UML - Diagram nasazení

UML Deployment Diagram

Martin Komárek

Diagram nasazení

- popisuje:
  - HW architekturu systému
  - nasazení SW na HW
- základním elementem je UZEL (NODE)
- základní vazbou je ASOCIACE (ASSOCIATION)

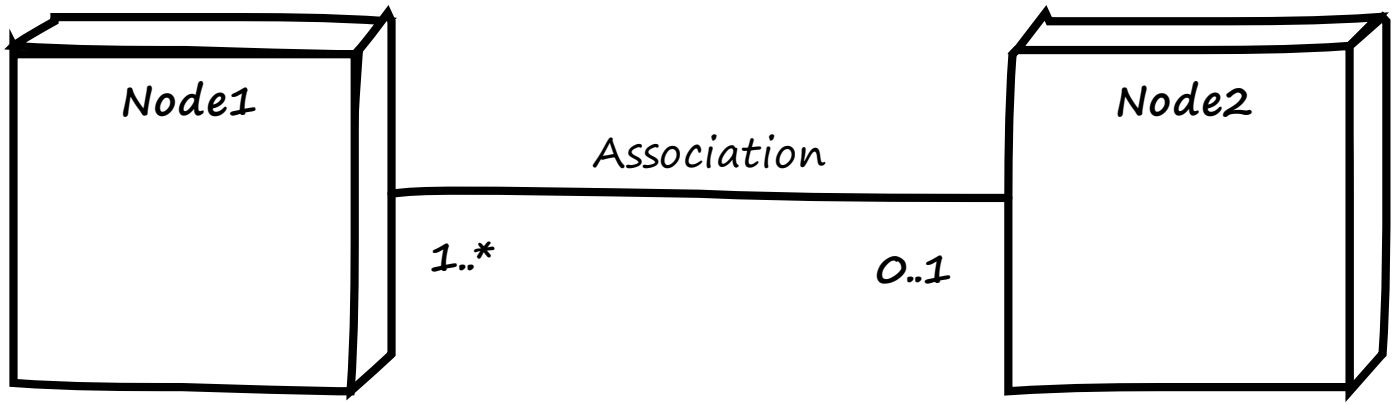
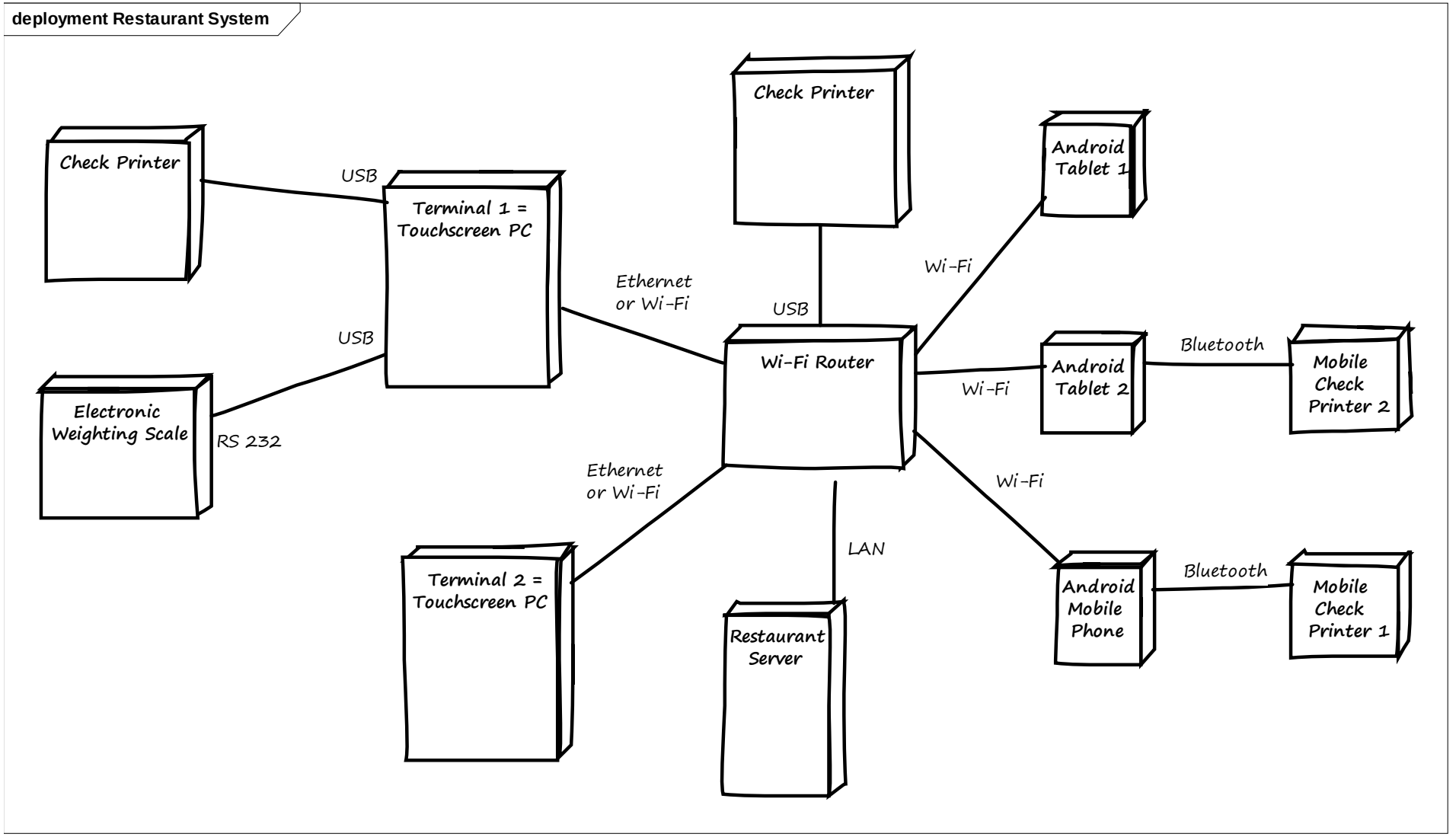
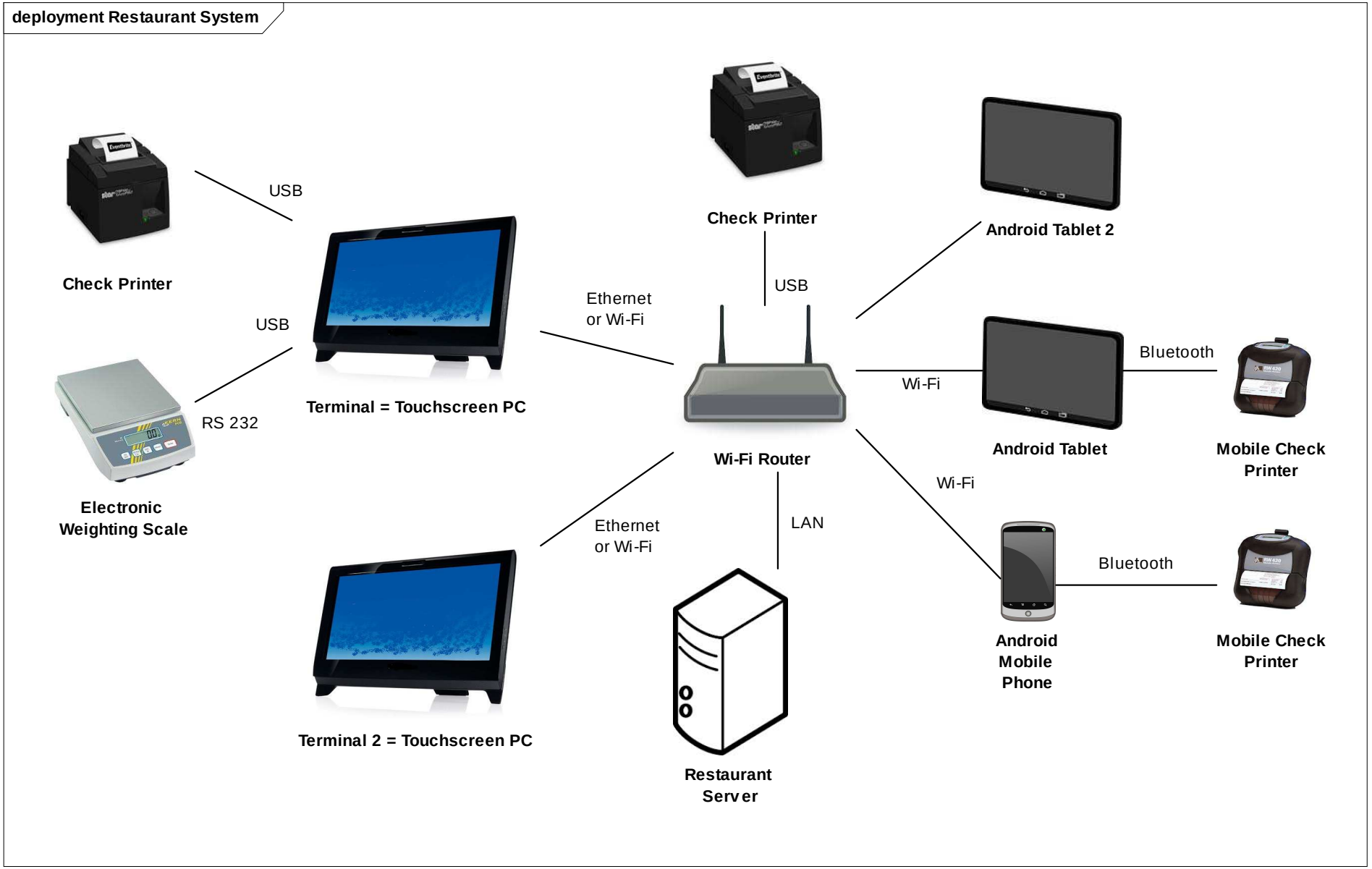


