorenie prázdnej scény po kliknutí na tlačidlo. Nakoľko vytvorenie kompletnej scény je implementované v “aktivity” časti prototypu, jediné čo bolo treba zmeniť, boli jednotlivé elementy diagramu aktivít za elementy sekvenčného diagramu. V menu sa teda nezobrazia elementy ako aktivity, ale budú nahradené napríklad čiarou života a správou. Za týmto účelom sme vytvorili XML súbory, v ktorých definujeme vlastnosti tlačidiel úvodného menu. V týchto súboroch sme definovali pozíciu, veľkosť a názov tlačidiel pre naše čiary života, správy a vrstvy. Rovnako sme týmto tlačidlám určili nadelement, v ktorom sú obsiahnuté. Pre všetky tieto tlačidlá to bolo menu v pravom hornom rohu obrazovky. Pri inicializácii scény tieto XML súbory spracujeme a jednotlivé tlačidlá

si uložíme do vhodných dátových štruktúr a priradíme ku každej `controller`. Po vykonaní spomenutých činností máme vykreslenú prázdnu scénu, v ktorej pravom hornom rohu sa nachádza menu obsahujúce nefunkčné tlačidlá elementov sekvenčného diagramu.

3.3.2 Algoritmus vykreslenia vrstiev

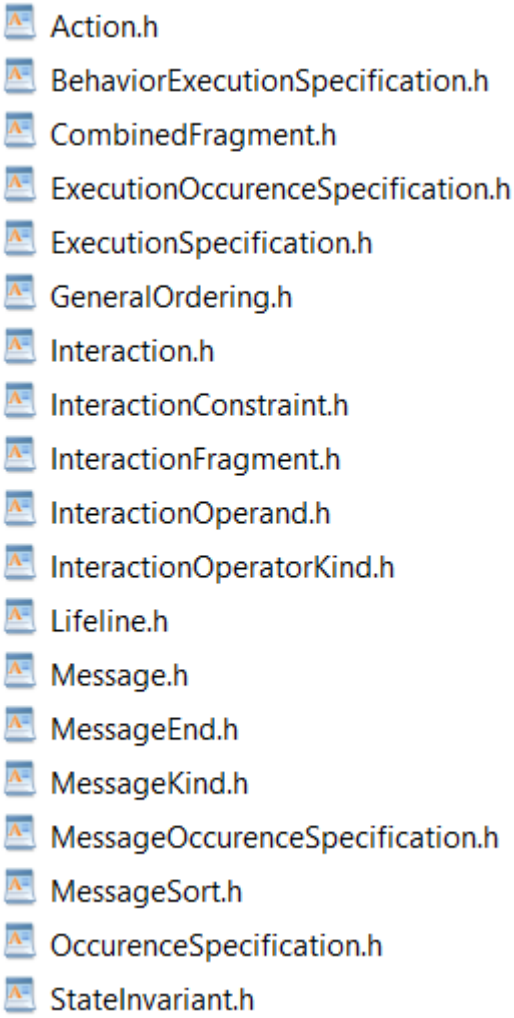
Vykreslenie vrstiev, pridávanie ďalších vrstiev je v prototype korektne vytvorené. Rozhodli sme sa používať už vytvorené komponenty prototypu diagramu aktivít. Vykreslenie jednotlivých vrstiev sme implementovali pomocou vzoru metódy `createLayer` triede `DiagramContext`, ktorá inicializuje vykreslenie vrstiev pre diagram aktivít. Táto metóda je dekorovaná o funkcionalitu špecifickú pre sekvenčný diagram v metóde `createSequenceLayer`. Nasledujúci obrázok (obrázok 65) zobrazuje reálnu implementačnú časť, ktorá zodpovedá za vykreslenie prázdnej scény. Pohyb medzi vrstvami a prepínanie aktuálnej scény bolo dotvorené podľa implementácii v triede `Gui`.

```
}else{
    lay1 = dataM->createSequenceLayer();
    layers.push_back(lay1);
}
```

Obrázok 65: Volanie vykresľovacieho algoritmu

3.3.3 Prototyp s implementovaným Metamodelom komponentom

Po navrhnutí `Metamodel` komponentu pre sekvenčný diagram sme implemetovali do štruktúry `Core/SequenceMetamodel/` určené triedy. Dedením od triedy NamedElement sme zabezpečili rozšírenie vlastností a správania. Trieda `NamedElement` obsahuje taktiež referenciu na triedu `ElementGraphics`, čím pridávame grafické informácie prvku. Na obrázku nižšie sú zobrazené hlavičkové súbory vytváraných tried.



Obrázok 66: Hlavičkové súbory vytváraných tried

3.3.4 Čiary života

Pre vloženie `Lifeline` komponentu bolo potrebné modifikovať časti `3D controller-a`, `3D model-u`, a `View` komponentu. Následne boli vytvárané potrebné triedy pre správne uloženie a následné vykreslenie čiary života. Pre vytvorenie bolo potrebné vykonať nasledujúce akcie:

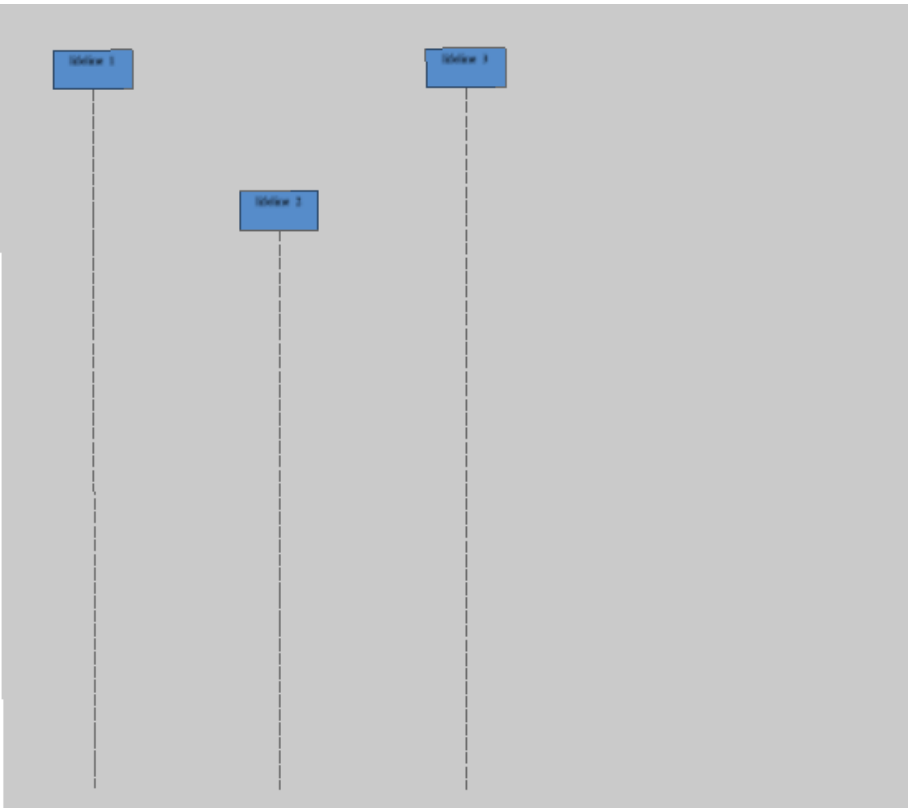
1. Definovať v metamodeli a modeli `Lifeline` triedu.
2. Vytvoriť `LifelineFactory` triedu, ktorá by pracovala s a vytvárala metamodelový komponent lifeline a prideľovala grafické informácie.
3. Vytvoriť triedu `LifelineGraphics`, ktorá obsahuje informácie ohľadom vykreslenia čiary života.
4. Vytvoriť `LifelineAlgorithm`, v ňom je implementovaná metóda `draw(std::string)`, ktorá vykreslí čiaru života.
5. V triede `SequenceManager` pridať metódu `createLifeline`, ktorá inicializuje `LifelineFactory` a `SequenceElementFactory`.

`SequencElementCollection` vloží čiaru života do vektorovej štruktúri, odkiaľ bude dostupná celému programu.

6. Modifikovať `SequenceDiagramContext`, aby inicializoval vytváranie komponentu a zobrazenie korektného editovacieho okna.

3.3.4.1 Algoritmus vloženia a vykreslenia komponentu čiary života

Proces vkladania čiary pozostáva z určenia výslednej polohy čiary života po kliknutí na vrstvu, vytvorenia objektu čiary života a výsledným vykreslením čiary života. Pre určenie výslednej polohy sme v prototype zaviedli premennú `horizontalBox`. Ide o premennú, ktorá predstavuje stĺpec vrstvy diagramu. Tento stĺpec má nemennú šírku. Aktuálne je každá vrstva rozdelená na 6 fiktívnych stĺpcov. Výsledná poloha čiary života sa bude nachádzať v niektorom z týchto stĺpcov. Po kliknutí na vrstvu sa na základe x a y súradnice kliknutia určí hodnota premennej `horizontalBox` (napr. 2 pre tretí stĺpec zľava, 3 pre štvrtý atď.). Na záver sa už len zavolá metóda, ktorá dokáže na základe horizontal boxu umiestniť čiaru života do scény.



Obrázok 67: Vykreslenie čiar života

Horizontálne sa čiara života bude nachádzať vždy v strede svojho stĺpca. Premenná `horizontalBox` je obzvlášť výhodná pre vkladanie čiary života medzi dve už existujúce čiary života. Jednoducho určíme stĺpec, v ktorom sa čiara života má nachádzať, a pozrieme sa, v ktorej polovici tohto stĺpca sa element bude nachádzať. Uvažujme príklad na **obrázku 67,** kde používateľ klikol medzi prvú a druhú čiaru života bližšie k prvej. Na základe súradníc vieme určiť, že čiara života bude mať `horizontalBox` rovný 0. Vieme však taktiež, že súradnice používateľovho kliknutia sa nachádzajú v pravej polovici tohto stĺpca. To znamená, že nová čiara

života sa bude nachádzať za prvou čiarou života z obrázka. `HorizontalBox` novo vznikajúceho elementu preto o jedna zvýšime a takisto o jedna zvýšime aj všetky horizontalBoxy, ktorých hodnota je väčšia alebo rovná hodnote `horizontalBoxu` novovznikajúcej čiary života. Na záver tohto procesu vieme na základe hodnôt `horizontalBoxov` vykresliť všetky elementy na správnu pozíciu. Vertikálna poloha nie je vo finálnej podobe nijak obmedzovaná a čiara života bude umiestnená vo výške, na akú používateľ klikol. Výsledné vykreslenie je zobrazené na **obrázku 17**. Vkladanie je možné na rôzne vrstvy.

3.3.4.2 Algoritmus pohybu s komponentom čiary života

Pri pohybe čiarou života sme využili metódu Drag & Drop. Používateľ chytí čiaru života, pohybuje s ňou podľa potreby a pri pustení tlačidla sa čiara života umiestni na výslednú pozíciu. Jadrom celého procesu je metóda `LifelineAlgorithm::translate`, ktorá je volaná vždy, keď je zachytený pohyb myšou, tlačidlo myši je stlačené a označený objekt je čiara života. Vstupom tejto metódy sú okrem iného aj x a y súradnice (z súradnica sa nemení), na ktorých sa práve kurzor myši nachádza. Na túto pozíciu prekreslíme označenú čiaru života, čím spôsobíme, že čiara života sa bude hýbať neustále spolu s kurzorom myši. Počas pohybu myšou v metóde translate kontrolujeme, či čiara života neprešla do vedľajšieho stĺpca a nie je potrebné zmeniť hodnotu premennej horizontalBox. Podľa x a y súradníc taktiež vieme určiť, či nedochádza ku kolízii medzi dvoma čiarami života. Ak ku kolízii dôjde, dané dve čiary života sa vymenia. V kóde to znamená, že hodnoty premenných `horizontalBox` si tieto dve čiary života vymenia. Následne sa všetky čiary života na danej vrstve okrem označenej prekreslia podľa hodnoty ich `horizontalBoxu`. Súčasťou pohybu s čiarou života, je aj pohyb správ, ktoré z danej čiary života vychádzajú a vchádzajú. Pre túto činnosť slúži metóda `translateMessagesForLifeline`, ktorá dynamicky pre každú správu určí, či sa posúva jej zdroj alebo cieľ, a následne prepíše hodnotu zvoleného zdroja alebo cieľa na novú. Po určení novej polohy potom stačí správu nanovo prekresliť.

3.3.4.3 Algoritmus výmeny dvoch čiar života

V prípade pohybu vybranej čiary života po scéne môže nastať situácia, kedy prechádzame vybranou čiarou života cez stredový bod inej čiary života. V takomto prípade, aby sme zachovali koncepciu usporiadania všetkých komponentov na danej vrstve, vymeníme pozície týchto dvoch čiar života. Samotná výmena je implementovaná v triede `LifelineAlgorithm`. V tejto triede sa nachádza metóda `translate`, ktorá sa využíva pri pohybe vybranej čiary života po scéne. V tejto metóde sa sleduje aj pohyb vybranej čiary života, ktorý je potrebné sledovať pre samotnú výmenu.

Pohyby, ktoré nás zaujímajú, sú pohyby doprava a doľava. Informáciu o pohybe dokážeme zistiť podľa klesajúcej resp. rastúcej x-ovej osi. Táto informácia je pre nás dôležitá hlavne z toho dôvodu, aby sme vedeli sledovať situáciu, kedy hodnota x-ovej súradnice vybranej čiary života presiahne hodnotu inej čiary života na vrstve a zároveň budeme ignorovať pozície ostatných čiar života, ktoré sa nachádzajú v opačnom smere jej pohybu. Napr. ak sa pohybujeme smerom doľava, porovnávame hodnotu x-ovej súradnice vybranej čiary života s hodnotami x-ových súradníc čiar života, ktoré sa nachádzajú v jej ľavej časti. Ak nastane stav, kedy je x-ová súradnica vybranej čiary života menšia ako hodnota jej suseda, dochádza k výmene. V prípade,

že by sme neignorovali ostatné čiary života v opačnom smere pohybu (v našom príklade napravo, ak nejaké existujú), táto podmienka by bola vždy platná, a preto by sme nedosiahli stav, ktorý požadujeme.

Pri samotnej výmene je potrebné zabezpečiť:

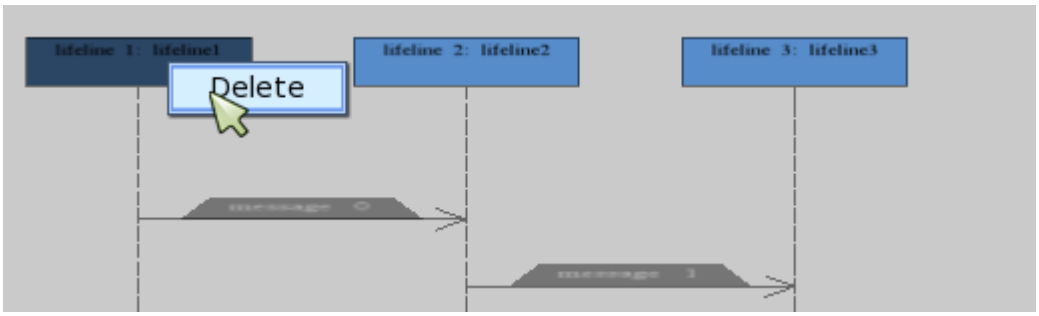
1. **Zmena pozície čiar života v metamodeli**
   Pri výmene je potrebné vymeniť referencie vymieňajúcich sa čiar života v metamodeli. Pomocou metódy `InteractionFragment::getLifelineVector` získame vektor s reálnymi inštanciami čiar života. V tomto vektore zabezpečíme výmenu referencií dvoch vymieňajúcich čiar života na základe ich indexov v tomto vektore.

2. **Prekreslenie správ obsiahnutých vo vymenených čiarách života a prekreslenie ich smeru**
   Po uskutočnení výmeny je potrebné tejto výmene prispôsobiť aj prekreslenie správ, ktoré vchádzajú alebo vychádzajú z vymenených čiar života. Samotné prekreslenie týchto správ vykonáva metóda `translateMessagesForLifeline()` triedy `LifelineAlgorithm`, ktorá na základe hodnôt posunu, vzniknutého počas výmeny, prekreslí dané správy a rovnako prekreslí aj ich smer.

3. **Prekreslenie vertikálnej čiary života podľa veľkosti vrstvy**
   V prípade, že vznikne hodnota posunu vo vertikálnom smere, je potrebné prekresliť vertikálnu čiaru čiary života k danej vrstve. Prekresľovanie tejto vertikálnej čiary zabezpečuje metóda `redrawLifelineVerticalLine()` triedy `LifelineAlgorithm`.

3.3.4.4 Algoritmus vymazania komponentu čiary života

Pre vymazanie je potrebné označiť danú čiaru života, kde sa po kliknutím pravého tlačidla na myši zobrazí pop-up okno pre vymazanie.



Obrázok 68: Vymazanie čiary života zo scény

Pri tejto akcii sa musia vykonať nasledujúce kroky:

1. **Čiara života sa musí graficky odstrániť zo scény**
   Vymazanie čiary života zo scény vykonáva trieda `SequenceDrawManager`. V tejto triede sa nachádza metóda `deleteLifeline()`, ktorá vymaže všetky grafické informácie objektu a zároveň vymaže aj samotný grafický objekt.

2. **Čiara života sa musí odstrániť z metamodelu**

Odstránenie elementu z metamodelu vykonáva pre každý element zodpovedná *factory* metóda. V prípade čiary života je to trieda `LifelineFactory`, ktorá z danej interakcie vymaže reálne vytvorenú inštanciu čiary života, ktorá sa nachádza v metamodeli.

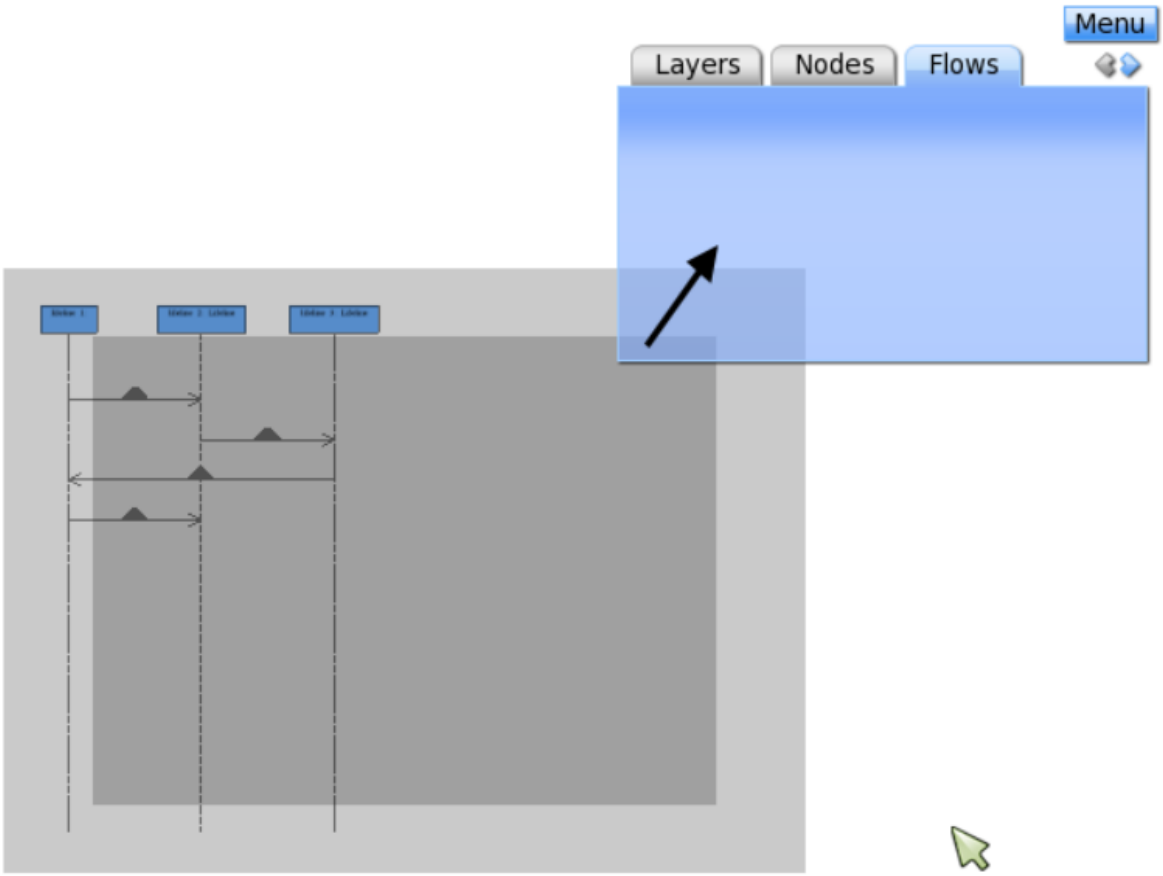
Dodatočné podmienky pri vymazávaní čiary života:

- Každá čiara života obsahuje v sebe zoznam všetkých `OccurenceSpecification`, pomocou ktorých vieme identifikovať všetky správy, ktoré vchádzajú alebo vychádzajú z danej čiary života. Počas procesu jej vymazania zo scény musíme vymazať aj všetky správy, ktoré sú s ňou prepojené.

- Čiara života obsahuje správy, ktoré sa nachádzajú v kombinovanom fragmente. Nakoľko fragmenty predstavujú novú vrstvu, do ktorej sa vykresľujú ďalšie elementy (napr. správy, operandy a pod.), nie je možné jednoducho zmazať čiaru života spolu s týmto fragmentom. Preto sme tu vytvorili obmedzenie, ktoré zabraňuje vykonaniu takejto akcie. Prvotným krokom je preto potrebné najskôr zmazať vytvorený kombinovaný fragment, ktorý je prepojený s vybranou čiarou života. Po úspešnom vymazaní fragmentu zo scény je možné následne vykonať akciu vymazania čiary života spolu s jej správami (za predpokladu, že už nie je prepojená so žiadnym iným kombinovaným fragmentom).

3.3.5 Správy

Pre pridanie správy v sekvenčnom diagrame bolo potrebné pridanie obrázku správy do menu, zabezpečiť označovanie čiar života a následne korektné ukladanie. Správa interaguje s dvoma čiarami života, z jednej vychádza a smeruje k druhej. V správe preto musí byť ukladaná informácia o tom, z ktorej čiary života správa odchádza a ku ktorej smeruje. Taktiež musí obsahovať informáciu o mieste, na ktorom sa má vykresliť. Nakoniec je potrebné pripraviť vykresľovací algoritmus na korektné vykreslenie správy.

