7. Prototípus koncepciója

5 – runtime_error

Konzulens:
Dobos-Kovács Mihály

Csapattagok

Mizser Ádám Zoltán	SHKGZW	mizser.adam@gmail.com
Tepliczky Olivér	WB6LC5	tepliczkyo@edu.bme.hu
Fekete Álmos Valér	KR5WPC	feketealmos0@gmail.com
Váradi Kristóf	BP17IB	kristofvaradi@edu.bme.hu
Sasvári Szabolcs Attila	**TWOZG6**	**szabolcs.attila.sasvari@edu.bme.hu**

2023.04.24.

7. Prototípus koncepciója

7.0 Változás hatása a modellre

*A változásokat **vastagon szedett** szövegekkel jelöljük a dokumentumban, hogy kitűnjenek a meglévő leírásokból.*

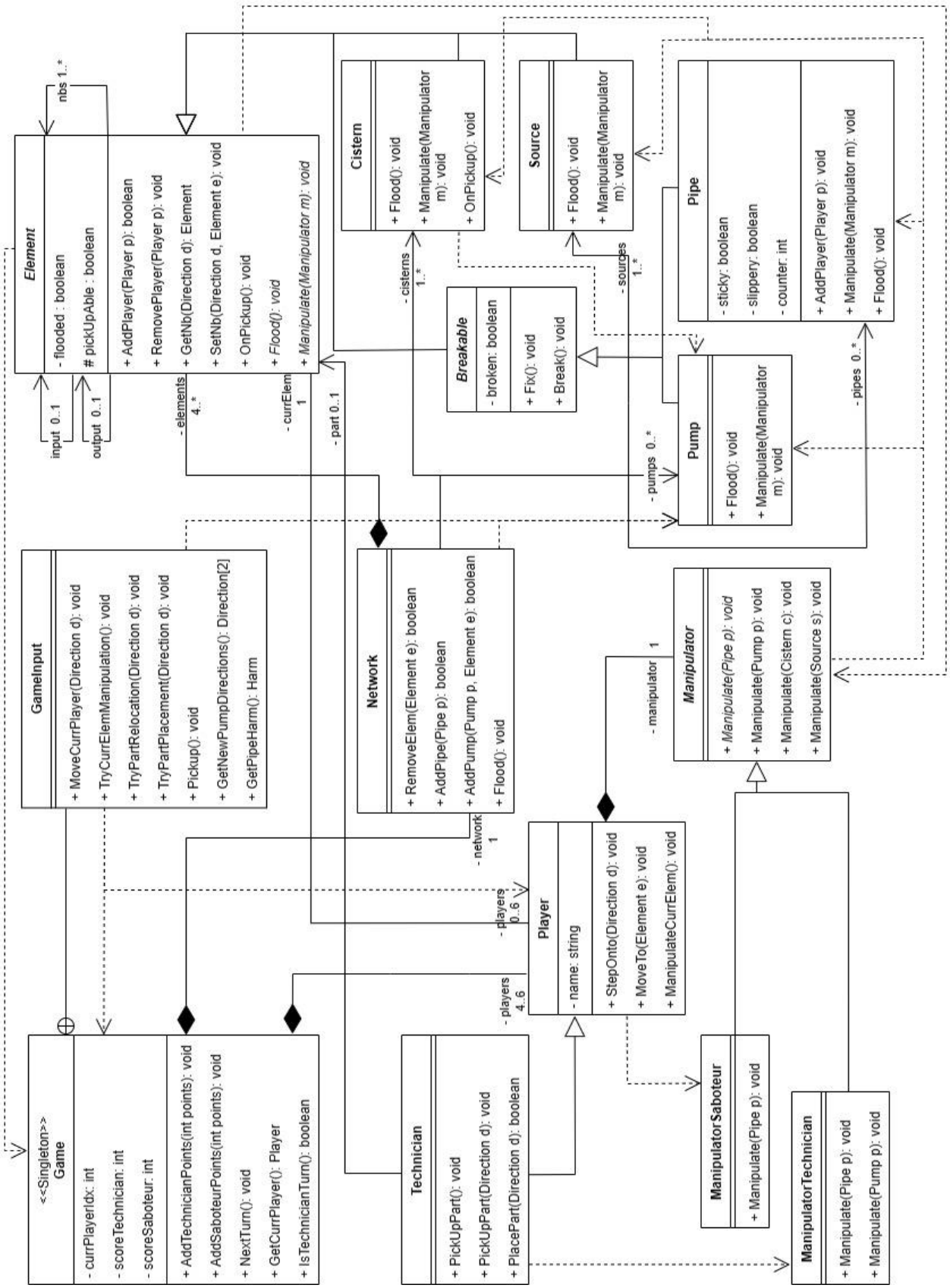
A változások listája, és referenciák a dokumentumban, hogy hol és hogyan orvosoljuk:

- *Szerelő is tud lyukasztani.*
  - **lásd 7.0.2.2**
- *Foltozott cső véletlen hosszúságú ideig nem lyukasztható ki.*
  - **lásd 7.0.2.1**
- *A csöveknek mindkét vége egyidőben le lehet csatolva.*
  - Mivel nálunk a csövek „tile” szerűek (két aktív elemet összekötő „cső” is csövekből áll, csövek is lehetnek egymás mellett), ezért ez a módosítás furcsa módon valójában könnyítésként érint minket.
    Eddig volt olyan követelményünk, hogy ez nem fordulhatott elő, de így, hogy a megrendelő kérésére ez már előfordulhat, most már nem lehet megakadályozni, hogy a játékosok „lecsatolják magukat” a pályáról, és a pálya, mint gráf komponenseire essen. **Innentől kezdve a játékosok felelősége lesz, hogy ne idézzenek elő ilyen helyzetet**.
- *A szabotőr azt a csövet, amin áll, rövid időre csúszóssá tudja tenni. Ilyenkor aki rálép, véletlenszerűen a cső valamelyik végéhez kapcsolódó elemre kerül.*
  - **lásd 7.0.2.1 és 7.0.2.3**
- *Mind a szabotőrök, mind a szerelők azt a csövet, amin állnak, rövid időre ragadóssá tudják tenni. Aki legközelebb rálép, egy ideig nem tud továbblépni.*
  - **lásd 7.0.2.1 és 7.0.2.2**

Egyéb változások:

- *Player (Saboteur) → Technician downcast szükségességének megszüntetése*
  - **Egy üres törzsű SetPart() és egy null-t visszaadó GetPart() függvény felkerül Player-be**, amit a jelenlegi Technician.SetPart() és GetPart() felülír.
    A szabotőr így nem tárol feleslegesen, és downcast-ra sem lesz szükség.
- *Element → Breakable downcast szükségességének megszüntetése*
  - **Technician nem Breakable-t, hanem Element-et tárol.**
    A Manipulator Dynamic Dispatch-ének hála nem lesz más komplikáció.