Diagram nasazení – HW architektura



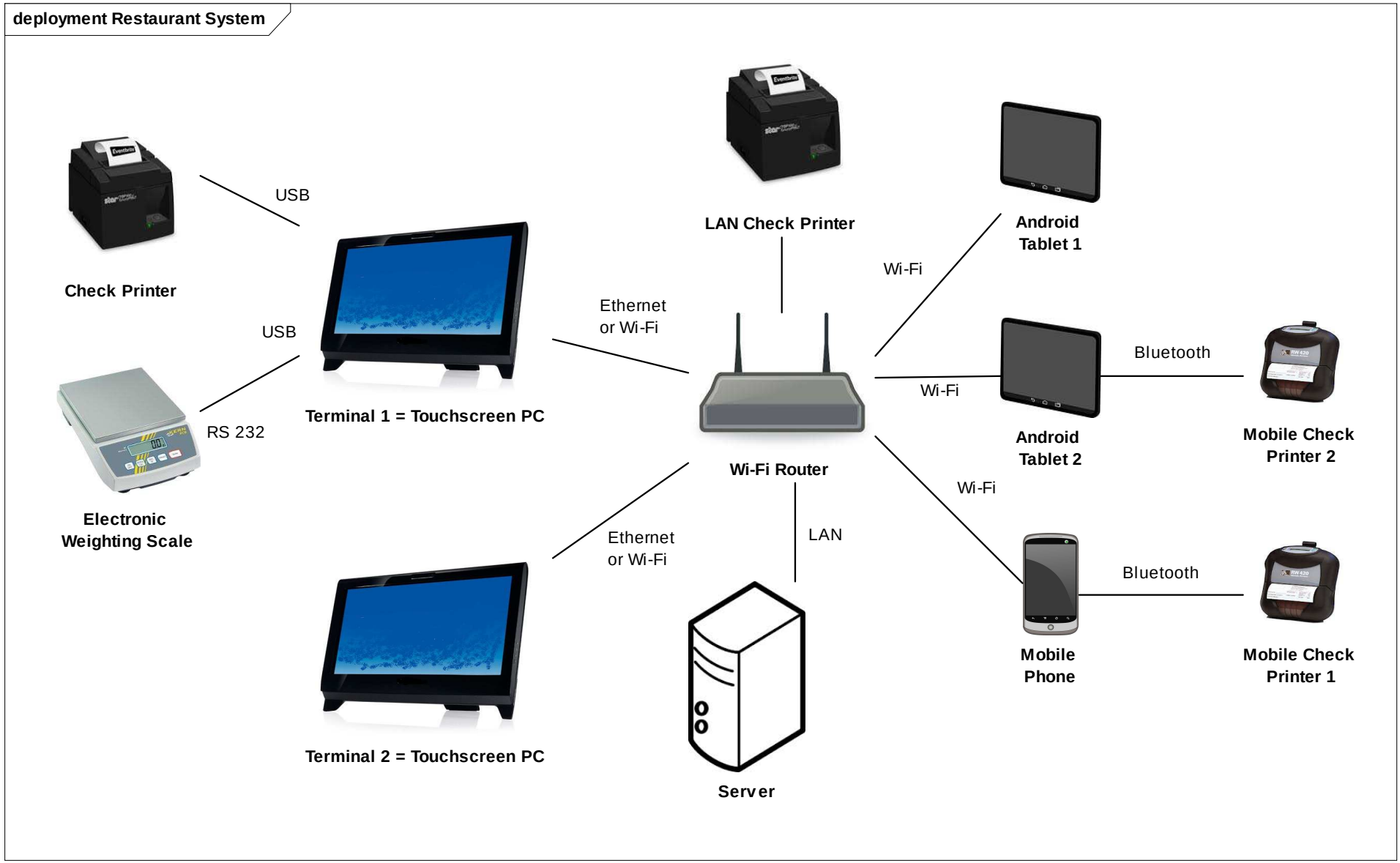
Pozor! Není zcela správně dle UML, vysvětlení později.