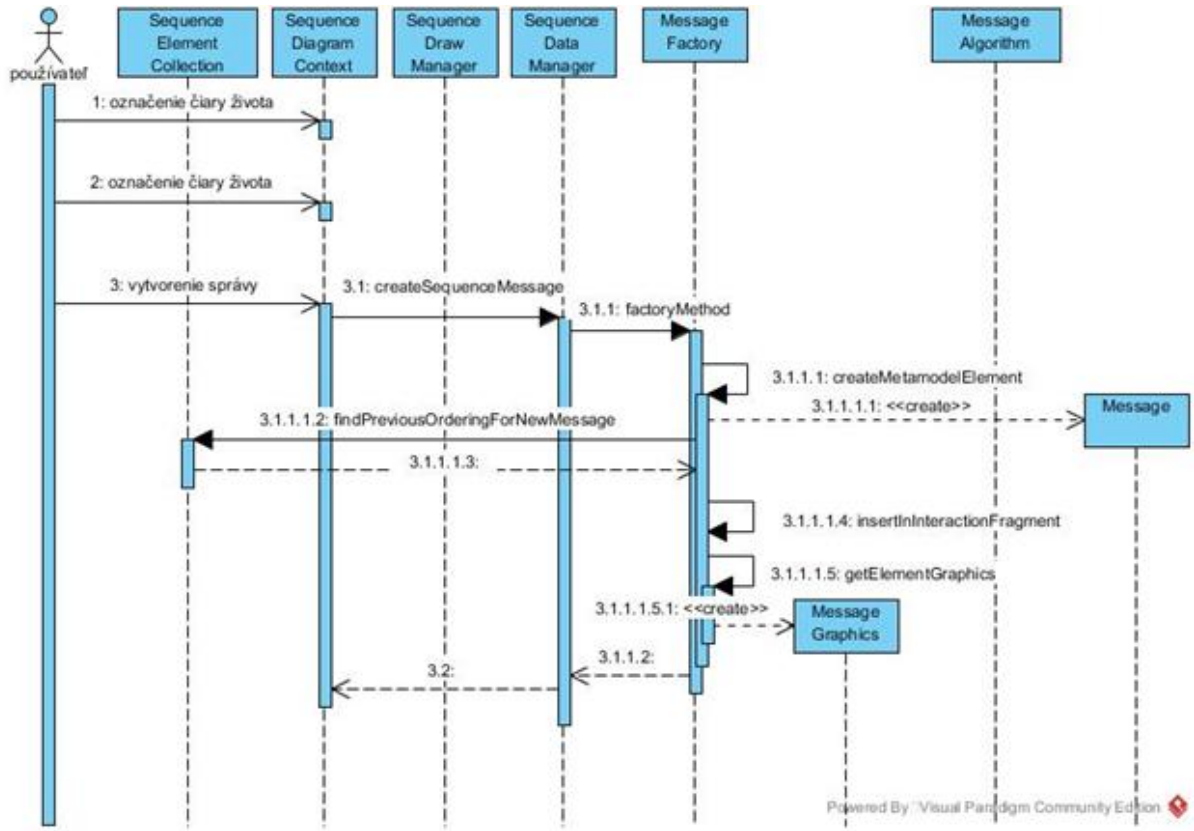


Obrázok 69: Vykreslenie `Message`

3.3.5.1 Algoritmus vloženia správy

Pre vloženie `Message` komponentu bolo potrebné modifikovať časti `3D controller-a`, `model`-u, a *View* komponentu. Algoritmus vykreslenie správ. Akcia vkladania správy je iniciovaná výberom zdrojovej a cieľovej čiary života, názvom správy a výberom miesta umiestnenia správy. Prvoradou časťou je zistenie správneho umiestnenia správy v rámci `metamodelu`. Celý proces sa deje v metóde `factoryMethod()`, ktorá inicializuje akciu vloženia správy do metamodelu volaním internej funkcie `createMetamodelElement()`. Táto metóda vytvára element správy so začiatočným a konečným elementom `MessageOccuranceSpecification` a naviazaným `GeneralOrdering`-om. Táto štruktúra sa vkladá do `metamodelu` pomocou volania metódy `insertInInteractionFragment()`. Táto metóda rekurzívne vyhľadá vhodný `interactionFragment`, do ktorého vloží spomínanú štruktúru. Vloženie sa môže prebiehať v troch prípadoch a to vloženie na začiatok fragmentu, vloženie medzi dve správy vo fragmente, alebo vloženie na koniec fragmentu. Táto akcia sa deje volaním metódy `findPreviousOrderingForNewMessage()`, ktorá nájde predošlí `generalOrdering` a naviaže naň `generalOrdering` vkladanej správy. Metóda je špecifická tým, že vráti NULL v prípade prvej správy. V prípade vkladania správy na prvé miesto v `InteractionFragment-e` je potrebné poznať aj príslušný `InteractionFragment`, kvôli zmene referencie `InteractionFragment-u` na prvý `GeneralOrdering`.

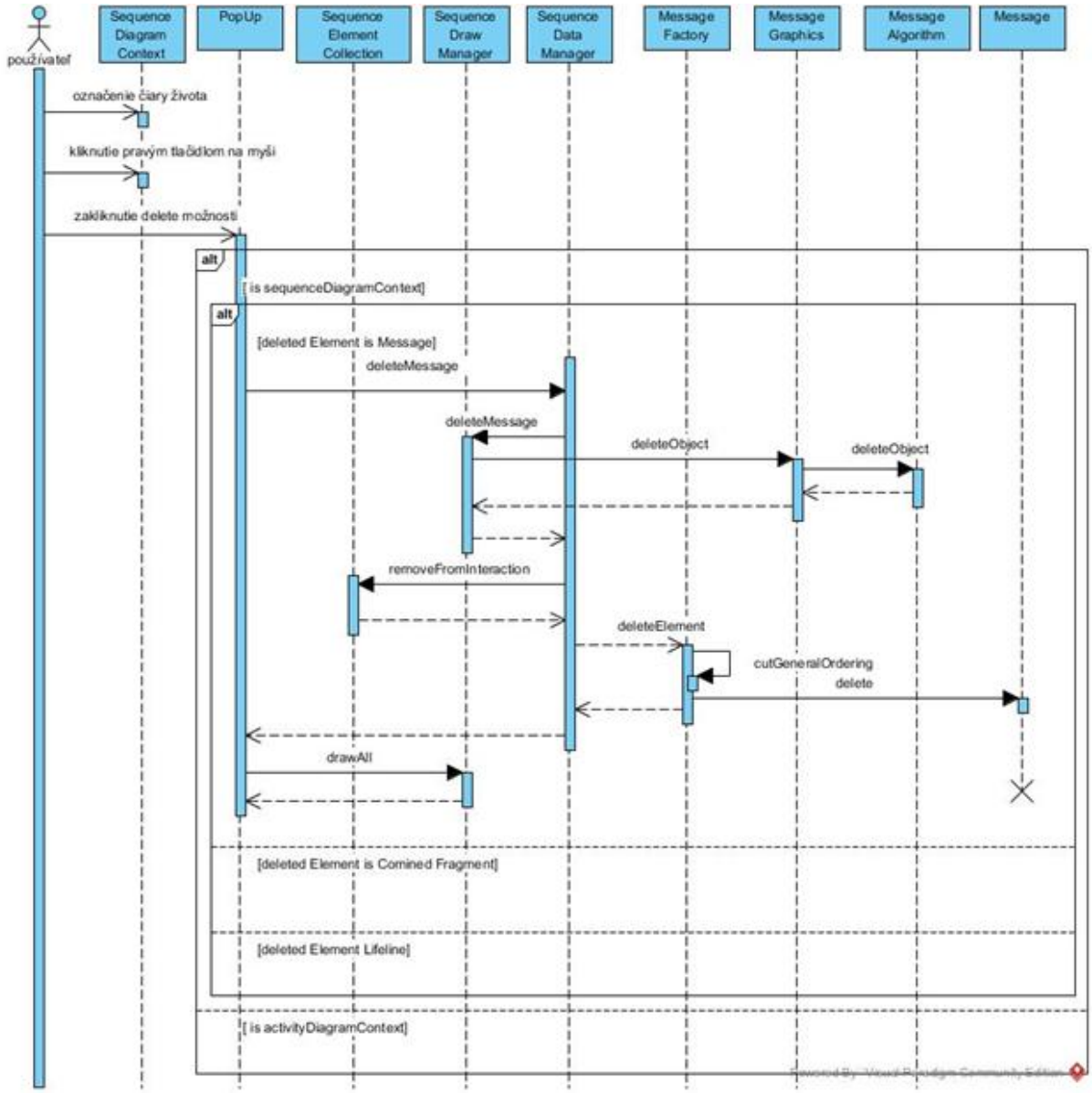
Nasledujúci obrázok demonštruje vkladanie správy pomocou sekvenčného diagramu. Po následom vložení do `metamodel` komponentu bude správa vykreslená metódou `drawAll()` v triede `SequenceDrawManager`.



Obrázok 70: Algoritmus vloženia správy

3.3.5.2 Vymazanie správy

Vymazanie elementu je inicializované v triede `PopUp`. V prípade správy sa odstraňuje objekt správy s dvoma objektami `MessageOccurancespecification` a predošlým `GeneralOrdering-om`. Táto štruktúra sa vytiahne z príslušného naviazaného `GeneralOrderingu`. V tomto prípade je potrebné zachovávať správnu štruktúru `metamodel-u` následným pospájaním `generalOrdering-ov`. V prípade mazania prvej správy v `InteractionFragment-e` je potrebné zmeniť prepojenie `InteractionFragmentu` s prvým `GeneralOrdering-om`. Pre túto akciu je potrebné poznať aj `InteractionFragment` z ktorého vyťahujeme `GeneralOrdering`. Celý proces sa vykonáva v metóde `cutGeneralOrdering()`. Taktiež `Interaction` element obsahuje referenciu na správu, ktorú je potrebné tiež zrušiť, a to zaobstaráva metóda `removeFromInteraction()`. Po odstránení všetkých referencií je sa objekty dealokujú z dynamickej pamäte. Obrázok nižšie demonštruje vykonanie vymazania pre správu.



Obrázok 71: Vymazanie správy

3.3.5.3 Posun správy hore dole

Pre vykonanie akcie posunu správy hore dole je dotvorená funkcionalita chytenia. Ta sa vykonáva rozšírením akcie `MouseManager-a` o podmienky kontrolujúce vyvolanie metódy `mouseDragged()`. Tento komponent si pamätá predošlí stav súradnice kurzora na obrazovke a v prídade ak bolo vykonané stlačenie pravého tlačidla na myši a zmenené súradnice, volá metódu `mouseDragged()`. Tento úsek kódu je reprezentovaný na obrázku 72.

```
bool MouseManager::mouseMoved( const OIS::MouseEvent &arg )
{
        if (mousePress)
        {
                If (xCoordinate != arg.state.X.abs &&
                        yCoordinate != arg.state.Y.abs){
                        dragging = true;
                        mouseDragged(arg);
                }
        }
```

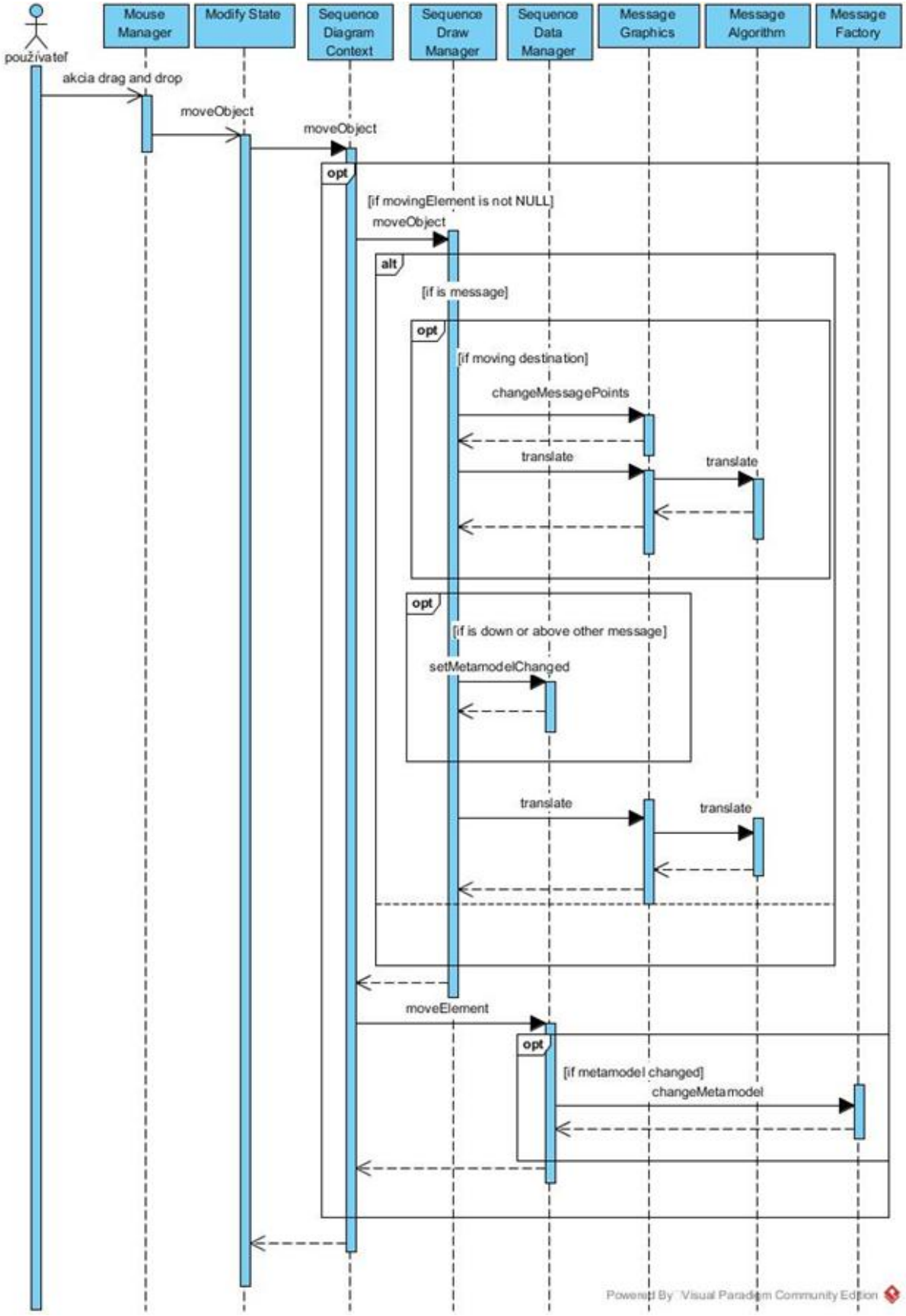
Obrázok 72: kód vyvolania metódy `mouseDragged()`

Najdôležitejším prvkom v prípade presunu správy je zachovanie korektnej `metamodel` štruktúry. Akcia sa vykonáva v metóde `moveObject()` triedy `SequenceDrawManager`. Tá mení grafické informácie elementu a porovnáva ich s grafickými informáciami predošlej a nasledovnej správy v rámci `GeneralOrdering-u`. Pokiaľ sa správa hýbe len v rámci intervalu vymedzeného týmito grafickými informáciami, menia sa len jej grafické informácie v `MessageElementGraphics` a `centerPoint`. V prípade prekročenia vymedzeného intervalu sa v `SequenceDataManager`-i zmení príznak inicializujúci potrebnú zmenu `metamodel-u` metódou `setMetamodelChanged()`. Po vykonaní tejto funkcionality sa vykonáva metóda `moveElement()` v triede `SequenceDataManager`. Tá v prípade nastaveného príznaku zavolá metódu `ChangeMetamodel`(). Tento proces je zobrazení vo forme diagramu na obrázku X. V metóde `ChangeMetamodel()` sa využívajú funkcie akcie vloženia a vymazania správy. Daná správa vystrihne metódou `cutGeneralOrdering()` a následne sa vloží do `metamodel-u` metódou `insertInInteractionFragment()`. Na obrázku 73 je zobrazený kód metódy `ChangeMetamodel()`.

```
void MessageFactory::moveMessage(SequenceMetamodel::Message* message,
        SequenceMetamodel::CombinedFragment* Layer, int x, int y)
{
    DEF_BEGIN
    SequenceMetamodel::MessageOccurenceSpecification * mos =
            dynamic_cast<SequenceMetamodel::MessageOccurenceSpecification*>
            (message->getFirstMessageEnd());
    SequenceMetamodel::GeneralOrdering * go = cutGeneralOrdering(message);
    SequenceMetamodel::InteractionFragment * fragment =
            SequenceElementCollection::findFragment(x, y, Layer);
    SequenceMetamodel::GeneralOrdering * pGO =
            SequenceElementCollection::findPreviousOrderingForNewMessage(fragment, y);

    insertInInteractionFragment(pGO, go, Layer, x, y);
    DEF_END
}
```

Obrázok 73: metóda moveMessage()



Obrázok 74: Akcia pre posun správy

3.3.5.4 Posun cieľa správy

Pre akciu posunu cieľa správy je potrebné zaobstarať rozdielne správanie pri posune správy. Preto je zaznamenávané aj zachytenie cieľa správy. Tento proces je rozoznávaný pomocou nového príznaku `selectedDestination` v triede `MessageGraphics`. Posun prebieha len zmenou grafických informácií správy a `centerPoint-u`. Metamodelová zmena sa v prípade zmeny cieľa vykonáva stále, a to len zmenením referencie cieľového `MessageOccuranceSpecification` na inú čiaru života a odstránením zo zoznamu zo zoznamu referencií `MessageOccuranceSpecification` patriacemu starej čiary života.

3.3.6 Kombinované fragmenty

3.3.6.1 Pridávanie fragmentov

Pre pridávanie nových fragmentov do sekvenčného diagramu bolo potrebné modifikovať menu obsahujúce rôzne typy fragmentov (napr. SEQ, ALT, LOOP). Na základe označenia fragmentu v menu vieme identifikovať, ktorý typ fragmentu chceme pridať. Pred samotným pridaním nového fragmentu je potrebné označenie čiar života, cez ktoré bude komponent fragmentu vykreslený. Pomocou týchto čiar života vieme identifikovať, od ktorého bodu ku ktorému sa daný fragment vykreslí v rámci x-vej osi. Posledným krokom je určenie pozície začiatku vykresľovania fragmentu pre y-vú os (kliknutím na vybranú pozíciu v scéne). Po kliknutí sa nám zobrazí okno, pomocou ktorého bližšie špecifikujeme vlastnosti fragmentu.

3.3.6.2 Okno na vytváranie fragmentov

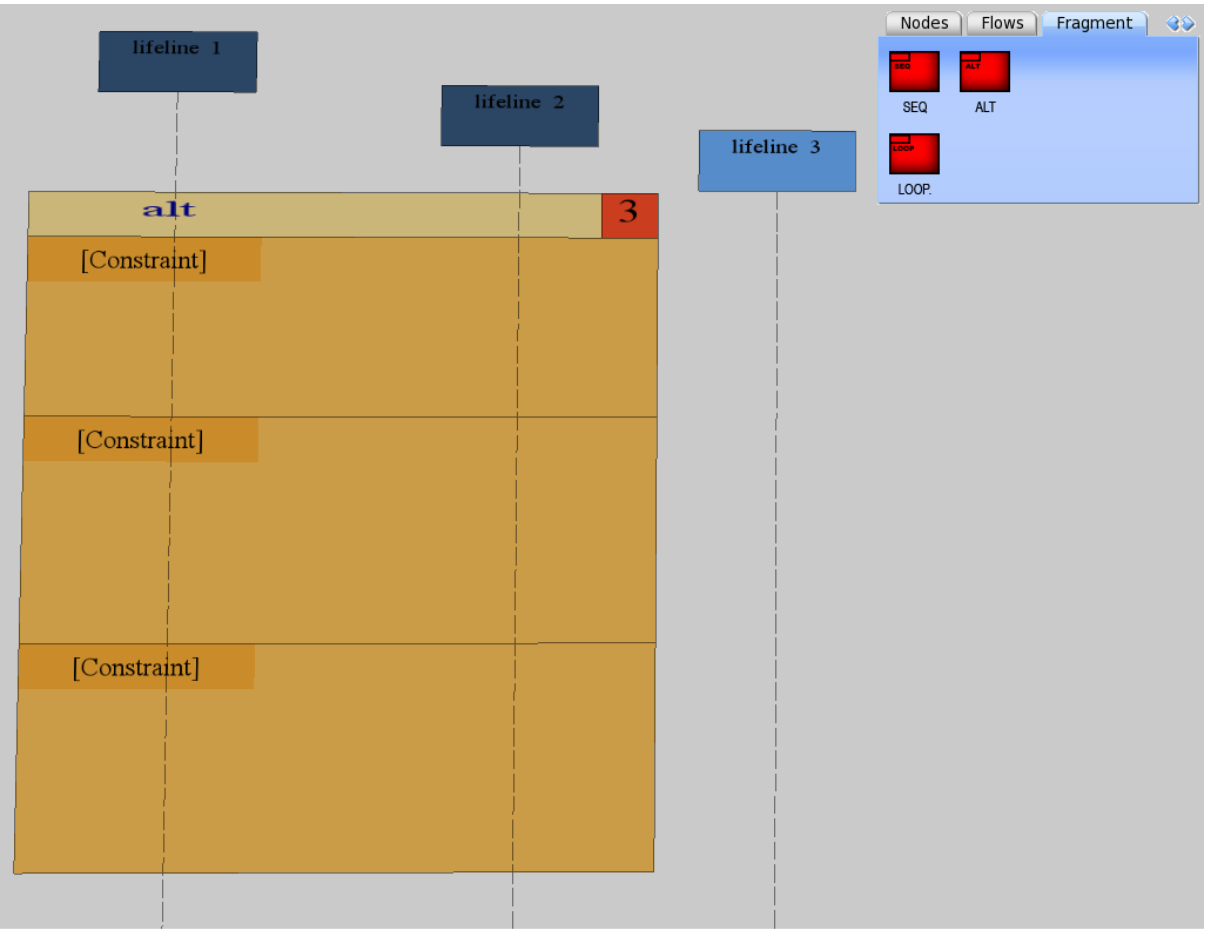
Pre vytvorenie fragmentov bolo potrebné pridať nový typ okna, ktoré obsahovalo podmienky zobrazené vo fragmente. Okno už neobsahuje typ ako v spájanom prototype, ale typ sa zvolí podľa tlačidla, ktoré používateľ označil v menu. Každý nový typ fragmentu musí mať teda samostatné tlačidlo. Pred kliknutím na tlačidlo fragmentu sa musia označiť čiary života, pre ktoré bude fragment určený.

3.3.6.3 Algoritmus vloženia a vykreslenia Fragment komponentu

Pri vkladaní fragmentov do existujúcej scény je potrebné, aby existovala aspoň jedna vrstva, na ktorej sú vytvorené najmenej dve čiary života. Až po splnení tohto predpokladu je možné pracovať s vkladaním fragmentu. Aby vytváranie a vykresľovanie fragmentov fungovalo správne v navrhnutej architektúre, bolo potrebné modifikovať 3D Controller, 3D Model a View komponenty. V rámci každého komponentu boli vytvorené triedy, ktoré sa starajú o vytvorenie samotnej inštancie fragmentu, nastavenie všetkých vstupných dát pre fragmenty a samotné vykreslenie tohto elementu. Diagram tokov používateľa pre vytvorenie fragment elementu:

1. Definovať v metamodeli a modeli `SequenceFragment` triedu
2. Vytvorenie triedy `InteractionFragmentFactory`, ktorá vytvára `SequenceFragment` komponent na základe jeho definície v metamodeli. Rovnako táto trieda vytvára a nastavuje grafické parametre.

3. Vytvorenie triedy `InteractionFragmentGraphics`, ktorá obsahuje všetky informácie potrebné pre vykreslenie fragmentu.
4. Vytvorenie triedy `InteractionFragmentAlgorithm`. Táto trieda obsahuje metódu `draw`, v ktorej je implementovaná funkcionalita pre výpočet začiatočného a koncového bodu fragmentu a získava sa v nej element s nastavenými potrebnými dátami pre vykreslenie. Samotný algoritmus a pravidlá pre vykreslenie fragmentu v scéne zabezpečuje metóda `drawFragment`.
5. Modifikácia triedy `SequenceDataManager`. V tejto časti bolo potrebné pridať novú metódu, ktorej účelom je vytvoriť nový element pre fragment a rovnako aj nastaviť pre tento element všetky dáta.
6. Modifikácia triedy `SequenceDiagramContext`, ktorá spúšťa editovacie okno pre fragmenty. V tejto triede prebieha inicializácia pre vytváranie komponentov sekvenčného diagramu.



Obrázok 75: Vykreslenie fragmentu typu ALT

3.3.6.4 Algoritmus pre zmenu veľkosti fragmentu

Pre zmenu veľkosti fragmentu bolo potrebné pridať niektoré menšie funkcie. Fragmenty sa doteraz nedali označiť kliknutím ľavého tlačidla myši ako je možné u čiar života a správ. Po pridaní tejto funkcie sa fragmenty už dajú označiť kliknutím do stredu ich hlavičky. Po označení

fragmentu sa zvýraznia rohové body fragmentu, ktoré môže používateľ nasledovne chytiť a ťahať, čím sa mení veľkosť fragmentu. Po pustení tlačidla myši sa upraví metamodel podľa nových elementov, ktoré fragment teraz zahŕňa. Element sa uznáva ako zahrnutý vo fragmente, ak je jeho stred vnútri plochy fragmentu. Pre iné fragmenty sa ako stred považuje miesto, kde sa dajú označiť, teda stred ich hlavičky. Po pustení tlačidla sa veľkosť fragmentu automaticky zmení tak, aby fragment pokrýval všetky zahrnuté elementy. V prípade, že po zmene veľkosti fragment neobsahuje žiadny element, je možné jeho veľkosť meniť voľne a nie je viazaný na veľkosti vnútorných elementov. Algoritmus funguje pre zväčšovanie aj zmenšovanie fragmentu a jeho jediné pravidlo je, že veľkosť fragmentu nesmie presahovať veľkosť jeho rodiča, teda iného fragmentu.

3.3.6.5 Odstraňovanie fragmentov

Odstraňovanie fragmentov je funkcia, ktorá bola veľmi jednoducho implementovaná do nášho prototypu. S využitím novo pridaného označovania fragmentov stačí, keď používateľ po označení daného fragmentu klikne na pravé tlačidlo myši, čo vyvolá vyskakovacie okno. V tomto okne sa nachádza možnosť DELETE. Z hľadiska metamodelu odstránenie fragmentu vyzerá nasledovne:

1. Všetky operandy, ktoré vymazaný fragment obsahoval, sa pridajú medzi operandy jeho rodiča rovno na pozíciu, kde bola daná cesta k tomuto vymazanému fragmentu
2. Po tomto premiestnení operandov sa zavolá funkcia na spájanie operandov. V prípade, že sa nachádzajú vedľa seba 2 alebo ešte väčšie množstvo operandov, ktorých zlúčenie nespôsobí žiadnu stratu údajov, tieto operandy sa spoja. Táto funkcia sa používa takmer pri všetkých editačných funkciách fragmentu a slúži na to, aby po viacerých editáciách nevznikalo zbytočne veľké množstvo operandov.

