Informatikai rendszertervezés (VIMIAC01)		VIZSGA	2019. 01. 14.
Név:		**Összpontszám:**	/ 40
NEPTUN:			

A vizsga hossza 90 perc. A vizsgán legalább 40%-ot kell elérni.

I.	**Elméleti kérdések** (minden kérdés 1 pontot ér)	**/ 16**

1. Mit jelent a traceability, miért van rá szükség?

2. Mi a különbség az include és extend reláció között a használati eset diagramok esetében?

3. Milyen csoportokra oszthatjuk a követelményeket? Soroljon fel legalább ötöt!

4. Struktúramodellezés során mire tudjuk használni a portokat bottom-up megközelítésben?

5. Mi az a jólformáltsági kényszer? Írjon rá egy példát!

6. 6. Milyen előnyei és hátrányai vannak a top-down megközelítésnek?

7. Mi a különbség a hazárd és a meghibásodás között?

8. 8. Hagyományosan milyen oszlopai vannak az FMEA elemzési táblázatnak?

9. Milyen formái vannak a redundanciának?

10. Mi a szemantikája a Join elemnek az activity diagramok esetében?

11. Mire érdemes használni az interakció diagramokat? Soroljon fel legalább három esetet?

12. Mit jelent, ha egy állapotgép teljes?

13. Mi a platform-modell, és mik a fő fázisai a definiálásuknak?

14. SysML-ben milyen eszközökkel lehet definiálni az allokációkat?

15. Tesztelés esetében mit jelent az ekvivalencia osztály? Adjon rá példát!

16. Tesztmodellezés esetében koncepcionálisan milyen modellekre van szükség (3-4 csoport)?

Informatikai rendszertervezés (VIMIAC01)		VIZSGA	2019. 01. 14.
Név:		**NEPTUN:**	

II.	Gyakorlati feladatok	/ 24

1. Verifikáció és validáció / 6

Okos üvegház rendszert fejlesztünk. A rendszer egész évben üzemel majd. Jelenleg azt a komponenst vizsgáljuk, ami azonosítja, hogy mikor kell mindenképp beavatkoznia a rendszernek. Ha a hőmérséklet 30° fölé nő, akkor el kell kezdeni a szabályzó folyamatokat, ha 40° fölé nőne, akkor riasztást kell küldeni. A növényeknek nem szabad megfagyniuk sem. A páratartalmat is figyelni kell, a paprika esetén például az alacsony páratartalom korlátozza a növekedést. A rendszernek fel kell készülnie arra, hogy a használt szenzorok nem túl megbízhatóak. a) Milyen be- és kimenő paraméterei vannak a komponensnek? Az egyes paraméterekhez milyen ekvivalencia osztályokat definiálna? b) Milyen teszteseteket definiálna a komponens teszteléséhez a fenti specifikáció alapján?

2. Szolgáltatásbiztonság / 6

Egy vállalat levelező (email) szolgáltatásának működéséről a következőket tudjuk:

- A levél küldése a munkatárs *munkaállomásáról* történik és azt a vállalati *mail szervere* továbbítja a címzettnek.
- A vállalat munkaállomásai és szerverei egy *belső hálózaton* keresztül vannak összekötve.
- A munkaállomásról a vállalati mail szerver eléréséhez, valamint a kimenő levelek továbbküldéséhez ismerni kell a célszerverek IP címeit. A vállalaton belül melegtartalékoltan *2 DNS szerver* van (elsődleges és másodlagos).
- A belső hálózatról az internet elérése egy *útválasztón* (router) keresztül történik. A vállalatnak 2 internet szolgáltatóval van kapcsolata, ezek mint *elsődleges* illetve *másodlagos internet kapcsolat* használhatók. Az összeköttetések megbízhatósági adatai ismertek.

Mivel a kimenő levelek további sorsáról nincs ismeretünk, a megbízhatósági analízist eddig a lépésig végezzük el.

Az egyes szolgáltatások megbízhatósági adatai a következők (órában mérve):

	Munka-állomás	Belső hálózat	Mail server	DNS szerverek	Útválasztó (router)	Elsődleges internet kapcsolat	Másodlagos internet kapcsolat
MTTF	4500	45000	4000	4500	9000	13500	5400
MTTR	2	3	2	2	1	4	6

Feladatok:

a) Rajzoljuk meg a rendszer megbízhatósági blokkdiagramját!

b) Számítsuk ki a levelezés szolgáltatás rendelkezésre állását (egy munkatárs szemszögéből) a fenti adatokat felhasználva!