Diagram nasazení – vlastní ikony



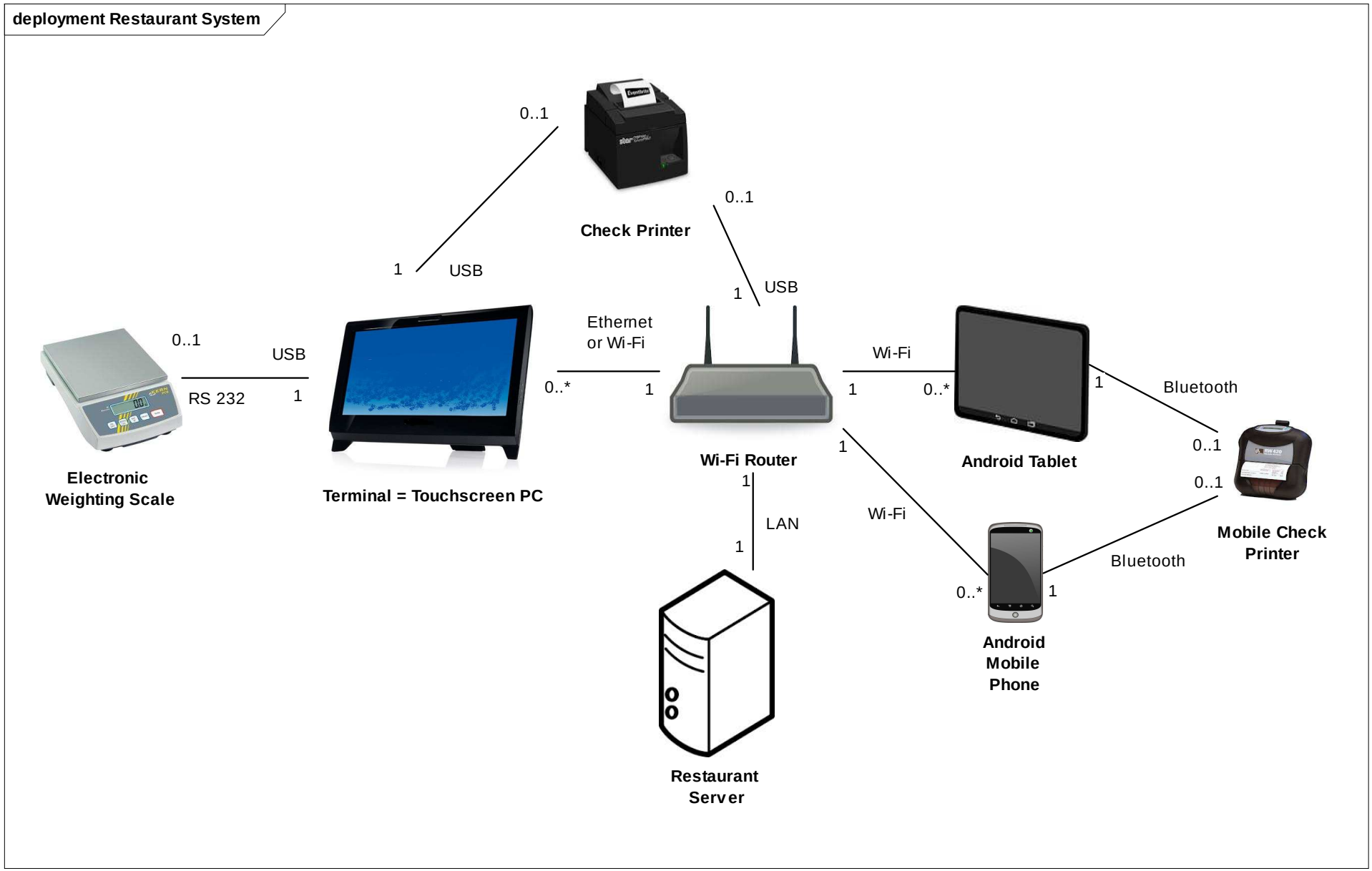
Pozor! Není zcela správně dle UML, vysvětlení později.

Diagram **konkrétního** nebo **obecného** nasazení?



Pozor! Není zcela správně dle UML, vysvětlení později.

Diagram nasazení – správné použití násobností



Správně dle UML, ale pro neznalce UML hůře pochopitelné.

