KUTATÁSI BESZÁMOLÓ

Krónikus hemodialízisgép dializáló oldat előkészítést vezérlő szoftverének objektumorientált szemléletű fejlesztése

Készítette

Horváth Gábor

Gyakornok, B.Braun Medical Kft.

2014. május 20.

Tartalomjegyzék

Bevezető

Napjainkban Magyarországon minden ezredik ember él olyan veseelégtelenséggel, amellyel folyamatos vesepótló kezelésre, azaz dialízisre szorul. Erre a célra kifejlesztett gépeket már több évtizede használunk, fejlesztésük azonban ma is folyik. A technológiai fejlődés lehetőséget biztosít kényelmesebb és biztonságosabb terápiák automatizált végzésére. A legfőbb cél az emberi hibázások kiküszöbölése és a precizitás fokozása. A közelmúltban számos dialízisgépet gyártó cég jelent meg a piacon. A tapasztalat szerint a gépek hardverelemeiben jelentős változtatásra nem igen lehet számítani a jövőt tekintve, a verseny tehát a szoftverfejlesztés terén dőlhet el.

Az általam vizsgált jelenlegi gépek hasonló feladatokat látnak el és hardverelemeik is közel megegyeznek, szoftverarchitektúrájuk azonban teljesen eltérő és mondhatni, hogy elavult. A jelenlegi tendencia szerint egy teljesen új prototípus kifejlesztésekor a szoftvertervezés szinte teljes egészében tiszta lapokkal indul, a korábbi megoldásokból nem igen emelhető át semmi, csakis komoly változtatások árán. Az elmúlt évtizedekben jelentős fejlődésen mentek keresztül a szoftverarchitektúrák. Számos ötlet és technológia született az optimalizálás és újrafelhasználhatóság jegyében, amelyeket érdemes szem előtt tartani.

Munkám célja, hogy modern technológiák és szemléletek alkalmazásával egy újragondolt szoftverarchitektúra alapjai valósuljanak meg figyelembe véve az újrafelhasználhatóságot, valamint azt, hogy a géppel szemben támasztott orvosi és egyéb követelmények alapján legyenek létrehozva az egyes szoftverkomponensek.

Munkám során tanulmányoztam a dialíziskezelés folyamatát, valamint néhány jelenlegi dialízisgép működését és szoftverük belső felépítését. Megvizsgáltam a legjelentősebb mai modern újrafelhasználási technológiákat, amelyek beágyazott rendszerekben használatosak és összevetettem a dialízisgépekben használt megoldásokkal. További munkám során a célom, hogy egy jól átgondolt, hatékony módszer alapján egy lehetséges megoldást találjak a dialízisgépek későbbi szoftverfejlesztésére, azon belül is a dializáló oldat előkészítést vezérlő szoftverre fókuszálva.

1. fejezet

Dialíziskezelés és hemodialízis

1.1. A kiválasztásért felelős szerv, a vese

A vese bab alakú páros szerv, melynek elsősorban a kiválasztásban van fontos szerepe, úgy mint a vízoldékony anyagcsere-végtermékek eltávolításában, a só- és vízháztartás szabályozásában, valamint a fehérjék és nukleinsavak bomlástermékeinek kiválasztásában. Amennyiben a vesék nem működnek megfelelően, úgy nem képesek ezen feladatok maradéktalan ellátására. Súlyos esetben transzplantációra van szükség az életben maradáshoz, ám erre nincs mindig lehetőség. Azt az eljárást, melynek során a veseelégtelenséggel rendelkező betegek vére a különféle salakanyagoktól megtisztításra kerül, dialízisnek hívjuk.