Funkcia DELETE funguje nielen pre fragmenty, ale aj všetky ostatné vytvoriteľné elementy. Je tiež možné vymazať viacero elementov jedným zavolaním tejto funkcie, stačí len, aby boli všetky tieto elementy označené.

3.3.6.6 Pridávanie/Odstraňovanie operandov

Pridávanie operandov je ďalšia z editačných funkcií pridaných do nášho prototypu. Po zmene okna pre vytváranie fragmentu tak, aby obsahovalo len názov, typ a prvú podmienku, bolo potrebné túto editačnú funkciu vytvoriť. Používateľ má možnosť pridávať ďalšie operandy alebo prepisovať existujúce tak, že si označí ľubovoľnú správu alebo fragment a po zvolení funkcie pridania operandu v menu klikne nad alebo pod tento označený element. Táto informácia určuje, kde presne sa má operand vykresliť. Pri implementácii tejto funkcie došlo k veľkým problémom hlavne z hľadiska limitácií používanej grafickej knižnice, takže toto bol jediný dostatočne jednoduchý spôsob pre používateľa ako túto funkciu zrealizovať.

Pri odstraňovaní operandov sme tiež narazili na podobné limitácie, konkrétne nemožnosť označenia konkrétneho operandu, ktorý by sme chceli zmazať. Odstraňovanie funguje preto iným spôsobom. Po zvolení funkcie odstránenia operandu sa používateľovi zobrazí okno, do ktorého musí používateľ vložiť poradie operandu, ktorý chce zmazať v danom označenom fragmente.

3.3.6.7 Pohyb fragmentu

Pohyb fragmentu je experimentálna funkcia, ktorá bola pridaná do nášho prototypu, ale nebola úspešne dokončená. Pohybovanie fragmentov je umožnené tak, že používateľ označí fragment a následne ho ťahá za jeho hlavičku. Pohybovanie fragmentov horizontálne nebolo úplne premyslené, pretože, pri jeho pohybe dochádza k veľkým zmenám v metamodeli. Jediné povolené pohybovanie je vertikálne a funguje tak, že všetky vnútorné elementy fragmentu sa pohybujú spolu s ním. Pri vertikálnom posúvaní môže tiež dôjsť k pokazeniu metamodelu, pretože nebolo úplne doladené a existujú v ňom drobné chyby. Pre budúcu prácu ďalšieho tímu, ktorý tento prototyp bola táto funkcia v zdrojovom kóde prototypu ponechaná aj vzhľadom na jej nedostatky. Pre používateľa je však v momentálnej verzii nedostupná.

3.4 Testovanie

Priebežne s implementáciou prebiehalo aj testovanie prototypu. Pre otestovanie implementovaných častí sme si vytvorili niekoľko nižšie uvedených testovacích scenárov. Niektoré scenáre sú určené aj pre funkcionalitu, ktorá mala byť implementovaná už pred našimi zmenami a v prípade, že test nefunguje pre elementy v scéne sekvenčného diagramu, tak nefunguje ani pre ostatné (diagram aktivít elementy).

3.4.1 Základná funkcionalita

Táto časť kapitoly obsahuje testovacie scenáre a ich výsledky pre základnú funkcionalitu. Pod touto sa rozumie fungovanie základných okien aplikácie, pridávanie a práca s vrstvami a tiež pridávanie základných elementov sekvenčného diagramu.

3.4.1.1 Testovacie scenáre

1. **Zobrazenie prázdnej scény pre sekvenčný diagram**
    a. Spustenie aplikácie.
    b. Výber sekvenčného diagramu v menu.
    c. Finálny stav - Vytvorí sá prázdna scéna sekvenčného diagramu.
2. **Zobrazenie čiary života v menu**
    a. Výber sekvenčného diagramu v menu.
    b. Otvorenie záložky “Nodes” v pravom hornom menu, ktoré sa zobrazí po načítaní scény.
    c. Finálny stav - Záložka obsahuje element s názvom “Lifeline”.
3. **Načítanie okna pre vytvorenie čiary života**
    a. Výber sekvenčného diagramu v menu.
    b. Pridanie aspoň jednej vrstvy v záložke “Layers”.
    c. Výber čiary života zo záložky “Nodes”.
    d. Kliknutie do pridanej vrstvy.
    e. Finálny stav - zobrazí sa okno pre vytvorenie čiary života.
4. **Nenačítanie okna pre vytvorenie čiary života v prípade, že neexistuje ešte žiadna vrstva**

a. Výber sekvenčného diagramu v menu.
b. Výber čiary života zo záložky “Nodes” v Menu.
c. Kliknutie na akékoľvek miesto na scéne.
d. Finálny stav - Okno pre vytvorenie čiary života sa nezobrazí.

5. Pridanie čiary života do scény

a. Výber sekvenčného diagramu v menu.
b. Pridanie aspoň jednej vrstvy v záložke “Layers”.
c. Výber čiary života zo záložky “Nodes”.
d. Kliknutie do pridanej vrstvy.
e. Vyplnenie poľa “name” v okne pre pridanie čiary života
f. Kliknutie na “OK” .
g. Finálny stav - Na vrstve sa zobrazí nová čiara života so zvoleným menom.

6. Nepridanie čiary života do scény mimo vrstvy

a. Výber sekvenčného diagramu v menu.
b. Pridanie aspoň jednej vrstvy v záložke “Layers”.
c. Výber čiary života zo záložky “Nodes”.
d. Kliknutie mimo akejkoľvek pridanej vrstvy.
e. Vyplnenie poľa “name” v okne pre pridanie čiary života.
f. Kliknutie na “OK”.
g. Finálny stav - čiara života sa nevytvorí. Možná alternatíva - Nezobrazí sa už ani okno pre vytvorenie elementu.

7. Pridanie čiary života do scény do práve zvolenej vrstvy

a. Výber sekvenčného diagramu v menu.
b. Pridanie aspoň jednej vrstvy v záložke “Layers”.
c. Výber jednej z vrstiev cez menu alebo tlačidlami “R” a “F” (bez výberu sa predvolene použije najnovšie vytvorená vrstva).
d. Výber čiary života zo záložky “Nodes”.
e. Kliknutie do pridanej vrstvy.
f. Vyplnenie poľa “name” v okne pre pridanie čiary života.
g. Kliknutie na “OK” .
h. Finálny stav - Na vybranej vrstve sa zobrazí nová čiara života so zvoleným menom.

8. Zrušenie okna pre vytvorenie čiary života

a. Výber sekvenčného diagramu v menu.
b. Pridanie aspoň jednej vrstvy v záložke “Layers”.
c. Výber čiary života zo záložky “Nodes”.
d. Kliknutie do pridanej vrstvy.
e. Vyplnenie poľa “name” v okne pre pridanie čiary života.
f. Kliknutie na “Cancel”.
g. Finálny stav - Okno sa zruší a čiara života sa nevytvorí.

9. Vymazanie čiary života

a. Výber sekvenčného diagramu v menu.
b. Pridanie aspoň jednej vrstvy v záložke “Layers”.
c. Výber čiary života zo záložky “Nodes”.

d. Kliknutie do pridanej vrstvy.
e. Vyplnenie poľa “name” v okne pre pridanie čiary zivota.
f. Kliknutie na “OK”.
g. Kliknutie pravým tlačidlom myši na vytvorený element.
h. Kliknutie na možnosť “Delete” v novozobrazenom menu.
i. Finálny stav - element je vymazaný.

10. Pridanie správy

a. Výber sekvenčného diagramu v menu.
b. Pridanie aspoň jednej vrstvy v záložke “Layers”.
c. Vytvorenie 2 čiar života
d. Označenie 2 čiar života
e. Výber správy zo záložky “Flows”.
f. Vyplnenie poľa “name” v okne pre pridanie správy
g. Kliknutie na “OK” .
h. Finálny stav - Na vrstve sa zobrazí nová správa so zvoleným menom.

11. Pridanie správy medzi rôznymi vrstvami

a. Výber sekvenčného diagramu v menu.
b. Pridanie aspoň 2 vrstiev v záložke “Layers”.
c. Vytvorenie 2 čiar života (každý v inej vrstve).
d. Označenie 2 čiar života.
e. Výber správy zo záložky “Flows”.
f. Vyplnenie poľa “name” v okne pre pridanie správy
g. Kliknutie na “OK” .
h. Finálny stav - Na vrstve sa zobrazí nová správa so zvoleným menom.

12. Pridanie kombinovaného fragmentu

a. Výber sekvenčného diagramu v menu.
b. Pridanie aspoň jednej vrstvy v záložke “Layers”.
c. Pridanie aspoň jednej správy.
d. Výber kombinovaného fragmentu zo záložky “Fragments” (na type fragmentu nezáleží).
e. Kliknutie do pridanej vrstvy.
f. Vyplnenie polí “Constraint count” a jednotlivých podmienok.
g. Kliknutie na “OK” .
h. Finálny stav - Na vrstve sa zobrazí nový kombinovaný fragment so zvoleným menom.

3.4.1.2 Výsledky testovacích scenárov

#	Názov testovacieho scenára	Stav testu	Poznámka
1.	Zobrazenie prázdnej scény pre sekvenčný diagram	Úspech	Sekvenčná scéna sa úspešne zobrazí

2.	Zobrazenie čiary života v menu scény	Úspech	Element je zobrazený v menu v záložke “Nodes”
3.	Načítanie okna pre vytvorenie čiary života	Úspech	Okno sa zobrazí a je editovateľné
4.	Nenačítanie okna pre vytvorenie čiary života v prípade, že neexistuje ešte žiadna vrstva	Neúspech	Pri kliknutí do prázdnej scény po vybratí elementu prestane program pracovať (program očakáva aspoň jednu vrstvu). Tento testovací scenár nefunguje pre žiadny element
5.	Pridanie čiary života do scény	Úspech	Element sa úspešne vykreslí
6.	Nepridanie čiary života do scény mimo vrstvy	Neúspech	Element sa zobrazí mimo vrstvy. Tento testovací scenár nefunguje pre žiadny element
7.	Pridanie čiary života do scény do práve zvolenej vrstvy	Úspech	Element sa vloží do práve zvolenej vrstvy
8.	Zrušenie okna pre vytvorenie čiary života	Úspech	Okne sa zatvorí.
9.	Vymazanie čiary života	Úspech	Čiara života je zmazaná.
10.	Pridanie správy	Úspech	Element sa úspešne vykreslí
11.	Pridanie správy medzi rôznymi vrstvami	Úspech	Element sa úspešne vykreslí
12.	Pridanie kombinovaného fragmentu	Úspech	Element sa úspešne vykreslí

Tabuľka 2: Vyhodnotenie testovacích scenárov

3.4.2 Čiary života

V tejto časti sa nachádzajú detailnejšie testovacie scenáre a ich výsledky pre funkcionalitu súvisiacu s čiarami života. Jedná sa o pridávanie nových čiar života do diagramu, ich posúvanie po scéne a nakoniec ich mazanie zo scény.

3.4.2.1 Testovacie scenáre

1. **Pridanie čiary života do prázdnej scény**
    a. Predpoklad: sekvenčný diagram s aspoň jednou vrstvou.
    b. Kliknutie na záložku “Nodes”.
    c. Výber čiary života.
    d. Vyplnenie názvu čiary života.
    e. Kliknutie na miesto scéne, kam sa má čiara života pridať.
2. **Pridanie čiary života do neprázdnej scény mimo existujúcich elementov**
    a. Predpoklad: sekvenčný diagram so správami a/alebo kombinovanými fragmentami.
    b. Kliknutie na záložku “Nodes”.
    c. Výber čiary života.
    d. Vyplnenie názvu čiary života
    e. Kliknutie na miesto v scéne mimo existujúcich elementov.
3. **Pridanie čiary života medzi dve existujúce čiary života bez elementov medzi nimi**
    a. Predpoklad: sekvenčný diagram s dvomi čiarami života bez elementov medzi nimi.
    b. Kliknutie na záložku “Nodes”.
    c. Výber čiary života.
    d. Vyplnenie názvu čiary života.
    e. Kliknutie na miesto v scéne medzi dve čiary života.
4. **Pridanie čiary života medzi dve existujúce čiary života so správami medzi nimi**
    a. Predpoklad: sekvenčný diagram s dvomi čiarami života so správami medzi nimi.
    b. Kliknutie na záložku “Nodes”.
    c. Výber čiary života.
    d. Vyplnenie názvu čiary života.
    e. Kliknutie na miesto v scéne medzi dve čiary života.
5. **Pridanie čiary života medzi dve existujúce čiary života so správami a kombinovanými fragmentami medzi nimi**
    a. Predpoklad: sekvenčný diagram s dvomi čiarami života so správami a kombinovanými fragmentami medzi nimi.
    b. Kliknutie na záložku “Nodes”.
    c. Výber čiary života.
    d. Vyplnenie názvu čiary života.
    e. Kliknutie na miesto v scéne medzi dve čiary života.
6. **Zmazanie čiary života bez elementov**
    a. Predpoklad: sekvenčný diagram s čiarou života bez elementov.
    b. Označenie čiary života.

c. Kliknutie pravým tlačidlom myše do scény.
d. Výber položky “Delete” z kontextového menu.

7. **Zmazanie čiary života s elementami**
   a. Predpoklad: sekvenčný diagram s čiarou života so správami.
   b. Označenie čiary života.
   c. Kliknutie pravým tlačidlom myše do scény.
   d. Výber položky “Delete” z kontextového menu.
8. **Posun čiary života bez elementov po scéne**
   a. Predpoklad: sekvenčný diagram s čiarou života bez elementov.
   b. Označenie čiary života.
   c. Ťahanie myšou so stlačeným ľavým tlačidlom v ľubovoľnom smere.
   d. Pustenie tlačidla myše.
9. **Posun čiary života s elementami po scéne**
   a. Predpoklad: sekvenčný diagram s čiarou života s elementami.
   b. Označenie čiary života.
   c. Ťahanie myšou so stlačeným ľavým tlačidlom v ľubovoľnom smere.
   d. Pustenie tlačidla myše.
10. **Výmena dvoch čiar života**
    a. Predpoklad: sekvenčný diagram s aspoň dvomi čiarami života so správami a/alebo kombinovanými fragmentami.
    b. Označenie čiary života.
    c. Ťahanie myšou so stlačeným ľavým tlačidlom v takom smere, aby došlo k výmene s inou čiarou života.
    d. Pustenie tlačidla myše.

3.4.2.2 Výsledky testovacích scenárov

#	Názov testovacieho scenára	Stav testu	Poznámka
1.	Pridanie čiary života do prázdnej scény	Úspech	Čiara života je úspešne pridaná na kliknuté miesto.
2.	Pridanie čiary života do neprázdnej scény mimo existujúcich elementov	Úspech	Čiara života je úspešne pridaná na kliknuté miesto.
3.	Pridanie čiary života medzi dve existujúce čiary života bez elementov medzi nimi	Úspech	Čiara života sa pridá, metamodel je OK.
4.	Pridanie čiary života	Úspech	Čiara života je úspešne pridaná.

	medzi dve existujúce čiary života so správami medzi nimi		
5.	Pridanie čiary života medzi dve existujúce čiary života so správami a kombinovanými fragmentami medzi nimi	Úspech	Čiara života je úspešne pridaná.
6.	Zmazanie čiary života bez elementov	Úspech	Čiara života je úspešne zmazaná.
7.	Zmazanie čiary života s elementami	Úspech	Čiara života je úspešne zmazaná spolu so správami.
8.	Posun čiary života bez elementov po scéne	Úspech	Posunutie úspešné.
9.	Posun čiary života s elementami po scéne	Úspech	Posunutie úspešné. Spolu s čiarou života sa hýbu aj jednotlivé elementy.
10.	Výmena dvoch čiar života	Úspech	Čiary života sa úspešne vymenia.

Tabuľka 3: Vyhodnotenie testovacích scenárov pre čiary života

3.4.3 Správy

V tejto časti sa nachádzajú detailnejšie testovacie scenáre a ich výsledky pre funkcionalitu súvisiacu so správami. Jedná sa o pridávanie nových správ do diagramu, ich posúvanie po scéne a nakoniec ich mazanie zo scény.

3.4.3.1 Testovacie scenáre

1. **Pridanie správy medzi dve čiary života v smere zľava doprava**
    a. Predpoklad: sekvenčný diagram s aspoň dvomi čiarami života.
    b. Označenie dvoch čiar života v poradí zľava doprava.
    c. Kliknutie na záložku “Flows”.
    d. Výber správy.
    e. Napísanie obsahu správy.
    f. Vloženie správy do scény.

2. **Pridanie správy medzi dve čiary života v smere sprava doľava**
    a. Predpoklad: sekvenčný diagram s aspoň dvomi čiarami života.
    b. Označenie dvoch čiar života v poradí sprava doľava.
    c. Kliknutie na záložku “Flows”.
    d. Výber správy.
    e. Napísanie obsahu správy.
    f. Vloženie správy do scény.
3. **Pridanie správy do kombinovaného fragmentu**
    a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom.
    b. Označenie dvoch čiar života.
    c. Kliknutie na záložku “Flows”.
    d. Výber správy.
    e. Napísanie obsahu správy.
    f. Vloženie správy do kombinovaného fragmentu.
4. **Pridanie správy medzi viacero čiar života**
    a. Predpoklad: sekvenčný diagram s viacero čiarami života.
    b. Označenie dvoch čiar života tak, aby medzi nimi bola aspoň jedna iná čiara života.
    c. Kliknutie na záložku “Flows”.
    d. Výber správy.
    e. Napísanie obsahu správy.
    f. Vloženie správy do scény.
5. **Pridanie správy na úroveň kombinovaného fragmentu vedľa neho**
    a. Predpoklad: sekvenčný diagram s viacerými čiarami života a s aspoň jedným kombinovaným fragmentom, ktorý pokrýva správy medzi ľubovoľnými dvomi správami života.
    b. Označenie dvoch čiar života tak, aby sa správa pridala vedľa kombinovaného fragmentu (vľavo alebo vpravo).
    c. Kliknutie na záložku “Flows”.
    d. Výber správy.
    e. Napísanie obsahu správy.
    f. Vloženie správy do scény.
6. **Presun správy nad inú správu alebo správy**
    a. Predpoklad: sekvenčný diagram s viacerými správami.
    b. Označenie správy na presun.
    c. Ťahanie správy držaním ľavého tlačidla myše a jej ťahaním nad inú správu alebo správy.
    d. Pustenie tlačidla myše.
7. **Presun správy pod inú správu alebo správy**
    a. Predpoklad: sekvenčný diagram s viacerými správami.
    b. Označenie správy na presun.
    c. Ťahanie správy držaním ľavého tlačidla myše a jej ťahaním pod inú správu alebo správy.
    d. Pustenie tlačidla myše.
8. **Presun správy do kombinovaného fragmentu**

a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom.
b. Označenie správy na presun.
c. Ťahanie správy držaním ľavého tlačidla myše a jej ťahaním do kombinovaného fragmentu.
d. Pustenie tlačidla myše.

9. Presun správy z kombinovaného fragmentu

a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom, ktorý obsahuje aspoň jednu správu.
b. Označenie správy v kombinovanom fragmente.
c. Ťahanie správy držaním ľavého tlačidla myše a jej ťahaním z kombinovaného fragmentu.
d. Pustenie tlačidla myše.

10. Zmazanie správy mimo kombinovaného fragmentu

a. Predpoklad: sekvenčný diagram s aspoň jednou správou.
b. Označenie správy na zmazanie.
c. Kliknutie pravým tlačidlom myše do scény.
d. Výber položky “Delete” z kontextového menu.

11. Zmazanie správy z kombinovaného fragmentu

a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom, ktorý obsahuje aspoň jednu správu.
b. Označenie správy v kombinovanom fragmente.
c. Kliknutie pravým tlačidlom myše do scény.
d. Výber položky “Delete” z kontextového menu.

12. Rozšírenie správy

a. Predpoklad: sekvenčný diagram s aspoň jednou správou medzi dvomi čiarami života a s aspoň tromi čiarami života.
b. Označenie správy pre zmenu cieľa.
c. Ťahanie správy držaním ľavého tlačidla myše a jej ťahaním tak, aby došlo k jej rozšíreniu a prepojeniu na inú čiaru života.
d. Pustenie tlačidla myše.

13. Skrátenie správy

a. Predpoklad: sekvenčný diagram s aspoň jednou správou medzi dvomi čiarami života tak, aby prechádzala cez tretiu čiaru života.
b. Označenie správy pre zmenu cieľa.
c. Ťahanie správy držaním ľavého tlačidla myše a jej ťahaním tak, aby došlo k skráteniu a prepojeniu na inú čiaru života.
d. Pustenie tlačidla myše.

14. Presun správy nad kombinovaným fragmentom pod kombinovaný fragment

a. Predpoklad: sekvenčný diagram s viacerými správami a s aspoň jedným kombinovaným fragmentom.
b. Označenie správy na presun nad kombinovaným fragmentom.
c. Ťahanie správy držaním ľavého tlačidla myše a jej ťahaním pod kombinovaný fragment.
d. Pustenie tlačidla myše.

15. Presun správy pod kombinovaným fragmentom nad kombinovaný fragment

a. Predpoklad: sekvenčný diagram s viacerými správami a s aspoň jedným kombinovaným fragmentom.
b. Označenie správy na presun pod kombinovaným fragmentom.
c. Ťahanie správy držaním ľavého tlačidla myše a jej ťahaním nad kombinovaný fragment.
d. Pustenie tlačidla myše.

3.4.3.2 Výsledky testovacích scenárov

#	Názov testovacieho scenára	Stav testu	Poznámka
1.	Pridanie správy medzi dve čiary života v smere zľava doprava	Úspech	Správa je úspešne pridaná so správnou orientáciou.
2.	Pridanie správy medzi dve čiary života v smere sprava doľava	Úspech	Správa je úspešne pridaná so správnou orientáciou.
3.	Pridanie správy do kombinovaného fragmentu	Úspech	Správa je úspešne pridaná do kombinovaného fragmentu, metamodel je príslušne upravený.
4.	Pridanie správy medzi viacero čiar života	Úspech	Správa je úspešne pridaná medzi viacero čiar života, má jeden zdroj a jeden cieľ a prechádza cez viacero čiar.
5.	Pridanie správy na úroveň kombinovaného fragmentu vedľa neho	Úspech	Kombinovaný fragment sa adekvátne rozšíri tak, aby pokryl novú správu.
6.	Presun správy nad inú správu alebo správy	Úspech	Správa sa presunie na správne miesto, metamodel je OK.
7.	Presun správy pod inú správu alebo správy	Úspech	Správa sa presunie na správne miesto, metamodel je OK.
8.	Presun správy do kombinovaného fragmentu	Úspech	Kombinovaný fragment obsiahne správu.

9.	Presun správy z kombinovaného fragmentu	Úspech	Správa nie je obsiahnutá kombinovaným fragmentom.
10.	Zmazanie správy mimo kombinovaného fragmentu	Úspech	Správa je úspešne zmazaná, metamodel je OK.
11.	Zmazanie správy z kombinovaného fragmentu	Úspech	Správa úspešne zmazaná. Metamodel je OK.
12.	Rozšírenie správy	Úspech	Cieľ správy sa zmení na inú čiaru života. Správa sa korektne rozšíri.
13.	Skrátenie správy	Úspech	Cieľ správy sa zmení na inú čiaru života. Správa sa korektne skráti.
14.	Presun správy nad kombinovaným fragmentom pod kombinovaný fragment	Úspech	Správa je úspešne presunutá. Metamodelovo je pridaná do správneho operandu.
15.	Presun správy pod kombinovaným fragmentom nad kombinovaný fragment	Úspech	Správa je úspešne presunutá. Metamodelovo je pridaná do správneho operandu.

Tabuľka 4: Vyhodnotenie testovacích scenárov pre kombinované fragmenty

3.4.4 Kombinované fragmenty

3.4.4.1 Testovacie scenáre

1. **Pridanie jednoduchého kombinovaného fragmentu so všetkými správami**
    a. Predpoklad: sekvenčný diagram s aspoň dvomi čiarami života a viacerými správami.
    b. Označenie všetkých správ.
    c. Kliknutie na záložku “Fragments”.

d. Výber kombinovaného fragmentu.
e. Napísanie typu kombinovaného fragmentu do okna.
f. Vloženie fragmentu do scény.