Diagram nasazení – klasifikátory a instance uzlů

- obdoba tříd a objektů

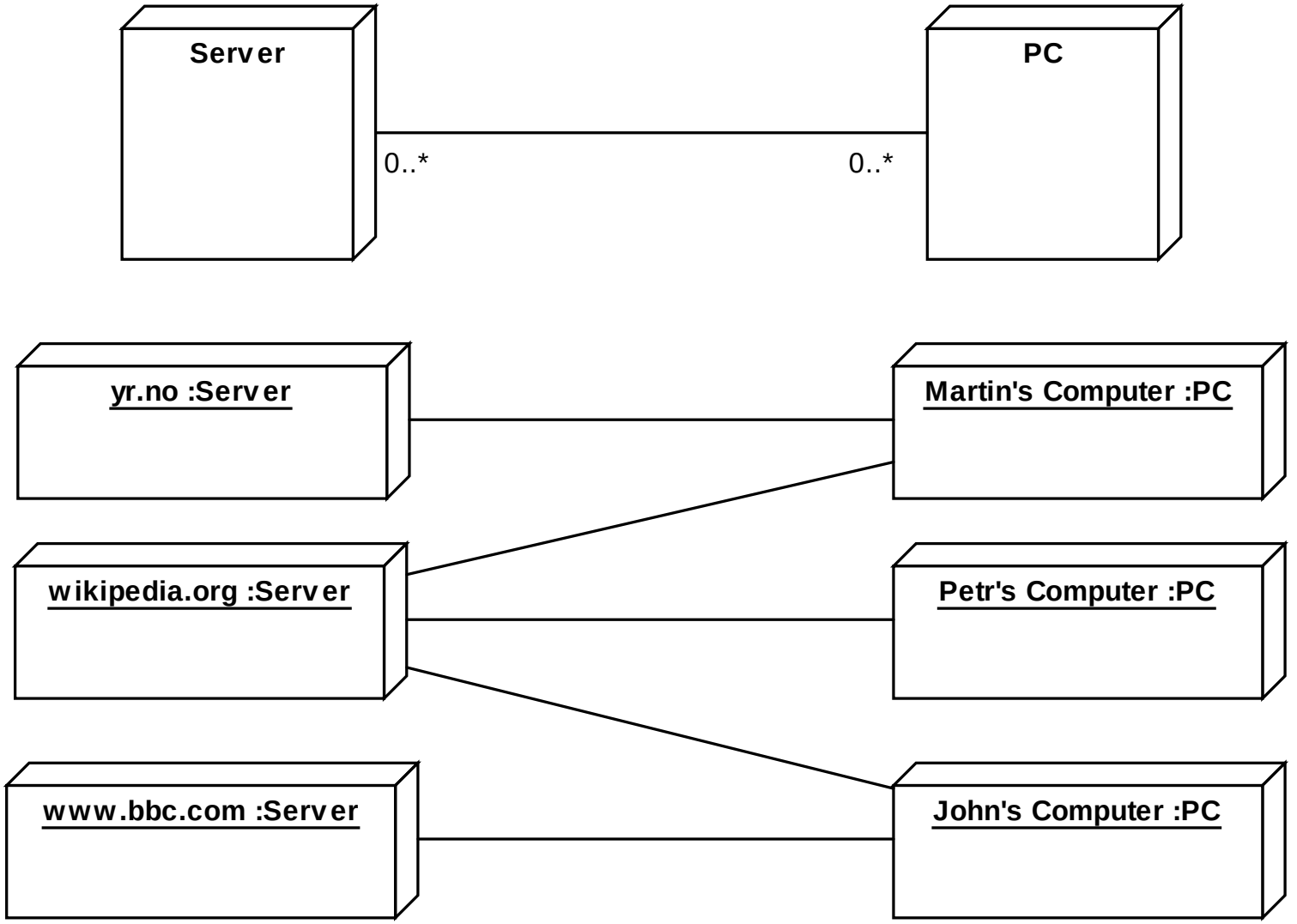


Diagram (konkrétního) nasazení – instance uzlů

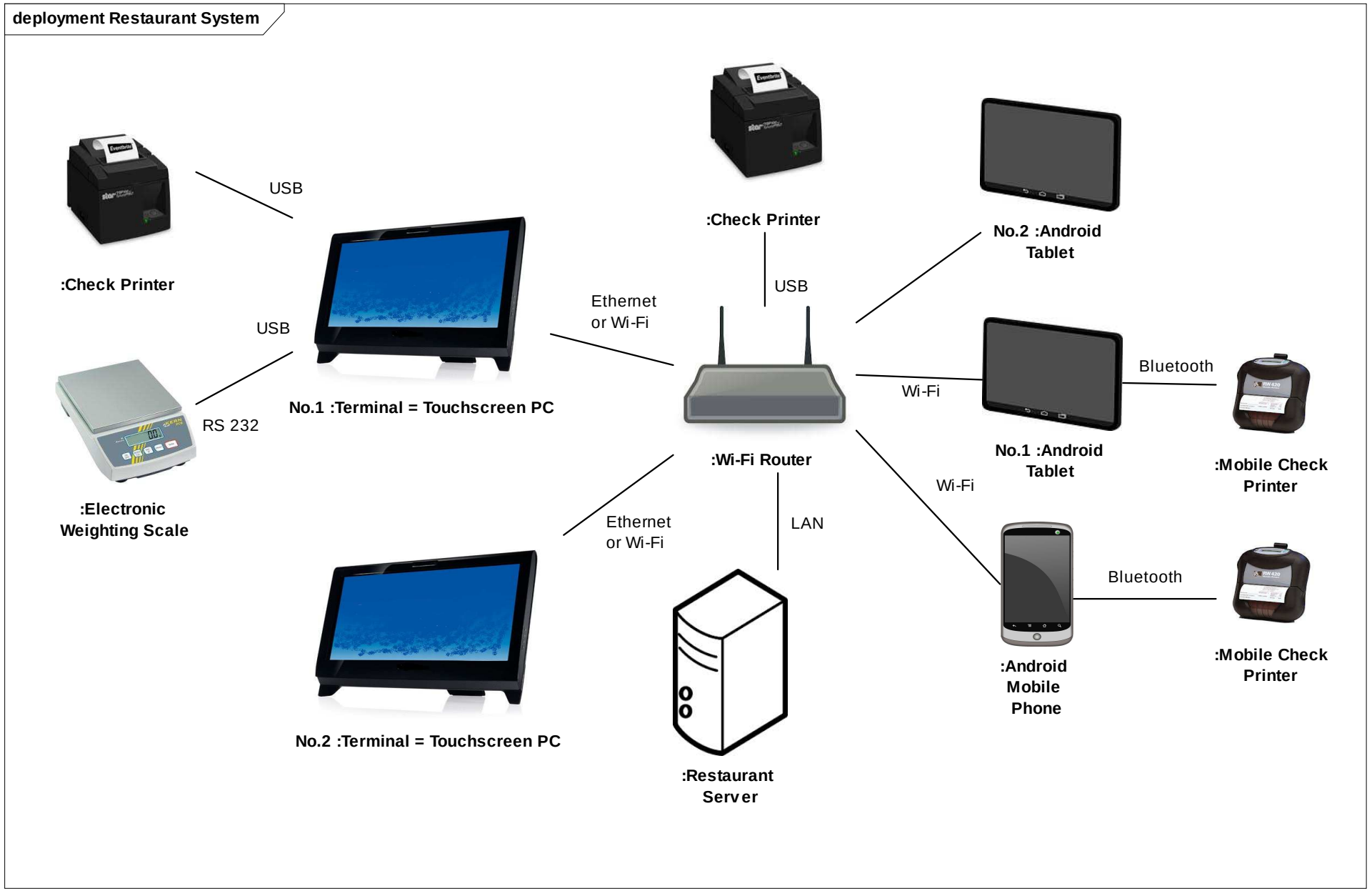


Diagram (konkrétního) nasazení – instance uzlů