1.2. Dialízis

Alkalmaznak testen belüli és testen kívüli dialízis módszereket is, ám utóbbi jóval elterjedtebb. Testen belüli módszerek esetén beszélhetünk például peritoneális dialízisről, melynek során a hashártya használatos féligáteresztő rétegként. A módszer előnye, hogy a beteg egymaga is végezheti otthonában, tehát akár el is utazhat hosszabb időre. Létezik kézi és gépi módszer is, de mindkettőben közös, hogy katétert ültetnek a páciens hasüregébe, melyen keresztül dialízisoldattal töltik fel az üres térrészt. Átlagosan fél óra várakozást követően leengedik, majd újratöltik a folyadékot. A vér megtisztítása úgy történik, hogy a kis molekula-méretű anyagok a hashártya ereiből a dialízisoldatba diffundálnak, amely később le lesz engedve. A módszer előnye, hogy kíméli a vérkeringést, amely idős betegek esetén fontos szempont, viszont hátrányként említhető meg, hogy a hasüreg nyitott állapotban van, tehát fokozottabb figyelmet kell fordítani a fertőzések elkerülésére, valamint a hashártya átereszti a fehérjéket is, amelyek ezáltal távoznak a szervezetből. Ennek orvoslására a betegnek mesterséges fehérje pótlásra van szüksége.

1.2.1. Hemodialízis

Testen kívüli módszerek esetén számos típus ismeretes, melyek közül az egyik legsűrűbben alkalmazott módszer a hemodialízis. Megkülönböztetjük akut és krónikus fajtáját, melyekben közös, hogy orvosi felügyelet mellett végzik dialízisközpontokban. Akut veseelégtelenség esetén csak véges ideig szükséges a kezelés folytatása, mert a vesék működése idővel helyre áll, viszont krónikus esetben a páciensnek élete végéig szüksége van a terápia alkalmazására. Egy-egy ilyen kezelés akár 4–5 órán át is tarthat a terápia paramétereitől függően és hetente átlagosan 3–4 alkalommal végzendő.

A hemodialízis szintén a már korábban említett diffúzió és ozmózis elve alapján működik. A méreganyagokkal, bomlástermékekkel és fontos ásványi anyagokkal teli vér egy féligáteresztő hártyával van elválasztva a csíraszegény, méreg- és salakanyagmentes dialízisoldattól. Ez a hártya egy olyan kis lyukacsos felület, amelynek pórusmérete olyan kicsi, hogy a vér kis molekulaméretű alkotóit – úgy mint, a víz, ásványi anyagok és salakanyagok – átengedi a dializáló oldat oldalára, ám visszatartja a nagy méretű sejteket és molekulákat (pl. fehérjéket). A dialízisgépek azon tartozékát, amelyben ez a folyamat végbemegy dializátornak nevezzük. Időben annál hatékonyabb a hemodialízis, minél nagyobb felületen történik meg a tisztítási folyamat. Ennek érdekében úgynevezett kapillárisokat használnak a dializátorokban, amelyek nem mások, mint hajszálvékony lyukacsos oldalfalú csövek. A vér a belsejükben, míg a dializáló oldat körülöttük helyezkedik el, így az ozmózis a csövek falán keresztül játszódik le. A kezelés hatékonyságát csökkenti az, hogy idővel egyre több pórus tömődik el a vér nagyobb molekulái által, csökkentve ezzel az áteresztő felület nagyságát. A folyamat hatékonyságának növelése érdekében a két folyadékot pupák segítségével ellentétes irányban folyatják, ezáltal nyomáskülönbséget hozva létre a féligáteresztő hártya két oldala között. A nyomások változtatásával a beteg vízháztartását is lehet szabályozni. A kezelő orvos meghatározhatja, hogy vizet elvenni, vagy hozzáadni szeretne-e a páciens szervezetéhez. Ez természetesen több dolog függvénye, többek közt az eddig fogyasztott folyadék és a képzett vizelet mennyiségéé. Minden esetben a lassú, folyamatos véghezvitel alkalmazandó, mert a szervezet nem képes túl gyorsan reagálni a változásokra. A vérnyomás hirtelen lezuhanhat vagy az izmok begörcsölhetnek.