7.0.1 Módosult osztálydiagram



7.0.2 Új vagy megváltozó osztályok és metódusaik

7.0.2.1 Pipe

- ***Felelősség***

Olyan elromolható, felvehető elem, amelyen legfeljebb egy játékos tartózkodhat. Ki lehet lyukasztani, a lyukas csövet pedig meg is lehet javítani, ***ekkor egy (véletlenszerű) ideig nem lehet kilyukasztani ismét****. Ha a cső nem lyukas akkor a kimenetéhez vizet tud juttatni.*
Ha lyukas, vagy a kimenete üres, akkor a szabotőrök pontot kapnak, amikor víz érkezik belé.
Ha nem lyukas, akkor csúszóssá vagy ragadóssá lehet tenni egy időre.
Ha csúszós a cső, akkor aki rálép véletlenszerűen az egyik végére csatlakoztatott elemre csúszik át. Ilyenkor se lyukasztani, se ragadóssá tenni nem lehet az adott csövet.
Ha ragadós a cső, akkor aki rálép, annak a köre véget ér. Ebben az esetben se lyukasztani, se csúszóssá nem lehet tenni az adott csövet. Egy cső általában kevesebb ideig ragadós, mint csúszós.

- ***Attribútumok***
  - ***- sticky: boolean****: Változó amely a ragadósság állapotát tárolja.*
  - ***- slippery: boolean****: Változó amely a csúszósság állapotát tárolja.*
  - ***- counter: int****: egész típusú változó, amely egy visszaszámláló. Ez az éppen aktív tulajdonsághoz tartozik, vagy hogy meddig nem lehet kilyukasztani ismét a csövet.*
- ***Metódusok***
  Az új attribútomokhoz kapcsolódnak ***getterek és setterek****, amiket nem sorolunk fel külön, és nem jelölünk az osztálydiagrammon.*

7.0.2.2 ManipulatorTechnician

- ***Metódusok***
  - ***+ void Manipulate(Pipe p)*** *:* ***Ellenőrzi, hogy az adott cső éppen lyukas-e vagy sem. Ha lyukas megjavítja az átadott p csövet. Ha nem lyukas bekéri, hogy mit szeretne vele csinálni (ragadóssá tenni vagy kilyukasztani)****. Ezután pedig véget ér a jelenlegi játékos köre.*

7.0.2.3 ManipulatorSaboteur

- ***Metódusok***
  - ***+ void Manipulate(Pipe p)*** *:* ***Ellenőrzi, hogy az adott cső éppen lyukas-e vagy sem. Ha nem lyukas bekéri, hogy mit szeretne vele csinálni (csúszóssá tenni, vagy ragadóssá, ha egyik sem, akkor automatikusan kilyukasztja a csövet)****. Ezután pedig véget ér a jelenlegi játékos köre.*

7.0.2.4 GameInput

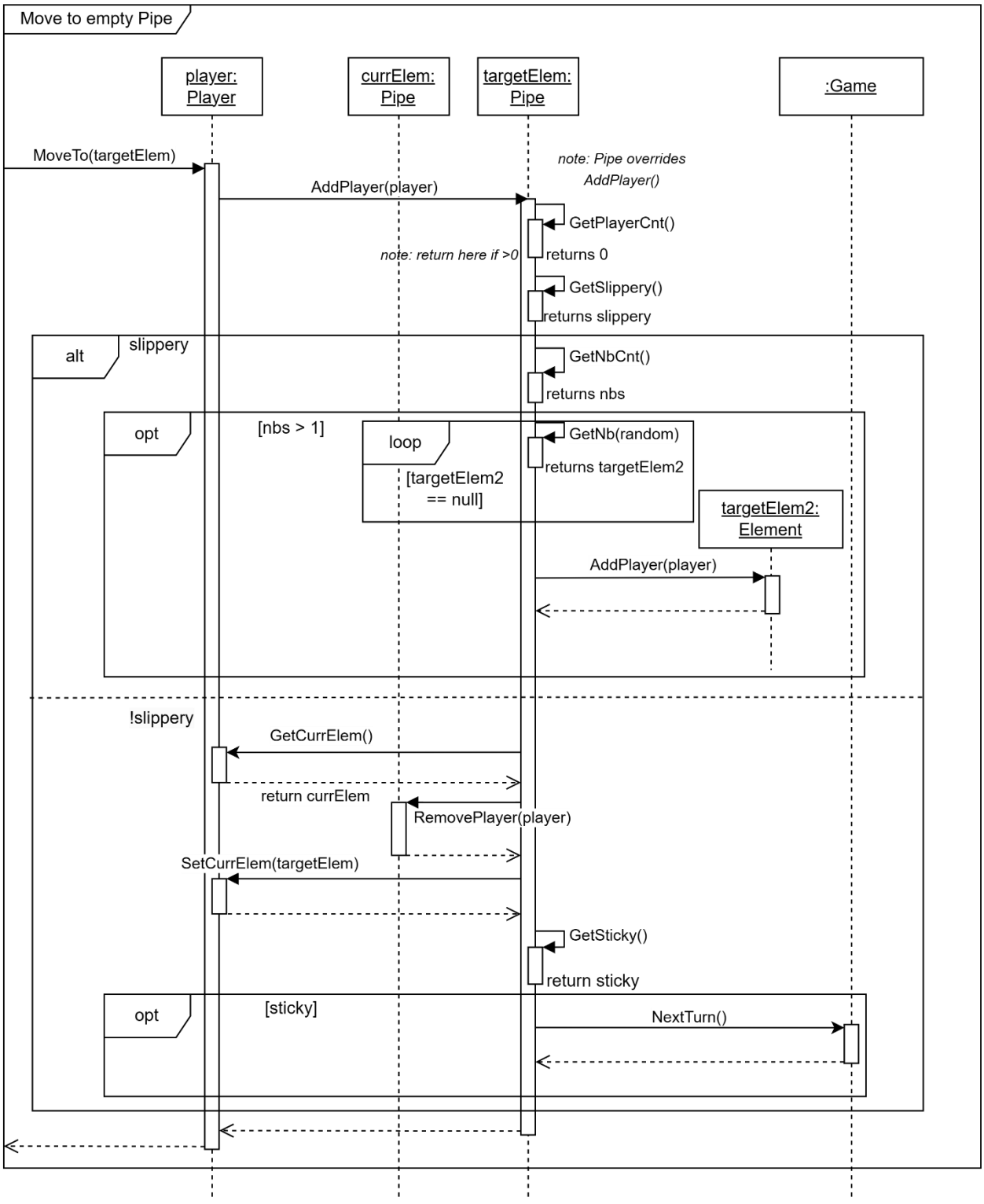
- ***Metódusok***
  - ***+ GetPipeHarm(): Harm****: Bekéri és visszaadja a soron lévő játékostól, hogy milyen módon akar kárt okozni a csövön, amin áll.*
    Harm egy enumeráció, amelynek lehetséges értékei: SLIPPY, STICKY, BROKEN
    Ezt a függvényt fogják hívni a manipulátorok Manipulate(Pipe p) függvényei.