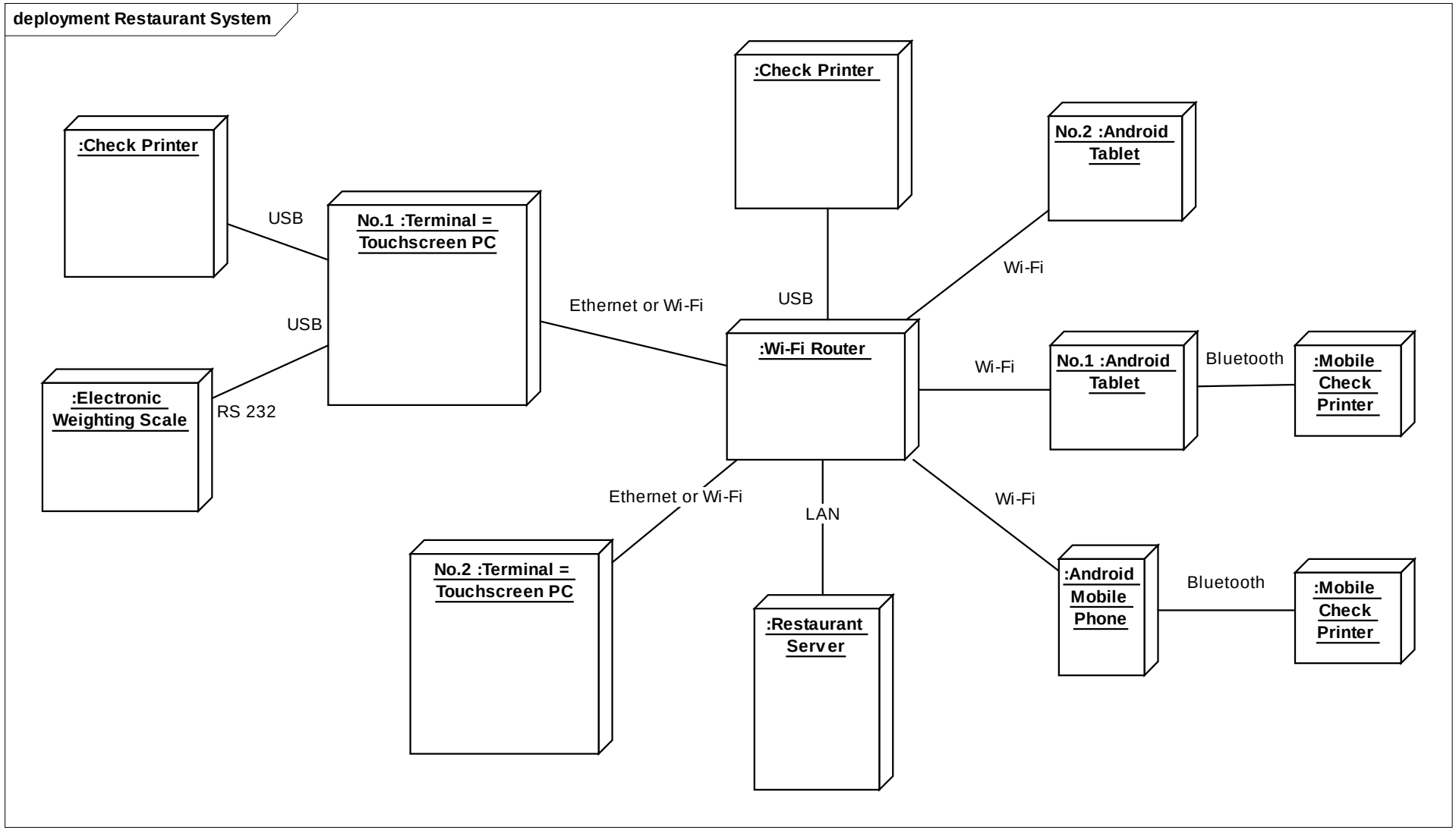
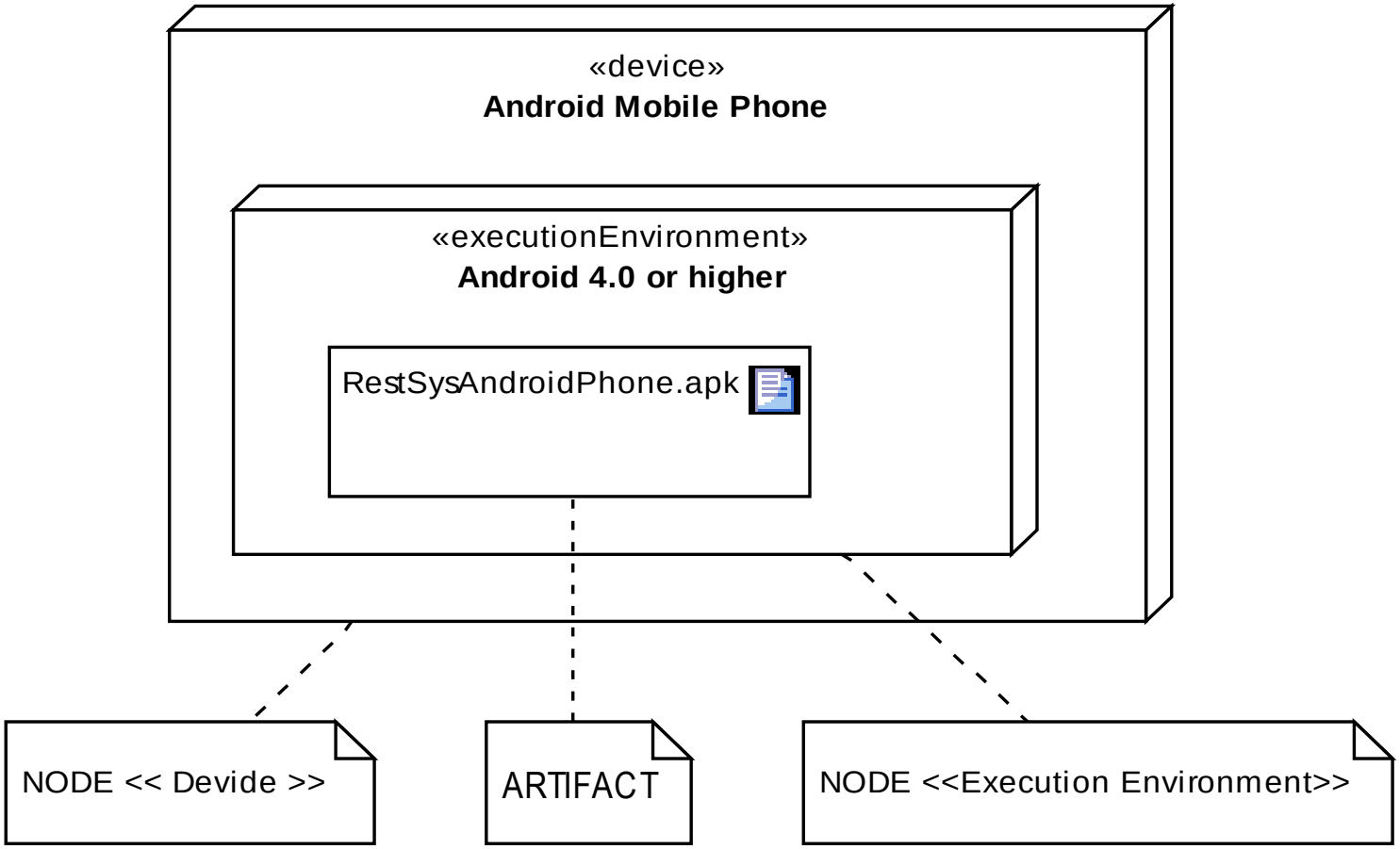


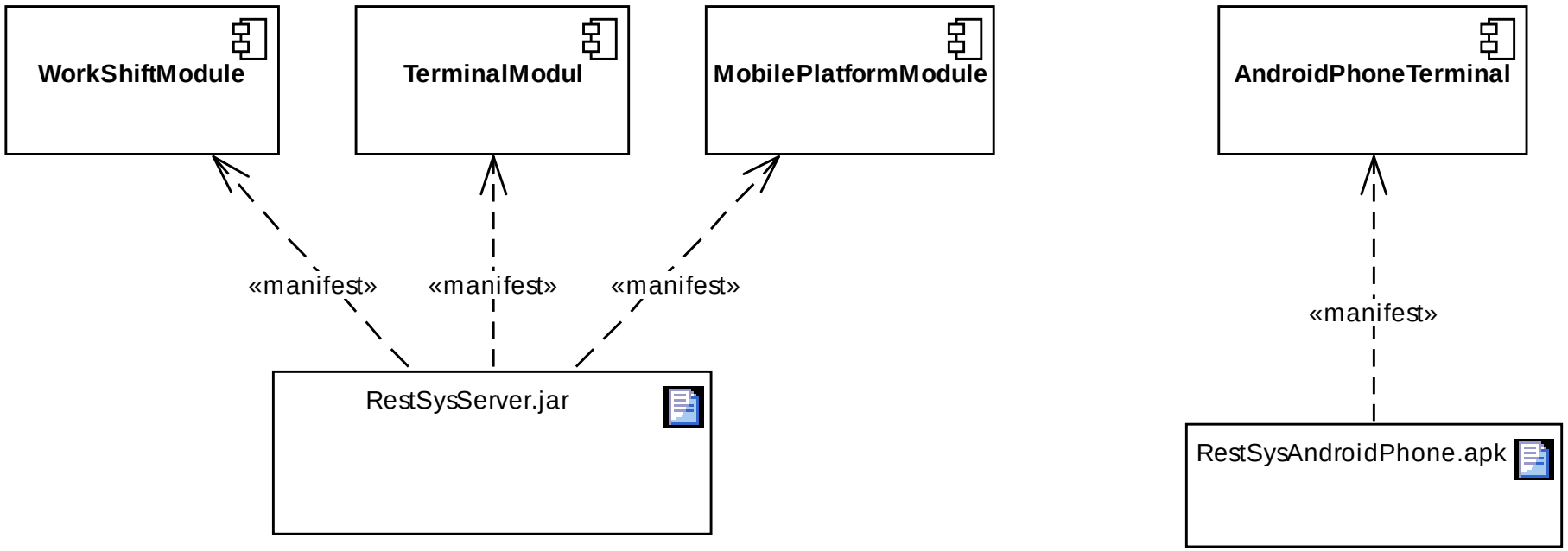
Diagram nasazení – umístění SW na HW

- grafický instalační manuál
- umístění jednotlivých SW částí a popis jejich vztahů