A hemodialízis alapfeltételei közé tartozik, hogy a páciens stabil vérkeringéssel és megfelelő vértérfogattal rendelkezzen. A legelterjedtebb esetben kari artériás-vénás söntöt, esetleg érprotézist alkalmaznak a vérkörhöz való könnyű és biztonságos csatlakozáshoz, de ritka esetekben a comb és a kulcscsont is szóba jöhet. Másik megoldásként használatos egy nagyobb vénába vezetett katéter alkalmazása, amely általában a nyaki vénát érinti.[1][2][3]

2. fejezet

Újrafelhasználási technológiák beágyazott környezetben

Ahogy majd később látni fogjuk, a jelenlegi krónikus hemodialízisgépben alkalmazott szoftverarchitektúra nem alkalmas hosszú távon hatékony továbbfejlesztésre. A különböző funkcionalitások karöltve, egymásra támaszkodva valósítják meg feladatukat, valamint az egyes mechanizmusok lezajlása nehezen követhető végig a szoftver egészén. A kellő fokú modularizálás hiánya egyrészt megnehezíti a csoportos fejlesztést, másrészt az egyes szoftverkomponensek a későbbi termékek esetén nem, vagy csak jelentős módosítással lesznek újra felhasználhatóak.

2.1. Referencia architektúrák

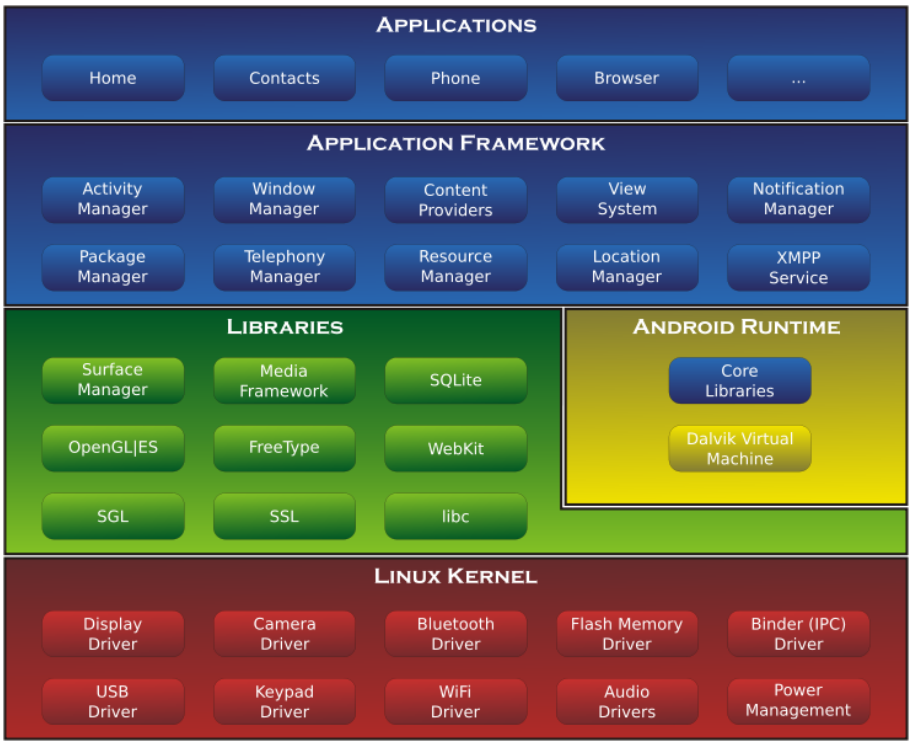
Az iparban több olyan beágyazott környezetben használatos szoftverarchitektúra is elterjedt, amelyek megoldást nyújtanak ezen problémákra. Számos tekintetben is hatékonynak tűnnek az úgynevezett referencia architektúrák, amelyek esetén az adott problémát szoftverkomponensekre bontjuk és definiáljuk azok kapcsolatait. A modularizálásnak köszönhetően az egyes feladatok jól szétválasztottak, amely számos előnyt jelent. A hordozhatóság és újrafelhasználhatóság mellett egy ügyes megtervezés esetén az egyes komponensek védettek lesznek egymással szemben. A rendszer egészének karbantarthatósága sokkal jobb, mint a jelenlegi architektúra esetén. A kialakítás lehetőséget biztosít közös elnevezés-használatra, amely egyértelmű, letisztult és kényelmes fejlesztést tesz lehetővé. A szoftvermodulokat logikai rétegekbe rendezve az architektúra határvonalai jól láthatóak, a köztes interfészek jól definiálhatóak. A referencia architektúra azonban néhány hátrányt is magával hordoz, amelyeket nem lehet figyelmen kívül hagyni. Beágyazott környezetben, azon belül is orvosi készülékekben elengedhetetlenül fontos szempont a real-time működés, amelyet az architektúra önmagában nem feltétlenül garantál, komoly odafigyelésre van szükség annak tervezése során. Amennyire sok előnnyel rendelkezik, olyan gondos megtervezést igényel. Ehhez olyan emberek szükségesek, akik komoly szaktudással rendelkeznek és ismerik a pontos feladatot, amelyet aztán képesek szétszedni megfelelő komponensekre. Az architekt(ek) feladata igen komoly, hiszen ha túl későn derül ki, hogy a funkciók