7.0.2.5 Network

- ***Attribútumok***
  - ***- pipes: Pipe[0..*]****: Külön csoportosítja minden elem közül a csöveket ez a gyűjtemény abból a célból, hogy a Game osztály* ***NextTurn()*** *függvénye be tudja őket járni, és le tudja kérdezni mindegyik csőnek a counter-jét, és csökkenteni tudja azt, ha nem nulla. Ha pont nullára változik, akkor az adott cső sticky és slippy attribútumát false-ra állítjuk a megfelelő setter-ekkel.*

7.0.3 Szekvencia-diagramok

7.0.3.1 Move to empty pipe

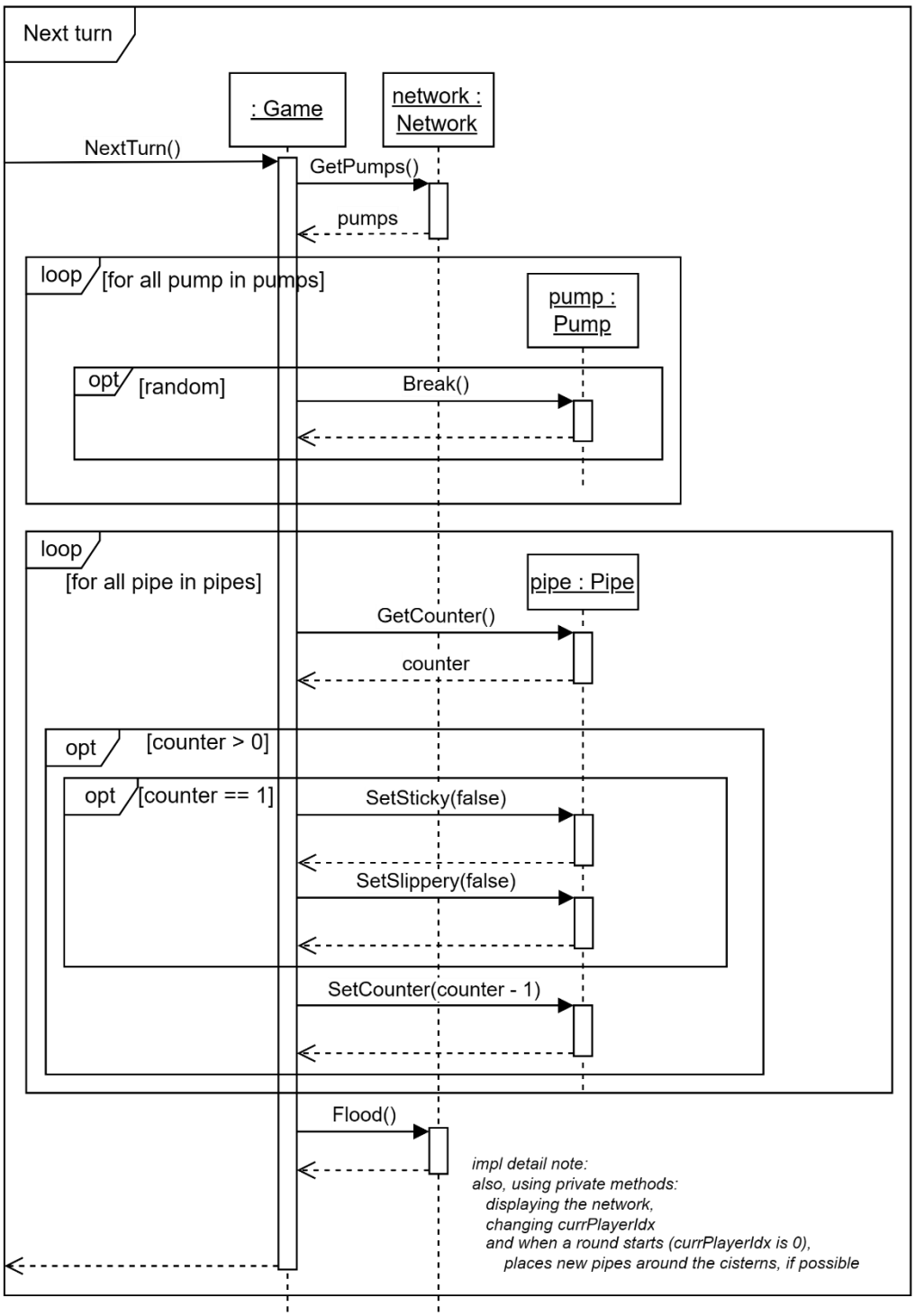


(3.4.2, 3.4.3 és 5.3.1 diagrammok végleges formája)

7.0.3.2 Relocate (un)relocatable pipe

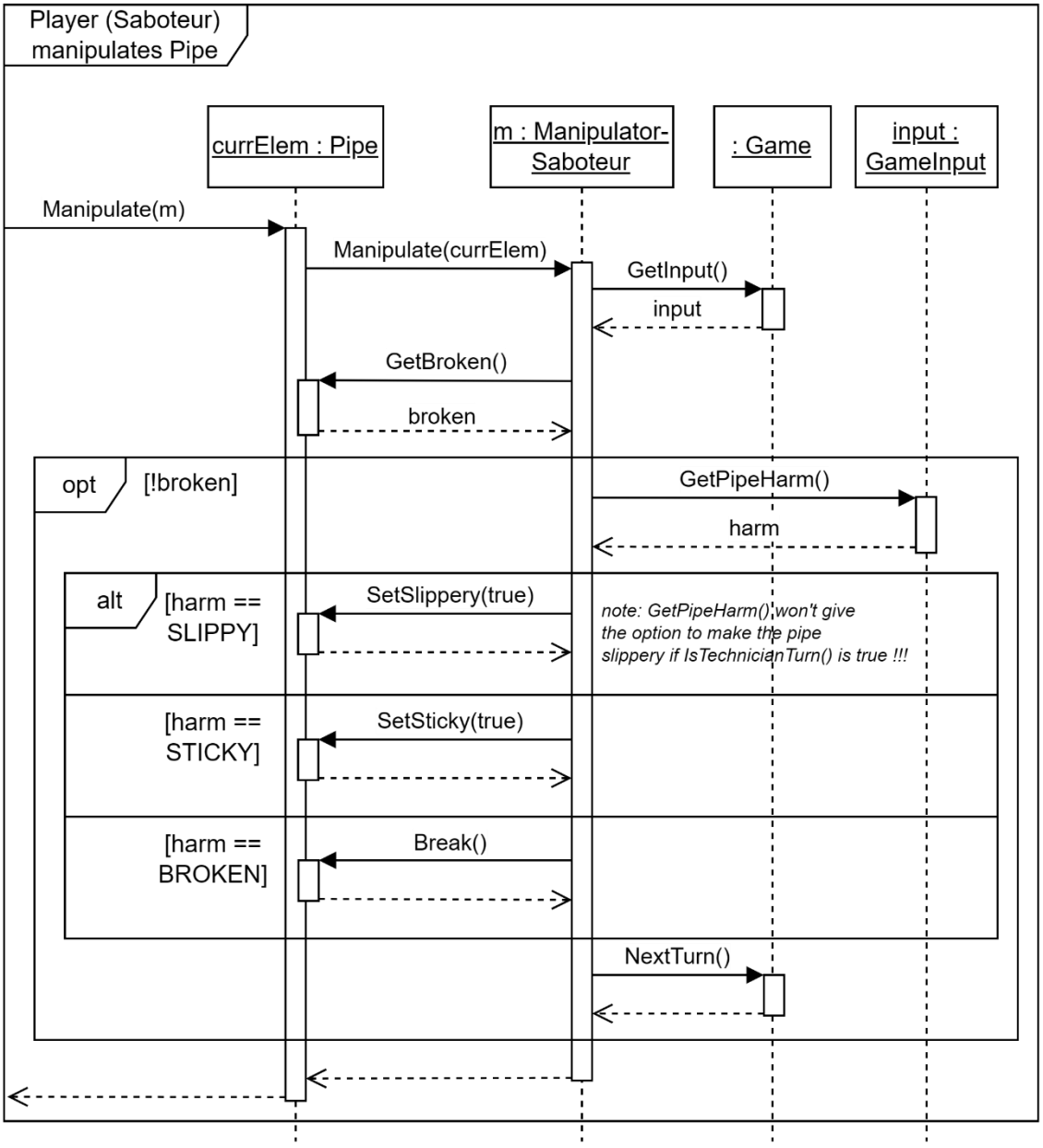
3.4.12, 5.3.14 és 5.3.15-ös diagrammokban csak annyi változik, hogy már nem szükséges vizsgálni, hogy a célpont csőnek legyen 2 szomszédja (már teljesen le lehet csatolni).