2. **Pridanie jednoduchého kombinovaného fragmentu s vybranými správami**
   a. Predpoklad: sekvenčný diagram s aspoň dvomi čiarami života a viacerými správami.
   b. Označenie vybraných správ.
   c. Kliknutie na záložku "Fragments".
   d. Výber kombinovaného fragmentu.
   e. Napísanie typu kombinovaného fragmentu do okna.
   f. Vloženie fragmentu do scény.
3. **Pridanie kombinovaného fragmentu s pokrytím správ v dvoch rôznych interakčných operandoch**
   a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom a jednou (alebo viacerými) správami nad alebo pod fragmentom.
   b. Označenie dvoch správ, jednej mimo fragmentu a druhej vo fragmente.
   c. Kliknutie na záložku "Fragments".
   d. Výber kombinovaného fragmentu.
   e. Napísanie typu kombinovaného fragmentu do okna.
   f. Vloženie fragmentu do scény.
4. **Pridanie vnoreného kombinovaného fragmentu**
   a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom.
   b. Označenie správ v kombinovanom fragmente.
   c. Kliknutie na záložku "Fragments".
   d. Výber kombinovaného fragmentu.
   e. Napísanie typu kombinovaného fragmentu do okna.
   f. Vloženie fragmentu do scény.
5. **Zmazanie nevnoreného kombinovaného fragmentu**
   a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom.
   b. Označenie kombinovaného fragmentu kliknutím na jeho stredný horný bod.
   c. Kliknutie pravým tlačidlom myše do scény.
   d. Výber položky "Delete" z kontextového menu.
6. **Zmazanie vnoreného kombinovaného fragmentu**
   a. Predpoklad: sekvenčný diagram s aspoň jedným vnoreným kombinovaným fragmentom.
   b. Označenie vnoreného kombinovaného fragmentu kliknutím na jeho stredný horný bod.
   c. Kliknutie pravým tlačidlom myše do scény.
   d. Výber položky "Delete" z kontextového menu.
7. **Zmazanie kombinovaného fragmentu s viacerými operandami**
   a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom, ktorý obsahuje viacero operandov.
   b. Označenie kombinovaného fragmentu kliknutím na jeho stredný horný bod.
   c. Kliknutie pravým tlačidlom myše do scény.

d. Výber položky “Delete” z kontextového menu.

8. **Zmazanie nevnoreného kombinovaného fragmentu pod iným kombinovaným fragmentom**
    a. Predpoklad: sekvenčný diagram s aspoň dvomi kombinovanými fragmentami usporiadanými tak, že ani jeden nie je vnorený v tom druhom a jeden sa nachádza nad alebo pod tým druhým.
    b. Označenie kombinovaného fragmentu kliknutím na jeho stredný horný bod.
    c. Kliknutie pravým tlačidlom myše do scény.
    d. Výber položky “Delete” z kontextového menu.
9. **Zmazanie prázdneho kombinovaného fragmentu**
    a. Predpoklad: sekvenčný diagram s aspoň jedným prázdnym kombinovaným fragmentom. Prázdny kombinovaný fragment môže vzniknúť vysunutím všetkých správ z neho.
    b. Označenie prázdneho kombinovaného fragmentu kliknutím na jeho stredný horný bod.
    c. Kliknutie pravým tlačidlom myše do scény.
    d. Výber položky “Delete” z kontextového menu.
10. **Pridanie operandu do kombinovaného fragmentu medzi dve správy**
    a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom.
    b. Označenie v kombinovanom fragmente.
    c. Kliknutie na záložku “Fragments”.
    d. Výber položky “Operand”.
    e. Napísanie podmienky do editačného okna.
    f. Kliknutie nad zvolenú správu v scéne.
11. **Pridanie operandu do kombinovaného fragmentu medzi iný kombinovaný fragment a správu**
    a. Predpoklad: sekvenčný diagram s aspoň jedným vnoreným fragmentom a správou mimo tohto fragmentu.
    b. Označenie vnoreného kombinovaného fragmentu.
    c. Kliknutie na záložku “Fragments”.
    d. Výber položky “Operand”.
    e. Napísanie podmienky do editačného okna.
    f. Kliknutie nad označený kombinovaný fragment.
12. **Zmazanie operandu v kombinovanom fragmente**
    a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom, ktorý obsahuje viac ako jeden operand.
    b. Označenie kombinovaného fragmentu s viac ako jedným operandom.
    c. Kliknutie na záložku “Fragments”.
    d. Výber položky “Operand”.
    e. Napísanie poradového čísla mazaného operandu (poradie začína od 1).
    f. Potvrdenie voľby.
13. **Editácia podmienky prvého operandu kombinovaného fragmentu**
    a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom.
    b. Označenie prvej správy v prvom operande kombinovaného fragmentu.

c. Kliknutie na záložku “Fragments”.
d. Výber položky “Operand”.
e. Napísanie podmienky do vytvoreného okna.
f. Kliknutie nad zvolenú správu.

14. Zväčšenie kombinovaného fragmentu so zahrnutím správ pod fragmentom

a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom, pod ktorým sa nachádzajú neobsiahnuté správy.
b. Označenie kombinovaného fragmentu kliknutím na jeho horný stredný bod.
c. Chytenie fragmentu za ľavý dolný alebo pravý dolný hybný bod.
d. Zväčšovanie fragmentu smerom nadol ťahaním myše so stlačeným ľavým tlačidlom.
e. Pustenie tlačidla myše.

15. Zmenšenie kombinovaného fragmentu s odobratím správ z neho

a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom.
b. Označenie kombinovaného fragmentu kliknutím na jeho horný stredný bod.
c. Chytenie fragmentu za ľavý alebo pravý dolný hybný bod.
d. Zmenšovanie fragmentom smerom nahor ťahaním myše so tlačeným ľavým tlačidlom.
e. Pustenie tlačidla myše.

16. Zväčšenie kombinovaného fragmentu so zahrnutím správ nad fragmentom

a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom, nad ktorým sa nachádzajú neobsiahnuté správy.
b. Označenie kombinovaného fragmentu kliknutím na jeho horný stredný bod.
c. Chytenie fragmentu za ľavý horný alebo pravý horný hybný bod.
d. Zväčšovanie fragmentu smerom nahor ťahaním myše so stlačeným ľavým tlačidlom.
e. Pustenie tlačidla myše.

17. Zväčšenie kombinovaného fragmentu so zahrnutím iných správ a iných kombinovaných fragmentov

a. Predpoklad: sekvenčný diagram s aspoň dvomi kombinovanými fragmentami a viacerými správami medzi nimi.
b. Označenie kombinovaného fragmentu kliknutím na jeho horný stredný bod.
c. Chytenie fragmentu za niektorý z pohybujúcich rohov.
d. Zväčšenie fragmentu smerom nahor alebo nadol ťahaním myše so stlačeným ľavým tlačidlom.
e. Obsiahnutie iného kombinovaného fragmentu (jeho hlavičky) a prípadných správ pri ťahaní.
f. Pustenie tlačidla myše.

18. Zmenšenie kombinovaného fragmentu s odobraním iných správ a iných kombinovaných fragmentov z neho

a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom, ktorý obsahuje iný vnorený kombinovaný fragment a správy.
b. Označenie vonkajšieho (nevnoreného) kombinovaného fragmentu kliknutím na jeho horný stredný bod.

c. Chytenie fragmentu za niektorý z pohybujúcich rohov.
d. Zmenšenie fragmentu smerom nadol alebo nahor ťahaním myše so stlačeným ľavým tlačidlom.
e. Pri ťahaní "odbalí" vonkajší fragment hlavičku vnútorného fragmentu (a prípadné iné správy v ňom).
f. Pustenie tlačidla myše.

19. Zväčšenie kombinovaného fragmentu so zahrnutím správ a kombinovaných fragmentov vľavo alebo vpravo od fragmentu (na iných čiarách života)

a. Predpoklad: sekvenčný diagram s viacerými kombinovanými fragmentami, s viacerými čiarami života a s takými správami a kombinovanými fragmentami, ktoré sa nachádzajú mimo kombinovaného fragmentu a mimo čiar života, ktoré sú zahrnuté kombinovaným fragmentom.
b. Označenie kombinovaného fragmentu kliknutím na jeho horný stredný bod.
c. Chytenie fragmentu za niektorý z pohybujúcich rohov.
d. Zväčšovanie fragmentu v ľubovoľnom smere tak, aby zahrnul iné správy a iné kombinované fragmenty.
e. Pustenie tlačidla myše.

20. Hýbanie kombinovaným fragmentom smerom nadol / nahor

a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom.
b. Označenie kombinovaného fragmentu kliknutím na jeho horný stredný bod.
c. Chytenie fragmentu za stredný horný pohybujúci bod.
d. Ťahanie fragmentu nahor alebo nadol ťahaním myše so stlačeným ľavým tlačidlom.
e. Pustenie tlačidla myše.

21. Hýbanie kombinovaným fragmentom vľavo / vpravo

a. Predpoklad: sekvenčný diagram s aspoň jedným kombinovaným fragmentom.
b. Označenie kombinovaného fragmentu kliknutím na jeho horný stredný bod.
c. Chytenie fragmentu za stredný horný pohybujúci bod.
d. Ťahanie fragmentu doľava alebo doprava ťahaním myše so stlačeným ľavým tlačidlom.
e. Pustenie tlačidla myše.

3.4.4.2 Výsledky testovacích scenárov

#	Názov testovacieho scenára	Stav testu	Poznámka
1.	Pridanie jednoduchého kombinovaného fragmentu so všetkými správami	Úspech	Kombinovaný fragment je úspešne vložený.
2.	Pridanie jednoduchého kombinovaného	Úspech	Kombinovaný fragment je úspešne vložený. Interakčný operand je v

	fragmentu s vybranými správami		metamodeli úspešne rozdelený.
3.	Pridanie kombinovaného fragmentu s pokrytím správ v dvoch rôznych interakčných operandoch	Neúspech	Očakávané správanie. Došlo by k prekrytiu dvoch kombinovaných fragmentov. Odporúča sa zmazať pôvodný fragment a vybrať správy nanovo.
4.	Pridanie vnoreného kombinovaného fragmentu	Úspech	Vnorený fragment je úspešne pridaný.
5.	Zmazanie nevnoreného kombinovaného fragmentu	Úspech	Fragment je úspešne zmazaný. Pôvodný obsah fragmentu sa úspešne zachováva. Metamodelovo sa operand správne presunul o úroveň vyššie.
6.	Zmazanie vnoreného kombinovaného fragmentu	Úspech	Fragment je úspešne zmazaný. Vnorenie sa znížilo o jednotku.
7.	Zmazanie kombinovaného fragmentu s viacerými operandami	Úspech	Fragment je úspešne zmazaný. Operandy sú úspešne zmazané.
8.	Zmazanie nevnoreného kombinovaného fragmentu pod iným kombinovaným fragmentom	Úspech	Kombinovaný fragment je úspešne zmazaný.
9.	Zmazanie prázdneho kombinovaného fragmentu	Úspech	Prázdny kombinovaný fragment je úspešne zmazaný.
10.	Pridanie operandu do kombinovaného fragmentu medzi dve správy	Úspech	Operand je úspešne pridaný. Metamodelovo je pôvodný operand úspešne rozdelený.
11.	Pridanie operandu do kombinovaného fragmentu medzi iný kombinovaný fragment a	Úspech	Operand je úspešne pridaný. Metamodel OK.

	správu		
12.	Zmazanie operandu v kombinovanom fragmente	Úspech	Operand je úspešne zmazaný. Metamodelovo nastalo úspešné zlúčenie operandov.
13.	Editácia podmienky prvého operandu kombinovaného fragmentu	Úspech	Podmienka bola úspešne zmenená.
14.	Zväčšenie kombinovaného fragmentu so zahrnutím správ pod fragmentom	Úspech	Kombinovaný fragment úspešne obsiahol zvolené správy.
15.	Zmenšenie kombinovaného fragmentu s odobratím správ z neho	Úspech	Správy boli z fragmentu úspešne odobrané.
16.	Zväčšenie kombinovaného fragmentu so zahrnutím správ nad fragmentom	Úspech	Správy nad fragmentom boli úspešne zahrnuté do fragmentu.
17.	Zväčšenie kombinovaného fragmentu so zahrnutím iných správ a iných kombinovaných fragmentov	Úspech	Iné kombinované fragmenty a prípadné iné správy boli úspešne zahrnuté do fragmentu.
18.	Zmenšenie kombinovaného fragmentu s odobraním iných správ a iných kombinovaných fragmentov z neho	Úspech	Iné kombinované fragmenty a prípadné iné správy boli úspešne odobrané z fragmentu.

19.	Zväčšenie kombinovaného fragmentu so zahrnutím správ a kombinovaných fragmentov vľavo alebo vpravo od fragmentu (na iných čiarách života)	Úspech	Iné kombinované fragmenty a prípadné iné správy boli úspešne zahrnuté do fragmentu.
20.	Hýbanie kombinovaným fragmentom smerom nadol / nahor	Neúspech	Iba čiastočne implementovaná funkcionalita. S fragmentom sa dá hýbať, ale občas sa preklopí okolo svojej X osi. Metamodel sa pri hýbaní rozruší.
21.	Hýbanie kombinovaným fragmentom vľavo / vpravo	Neúspech	Neimplementovaná funkcionalita.

Tabuľka 5: Vyhodnotenie testovacích scenárov pre kombinované fragmenty

3.4.5 Identifikované problémy mimo testovacích scenárov

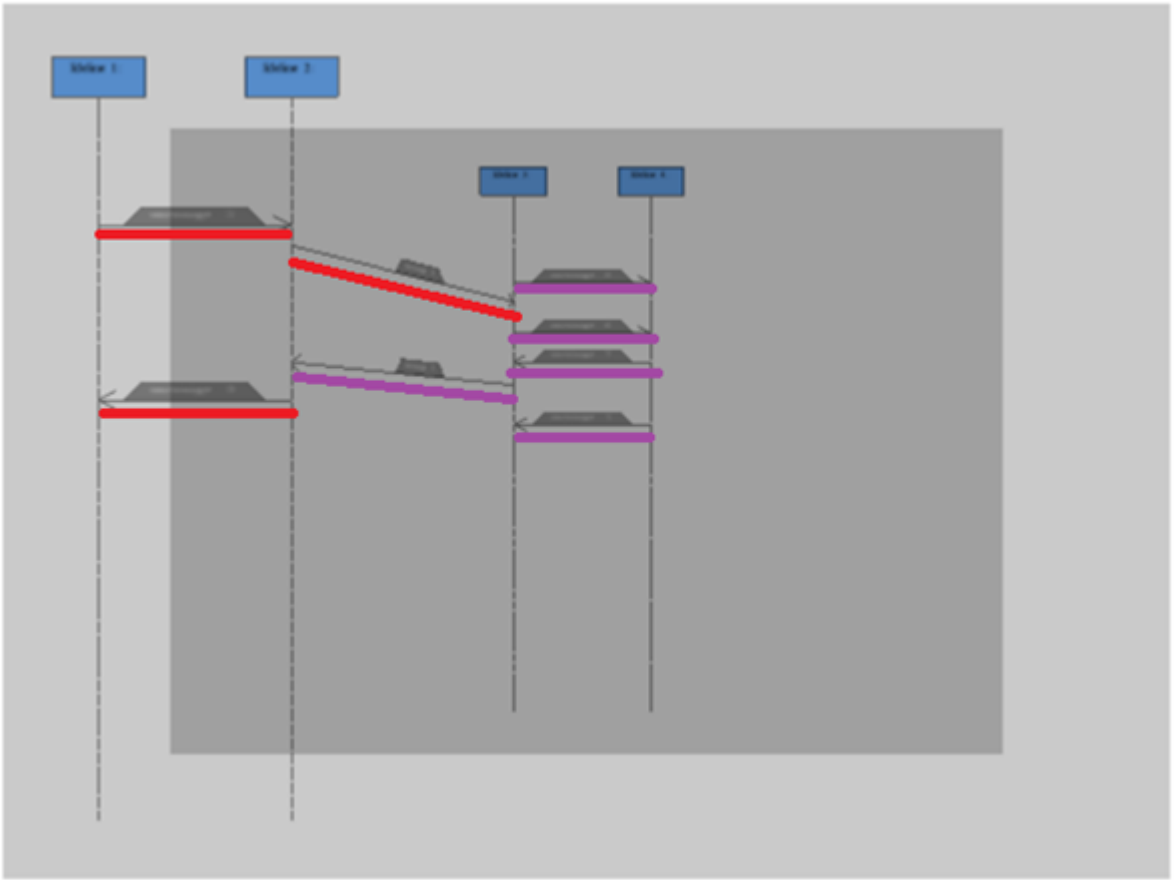
V tejto časti sa nachádzajú niektoré identifikované problémy, ktoré nie sú pokryté testovacími scenármi.

1. Vypnutie projektu iným spôsobom než kliknutím na tlačidlo “Esc” môže spôsobiť, že program prestane pracovať.
2. Pri inicializácii škálovania fragmentov je potrebné určitý čas krúžiť okolo voleného hybného bodu.

4 Výskumné výsledky

4.1 GeneralOrdering v 3D diagrame

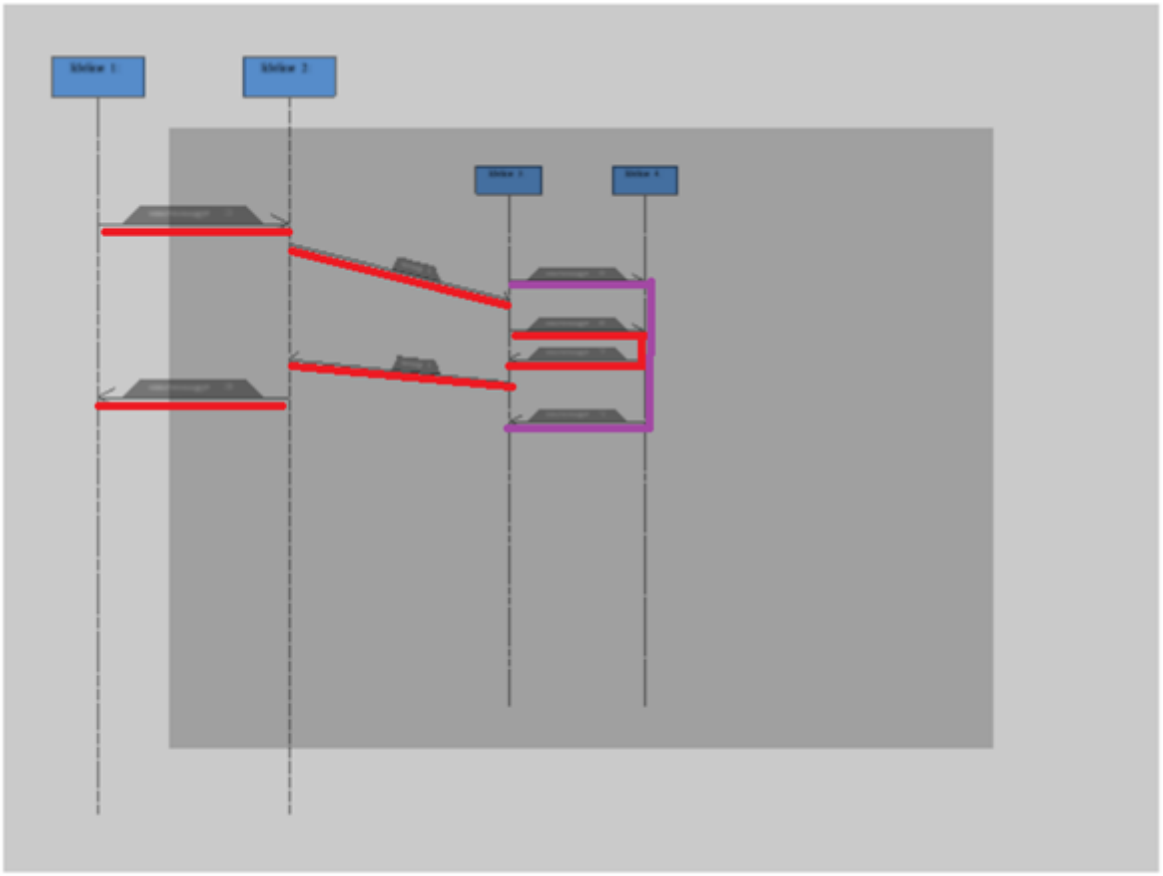
V prípade 3D správ naše riešenie priraďuje vrstve jeden `InteractionFragment`. Každej vrstve sú priradené čiary života a `GeneralOrdering` správ, ktoré majú zdroj na určenej vrstve (obrázok 76). Objektovo tento návrh nesedí, kvôli tomu, že metódy sú implementované v triede cieľa, nie zdroja. Táto filozofia má len grafickú výhodu pre použitie pre používateľa, že dokáže mať vytvorený model 3D model sekvenčného diagramu. V prípade pokiaľ by bola sekvenčnému diagramu pridávaná funkcionalita exportu do programovacieho jazyka vzniká ale problém, lebo nie je relevantné poradie správ podľa grafických informácií.



Obrázok 76: `GeneralOrdering` v rámci dvoch `InteractionFragmen-tov`

Kvôli nekompatibilite pri exporte diagramu do kódu je potrebné nedávať do súvisu `InteractionFragment` a vrstu sekvenčného diagramu. Modifikácia `GeneralOrdering-ov` pozostáva z prepojenia metód, ktoré vytvárajú istú postupnosť. Modifikovaný `GeneralOrdering` je zaznačený na obrázku 77. Každá štruktúra `GeneralOrdering` určuje

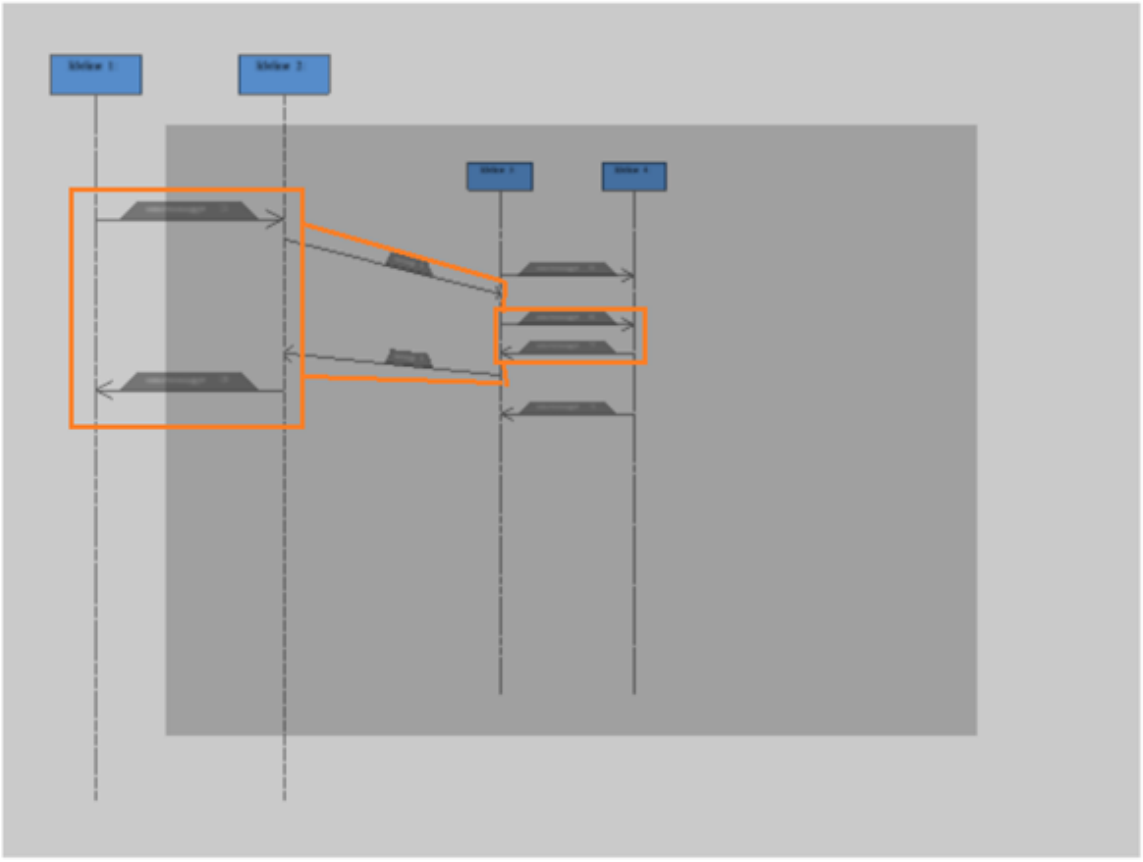
poradie správ podľa vertikálnej osi. Preto je potrebné aby v prípade vzniku nekonzistentnej udalosti boli oddelené `GeneralOrdering-y` ako napríklad na obrázku 77.