szétválasztását másképp kellett volna megoldani, akkor annak orvoslása igen nagy energia és anyagi többlettel jár.

Az iparban a referencia architektúrák csoportjában két jeles képviselő is szerepel, akiket érdemes megvizsgálni belső felépítésük és működésük szempontjából. Látni fogjuk, hogy ha teljes egészében nem is, de ésszerűen alkalmazva a belőlük levont tanulságokat és előnyöket, egy saját architektúra kialakítására van lehetőség, ami igen jól illeszkedik a dialízisgépek működéséhez.

2.1.1. Android

Napjainkban a mobil operációs rendszerek területén hatalmas teret hódító *Android* nevű szoftver a referencia architektúrák oszlopos tagja. Olyan beágyazott eszközökön találkozhatunk vele, mint az érintőképernyős okostelefonok és táblagépek, de egyre növekvő az érdeklődés az ipari automatizálás és a gépjárművek fedélzeti számítógépének fejlesztő cégei körében is. Létrehozásakor a cél az volt, hogy egy nyílt forrású operációs rendszert hozzanak létre mobil eszközökre, amely az alkalmazások elől elfedi a hardverspecifikus tulajdonságokat. Erre szolgál az Android magja, amely lényegében egy Linux kernel. A fejlesztőcég e fölé a kernel fölé hozott létre egy virtuális gépet, amely az alkalmazások futtatásáért és a felhasználói felület kezeléséért felelős. Az architektúra részletes felépítése a 2.1. ábrán látható.



2.1. ábra. Az Android architektúrája

A kernel felel tehát a hardverelemek működtetéséért, kezeléséért. Az architektúra ezen szintjén lévő szoftverkomponenseket maguk a hardvergyártók fejlesztik ki a saját készülékeikhez, amennyiben azt

szeretnék, hogy az adott eszközön Android operációs rendszer fusson. A kernelben az alapvető Linux kernel-funkciók érhetőek el úgy, mint a memória- és eszközkezelés, folyamatok és szálak ütemezése, valamint a teljesítménykezelés, ami az alacsonyabb energiafogyasztást igyekszik elősegíteni.