7.0.3.3 Next turn



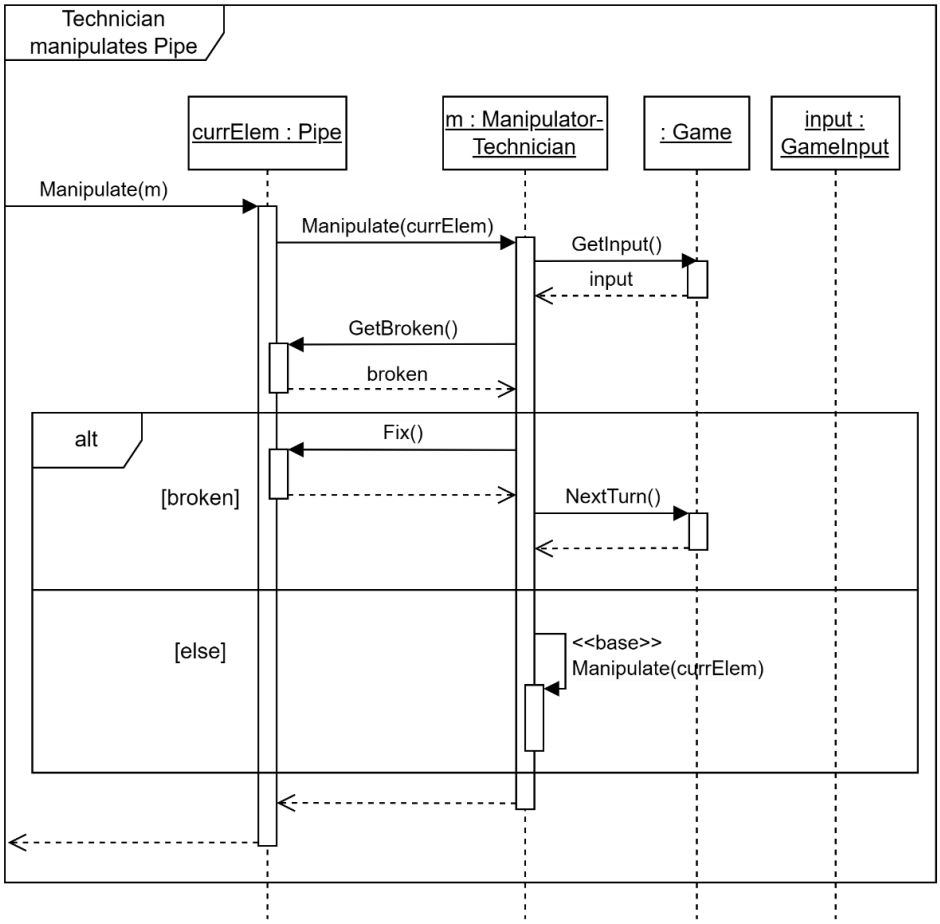
(3.4.2, 3.4.3 és 5.3.1 diagrammok végleges formája)

7.0.3.4 Player (Saboteur) manipulates Pipe



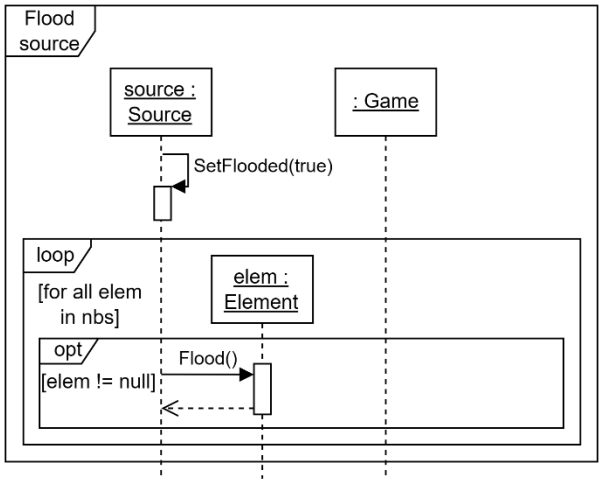
(Ez egy új diagram: több korábbi diagramot egyesít.
Figyelem: SLIPPY és STICKY esetén a privát counter változót is beállítja egy véletlenszerű, 0-tól nagyobb értékre. STICKY esetén a „rövid időre” megfogalmazás miatt az osztható tartomány valamennyivel kisebb számokból van, de ez implementációs részlet.)

7.0.3.5 Technician manipulates Pipe



(ez a diagram nem változott, konzisztencia céljából tüntetjük itt fel: működik az előzővel (a <<base>> hívás hatása az előző diagramról leolvasható))

7.0.3.6 Flood source



(A források elárasztása csak annyiban változik, hogy csak a rákötött bemeneteikbe eresztenek vizet. Mivel nem korlátolt az irányok száma a prototípusban, ellenkező esetben végtelen pontot osztana minden elképzelhető irányba, amely felé nincsen szomszédja. A ciszterna termést viszont nem változtatjuk, az első 4 irányba teremnek csak csövek a ciszterna körül.)

7.1 Prototípus interface-definíciója

7.1.1 Az interfész általános leírása

Szempontok: minél kevesebb parancs legyen, de fedjenek le mindent, ami a játék előkészítéséhez és játszásához szükséges.

A parancsok beolvasása történhet konzolról és bemeneti fájlokból is. A parancsok kimenete szintén. Ezek megadására két szöveges fájl szolgál:

- **input.txt**: ha üres, konzolról történik a parancsok beolvasása. Ha meg vannak adva benne további relatív elérési utak pl. test/test1.txt, test/test2.txt *soronként*, akkor az azoknak megfelelő tesztesetek parancsait is beolvassa, és végrehajtja.
- **output.txt**: ha üres, a konzolra kerül a parancsok kimenete. Ha meg vannak adva benne további relatív elérési utak pl. output/output1.txt, output/output2.txt *soronként*, akkor a megfelelő input fájlok parancsai a hozzájuk tartozó sorban lévő output fájlba lesznek írva. Fontos tehát, hogyha fájlokat adunk meg input.txt-ben, akkor mindegyikhez megadjuk egy kimeneti fájlt is output.txt-ben (ugyanannyi sorból kell állniuk!).

7.1.2 Bemeneti és kimeneti nyelv

7.1.2.1 Parancsok a játék (pálya) előkészítéséhez

megjegyzés: a kezdeti állapotban a *Network* osztály az *elements* gyűjteményében egyetlen forrást tárol a 0. indexen. Minden hozzáadott elem sorban a következő indexre kerül.
Úgy lehet felvenni új elemeket, hogy egy meglévőtől a megadott irányba helyezzük le.
Az elemet, amelyik mellé le szeretnénk helyezni egy újat, ezzel az indexével jelöljük ki.
Ennek az indexnek a számontartása a tesztelő feladata, de könnyíti a helyzetét az, hogy az elemhozzáadó parancsok kimenetében látszik a hozzáadott elem indexe. Alternatívan a pálya kiírásából is ki tudja olvasni ezeket az indexeket.