Artifact - Artefakt

- obvykle fyzickým projevem jedné či více komponent



Artifact - Artefakt

- standard UML má předdefinované stereotypy:

artifact stereotype	Význam stereotypu
«file»	fyzický soubor
«deployment spec»	specifikace nasazení (e.g. web.xml in J2EE)
«document»	dokument
«executable»	spustitelný soubor
«library»	statická nebo dynamicky link. knihovna (DLL) nebo Java Archive (JAR)
«script»	skript proveditelný pomocí interpretru
«source»	zkompilovatelný zdrojový soubor

- možno nadefinovat si vlastní stereotyp dle potřeby

Diagram nasazení – umístění SW na HW

- hierarchie pomocí vazby Dependency
- možno pojmenovat nebo vytvořit vlastní stereotyp

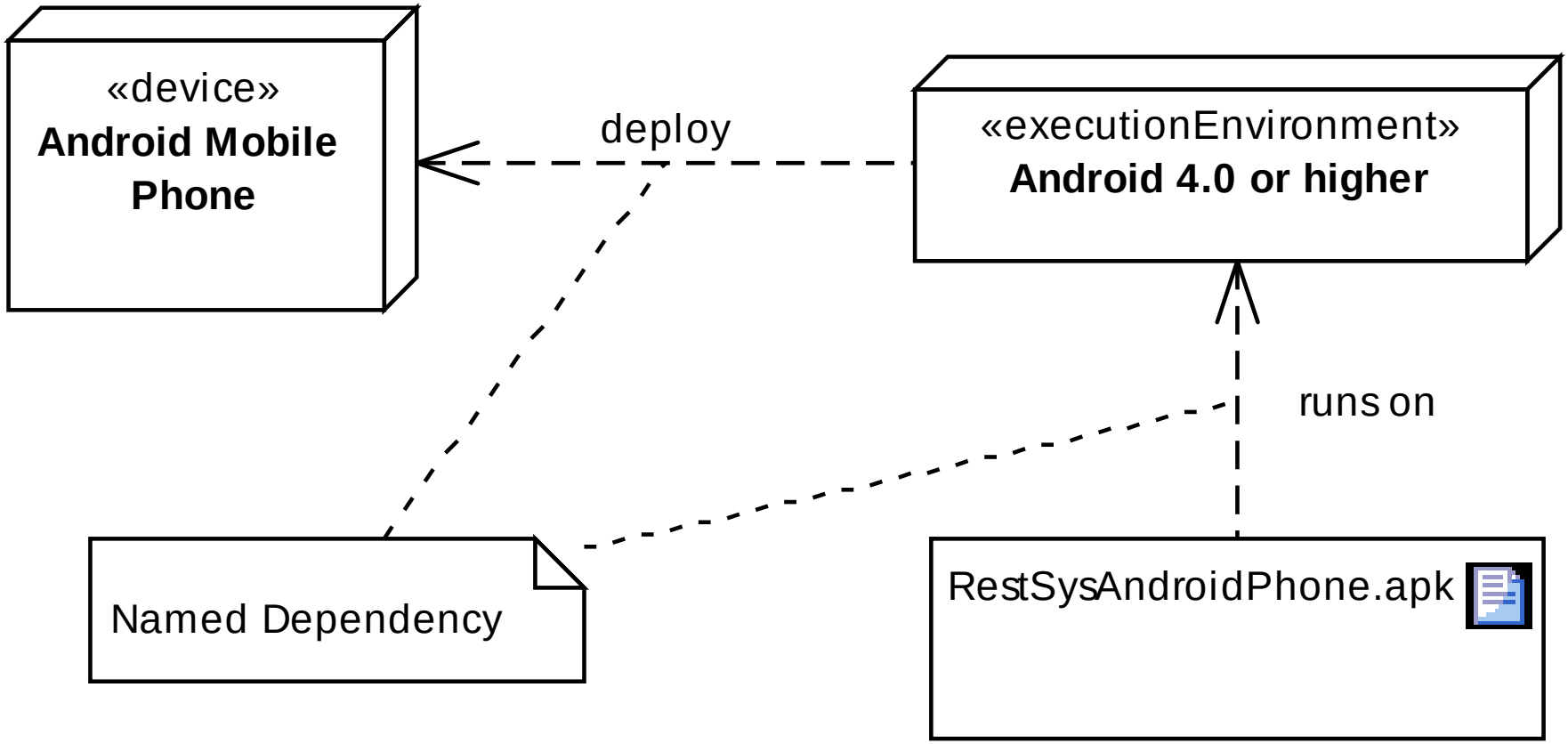


Diagram nasazení – umístění SW na HW

- hierarchie pomocí vazby Dependency
- možno pojmenovat nebo vytvořit vlastní stereotyp