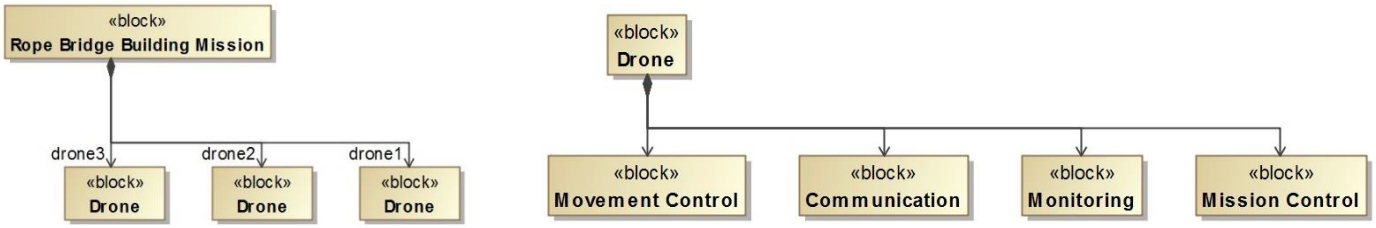
c) Adjuk meg, átlagosan hány óra kiesésre kell számítanunk egy évben!

Informatikai rendszertervezés (VIMIAC01)		VIZSGA	2019. 01. 14.
Név:		NEPTUN:	

3. Struktúramodellezés / 6

Kötélhidat építő drónokat szeretnénk tervezni. Ehhez már elkészült a drónok funkcionális dekompozíciója, illetve tudjuk, hogy három drónra lesz szükségünk (lásd mellékelt BDD-k). Következő lépésben egy drón magas szintű funckionális architektúráját szeretnénk megtervezni, illetve a drónok közötti kommunkiációt. A *Mission Control* egy előre megadott lépés sorozat végrehajtásáért felel. A lépés sorozatban lehetnek *checkpointReached* üzenetre való várakozások, amely üzenetben lehetnek más információk is. A drónok a *Communication* modulon keresztül kommunikálnak és adott időközönként valamilyen pozíció információkat is megosztanak magukról broadcast jelleggel.

Rajzolja le a *Drone* és *Rope Bridge Building Mission* IBD-jét és egy BDD-n a szükséges szignálokat és interfészeket. Diagram fejlécre nincs szükség. Elég a szövegben leírt funkcionalitást kidolgozni. Portokat csak ott vegyen fel, ahol szükséges. Neveket rövidíteni lehet, ha az egyértelmű.



4. Viselkedésmodellezés / 6

Feladatunk állapotgépek segítségével megtervezni egy biztonsági vezérlő logikáját. A vezérlőnek három vezérlési tartománya van: bekapcsolt állapotban (normális működés) az **ON** állapotban, kikapcsolt állapotban az **OFF** és hiba bekövetkeztekor az **ERROR HANDLING** (hibakezelés) állapotban van.

A vezérlő bejövő eseményei az alábbiak: *on, off, err, succ*.

A vezérlő az alábbi kimeneteket adhatja: *repair, return*.

a) A vezérlő kikapcsolt állapotból indul és onnan csak az *on* esemény hatására kezdi el a normális működés. Bármikor *off* esemény érkezik, kikapcsol. Ha egy *err* esemény érkezik, akkor átlép a hibakezelés állapotba és a kimenetén kiküld egy *repair* eseményt! A hibakezelés állapotból a *succ* esemény hatására egy *return* esemény generálódik és a rendszer visszatér a normális működési tartományba. Egészítse ki az alábbi állapotgép-vázat, hogy megfeleljen a specifikációnak!

Finomítsuk/módosítsuk a működést úgy, hogy már alább eseményeket <u>is</u> felhasználhatjuk:

Bejövő események: *a, b.*

Kimenő események: *ok, nok*.

b) Egészítse ki az **ON** állapotban található állapotgépet annak érdekében, hogy megvalósítsa az alábbi funkciót! A megvalósított funkció *a* és *b* események megfelelő sorrendben történő beérkezését vizsgálja és jelez vissza az *ok* és *nok* üzenetek segítségével. Az *a* és *b* üzeneteknek egymás után váltakozva kell bekövetkezniük, amelyre *ok* üzenet kiküldésével válaszolunk. Ha egy *a* üzenet után további *a* üzenetek érkeznek, ilyenkor *nok* üzeneteket küldünk ki válaszként egészen a következő *b* beérkező üzenetig. Ha a *b* üzenetből érkezik több egymás után anélkül, hogy *a* jönne, akkor nem adunk ki kimenetet, csak várunk a következő *a* üzenetre. A normál működés egy *a* üzenet beérkezésével kell, hogy kezdődjön!

c) Az **ERROR HANDLING** állapot finomításával tervezzen olyan funkciót, amely felügyeli, hogy pontosan háromszor próbálja a rendszer a *repair* esemény segítségével megjavítani az állapotát, és amennyiben mindhárom *repair* esemény után *err* válasz érkezett, kikapcsolja magát a vezérlő. Figyeljen arra, hogy a hibakezelés állapotba lépéskor is keletkez(het)nek *repair* esemény(ek)!

d) Módosítsa úgy a b) feladatrészben elkészített funkciót, hogy a funkció végrehajtását mindig ugyanonnan folytassa a rendszer, ahol legutóbb abbahagyta!