Obrázok 77: Korektný `GeneralOrdering` v prípade 3D sekvenčného diagramu

4.2 3D Kombinovaný fragment

V prípade 3D kombinovaného fragmentu je vykreslenie najlepšie formou zobrazenou na obrázku 78. V rámci jedného GeneralOrdering-u je kombinovaný fragment v 3D priestore najlepšie zakreslený pomocou viacerých grafických elementov (obdĺžnikov).



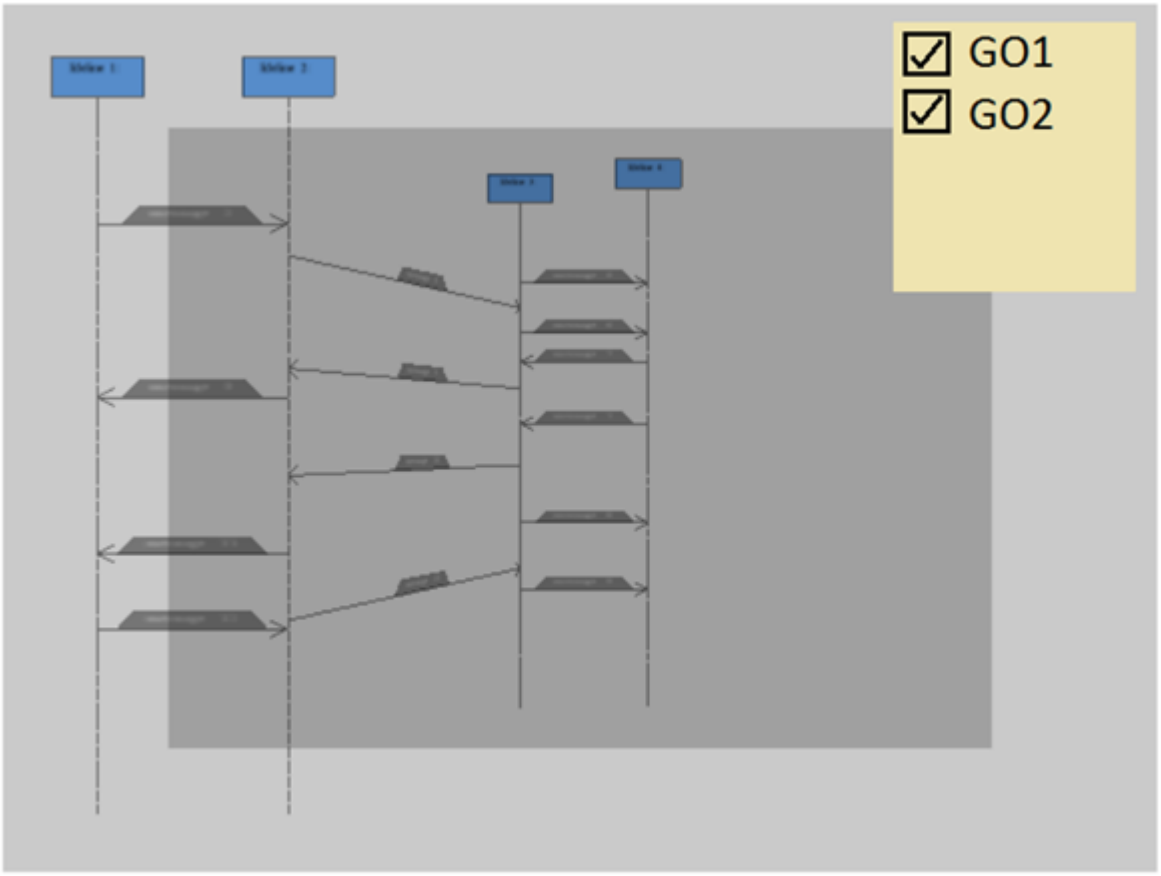
Obrázok 78: 3D kombinovaný fragment

4.3 Posun Fragmentu po vertikálnej osi

V prípade posunu fragmentu po vertikálnej osi je najlepším riešením vystrihnutie kombinovaného fragmentu aj s obsahom mimo metamodel. Graficky sa elementy môžu hýbať aj mimo fragment a v prípade, že sa nevedia naviazať na metamodel, tak túto akciu nevykonávajú, ale zobrazia sa červenou farbou, ktorá indikuje nekonzistentnosť v rámci metamodelu. Používateľ bude upozornený, že metamodel nie je konzistentný a presunie fragment na miesto s dostatočným priestorom. Po tejto akcii sa zobrazí kombinovaný fragment bez indikácie a naviaže sa na metamodel.

4.4 Zlepšenie prehľadnosti

Pre vylepšenie prehľadnosti je graficky vhodné zobrazovať len isté elementy v rámci `GenerealOrdering-u` jednoduchým zoznamom zaškrtávacími okienkami. Každá štruktúra `GeneralOrdering-ov` by bola istá akcia, ktorá by pomocou týchto okienok bola zobrazená alebo skrytá. V inom prípade by bol sekvenčný diagram nezrozumiteľný pre používateľa (obrázok 79).



Obrázok 79: Nezrozumiteľný sekvenčný diagram

Príloha A: Príručky

A.1 Inštalačná príručka

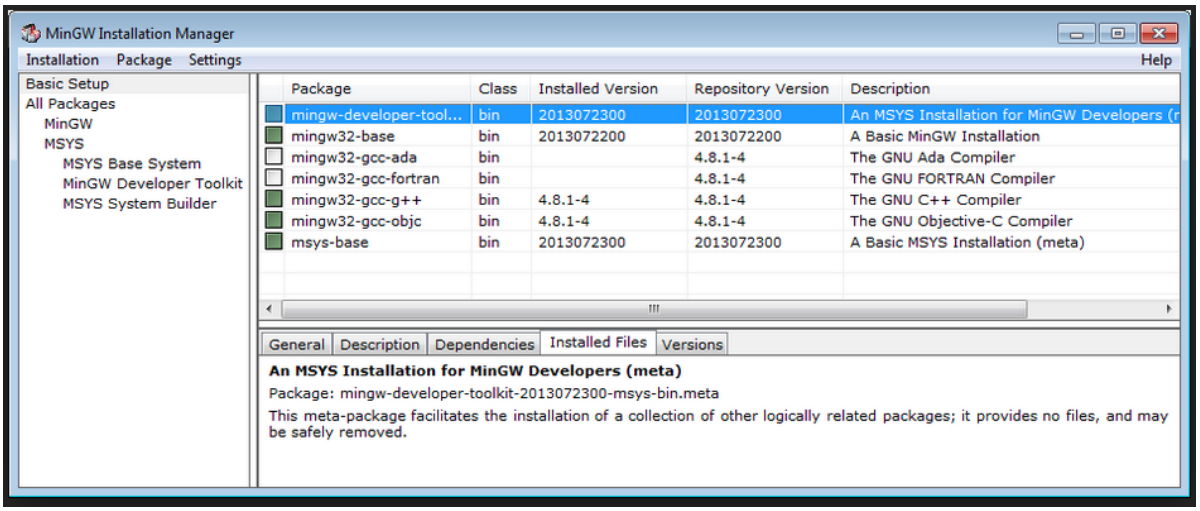
V úvodnej fáze projektu bolo potrebné zistiť, aké kroky je nutné pri spúšťaní projektu na vlastnom počítači vykonať. Preto sme vytvorili nasledujúcu inštalačnú príručku.

A.1.1 Inštalácia vývojového prostredia Eclipse pre C/C++

Prijateľná je akákoľvek 32-bitová verzia tohto prostredia. Pri 64-bitovej verzii môžu nastať problémy pri kompilácii 3D UML prototypu a preto sa jej používanie neodporúča. Používateľ postupuje pri inštalácii nástroja štandardne.

A.1.2 Inštalácia kompilačného systému MinGW

1. Pri inštalácii je odporúčané vybrať zložku "C:/MinGW"
2. Po dokončení úvodnej inštalácie sa kliknutím na tlačidlo "Continue" otvorí okno s dostupnými balíkmi. Potrebné balíky sú zobrazené na obrázku 80 nižšie (označené zelenou farbou).
3. Potvrdením výberu sa balíky nainštalujú.



Obrázok 80: Potrebné balíky pre korektnú inštaláciu MinGW

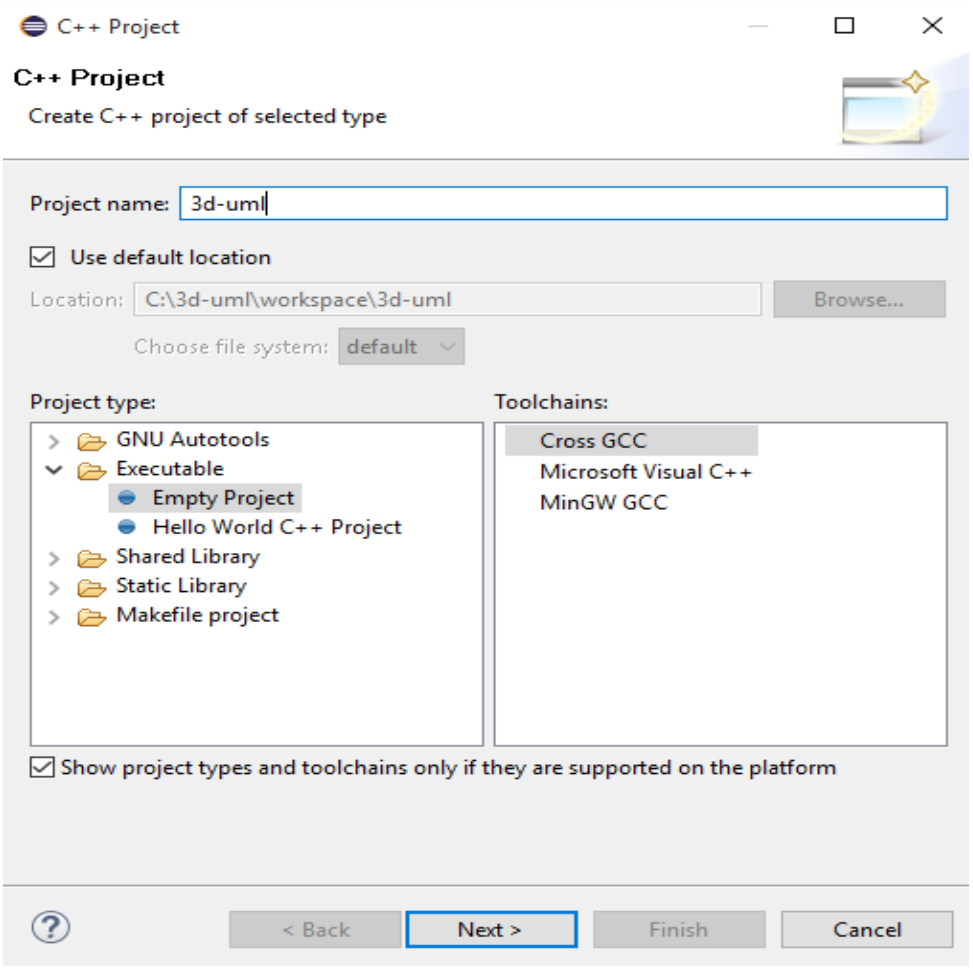
A.1.3 Stiahnutie prototypu z Bitbucketu

1. Najjednoduchší spôsob stiahnutia projektu je využitie open source programu SourceTree.
2. Po nainštalovaní SourceTree je potrebné v Bitbuckete zvoliť konkrétnu vetvu.
3. V novom okne potom stačí len kliknúť na "Check out -> Check out in SourceTree" v pravom hornom rohu. Tým sa spustí SourceTree.

4. V SourceTree je nasledovne potrebné vybrať zložku, kam sa má prototyp stiahnuť. Odporúčaná cesta je “C:\3d-uml”

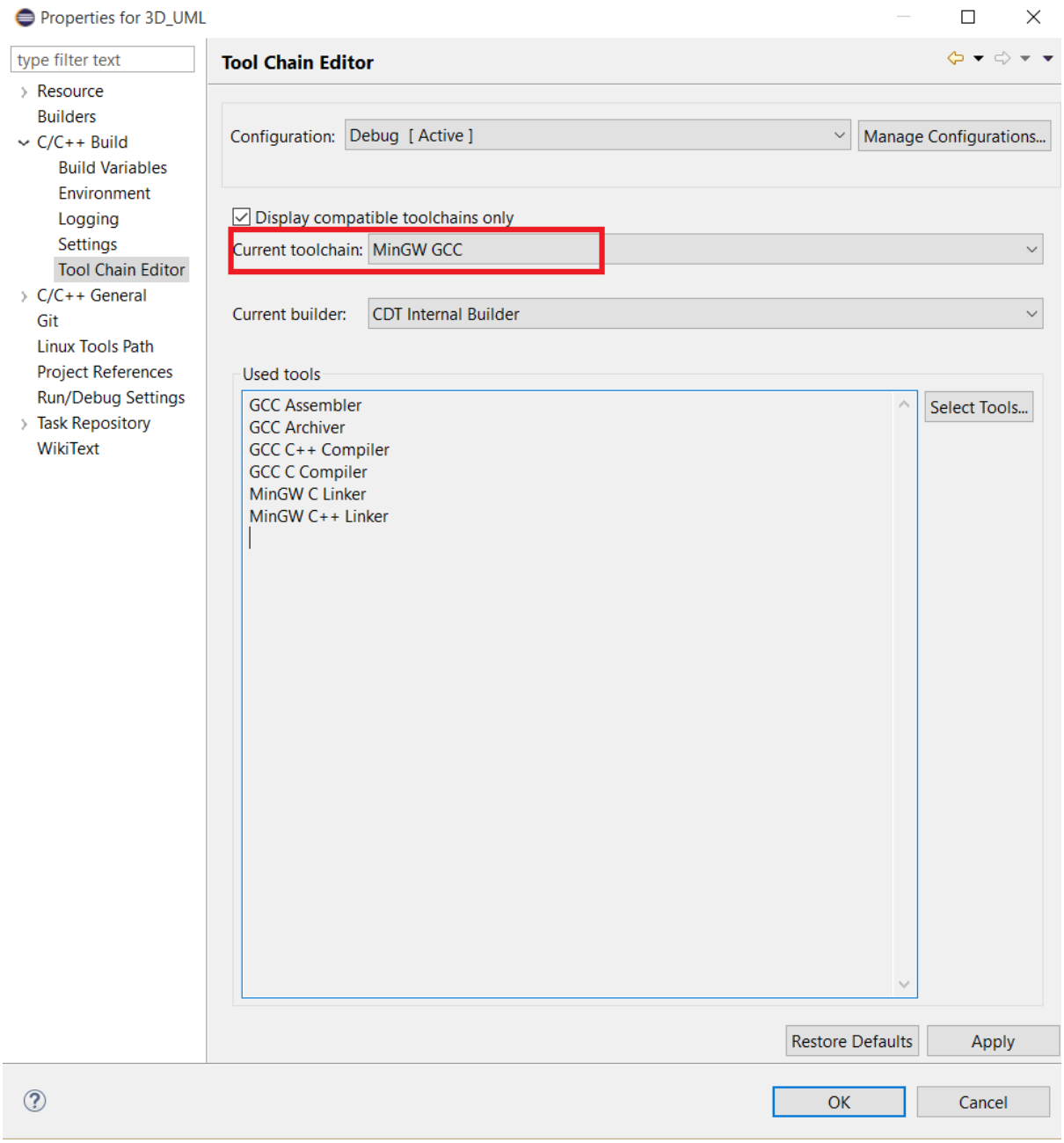
A.1.4 Úprava nastavení v Eclipse

1. Pri spustení Eclipse je potrebné vybrať *workspace*. V každom prototype sa *workspace* už nachádza, takže je potrebné ho len nájsť v zložke, kam sa celý prototyp stiahol.
2. V prípade, že sa projekt v Eclipse zobrazí, preskočte na krok 4. V prípade, že po zvolení existujúceho *workspacu* sa v Eclipse nezobrazí projekt, je potrebné vytvoriť nový. Vytvorenie projektu je zobrazené na obrázku nižšie.



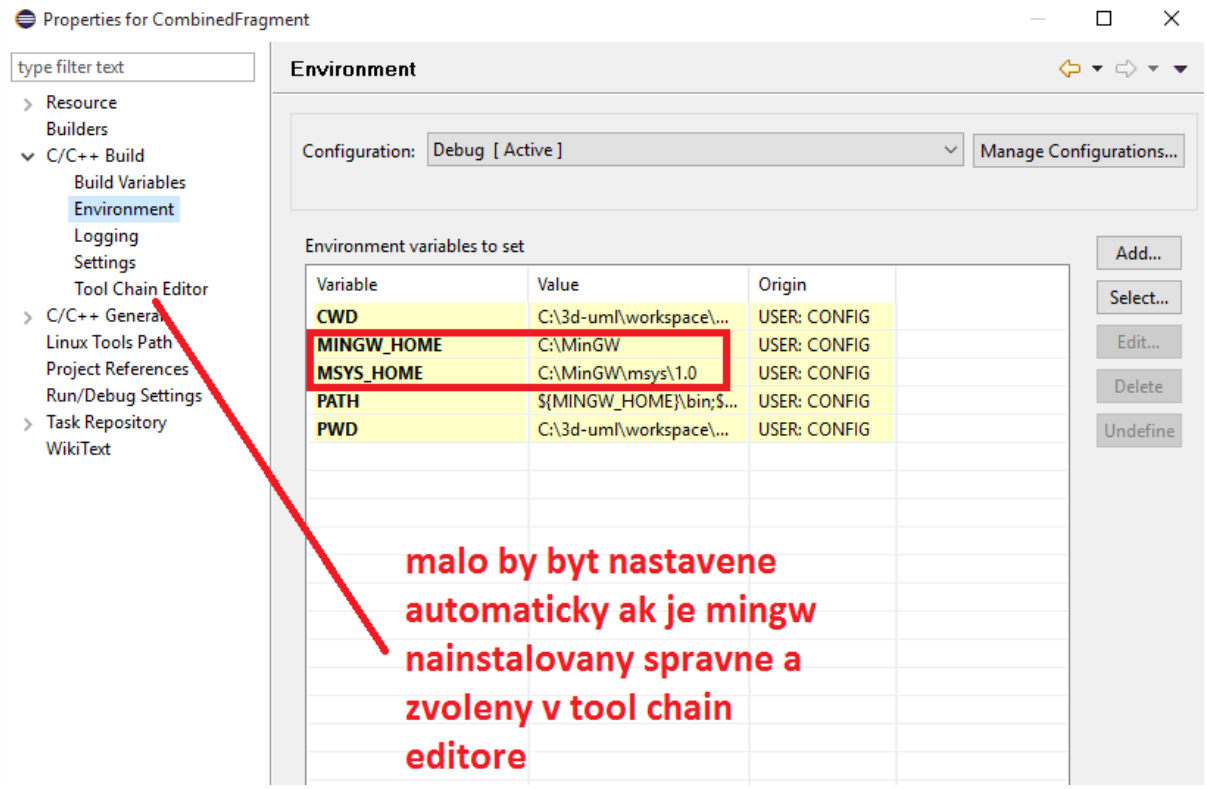
Obrázok 81: Ogre setup - krok 1

3. Po vytvorení projektu je potrebné prekopírovať zložky “Debug” a “src” z pôvodného projektu.
4. Následne je potrebné kliknúť pravým tlačidlom na projekt a zvoliť Properties.
5. V časti “C/C++ Build → Tool Chain Editor” je potrebné zmeniť používaný kompilátor na MinGW GCC. Zmena je zobrazená na obrázku nižšie.



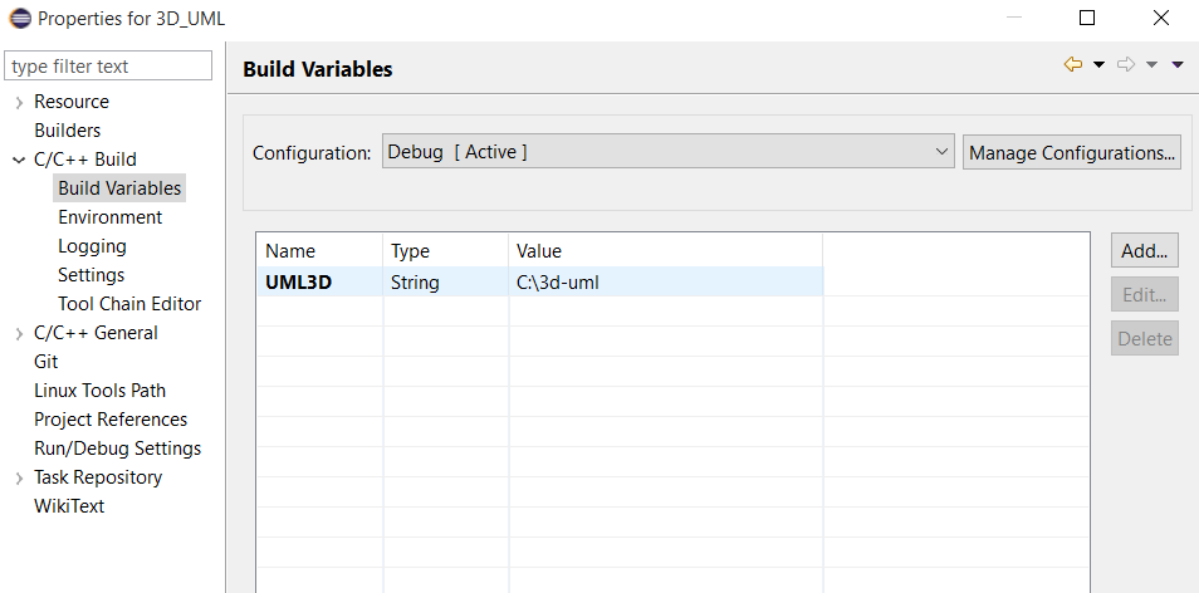
Obrázok 82: Ogre setup - krok 2

6. Po zvolení kompilátora by mali byť automaticky v "C/C++ Build → Environment" tieto premenné prostredia. V opačnom prípade je potrebné ich ručne pridať.