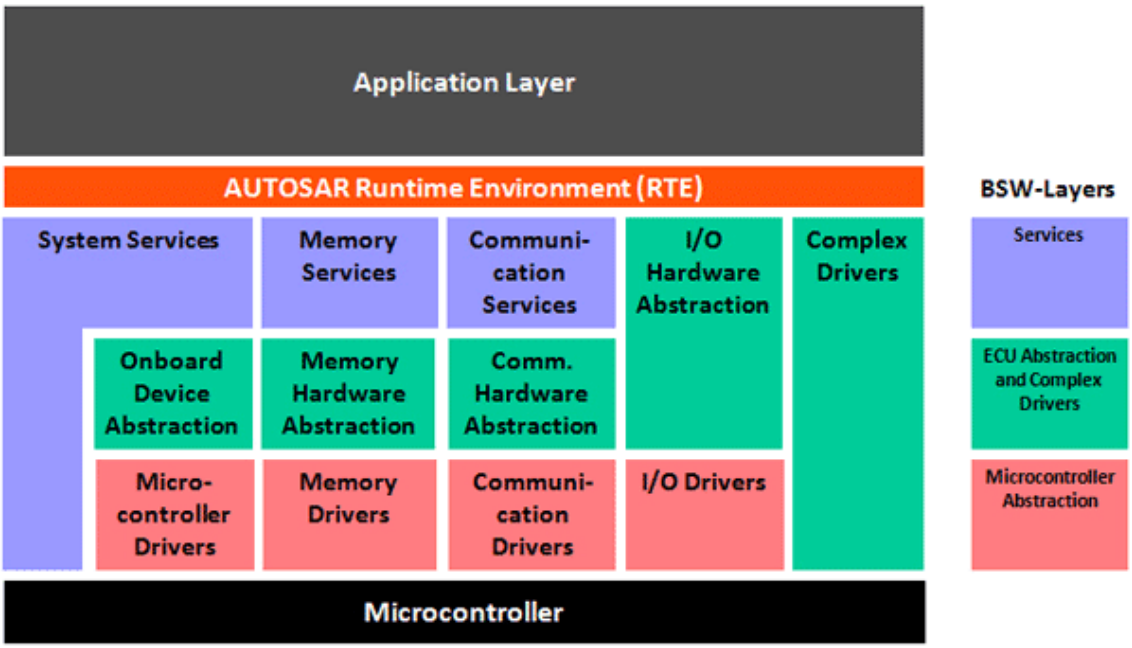
Ezen szolgáltatásokat használják az architektúrában logikailag egyel feljebb lévő programkönyvtárak, amelyek többnyire C/C++ nyelven íródtak és szintén a kernel részei, tehát még a Linuxból hagyományozódtak. A könyvtárak funkcióit maga a virtuális gép használja, amely elfedi a Linux fájlrendszert és egységes interfészt biztosít a magasabb szintű alkalmazások számára. Az alkalmazás keretrendszer és az alkalmazások már Java nyelven íródnak az egységes Android Runtime fájlrendszert használva. Világos tehát és figyelemre méltó, hogy a virtuális gép feletti rétegek elemei már függetlenek az eszközök típusaitól.[4]

2.1.2. AUTOSAR - (AUTomotive Open System ARchitecture)

Ahogy a nevében is benne van, szintén egy nyílt forrású, ám ezúttal egy elsősorban az autóiparban használatos beágyazott szoftverarchitektúráról van szó, ha az AUTOSAR-ról beszélünk. Rétegelt (*layer*) felépítése az Androidéhoz hasonló, amely lehetővé teszi az egyes szoftverkomponensek újrafelhasználhatóságát. Az informatikai automatizálás közelmúltban való egyre növekvő térnyerése az autóiparra is kiterjedt, amely magával hozta az egységes szoftverek igényét is, hiszen óriási előnnyel jár, ha az adott termékek mindenféle módosítás nélkül áthordozhatóak más-más járműplatformokra. Az AUTOSAR oly módon lett erre kitalálva, hogy az egyes autóipari szoftverkomponensek azonos interfészen keresztül kommunikálhassanak, csak a hardverkezelő egységek termékspecifikusak. Jelentős különbség az Androidhoz képest, hogy itt az alkalmazások nem feltétlen egy ugyanazon fizikai eszközhöz készülnek, hanem az adott járműben található különféle egységek (szenzorok, aktuátorok stb .) szoftveres reprezentációját valósítják meg, akik egy közös virtuális hálózaton kommunikálhatnak egymással. Az architektúra felépítését a 2.2. ábrán vizsgálhatjuk meg közelebbről.

Legalul helyezkedik el valamilyen hardveregység, amelyhez aztán specifikus drivereket kell készíteni. Ezeket általában a hardvergyártó cég írja meg a saját termékéhez. Az efölött lévő magasabb szintek igaz még mindig hardverspecifikusak, de ezt a tulajdonságot igyekszenek elfedni a felsőbb szintek számára. A következő szint a Runtime Environment (RTE), ahová az alsóbb szintek már egy jól definiált interfésszel kapcsolódnak. Az RTE feladata az, hogy kezelje a felette lévő szoftverkomponensek közti kommunikációt, valamint, hogy ezen szoftverkomponenseket összekapcsolja a megfelelő hardveregységekkel. Lényegében tehát egy felhőről beszélünk, amelyben az RTE biztosítja, hogy mindenki a megfelelő partnerrel kommunikálhat. A kialakítás nagy előnye, hogy az adott komponenseknek nem kell fixen összekapcsolva lenniük, az RTE megoldja az összeköttetéseket. Elvileg tehát bármely komponens kommunikálhat egy másikkal, hiszen azonos kommunikációs interfésszel rendelkeznek. Az RTE felett lévő szinten helyezkednek el maguk az alkalmazások, amelyek tehát egy–egy szoftverkomponensek. A lényegi funkcionalitás itt realizálódik, az alsóbb rétegek csak a hardverspecifikusság megszüntetéséért és a dinamikus kommunikáció megvalósításáért felelnek. Világos tehát, hogy ez egy jelentős szempontot képvisel az újrafelhasználhatóságot tekintve, ugyanis az