`add <elem_type> <nb_elem_nr> <dir_nr>`

Leírás: a megadott sorszámú elemtől a megadott irányba lehelyez egy új elemet

Opciók: **`elem_type:`** `pipe`, `pump`, `source` vagy `cistern`
figyelem: a tesztelőnek figyelnie kell, hogy 2 aktív elem ne legyen egymás mellett!
`nb_elem_nr`: az elem indexe *elements* gyűjteményben, amely mellé szeretnénk helyezni az új elemet
(az új elem indexe is meg fog jelenni a kimenetben)
figyelem: ha nincs ilyen indexű elem, akkor nem fog semmi történni (ez igaz a további parancsokra is.
`dir_nr`: egész szám, ennek megfelelő irányú szomszédja lesz az új elem

Kimenet: **`a pipe/pump/source/cistern was added as element <elem_nr>`**

`add <player_type> <name> <elem_nr>`

Leírás: egy megadott nevű új játékost helyez le a megadott sorszámú elemre

Opciók: **`player_type:`** `saboteur` vagy `technician`
figyelem: a tesztelőnek figyelnie kell, hogy csak felváltva legyenek hozzáadva játékosok a játékos típusokból, és MUSZÁJ szerelővel kezdeni a játékosok hozzáadását. (Más különben a currPlayerIdx paritását ellenőrző *IsTechnicianTurn()* függvény működése helytelen lesz.)
Pl. `technician`, `saboteur`, `technician`, `saboteur` egy jó sorrend
Ezen kívül már csak egy helyes sorrend van, az előző kiegészítve még kettő játékossal: `technician`, `saboteur` (lásd D14 és D26 követelmények – minimum 2-2 és maximum 3-3 játékos játszhat)
`name:` a játékos neve (célszerű használni a „t” és „s” prefixeket, hogy felismerhető legyen szerelőről vagy szabotőrről van szó). Egyes parancsok esetén és a kimenetben ezzel lehet majd azonosítani a játékost.
`elem_nr:` az elem indexe *elements* gyűjteményben, amelyre rá szeretnénk tenni a létrehozandó játékost.
figyelem: a tesztelőnek figyelnie kell, hogy egy csőre ne tegyen több mint egy játékost.

Kimenet: **`technician/saboteur <name> was added to element <elem_nr>`**

`controller toggle random`

Leírás: Ki-/bekapcsol minden véletlenszerű viselkedést a modellben, és helyettük a definiált (és módosítható) értékeket használja, hogy determinisztikus teszteseteket állíthassunk elő. Az alábbiakra van hatással:

- Pumpák elrontása a körök elején (manuálisan a **`controller break <elem_nr>`** paranccsal kell megadni majd a controller általi pumpaelrontást helyette, ha szeretnénk ilyet).
- A körök száma, ameddig egy cső csúszós és ragadós maradhat. Amennyiben ki van kapcsolva a véletlenszerű viselkedés, a **`controller set defaultCounter <turn_cnt>`** értékkel állítható ez be. Ilyenkor mindig a megadott számú körök erejéig maradnak csúszósak és ragadósak a csövek.
- Csúszós cső esetén, hogy melyik irányú szomszédja felé mozgatja el a játékost. Kikapcsolt véletlenszerű viselkedés esetén mindig a legelső sorszámú irányba fogja, amerre van szomszédja az adott csőnek.

Kimenet: **`random behaviour was turned off/on`**

`controller break <elem_nr>`

Leírás: Egy controller általi elemelrontást csinál, azaz olyat, ami nem eredményezi a jelenlegi játékos körének a végét, és így a víz folyására az nem is lesz hatással a következő kör elejéig.

Opciók: **`elem_nr:`** az elrontani kívánt elem sorszáma (indexe az *elements* gyűjteményben)

Kimenet: **`<elem_nr> was broken by controller`**

`controller set defaultCounter <turn_cnt>`

Leírás: Beállítja, hogy kikapcsolt véletlenszerű viselkedés esetén hány körig (turn) maradjanak csúszósak és ragadósak a csövek.

Opciók: **`turn_cnt:`** ennyi kör után fogják elveszteni a csúszós és ragadós állapotukat a csövek.
figyelem: a tesztelő felelősége, hogy ne adjon meg negatív számot.

Kimenet: **`defaultCounter was set to <turn_cnt>`**

`load state <rel_file_name>`

Leírás: A megadott fájlból beolvassa, és végrehajtja a parancsokat. Használható csak a játék előkészítésére, de komplett teszteset ellenőrzésére is.

Opciók: **`rel_file_name:`** a fájl relatív elérési útvonala, amelyikből szeretnénk beolvasni az új állapotot.
figyelem: a tesztelő felelősége, hogy létező útvonalat adjon meg, amelyen egy megfelelő parancsokat tartalmazó és formátumú fájl van.

Kimenet: **`loading state from <rel_file_name>`**
`<a fájlban lévő parancsoknak megfelelő sorok...>`

`reset state`

Leírás: Alapállapotba (egyetlen forrás, nem lesznek játékosok, stb.) állítja az egész játékot. A **`load state <rel_file_name>`** parancs előtt nincs szükség kiadni ezt a parancsot, mert alapjáraton megteszi ezt.

Kimenet: **`game state was reset`**

`start game`

Leírás: Elkezdi a játékot (A játékosok felváltva kerülnek majd sorra, minden kör elején folyik a víz, stb.).
figyelem: a tesztelő felelősége, hogy a parancs kiadása előtt érvényes kezdeti feltételekkel rendelkező pályát alkosson (pl. megfelelő számú, és sorrendben hozzáadott játékosok legyenek).

Kimenet: **`game started`**
A parancs kiadása után minden kör (turn) elején kiírja a következőt:
`it's <name>'s turn! (Points: saboteurs: <sab_points>, technicians: <tech_points>`

7.1.2.2 Parancsok a játék játszásához

A **`start game`** parancs kiadása után mindig a soron lévő játékost lehet irányítani. A következő parancsokat mindig a soron lévő játékos hajtja végre.

`move <dir_nr>`

Leírás: A megadott irányba próbál lépni a jelenlegi eleméről a játékos a hatására. Ha az adott irányban nincsen szomszédos elem, vagy olyan cső van, amelyen már állnak, akkor nem történik semmi.
Ha ragadós csőre lépett, akkor véget ér a köre, ellenkező esetben viszont ebből a típusú interakcióból bárhányat végrehajthat a játékos.
Ha csúszós csőre lépett, akkor véletlenszerűen egy másik szomszédjára (vagy ha ki van kapcsolva a véletlenszerű viselkedés, akkor a legkisebb sorszámú szomszédos elemére) tovább mozgatja a játékost.

Opciók: **`dir_nr:`** egész szám, irány amerre a jelenlegi elemről tovább szeretne lépni a játékos (ilyen irányban lévő szomszédjára kísérel lépni, ha van ilyen).