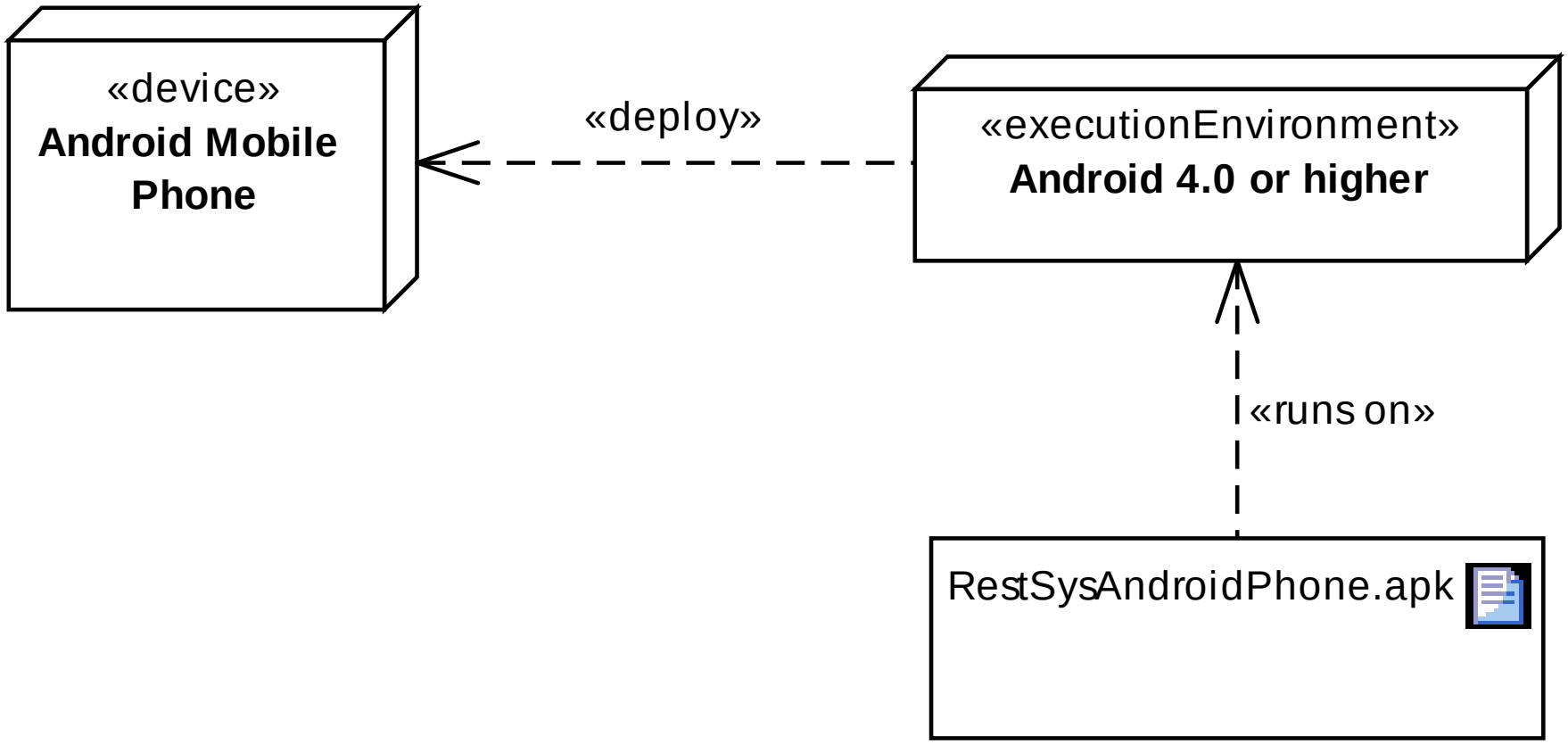
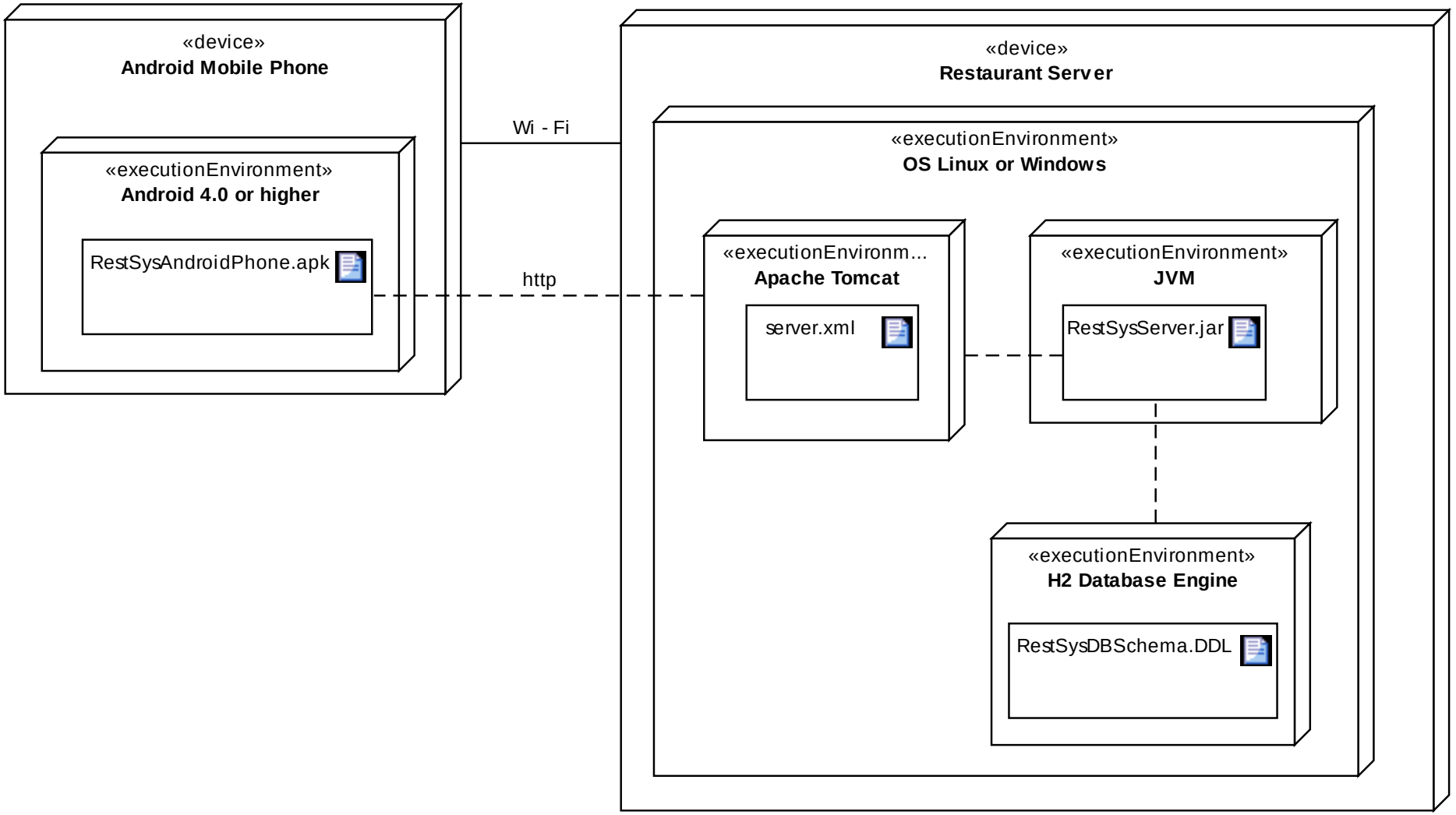


Diagram nasazení – umístění SW na HW



- místo **artefaktů** někdo používá přímo **komponenty**

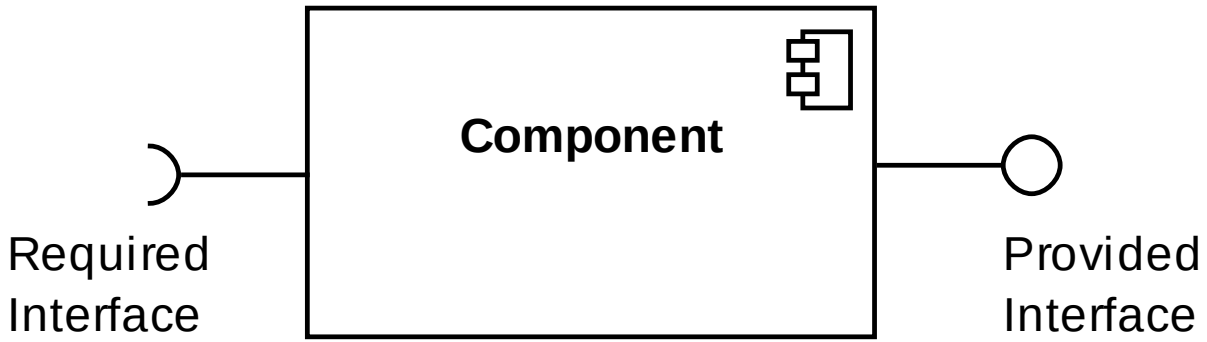
Úvod do
UML Diagram komponent

Intro to
UML Component Diagram

Martin Komárek

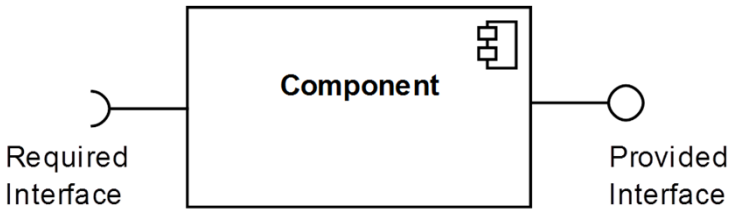
Diagram komponent

- popisuje logickou architekturu systému (tedy rozdělení systému na subsystémy/moduly)
- základním elementem je KOMPONENTA
- chování KOMPONENTY je definováno:
  - POSKYTOVANÝMI ROZHRANÍMI
  - POŽADOVANÝMIROZHRANÍMI



Co je komponenta?