alkalmazásréteg elemeit elegendő egyszer elkészíteni, mivel azok képesek lesznek akármilyen hardveren futni köszönhetően az alsóbb rétegeknek. A jövőben gyártott hardverekre csak az RTE alatti rétegeket kell elkészíteni (sőt, azokból se mindent), az alkalmazásréteg egy az egyben használható lesz.[5]



2.2. ábra. Az AUTOSAR architektúrája

3. fejezet

A jelenlegi gépekben alkalmazott szoftverarchitektúrák

Az általam vizsgált dialízisgépeken alkalmazott szoftverarchitektúrák jelentős mértékben eltérőek. A krónikus gép esetén alkalmazott megoldás közel 20 éve lett kitalálva és a mai napig kiegészítésekkel látják el. A korábbi tapasztalatokra alapozva, valamint a speciális igényeket figyelembe véve, az akut gép szoftverarchitektúrája jelentős különbséget mutat a krónikus gépben alkalmazottal. Mivel igen sok hasonlósággal rendelkezik a két gép mind fizikailag, mind az ellátandó feladatokat illetően, az az elképzelés született, hogy egy jól átgondolt architektúra esetén számos szoftverkomponenst újra fel lehetne használni. Ahhoz, hogy ezt végig gondoljuk, vizsgáljuk most meg a jelenleg alkalmazott megoldásokat mindkét esetben!

3.1. Krónikus gép

A krónikus hemodialízisgép hardverét tekintve első körben azt vehetjük észre, hogy hármas tagozódást mutat. Külön processzoron fut az alsóbb szintű vezérlő szoftver (VS), a felügyelő szoftver (FS) és a felsőbb szintű vezérlő szoftver (FVS). A VS és FS különválásának oka a nagyobb biztonság megteremtése, amely orvosi környezetben használt gépek esetén előírás. A működést a VS végzi, az FS feladata csak az ellenőrzés és szükség esetén a közbeavatkozás. A két komponens esetében biztosítani kell, hogy külön (és általában más típusú) processzoron fussanak, valamint felépítésük és működésük esetén érdemes más technológiákat alkalmazni, kiküszöbölve ezzel a közös hibázás lehetőségét. Az FVS felel a felhasználói inputok és FS kérések fogadásáért, kommunikál az alsóbb szintű szoftverkomponensekkel RS232-n, felelős a grafikus felhasználói interfész megjelenítéséért, valamint betölti a megfelelő konfigurációs adatokat az egyes elvégzendő műveletekhez. Munkám célja az alsóbb szintű vezérlő szoftver egyes részeinek újragondolása, így most csak a VS ismertetésére szorítkozom.

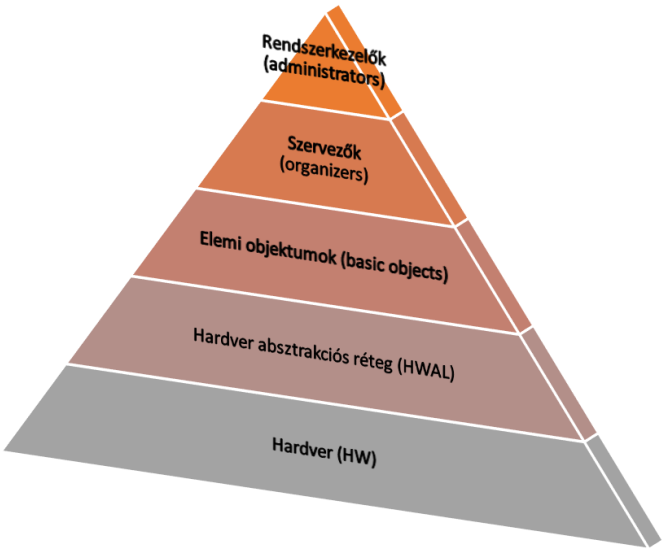
A VS indulásakor a Boot taszk felel a hardvertesztek lefuttatásáért (RAM és stack teszt), valamint a beágyazott operációs rendszer elindításáért. Amennyiben a boot-olás sikeres volt, elindul a Main