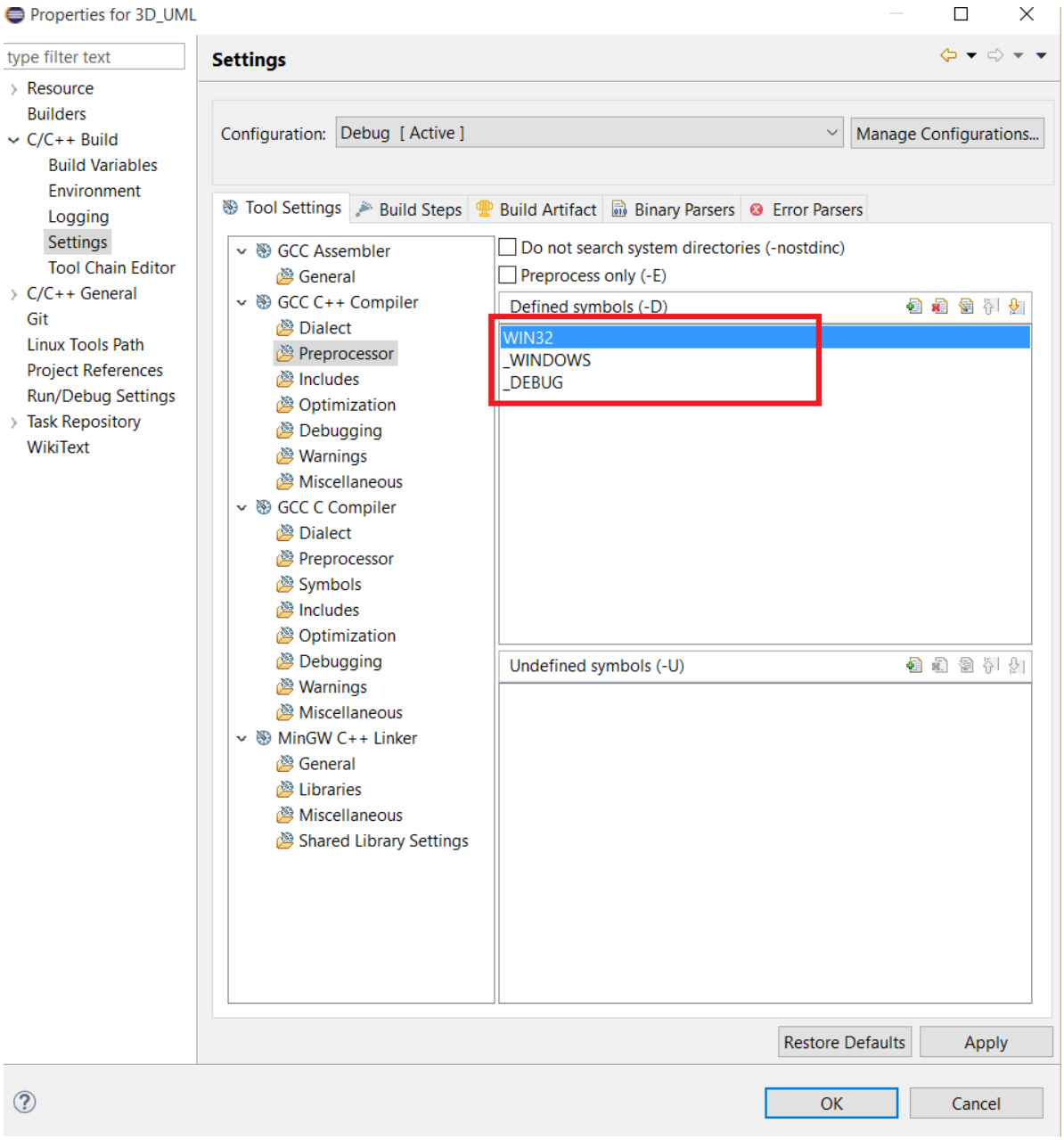
Obázok 83: Ogre setup - krok 3

7. V "C/C++ Build → Build Variables" vytvorte premennú s názvom "UML3D" a ako jej obsah dajte cestu k projektu.

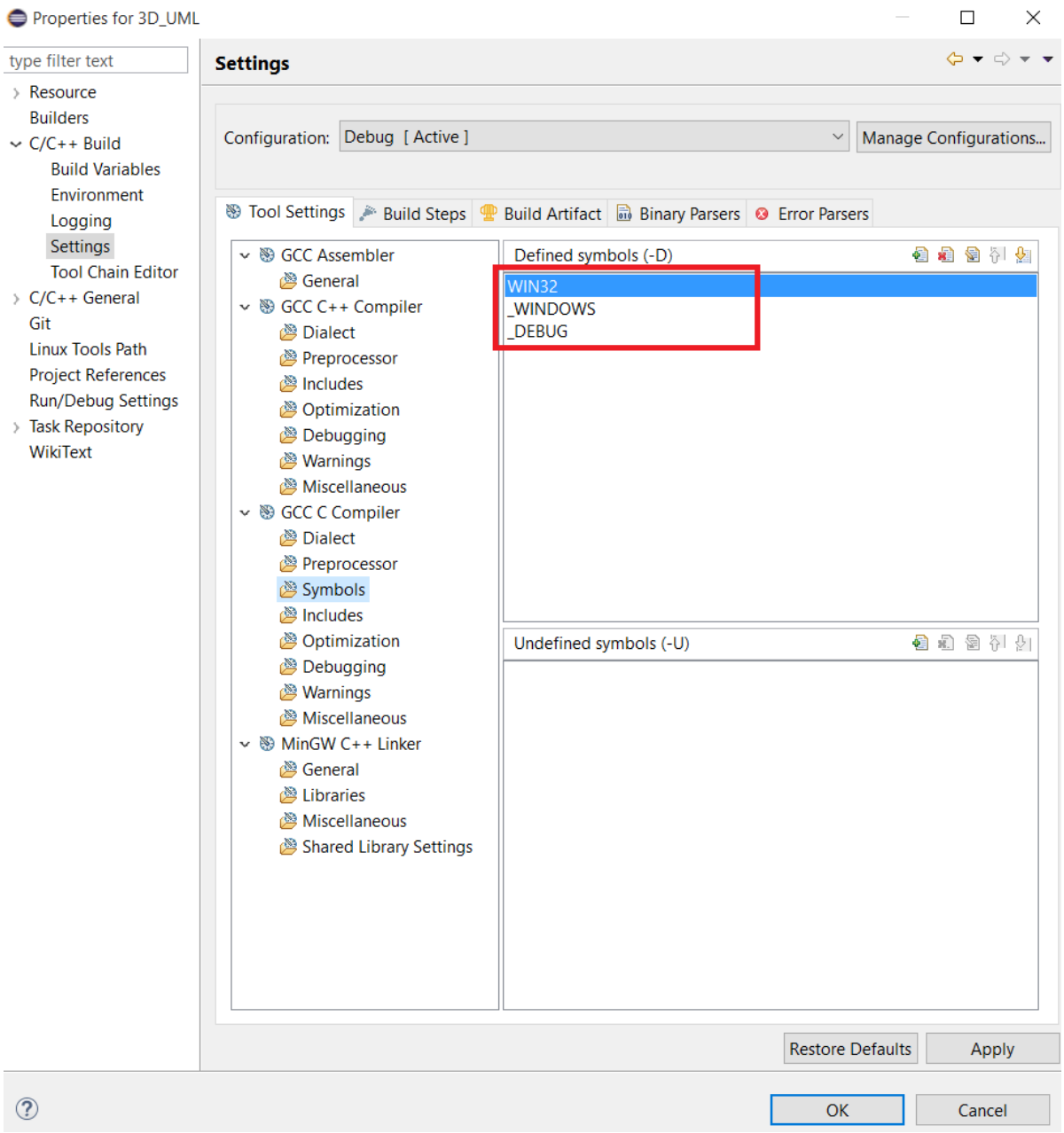


Obrázok 84: Ogre setup - krok 4

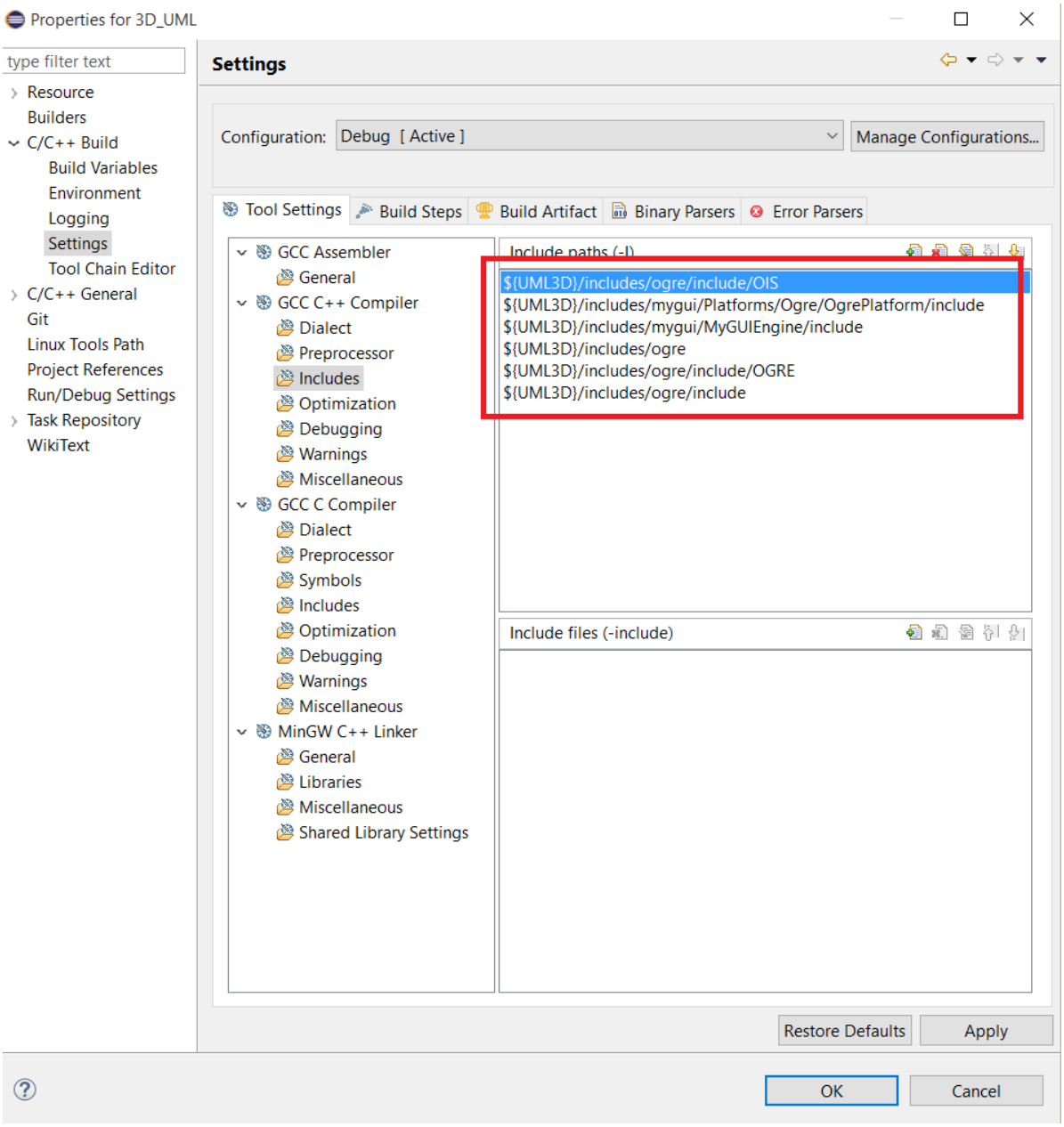
8. V časti "C/C++ Build → Settings" nasledovne vyplňte všetky údaje tak, ako je zobrazené na obrázkoch nižšie.



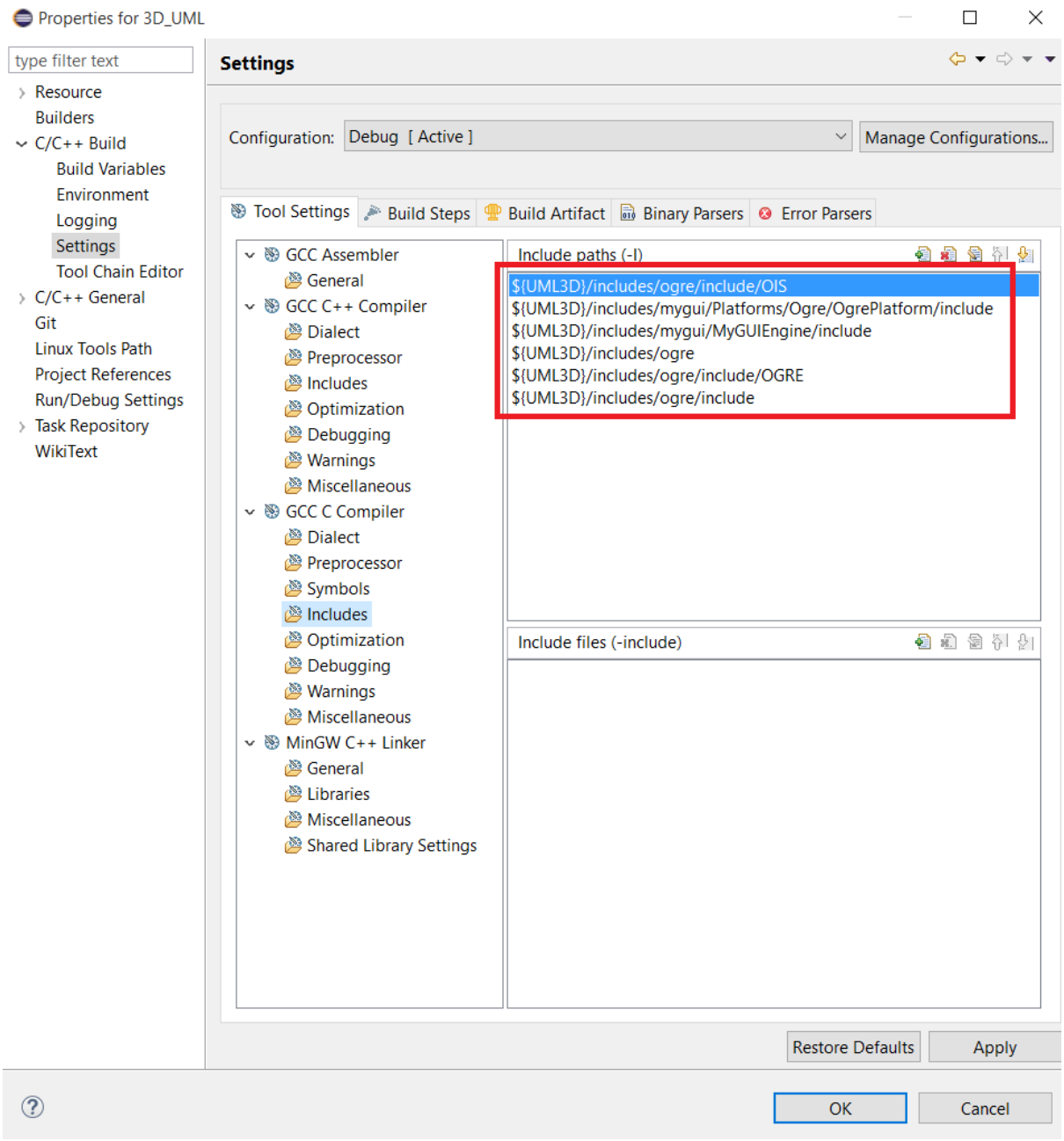
Obrázok 85: Ogre setup - krok 5



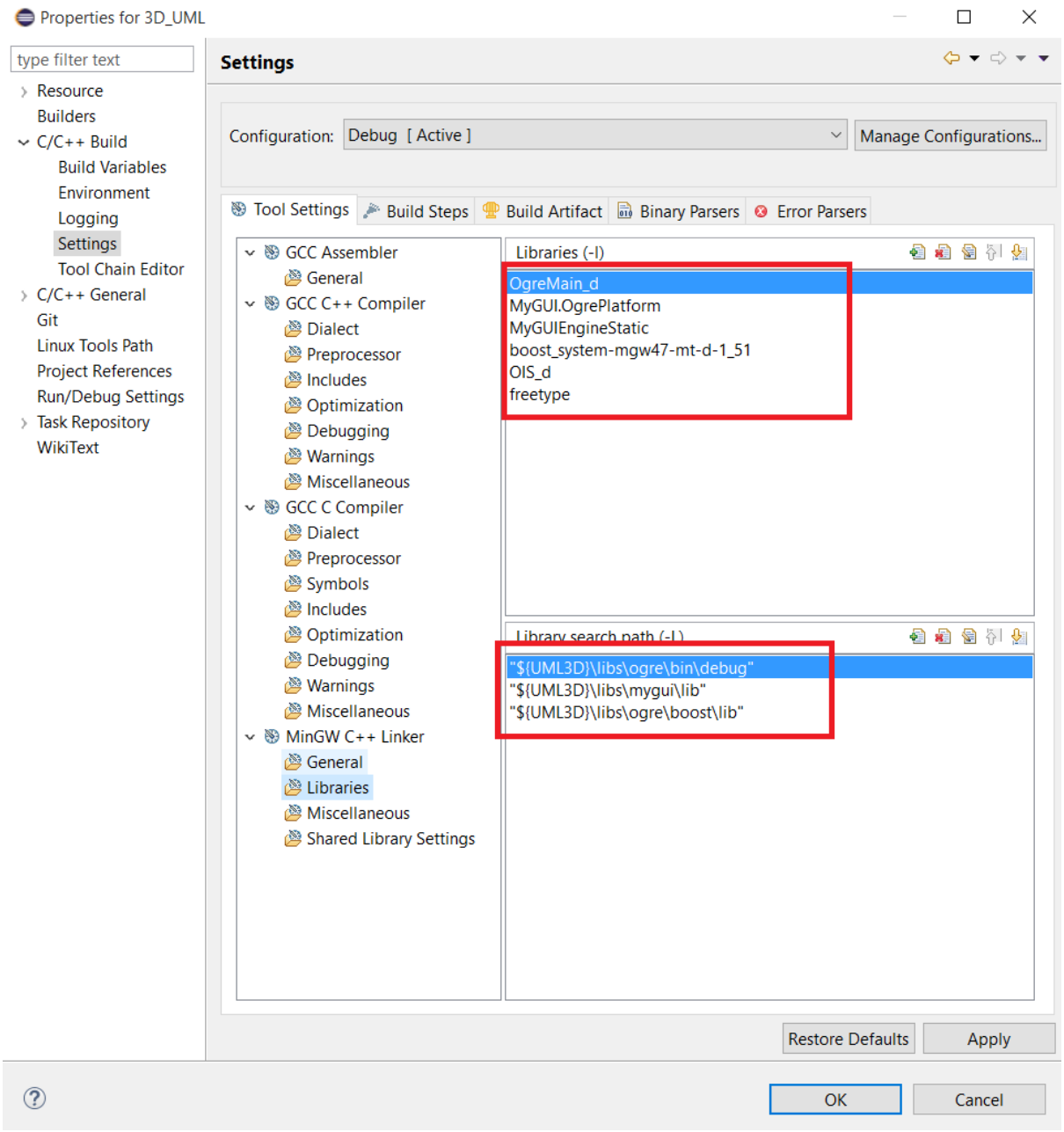
Obrázok 86: Ogre setup - krok 6



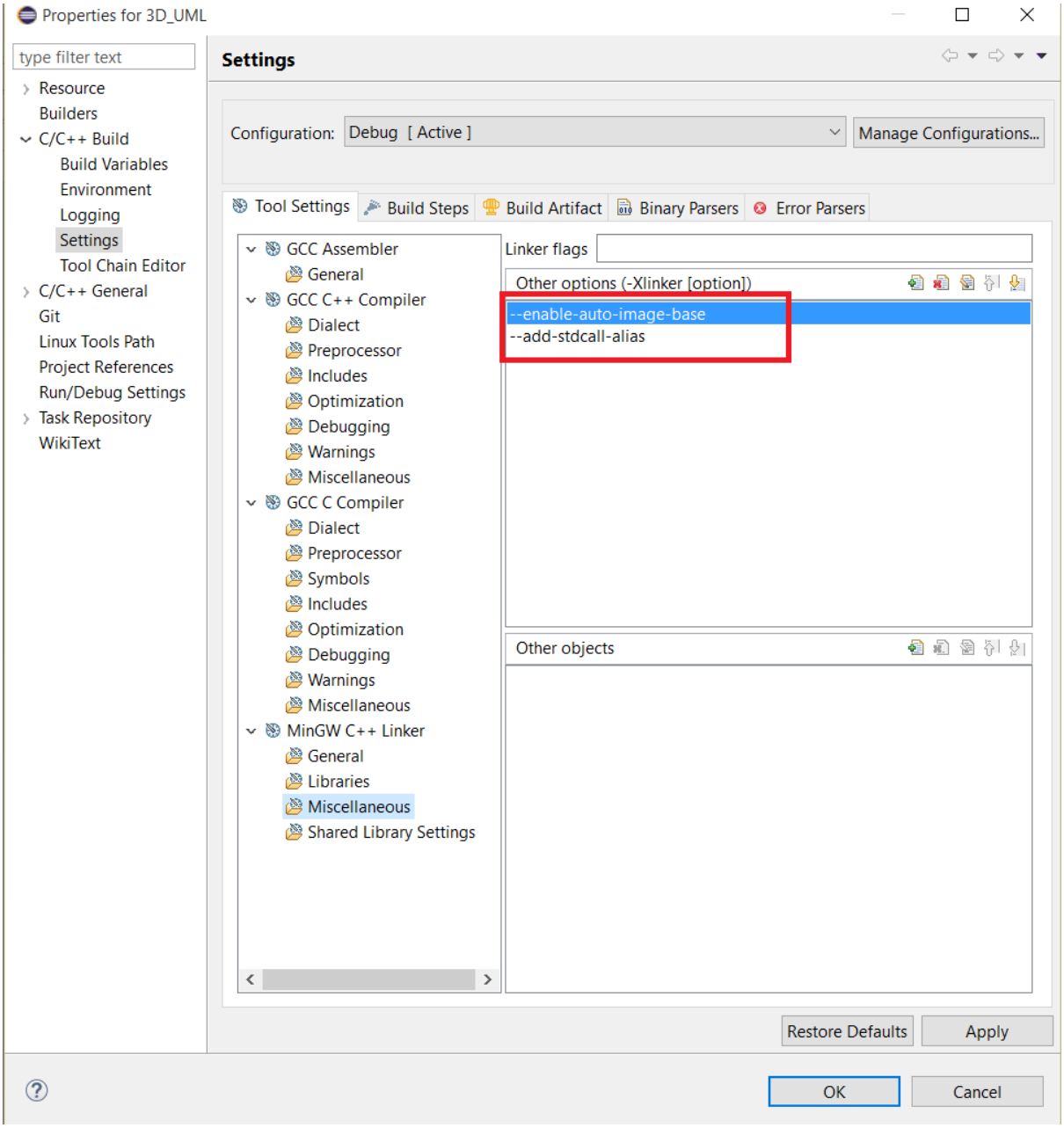
Obrázok 87: Ogre setup - krok 7



Obrázok 88: Ogre setup - krok 8

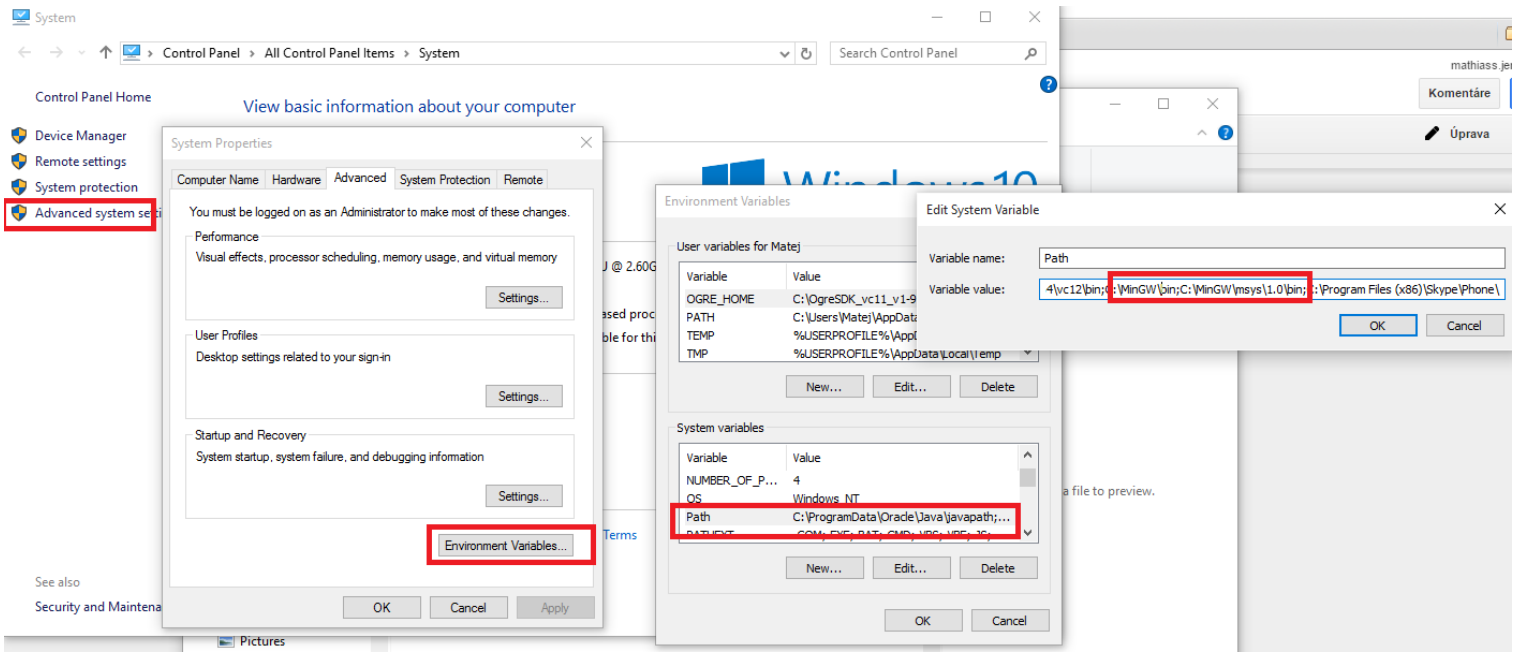


Obrázok 89: Ogre setup - krok 9



Obrázok 90: Ogre setup - krok 10

9. Následne nastavte systémové premenné tak, ako je znázornené na obrázku nižšie.



Obrázok 91: Nastavenie systémovej premennej ‘Path’

10. Po nastavení všetkých údajov reštartuje Eclipse. Projekt by mal byť v tomto stave kompilovateľný. Kompilácia projektu môže trvať rádovo niekoľko minút (na starších počítačoch aj 30 minút). Po prvej kompilácii je možné využívať inkrementálnu kompiláciu, ktorá je oveľa rýchlejšia (trvá max. jednotky minút).