Kimenet: **`player <name> moved to element <elem_nr>`**
`(csúszós csövek esetén még egyszer vagy többször)`

`manipulate`

Leírás: Ez a parancs szolgál arra, hogy a játékos manipulálja a képességeinek és az adott elemnek megfelelően azt az elemet, amelyen éppen áll.

szabotőr	cső	ha törött, nem csinál semmit különben: proto bekéri, hogy mit szeretne tenni, választási lehetőségek (parancsok): • **`stickify`**: ragadóssá teszi a csövet • **`slippify`**: csúszóssá teszi a csövet • **`break`**: kilyukasztja a csövet
	pumpa	átállítja a bemeneteit felszólításra a következő paranccsal: **`change input <dir_nr> output<dir_nr>`** (az új be- és kimeneti irányokat kell megadni)
	ciszterna	átkerül a következő ciszternára *(köre nem ér véget)*
	forrás	-
technikus	cső	ha törött, megjavítja különben: proto bekéri, hogy mit szeretne tenni, választási lehetőségek (parancsok): • **`stickify`**: ragadóssá teszi a csövet • **`break`**: kilyukasztja a csövet
	pumpa	ha törött megjavítja, különben átállítja a bemeneteit felszólításra a következő paranccsal: **`change input <dir_nr> output <dir_nr>`** (az új be- és kimeneti irányokat kell megadni)
	ciszterna	átkerül a következő ciszternára *(köre nem ér véget)*
	forrás	-

Kimenet: Értelemszerűen az alábbiak közül valamelyik:

```
element <elem_nr> pipe stickified by <name>
element <elem_nr> pipe slippified by <name>
element <elem_nr> pipe broken by <name>
element <elem_nr> pipe fixed by <name>
element <elen_nr> pump fixed by <name>
element <elem_nr> pump new input <dir_nr> and output <dir_nr>,
                                          change made by <name>

<name> crossed cisterns, new location: element <elem_nr>
```

(ez utóbbiból több is lehetséges, ha egymás utáni **`manipulate`** parancsokat ad meg amikor ciszternákon áll, mert nem eredményezi a kör végét)

A következő parancsok csak technikus játékos esetén működnek. Szabotőr játékos esetén nem tesznek semmit.

`pickup`

Leírás: Ha egy szerelő egy ciszternán áll, akkor ennek a parancsnak a hatására pumpa kerül a tárolójába, ha az jelenleg üres.
Ellenkező esetben hatástalan.

Kimenet: **`a new pump was added to <name>’s inventory`**

`relocate <dir_nr>`

Leírás: (A jelenlegi elemétől) a megadott irányba lévő elemet próbálja felvenni a szerelő a hatására. Csak akkor lehetséges, ha csőről van szó, nem törött, és nincs benne víz, illetve a tárolója üres.

Opciók: **`dir_nr:`** egész szám, irány amerre lévő szomszédos elemet kíséreli meg felvenni és tárolni a tárolójában.

Kimenet: **`element <elem_nr> pipe relocated from map to <name>’s inventory`**

`place <dir_nr>`

Leírás: A hatására (a jelenlegi elemétől) a megadott irányba lévő helyre próbálja letenni a tárolt elemét (ha van ilyen) a szerelő.
Cső esetén ez csak akkor lehetséges, ha nincs még arra más elem.
Pumpa esetén máshogy működik, lásd az opcióknál.

Opciók: **`dir_nr:`** egész szám, irány amerre le szeretné tenni a tárolt elemét a szerelő (ilyen irányú szomszédja lesz annak az elemnek, amelyen áll).
Ha pumpát tesz le, nem muszáj megadni. Ha pedig meg van adva, hatástalan, hiszen csak a jelenlegi, nem törött csövet tudja kicserélni rá, amelyen áll.

Kimenet: Egyik az alábbiak közül, attól függően, hogy csövet vagy pumpát helyez le:
`element <elem_nr> pipe placed`
`element <elem_nr> pipe replaced by new pump`

7.1.2.3 Parancsok az objektumok állapotának ellenőrzéséhez

`print inventory <technician_name>`

Leírás: Kiírja a megadott nevű szerelő tárolójának tartalmát, és annak jellemzőit.

Opciók: **`technician_name:`** a szerelő neve, akinek ki szeretnénk írni, hogy mi van a tárolójában

Kimenet: Az elem tulajdonságait írja ki, amit tárol, vagy annak tényét, hogy üres a tároló.

`print currElem <name>`

Leírás: Kiírja a megadott nevű játékos jelenlegi elemének (amin áll) jellemzőit.

Opciók: **`name:`** a játékos neve

Kimenet: Az elem tulajdonságait írja ki, amelyen áll a megadott játékos.

`print elem <elem_nr>`

Leírás: A megadott sorszámú (*elements*-beli indexű) elem jellemzőit írja ki.

Opciók: **`elem_nr:`** az elem indexe *elements* gyűjteményben, amelynek ki szeretnénk írni a jellemzőit

Kimenet: A megadott sorszámú elem tulajdonságait írja ki.

`print map`

Leírás: A pálya összes elemének kiírja a jellemzőit.

Kimenet: A pálya összes elemének kiírja a jellemzőit.

7.2 Összes részletes use-case

megjegyzés: a Proto-n kívül csak akkor van megadva aktor-ként a Tester, ha további bemenetet is kér a parancs tőle. (A parancs begépelése még nem a use-case része.)

Továbbá a redundancia elkerülése érdekében a forgatókönyvekben nem szerepel, hogy mit írnak ki a parancsok, az az előző fejezetben lett specifikálva, részletesen.

7.2.1 add element

Rövid leírás	Az **`add <elem_type> <nb_elem_nr> <dir_nr>`** parancshoz tartozó use-case. A megadott sorszámú elemtől a megadott irányba lehelyez egy új elemet.
Aktorok	Proto
Forgatókönyv	**1.** **`nb_elem_nr`** létező sorszámú elem. **1.A.1.** **`elem_type==pipe`** esetén lehelyezi **`dir_nr`** irányba **`nb_elem_nr`** sorszámú elemtől egy új csövet. **1.B.1.** **`elem_type==pump`** esetén lehelyezi **`dir_nr`** irányba **`nb_elem_nr`** sorszámú elemtől egy új pumpát. **1.C.1.** **`elem_type==source`** esetén lehelyezi **`dir_nr`** irányba **`nb_elem_nr`** sorszámú elemtől egy új forrást. **1.D.1.** **`elem_type==cistern`** esetén lehelyezi **`dir_nr`** irányba **`nb_elem_nr`** sorszámú elemtől egy új ciszternát.

7.2.2 add player

Rövid leírás	Az **`add <player_type> <name> <elem_nr>`** parancshoz tartozó use-case. Egy megadott nevű új játékost helyez le a megadott sorszámú elemre.
Aktorok	Proto
Forgatókönyv	**1.** **`elem_nr`** létező sorszámú elem. **2.** **`name`** létező játékosnév. **2.A.1.** **`player_type==saboteur`** esetén lehelyez az **`elem_nr`** sorszámú elemre egy új, **`name`** nevű szabotőrt. **2.B.1.** **`player_type==technician`** esetén lehelyez az **`elem_nr`** sorszámú elemre egy új, **`name`** nevű szerelőt.

7.2.3 controller toggle random

Rövid leírás	Az **`controller toggle random`** parancshoz tartozó use-case. Ki-/bekapcsol minden véletlenszerű viselkedést a modellben, és helyettük a definiált (és módosítható) értékeket használja, hogy determinisztikus teszteseteket állíthassunk elő. Az alábbiakra van hatással: • Pumpák elrontása a körök elején (manuálisan a **controller break <elem_nr>** paranccsal kell megadni majd a controller általi pumpaelrontást helyette, ha szeretnénk ilyet). • A körök száma, ameddig egy cső csúszós és ragadós maradhat. Amennyiben ki van kapcsolva a véletlenszerű viselkedés, a **controller set defaultCounter <turn_cnt>** értékkel állítható ez be. Ilyenkor mindig a megadott számú körök erejéig maradnak csúszósak és ragadósak a csövek. • Csúszós cső esetén, hogy melyik irányú szomszédja felé mozgatja el a játékost. Kikapcsolt véletlenszerű viselkedés esetén mindig a legelső sorszámú irányba fogja, amerre van szomszédja az adott csőnek.
Aktorok	Proto
Forgatókönyv	**1.** Kikapcsolja a véletlenszerű viselkedést a modellben.