taszk, azon belül is a Root folyamat. Ezen taszk segítségével van lehetőségünk egyes taszkokat inicializálni vagy újraindítani, illetve felelős a kezelés (Treatment) vagy szerviz mód (SZM) taszkok indítására. A teljesség igénye nélkül az SZM által nyújtott funkciók: a gép szenzorainak és aktuátorainak tesztelése és kalibrálása, a vízoldal épségének és a hardverelemek megbízhatóságának ellenőrzése, alsóbb szintű beállítások alkalmazása, valamint az egyes terápiák alapértelmezett paramétereinek és limitjeinek beállítása. Ezen feladatok egy nagy részhalmazának ellátása egészen hasonló Treatmentben is, mégis itt külön implementálva van. Egy esetleges változtatás tehát több helyen való forráskód módosítást, majd többszöri tesztelést von maga után. Maga a működés az operációs rendszer nyújtotta lehetőségek alapján történik, tehát taszkokat ütemezünk, alkalmazunk szemaforokat, mailbox-okat és message queue-kat. Munkám során az SZM újbóli felépítését, újrafelhasználható kialakításának lehetőségét vizsgáltam.

3.2. Akut gép

A krónikus gép esetében ismertetett hármas tagozódás szintén megtalálható az akut gép esetén is, azonban érdemes megvizsgálni a vezérlő szoftver architektúráját, mert jelentős különbség mutatkozik a krónikus gépben alkalmazottal.

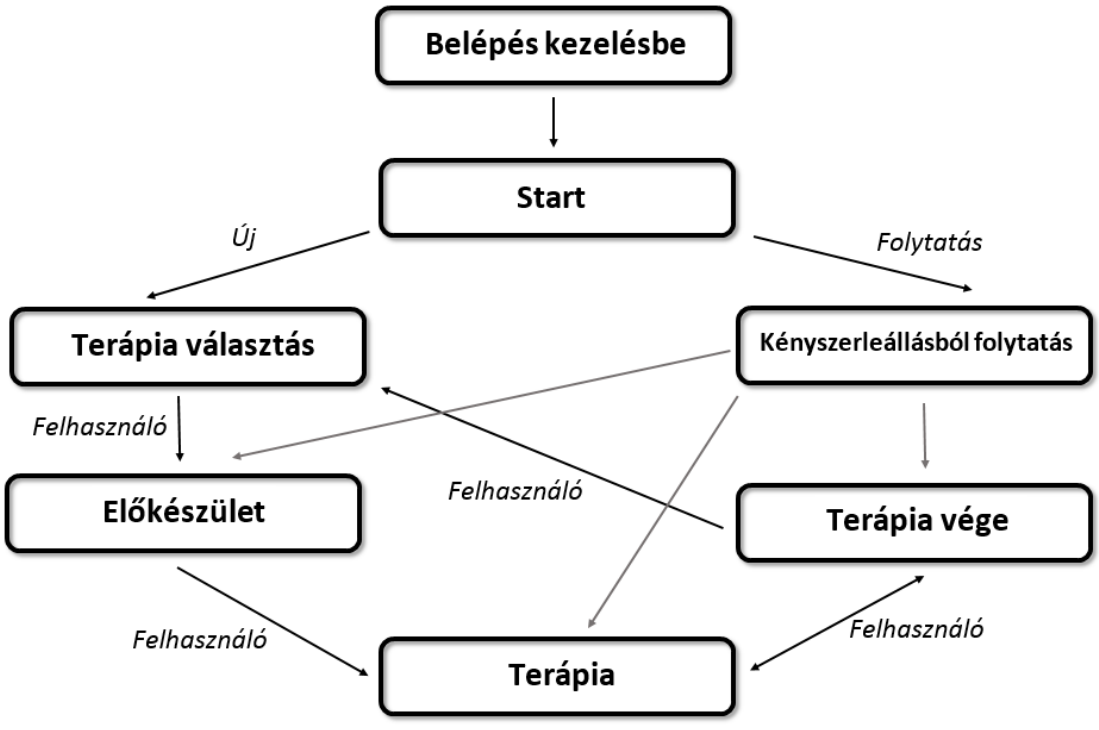
A szoftver felépítését a rétegezettség jellemzi. Az egyes rétegek a logikailag összetartozó szoftverkomponenseket tartalmazzák és egymástól igen jól elkülönülnek. Fejlesztéskor lehetőség van az egyes rétegek komponenseinek egyenkénti elkészítésére, így elkerülhető a fejlesztők egymásra várakozása. A továbbiakban megvizsgáljuk ezen rétegeket a 3.1. ábra segítségével.