A.1.5 Známe problémy

V tejto časti sa nachádzajú problémy, ktoré nastali pri kompilácii / spustení skompilovaného projektu jednému z členov tímu. Na všetky identifikované problémy sa tu nachádzajú aj riešenia.

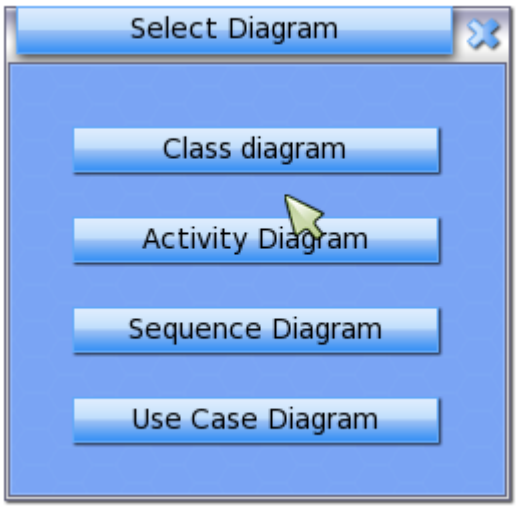
1. Pri kompilácii sa rovno na začiatku vypíše chyba typu “G++ compiler is missing” - Táto chyba znamená, že nie je správne nastavená cesta k MinGW. Musí byť preto pridaná do “Project → Settings → C/C++ Build → Environment” do premennej “Path”. Pridaná cesta by mala vyzerať nasledovne “${MINGW_HOME}\bin;${MSYS_HOME}\bin”.
2. Projekt je skompilovaný, ale po spustení exe súboru zahlási chybu typu “Program 3d-uml has stopped working”. Chyba je pravdepodobne v zložke “resources” a je potrebné ju nahradiť. Odkaz na funkčnú zložku: “**https://drive.google.com/file/d/0B0Tko-LCDS_BZlIwX19McDFOU0U/view?usp=sharing**”.
3. Projekt je skompilovaný, ale po spustení exe súboru zahlási chybu typu “Cannot run program. Missing dll”. Po kompilácii sa nemusia presunúť knižnice potrebné pre spustenie programu a preto ich treba prekopírovať ručne zo zložky “dll” do “Debug”.

Iné problémy by pri kompilácii / spustení nemali nastať. V opačnom prípade je potrebné prejsť si návod podrobne, či boli všetky kroky vykonané správne.

A.2 Používateľská príručka

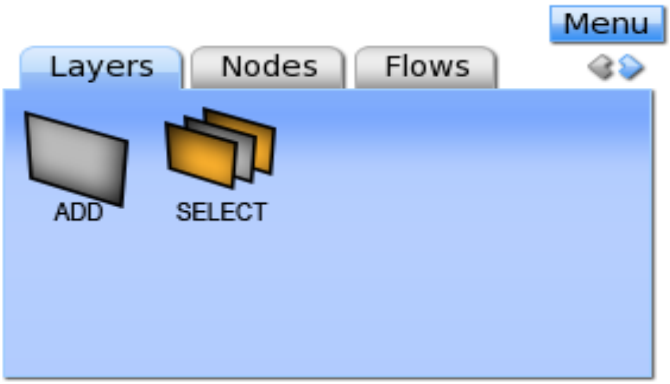
A.2.1 Sputenie aplikácie

Po spustení aplikácie sa nám zobrazí nasledujúce dialógové okno:



Obrázok 92: Úvodné dialógové okno

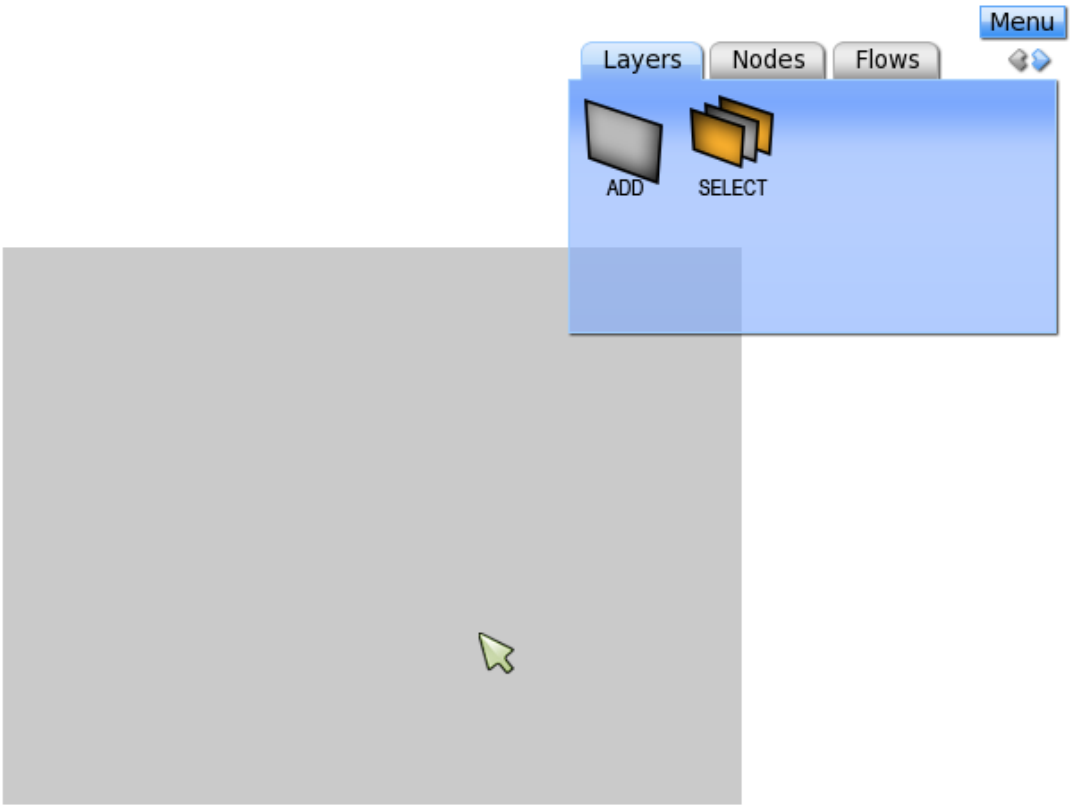
Pre modelovanie sekvenčného diagramu klikneme na tlačidlo “Sequence Diagram”. Následne sa nám zobrazí prázdna obrazovka s tlačidlom “Menu” v pravom hornom rohu. Pre kliknutie na toto tlačidlo, je potrebné sa prepnúť do editačného módu. Prepnutie sa do editačného módu vykonáme stačením klávesy medzerník. Po kliknutí na tlačidlo “Menu” sa nám v pravom hornom rohu zobrazí nasledujúce okno:



Obrázok 93: Menu so záložkami

Toto okno pozostáva z piatich záložiek. Konkrétne sa jedná o záložky na prácu s vrstvami, čiarami života, správami, fragmentami a záložkou na správu diagramu.

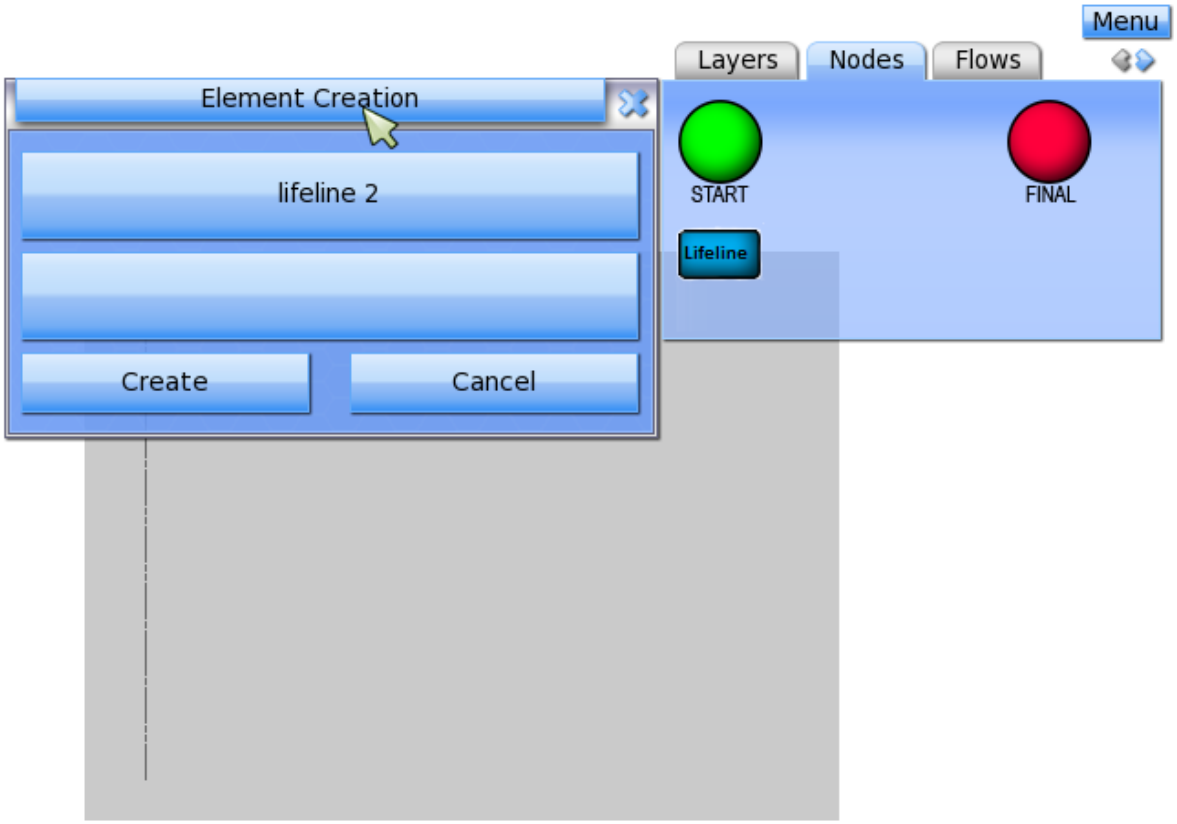
Keďže pracujeme v 3D priestore, pri začatí kreslenia diagramu si musíme zvoliť vrstvu na ktorej chceme pracovať. Pokiaľ sa žiadna vrstva v našom projekte nenachádza, musíme ju do projektu pridať. Vrstva sa do projektu pridáva kliknutím na položku “ADD” v záložke “Layers” ktorá sa nachádza v menu. Pridaná vrstva vyzerá nasledovne:



Obrázok 94: Zobrazenie vrstvy

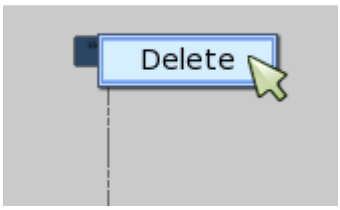
A.2.2 Čiary života

Čiaru života pridáme na vrstvu stlčením tlačidla “Lifeline”, ktorá sa nachádza v záložke “Nodes”. Po stlačení tlačidla klikneme na požadovanú vrstvu na ktorú chceme pridať čiaru života. Následne sa nám zobrazí vyskakovacie okno, do ktorého môžeme zadať identifikátor čiary života a jej názov.



Obrázok 95: Vloženie čiary života

Pokiaľ chceme čiaru života odstrániť, tak ju označíme jedným kliknutím ľavého tlačidla myši do stredu hlavičky čiary života. Následne stlačíme pravé tlačidlo myši. Zobrazí sa nám potvrdzovacie okno, do ktorého keď klikneme, potvrdí sa operácia zmazania a čiara života sa vymaže.

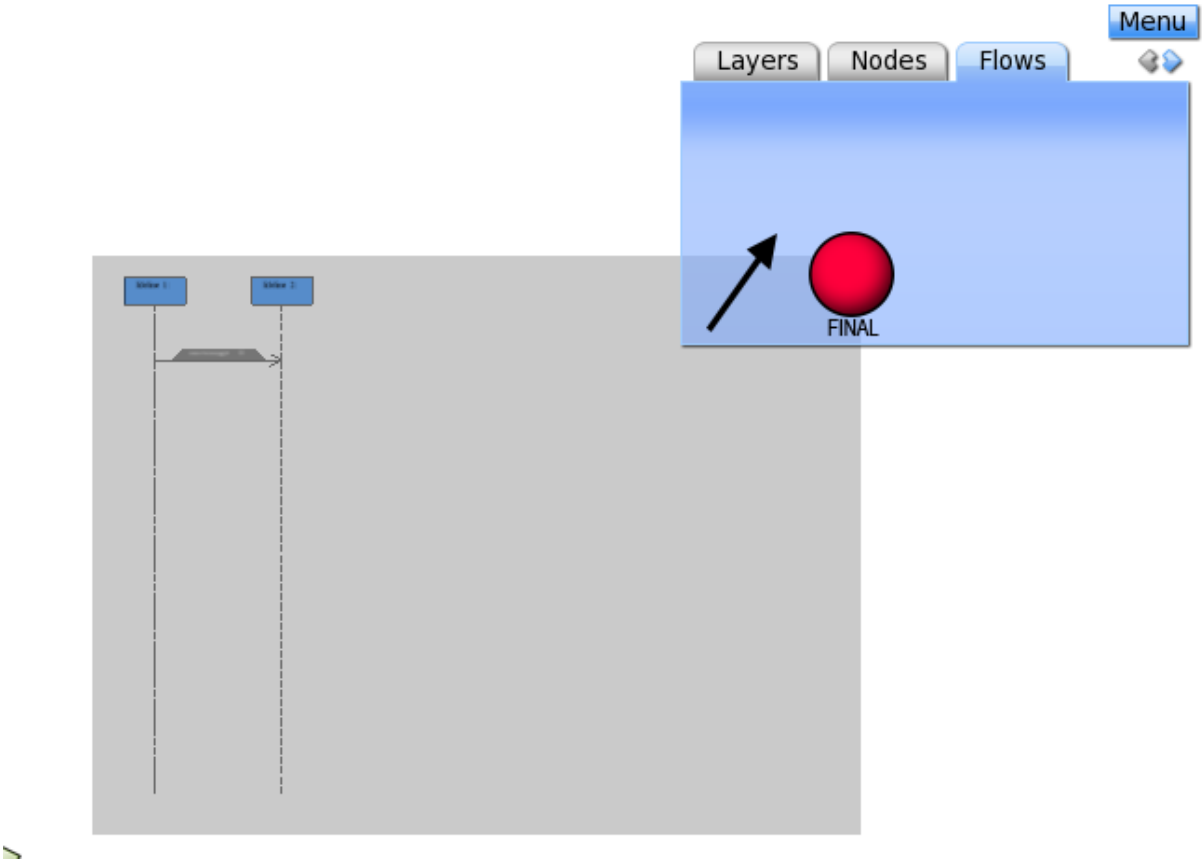


Obrázok 96: Mazanie čiary života

Čiare života je možné meniť umiestnenie na vrstve. Premiestnenie sa vykoná jej označením a následným potiahnutím na miesto, kde ju chceme presunúť. Na požadovanom mieste stačí pustiť tlačidlo myši a čiara života tam zostane.

A.2.3 Správy

Ak chce používateľ vložiť správu, musí označiť dve čiary života v poradí, v akom smere chce aby išla správa. Keď má čiary života označené, klikne na ikonu šípky, ktorá sa nachádza v záložke "Flows". Následne mu vyskočí dialógové okno, do ktorého zadá názov správy a operáciu potvrdí.

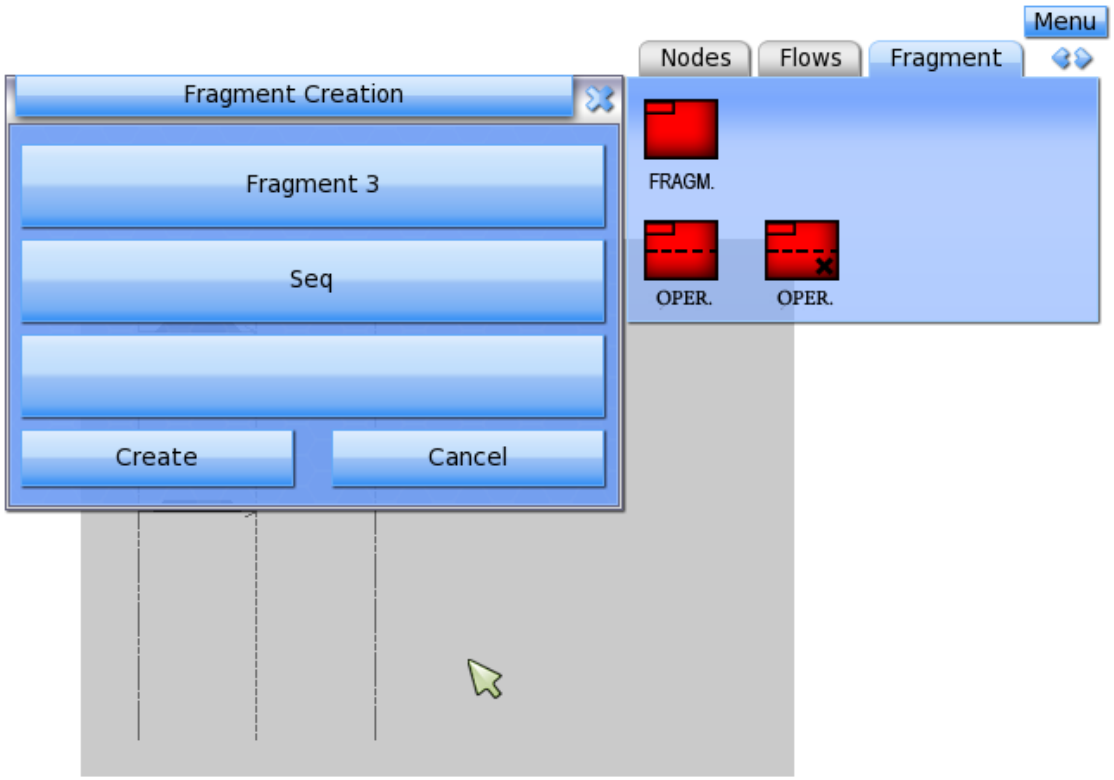


Obrázok 97: Vkladanie správy

Označenie správy sa robí podobne ako pri čiarach života. Používateľ musí kliknúť do stredu správy. Označená správa stmavne. Označenú správu môžeme presúvať zvisle alebo vodorovne medzi čiarami života. Pri mazaní správy musí mať používateľ označenú správu ktorú ide mazať. Kliknutím na pravé tlačidlo myši sa mu zobrazí potvrdzovacie okno o pre zmazanie správy.

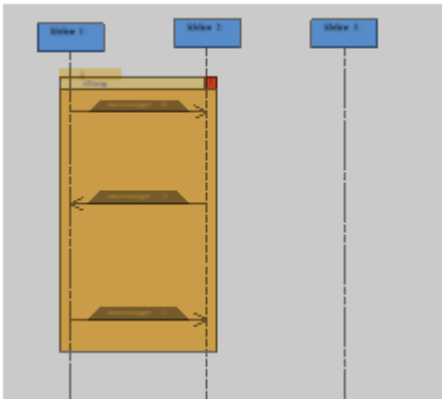
A.2.4 Kombinované fragmenty

Pri pridávaní kombinovaného fragmentu musí používateľa označiť všetky správy ktoré chce do neho vložiť. Pokiaľ sú správy zvisle nad sebou, tak stačí zvoliť prvú a poslednú správu a ostatné sa do fragmentu pridajú. Ak sú požadované správy označené, používateľ zvolí položku "FRAGM." v záložke "fragment". Následne klikne na vrstvu, na ktorú chce vložiť fragment. Po kliknutí vyskočí používateľovi okno v ktorom používateľ zadá názov fragmentu, jeho typ a podmienku ak je to potrebné.



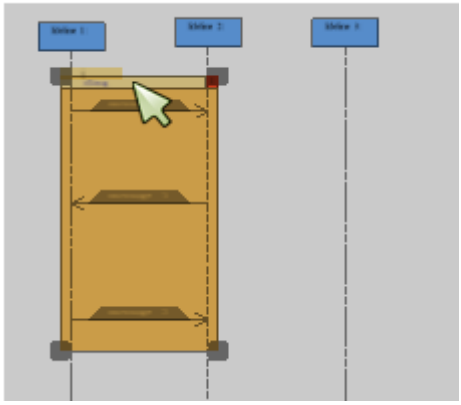
Obrázok 98: Vkladanie kombinovaného fragmentu

Po stlačení tlačidla "Create" sa vykreslí fragment.



Obrázok 99: Kombinovaný fragment

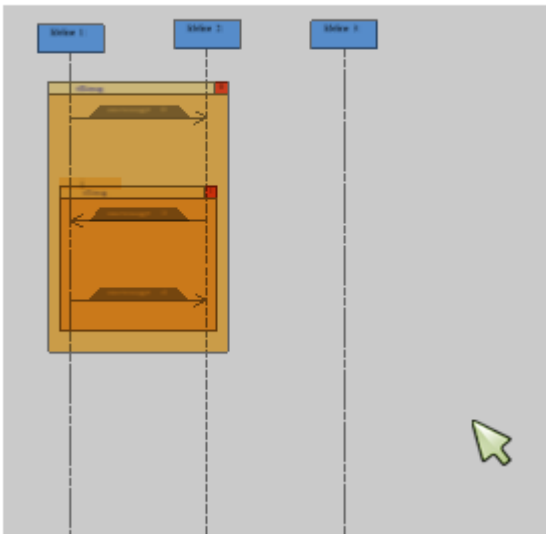
Kombinovaný fragment je možné označiť kliknutím do stredu lišty kde je vykreslený názov fragmentu. Pokiaľ sa nám po rohoch kombinovaného fragmentu vykreslia šedé štvorčeky, znamená to, že daný kombinovaný fragment je označený.



Obrázok 100: Označený kombinovaný fragment

Mazanie kombinovaného fragmentu sa vykonáva podobným spôsobom ako mazanie správ a čiar života. Pokiaľ je kombinovaný fragment označený, stačí kliknúť pravým tlačidlom myši a potvrdiť odmazanie.

Vkladanie vnoreného kombinovaného fragmentu je veľmi podobné, ako vkladanie obyčajného kombinovaného fragmentu. Používateľ označí správy ktoré sa nachádzajú v kombinovanom fragmente a postupuje rovnako, ako pri vkladaní kombinovaného fragmentu. Výsledok vyzerá nasledovne:

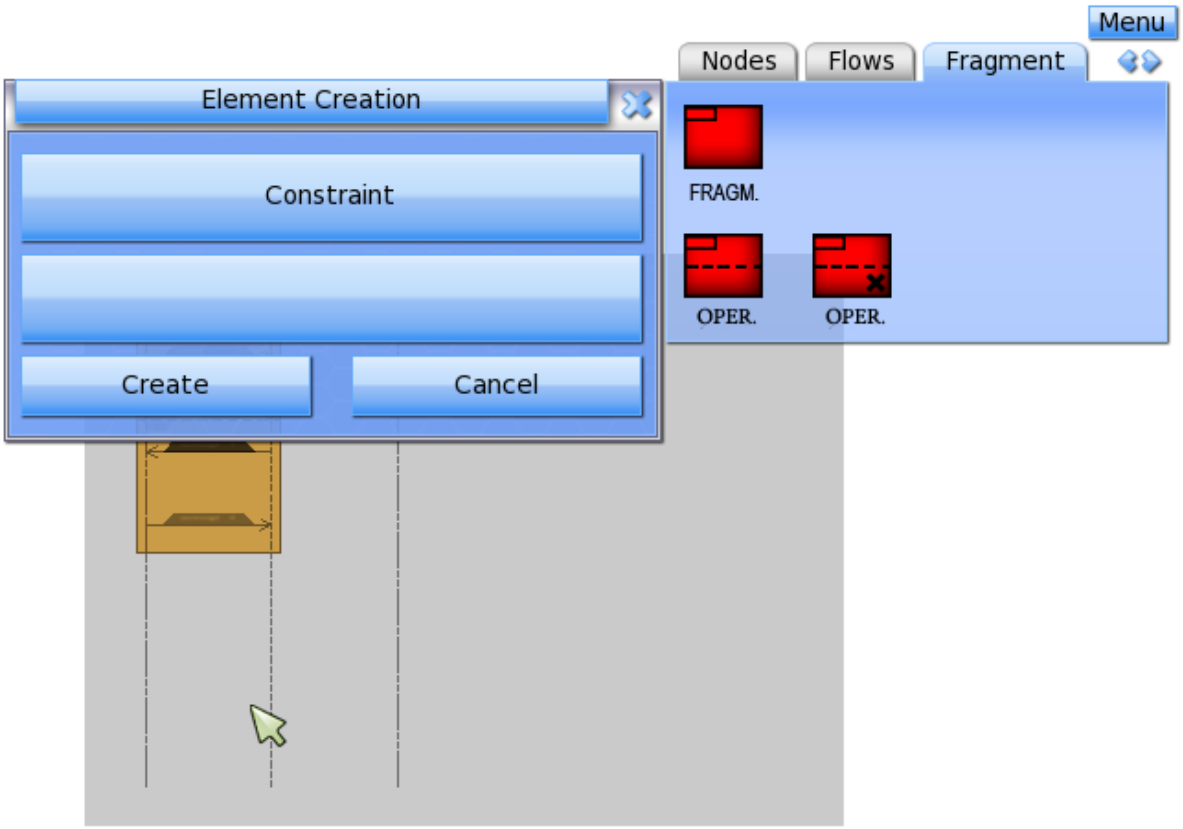


Obrázok 101: Vnorený kombinovaný fragment

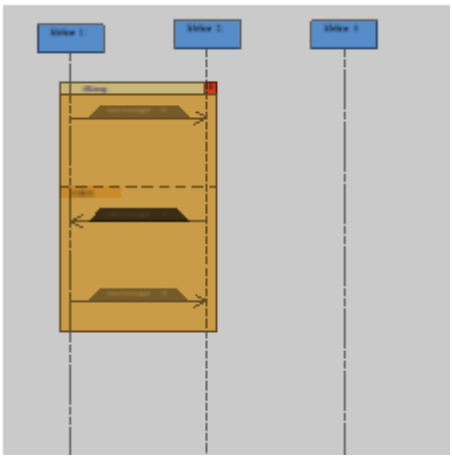
Rovnakým spôsobom ako obyčajný kombinovaný fragment môžeme odstrániť aj vnorený kombinovaný fragment.