7.2.4 controller break elem

Rövid leírás	Az **`controller break <elem_nr>`** parancshoz tartozó use-case. Egy controller általi elemelrontást csinál, azaz olyat, ami nem eredményezi a jelenlegi játékos körének a végét, és így a víz folyására az nem is lesz hatással a következő kör elejéig.
Aktorok	Proto
Forgatókönyv	**1.** **`elem_nr`** létező sorszámú elem. **2.** Elrontódik.

7.2.5 controller set defaultCounter

Rövid leírás	Az **`controller set defaultCounter <turn_cnt>`** parancshoz tartozó use-case. Beállítja, hogy kikapcsolt véletlenszerű viselkedés esetén hány körig (turn) maradjanak csúszósak és ragadósak a csövek.
Aktorok	Proto
Forgatókönyv	**1.** **`turn_cnt`**-ra állítja counter attribútum csúszóssá vagy ragadóssá tétel utáni alapértelmezett értékét.

7.2.6 load state

Rövid leírás	Az **`load state <rel_file_name>`** parancshoz tartozó use-case. A megadott fájlból beolvassa, és végrehajtja a parancsokat. Használható csak a játék előkészítésére, de komplett teszteset ellenőrzésére is.
Aktorok	Proto
Forgatókönyv	**1.** Létezik **`rel_file_name`** nevű/elérési útvonalú fájl. **2.** **`rel_file_name`** nevű fájl parancsait végrehajtja.

7.2.7 reset state

Rövid leírás	Az **`reset state`** parancshoz tartozó use-case. Alapállapotba (egyetlen forrás, nem lesznek játékosok, stb.) állítja az egész játékot. A **`load state <rel_file_name>`** parancs előtt nincs szükség kiadni ezt a parancsot, mert alapjáraton megteszi ezt.
Aktorok	Proto
Forgatókönyv	**1.** Visszaállítja a pontszámlálókat, és egyéb változókat. **2.** Kiüríti a pályát és a játékosok gyűjteményét. **3.** Egy kiinduló forrást helyez a pályára.

7.2.8 start game

Rövid leírás	A **`start game`** parancshoz tartozó use-case. Elkezdi a játékot (A játékosok felváltva kerülnek majd sorra, minden kör elején folyik a víz, stb.). **figyelem**: a tesztelő felelősége, hogy a parancs kiadása előtt érvényes kezdeti feltételekkel rendelkező pályát alkosson (pl. megfelelő számú, és sorrendben hozzáadott játékosok legyenek).
Aktorok	Proto
Forgatókönyv	**1.** Elindítja a játékot (játék főciklusát).

7.2.9 move player

Rövid leírás	A **`move <dir_nr>`** parancshoz tartozó use-case. A megadott irányba próbál lépni a jelenlegi eleméről a soron lévő játékos a hatására.
Aktorok	Proto
Forgatókönyv	**1.** A jelenlegi elemtől **`dir_nr`** irányban van szomszédos elem, ami nem egy olyan cső, amin már állnak. **1.A.1.** Csúszós csőre szeretne lépni. **1.A.2.** Egy véletlenszerűen választott vagy a legkisebb irányszámú szomszédjára kerül a csúszós csőnek (attól függően, hogy be van-e kapcsolva a véletlenszerű viselkedés). **1.B.1.** Ragadós csőre szeretne lépni. **1.B.2.** A ragadós csőre kerül. **1.B.3.** A köre véget ér. **1.C.1.** Átkerül a kiválasztott elemre.

7.2.10 manipulate current elem

Rövid leírás	A **`manipulate`** parancshoz tartozó use-case. Ez a parancs szolgál arra, hogy a játékos manipulálja a képességeinek és az adott elemnek megfelelően azt az elemet, amelyen éppen áll.
Aktorok	Proto, Tester
Forgatókönyv	**1.A.1.** A soron lévő játékos szabotőr. **1.A.1.A.1.** Az elem, amin áll, egy nem törött cső. **1.A.1.A.2.** Proto felszólítja Tester-t, hogy válasszon, hogy mit szeretne tenni a csővel. **1.A.1.A.2.A.1.** **`stickify`** parancs hatására ragadós lesz **1.A.1.A.2.B.1.** **`slippify`** parancs hatására csúszós lesz **1.A.1.A.2.C.1.** **`break`** parancs hatására lyukas lesz **1.A.1.B.1.** Az elem, amin áll, egy pumpa. **1.A.1.B.2.** Proto felszólítja Tester-t, hogy adja meg az új bemeneti és kimeneti irányt a **`change input <dir_nr> output <dir_nr>`** paranccsal. **1.A.1.B.3.** Tester a paranccsal megváltoztatja a pumpa be- és kimenetét. **1.A.1.C.1.** Az elem, amin áll, egy ciszterna. **1.A.1.C.2.** Átkerül a következő ciszternára. **1.B.1.** A soron lévő játékos technikus. **1.B.1.A.1.** Az elem, amin áll, egy törött cső. **1.B.1.A.2.** Megjavítja a törött csövet. **1.B.1.B.1.** Az elem, amin áll, egy nem törött cső. **1.B.1.B.2.** Proto felszólítja Tester-t, hogy válasszon, hogy mit szeretne tenni a csővel. **1.B.1.B.2.A.1.** **`stickify`** parancs hatására ragadós lesz **1.B.1.B.2.B.1.** **`break`** parancs hatására lyukas lesz **1.B.1.C.1.** Az elem, amin áll, egy törött pumpa. **1.B.1.C.2.** Megjavítja a törött pumpát. **1.B.1.D.1.** Az elem, amin áll, egy nem törött pumpa. **1.B.1.D.2.** Proto felszólítja Tester-t, hogy adja meg az új bemeneti és kimeneti irányt a **`change input <dir_nr> output <dir_nr>`** paranccsal. **1.B.1.D.3.** Tester a paranccsal megváltoztatja a pumpa be- és kimenetét. **1.B.1.E.1.** Az elem, amin áll, egy ciszterna. **1.B.1.E.2.** Átkerül a következő ciszternára.

	(a körök vége nincs feltüntetve, de a ciszternás eseteken kívül mindig úgy végződik, hogy véget ér a játékos köre)

7.2.11 pickup

Rövid leírás	A **`pickup`** parancshoz tartozó use-case. Ha egy szerelő egy ciszternán áll, akkor ennek a parancsnak a hatására pumpa kerül a tárolójába, ha az jelenleg üres. Ellenkező esetben hatástalan.
Aktorok	Proto
Forgatókönyv	**1.** A jelenlegi játékos szerelő, üres a tórolója és ciszternán áll. **2.** Egy új pumpa kerül a tárolójába.

7.2.12 relocation

Rövid leírás	A **`relocate <dir_nr>`** parancshoz tartozó use-case. (A jelenlegi elemétől) a megadott irányba lévő elemet próbálja felvenni a szerelő a hatására. Csak akkor lehetséges, ha csőről van szó, nem törött, nem áll rajta senki és nincs benne víz, illetve a tárolója üres.
Aktorok	Proto
Forgatókönyv	**1.** A jelenlegi elemtől **`dir_nr`** irányban van szomszédos, nem törött, víz és játékos szempontjából üres cső, illetve a jelenlegi játékos egy üres tárolójú szerelő. **2.** Az említett cső a szerelő tárolójába kerül, a pályáról eltűnik.