3.1. ábra. Az akut gép szoftverarchitektúrájának felépítése

Mivel fejlesztéskor és dokumentáláskor is az angol nyelv használatos, így az egyes elnevezéseket is angolul fogom használni a továbbiakban. A hardver réteg tartalmazza a szenzorokat, aktuátorokat,

kommunikációs csatornákat és az elektronikai komponenseket. A HWAL feladata elfedni a hardver réteget. Tartalmazza a hardverspecifikus drivereket, interfész függvényeket. A felsőbb rétegek a HWAL-nak köszönhetően egy uniformizált BIOS interfészen keresztül tudják elérni az egyes hardverelemeket, tehát a BIOS a HWAL-ban került implementálásra egy nagyon vékony, egyszerű rétegként. A basic object-ek olyan szoftverkomponensek, amelyek az egyes hardverelemeket realizálják szoftveres megjelenéssel. Tartalmazzák az összes funkcionalitást, amelyet az adott hardverkomponensnek tudnia kell, elfedik a hardverspecifikus tulajdonságokat és teljes mértékben kezelik a hozzájuk rendelt elemet. Az organizer-ek felelősek a kisebb szervező feladatok ellátására és ehhez a szükséges basic object-eket használják segítségül. Az organizer-ek összehangolt munkáját az administrators réteg elemei végzik. Itt találhatóak az állapotgép vezérlők. Négy fő állapotról beszélhetünk a bekapcsolást követően. Első lépésként önellenőrzést hajt végre a gép, azaz leellenőrzi, hogy minden feltétel adott-e az elvárt működéshez. Ha ezzel végzett három lehetőség adódik, úgy mint a verzió csere, szerviz üzemmódba lépés, valamint egy kezelés megkezdése. A 3.2. ábrán az administrators réteg egy komponensének folyamatábrája látható, amely egy kezelés lebonyolítását végzi.



3.2. ábra. Egy kezelés lebonyolításának menete

Az egyes állapotok vezérlése tehát az administrators réteg egyik elemének feladata, de az állapotokon belül már az organizer-ek veszik át az irányítást és a basic object-eken keresztül működtetik a gépet.

Jól látható, hogy ez egy letisztultabb, átgondoltabb felépítés a krónikus gépben alkalmazott megoldáshoz képest. A téma folytatásaként egy ehhez egészen hasonló, ám további hasznos ötleteket al-

kalmazó megoldást keresek, amellyel a jövőbeni dialízisgépek szoftverének fejlesztése remélhetőleg jelentősen kevesebb fejlesztési időt fog igénybe venni.

Irodalomjegyzék

[1] Gerd Breuch, Willi Servos, *„An Introduction to Dialysis”*, *Elsevier*, 4–9, 12., 13., 42–48, 77.

[2] `http://goo.gl/CN4gIF`, 2014. május 20., 10:17

[3] `http://goo.gl/Yrhci1`, 2014. május 20., 10:17

[4] `http://goo.gl/ln23s`, 2014. május 20., 10:15

[5] `http://en.wikipedia.org/wiki/AUTOSAR`, 2014. május 20., 10:15