Kombinovaný fragment je možné aj editovať a to takým spôsobom, že ho rozťahujem alebo sťahujeme. Stačí si označiť kombinovaný fragment a chytiť ho za jeden zo štyroch zobrazených štvorčekov na rohoch fragmentu. Kliknutím na štvorček a ťahaním do požadovaného smeru do neho priberáme alebo uberáme elementy v jeho okolí. Po pustení tlačidla myši sa rozťahovanie alebo sťahovanie fragmentu ukončí.

Používateľ môže do diagramu vložiť operand takým spôsobom, že vyberie element nad ktorý chce vložiť operand. Potom si v menu zvolí položku "OPER." v záložke "Fragment". Po kliknutí na položku "OPER." sa zobrazí okno v ktorom používateľ zadá podmienku. Po potvrdení tlačidlom "Create" sa zobrazí operand.



Obrázok 102: Vkladanie operandu



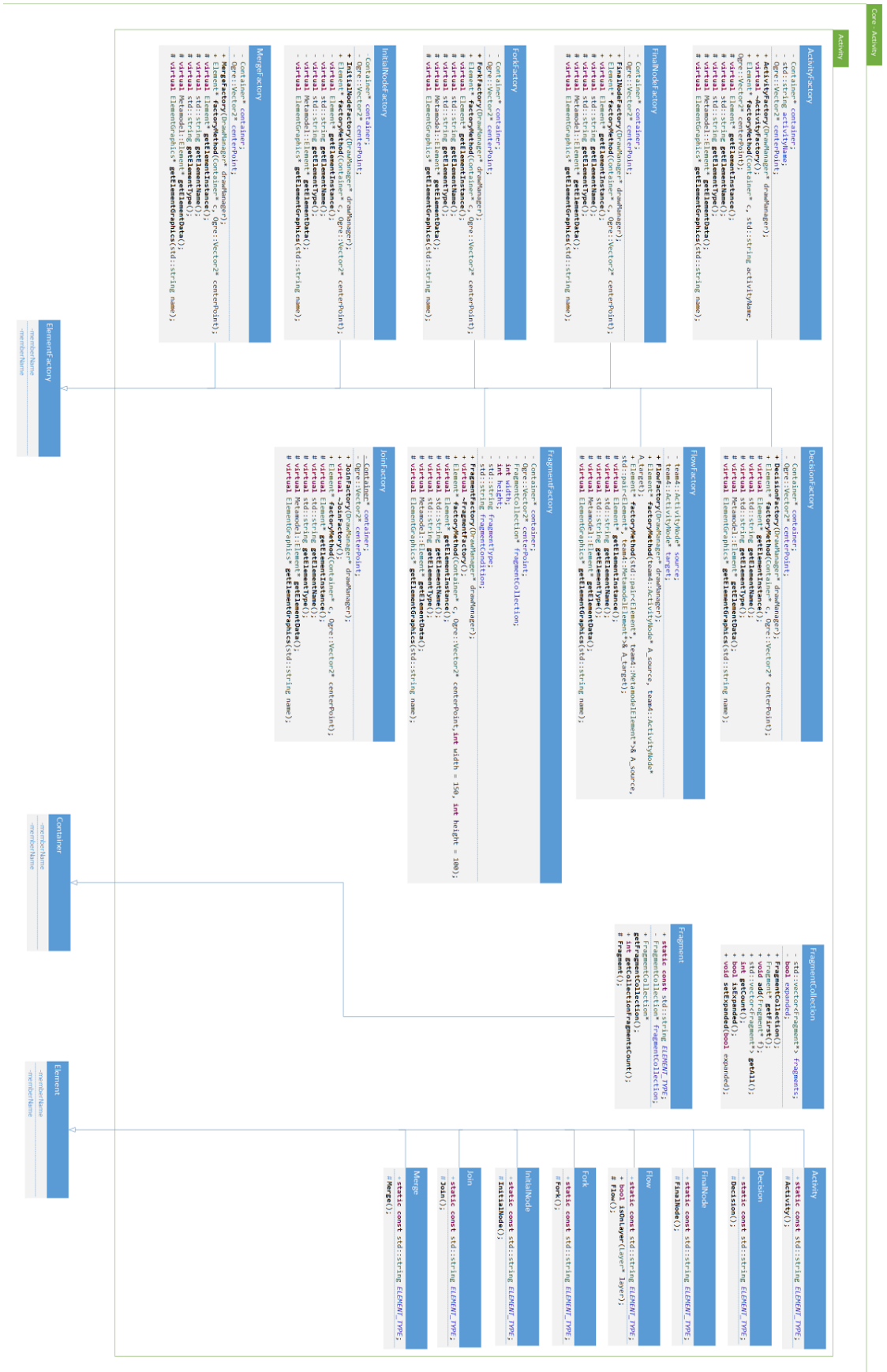
Obrázok 103: Kombinovaný fragmet s operandom

Pri mazaní operandu v kombinovanom fragmente si musí používateľ označiť kobinovaný fragment v ktorom chce mazať operand. Potom si v záložke “Fragment” zvolí položku “OPER.” s krížikom. Po kliknutí sa používateľovi zobrazí okno do ktorého zadá číslo, ktoré predstavuje poradie operandu v kombinovanom fragmente zhora na dol. Po kliknutí na tlačidlo potvrdenia sa operand odstráni.

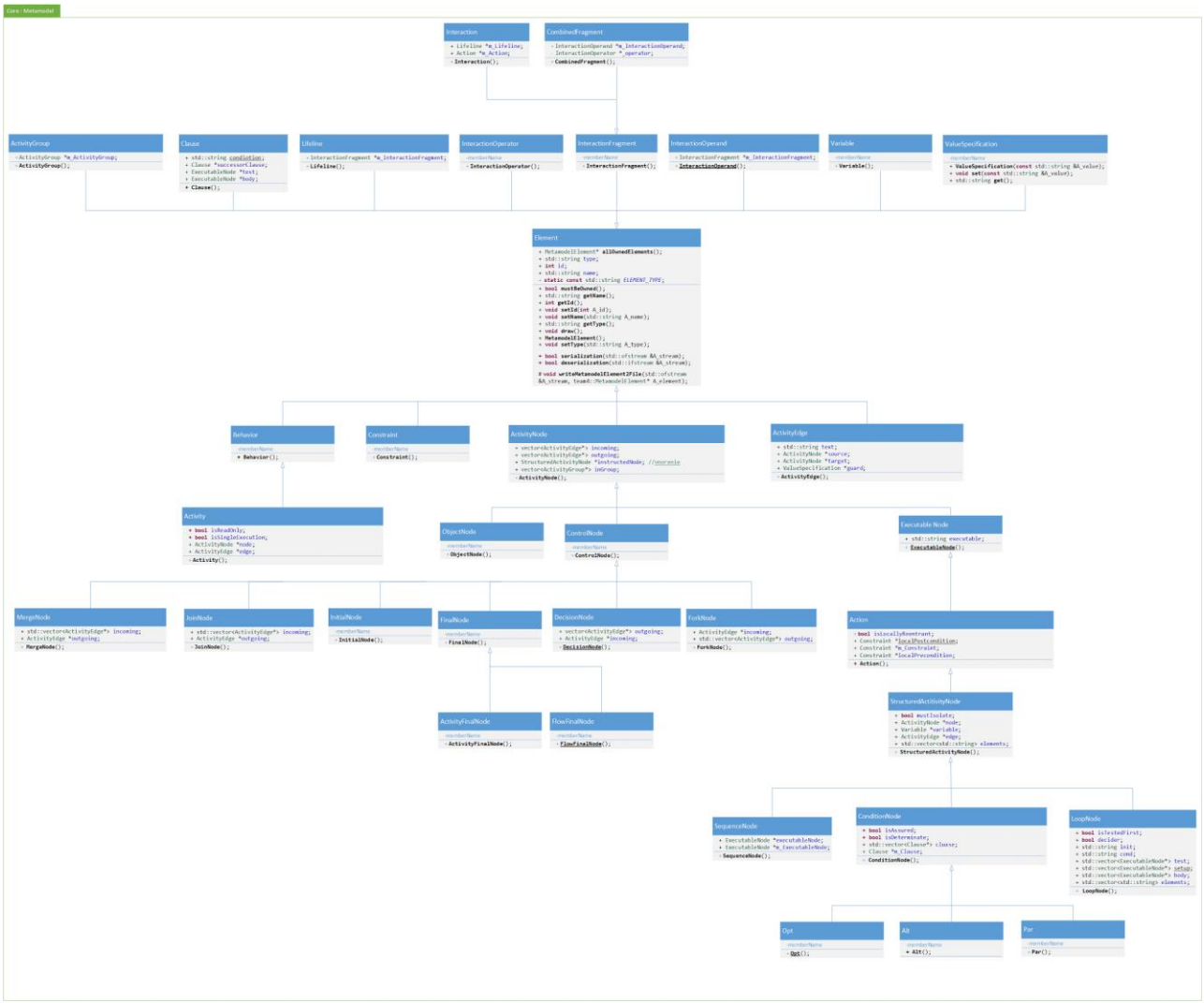
Príloha B: Technická dokumentácia

Pre lepšiu predstavu pri implementácii sme si v analytickej časti tímového projektu vytvorili diagramy pomocou Microsoft Visio[3], ktoré znázorňujú jednotlivé existujúce triedy, dedenie, premenné, metódy a prístupy k nim. Všetky vytvorené diagramy sa nachádzajú nižšie.

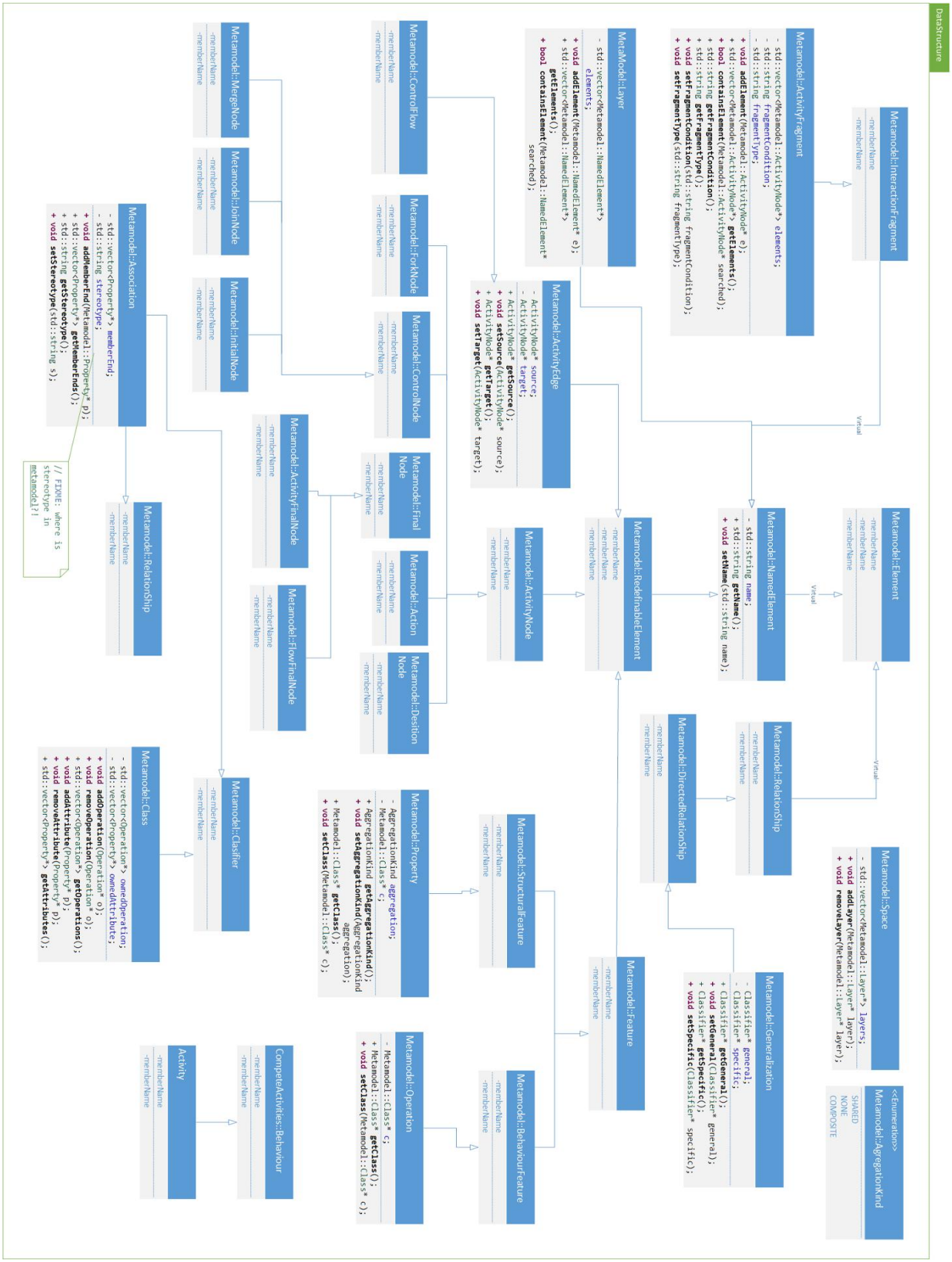
[3] https://products.office.com/en/visio/flowchart-software



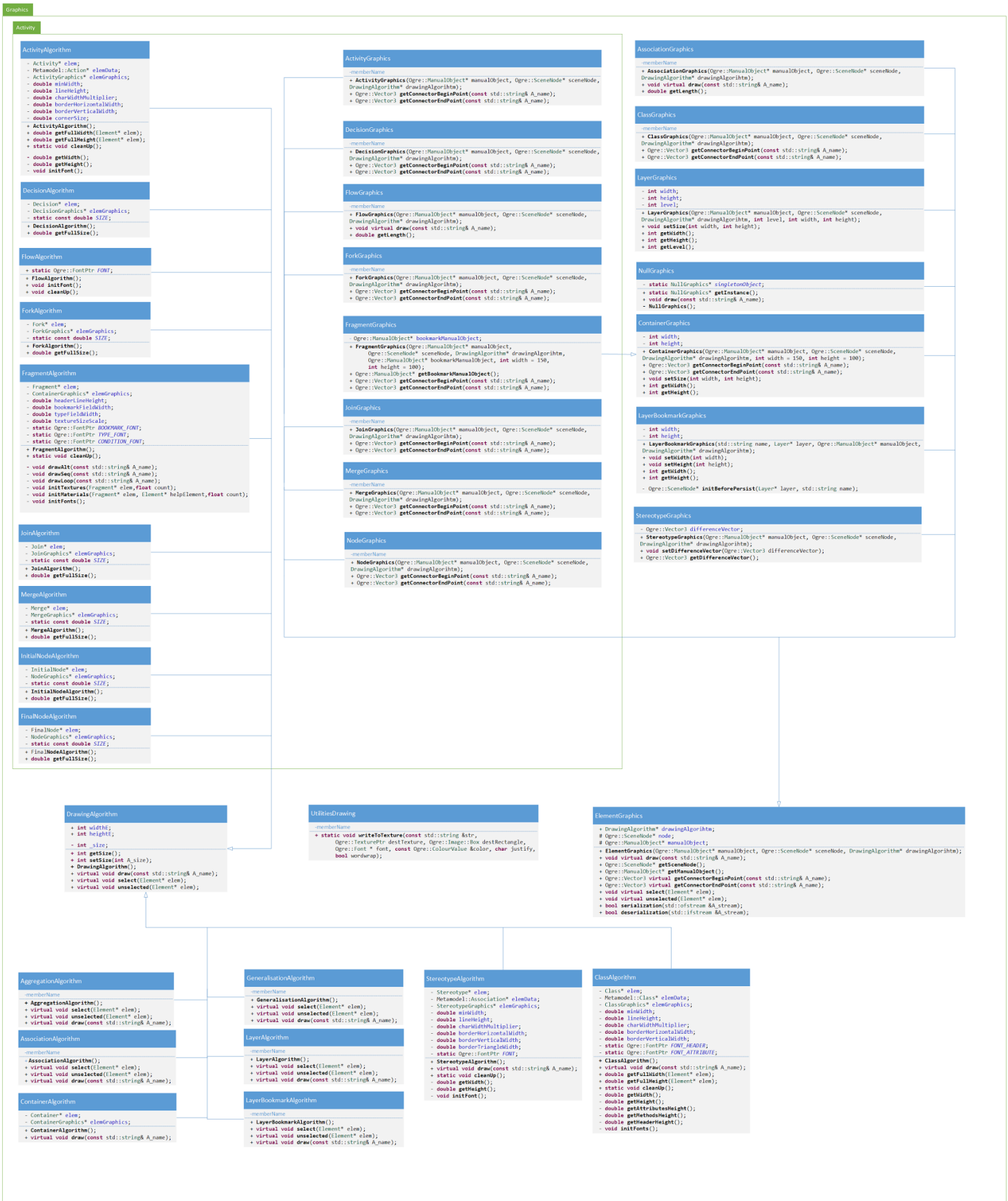
Obrázok 104: Core/Activity triedy



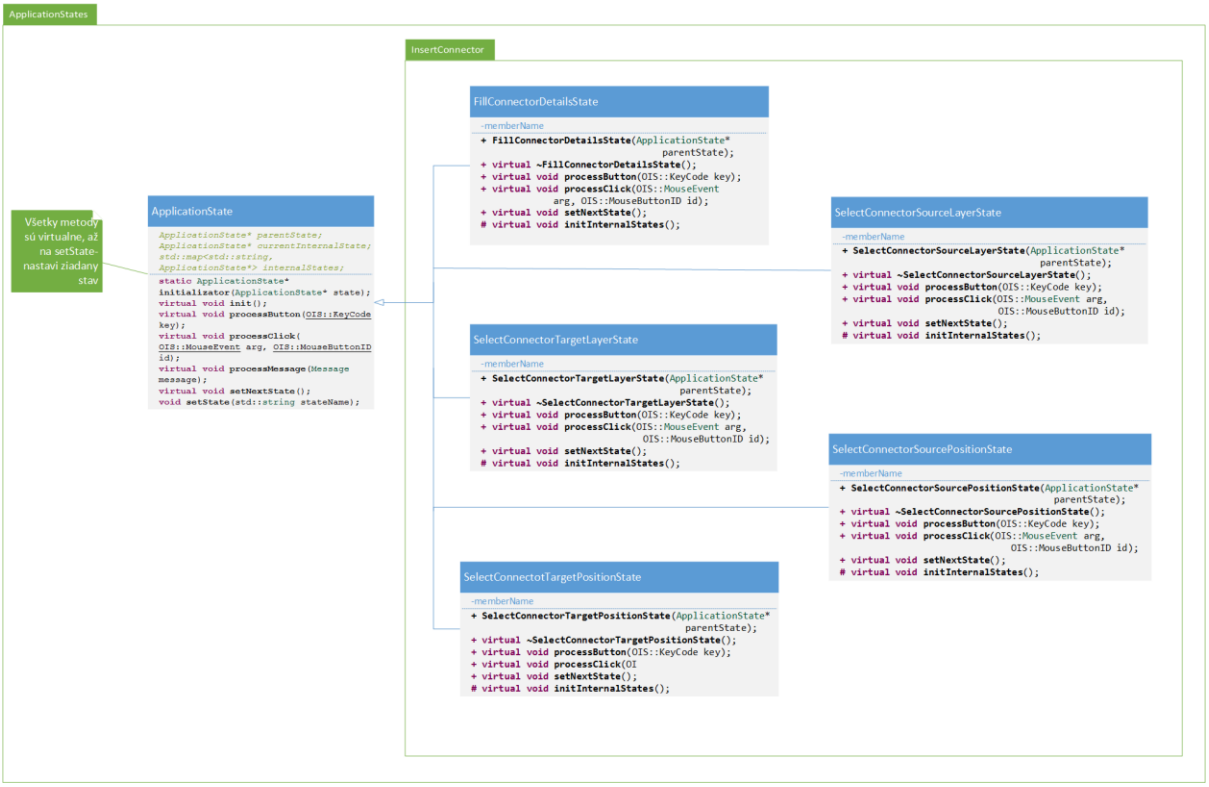
Obrázok 105: Core/Metamodel triedy



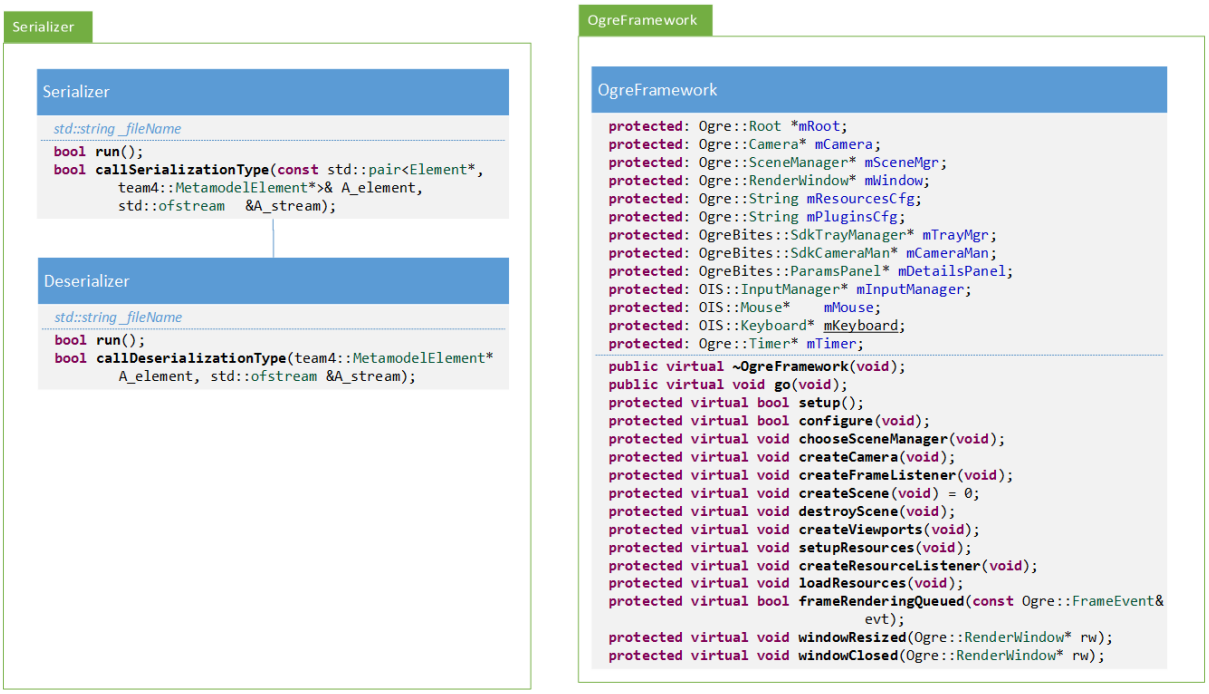
Obrázok 106: DataStructure triedy



Obrázok 107: Graphics triedy



Obrázok 108: InsertConnector triedy



Obrázok 109: OgreFramework serialization triedy