7.2.13 placement

Rövid leírás	A **`place <dir_nr>`** parancshoz tartozó use-case. A hatására (a jelenlegi elemétől) a megadott irányba lévő helyre próbálja letenni a tárolt elemét (ha van ilyen) a szerelő. Cső esetén ez csak akkor lehetséges, ha nincs még arra más elem. Pumpa esetén máshogy működik. Ilyenkor a nem törött csövet tudja kicserélni rá, amelyen áll.
Aktorok	Proto
Forgatókönyv	**1.A.1.** A jelenlegi játékos egy szerelő, akinek a tárolójában van egy cső. **1.A.2.** **`dir_nr`** irányban nincs jelenleg elem. **1.A.3.** A tárolóban lévő cső le lesz helyezve a kiválasztott helyre. **1.B.1.** A jelenlegi játékos egy szerelő, aki egy nem törött csövön áll, és van a tárolójában egy pumpa. **1.B.2.** A cső ki lesz cserélve a tárolóban tárolt pumpára. **2.** A szerelő tárolója üres lesz.

7.2.14 print inventory

Rövid leírás	A **`print inventory <technician_name>`** parancshoz tartozó use-case. Kiírja a megadott nevű szerelő tárolójának tartalmát, és annak jellemzőit.
Aktorok	Proto
Forgatókönyv	**1.** Létezik **`technician_name`** nevű szerelő. **2.** Az elem tulajdonságait írja ki, amit tárol, vagy annak tényét, hogy üres a tároló.

7.2.15 print currElem

Rövid leírás	A **`print currElem <name>`** parancshoz tartozó use-case. Kiírja a megadott nevű játékos jelenlegi elemének (amin áll) jellemzőit.
Aktorok	Proto
Forgatókönyv	**1.** Létezik **`name`** nevű játékos. 2. Kiírja a tulajdonságait annak az elemnek, amelyen áll.

7.2.16 print elem

Rövid leírás	A **`print elem <elem_nr>`** parancshoz tartozó use-case. A megadott sorszámú (elements-beli indexű) elem jellemzőit írja ki.
Aktorok	Proto
Forgatókönyv	**1.** Létezik **`elem_nr`** sorszámú elem. **2.** Kiírja a megadott sorszámú elem jellemzőit.

7.2.17 print map

Rövid leírás	A **`print map`** parancshoz tartozó use-case. A pálya összes elemének kiírja a jellemzőit.
Aktorok	Proto
Forgatókönyv	**1.** A pálya összes elemének kiírja a jellemzőit.

7.3 Tesztelési terv

7.3.1 test pipe breaking/fixing

Rövid leírás	2-2 játékos egy egyszerű pályán mozog, és a szerelők és szabotőrök felváltva tesztelik a csövek megjavítását és kilyukasztását.
Teszt célja	Tesztelt osztályok: Game, GameInput, Network, Player, Technician, ManipulatorSaboteur, ManipulatorTechnician, Element, Pipe, Source, Cistern Nem tesztelt osztály: Pump

7.3.2 test pump breaking/fixing/switching

Rövid leírás	2-2 játékos egy pumpákat tartalmazó pályán mozog. A kontroller elront pumpákat néha. A szerelők megjavítják és átállítják őket. A szabotőrök csak átállítják őket.
Teszt célja	Tesztelt osztályok: mindent „megmozgat”

7.3.3 test pipe stickifying/slippifying and cistern crossing

Rövid leírás	3-3 játékos egy komplexebb, minden elemből többet tartalmazó pályán mozog. Felváltva csúszóssá és ragadóssá teszik a csöveket, és kipróbálják őket úgy, hogy rájuk lépnek. Ciszternák között is „teleportálnak”.
Teszt célja	Tesztelt osztályok: mindent „megmozgat”

7.3.4 test storing/relocation/placement

Rövid leírás	2-2 játékos mozog egy minden elemet tartalmazó pályán. A szerelők tesztelik a csövek felvételét és lerakását, illetve pumpák felvételét a ciszternáknál és azok lerakását is. A szabotőrök ezalatt korábban már tesztelt szabotálásokat végeznek.
Teszt célja	Tesztelt osztályok: mindent „megmozgat”

7.3.5 test whole game

Rövid leírás	3-3 játékos játszik, egy komplexebb pályán, minél több funkciót tesztelve. Egy kimerítőbb teszt, amiben már korábban egyenként tesztelt funkciók újra tesztelésre kerülnek. Ami itt lesz és korábban nem volt tesztelve: egy teljes játék véget érése, azaz az egyik csapat el fogja érni az elérendő pontmennyiséget a győzelemhez.
Teszt célja	Tesztelt osztályok: mindent „megmozgat”

megjegyzés: a tesztek helyességének ellenőrzése „manuális” meggondolással, a kimenetek és a print-ek várt és kapott formájának összevetésével fog történni.

7.4 Tesztelést támogató segéd- és fordítóprogramok specifikálása

Nincs szükség egyéb segéd- és fordítóprogramokra.

7.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2023.04.20 18:00	1 óra	Sasvári Mizser Váradi Fekete Tepliczky	Értekezlet. Döntés: Az eheti feladatokat kollaboratív módon, specifikus felosztás nélkül végezzük. Módosítások magas szintű megbeszélése. ---------- **Teljes dokumentum**: határidő: 2023.04.24. 00:00 elkészült: 2023.04.23. 20:00
2023.04.21 10:30	4 óra	Tepliczky Fekete	ötletelés, Változott osztályok és metódusaik (7.0.2) osztálydiagram módosítása (7.0.1) Move to empy pipe szekvencia módosítása (7.0.3.1)
2023.04.21 15:00	3 óra	Tepliczky	teljes lecsatolhatóság átgondolása Next turn szekvenciadiagram (7.0.3.3) osztálydiagram javítások (inner class tartalmazás helyett, Network pipes asszociációja)
2023.04.21 15:15	3 óra	Sasvári Fekete	a csőmanipuláció inputjának átgondolása (7.0.2.4, 7.0.2.5) csőmanipulálás végleges szekvenciadiagrammjai (7.0.3.4, 7.0.3.5) osztálydiagramm véglegesítése
2023.04.22 15:00	7 óra	Mizser	eddigi munka ellenőrzése változások áttekinthető összegzése (7.0), szekvenciadiagram javítások castok megszüntetésének terve Flood source szekvencia módosítása (7.0.3.6), geometria kiszorítása Parancsok a játék (pálya előkészítéséhez) (7.1.2.1) Parancsok a játék játszásához (7.1.2.2)
2023.04.22 18:00	3 óra	Váradi	Be- és kimeneti parancsok (Mizserrel együtt)

2023.04.23. 15:00	1 óra	Mizser Sasvári	Parancsok az objektumok állapotának ellenőrzéséhez (7.1.2.3) Magas szintű tesztelési terv (7.3)
2023.04.23. 16:00	2 óra	Sasvári Váradi	Use-case leírások a parancsok alapján (7.2)
2023.04.23. 19:00	1 óra	Váradi	Dokumentumszerkesztés, ellenőrzések, diagramok képeinek beszúrása