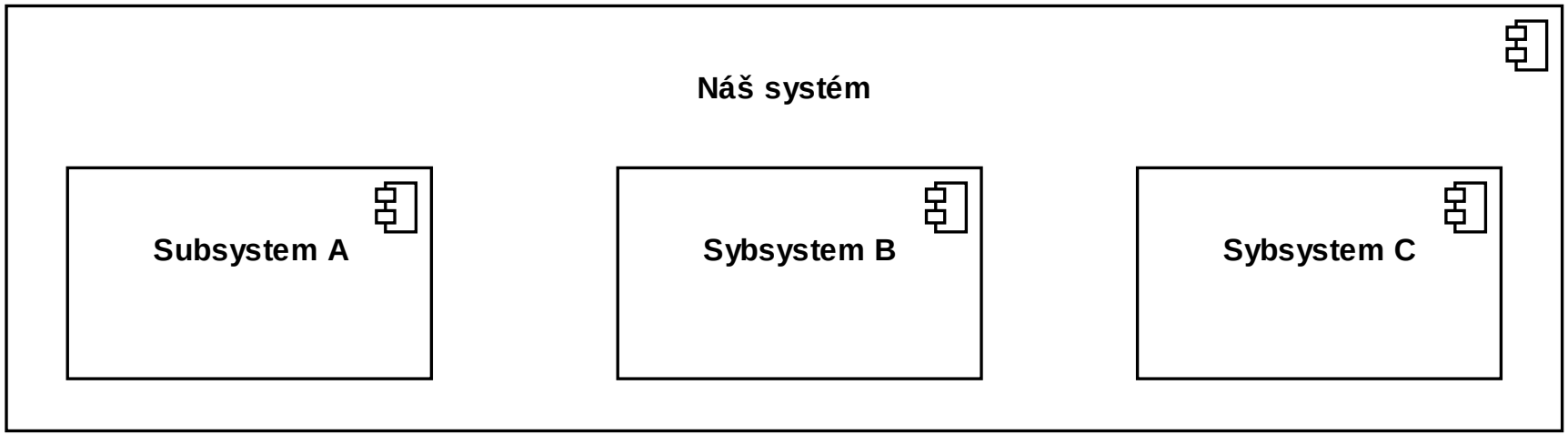
- zastupuje část modulárního systému
- ukrývá (zapouzdřuje) svůj obsah a její chování může být nahrazeno jinou komponentou se stejným chováním
- KOMPONENTY je „černá skříňka“ a její chování je definováno:
  - POSKYTOVANÝMI ROZHRANÍMI
  - POŽADOVANÝMIROZHRANÍMI



- je speciálním případem TŘÍDY

Komponenta

- zastupuje část modulárního systému
- může obsahovat další komponenty
- DIAGRAM KOMPONENT tedy může popisovat logickou architekturu systému



Rozhraní komponent – varianty zobrazení

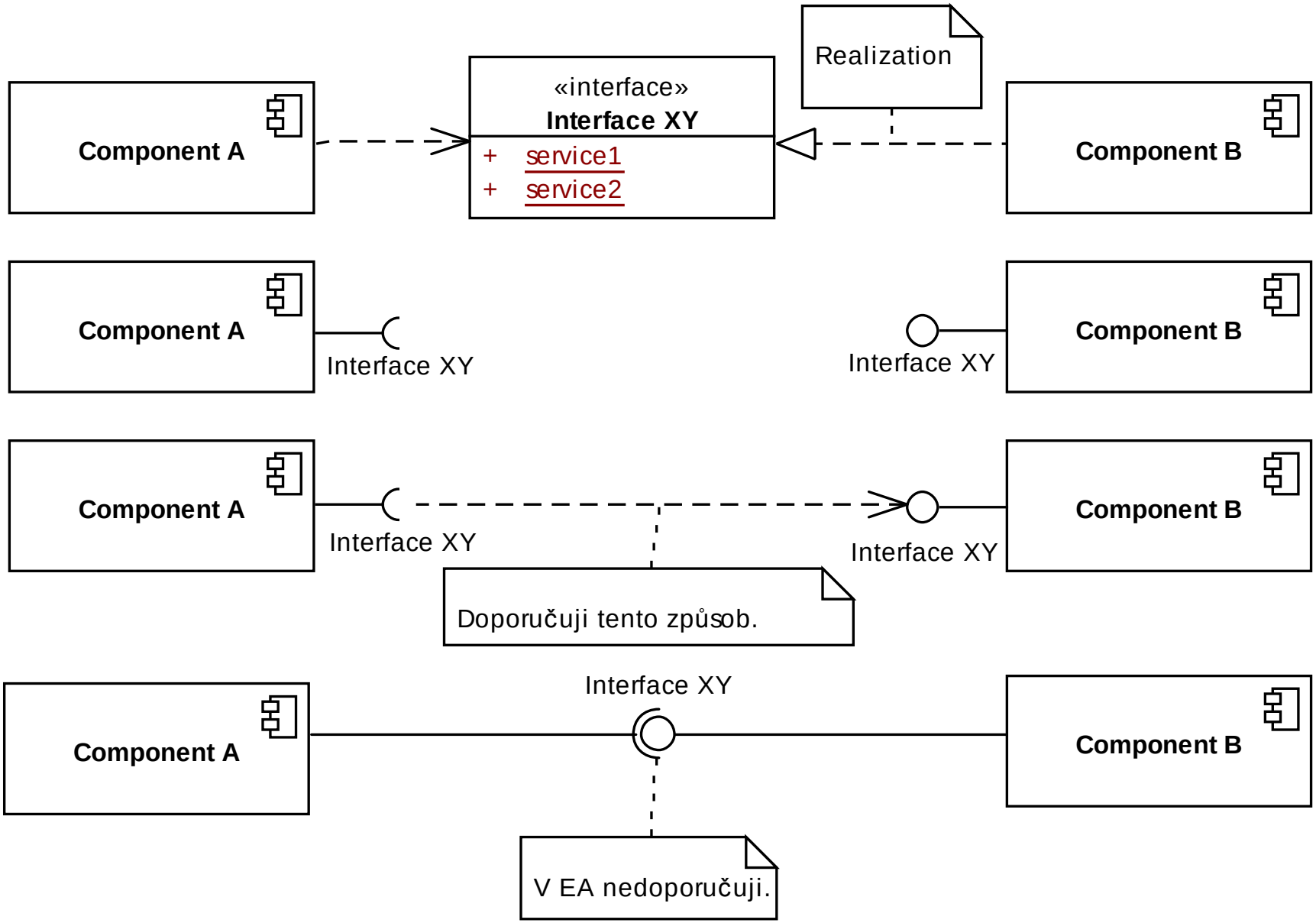


Diagram komponent

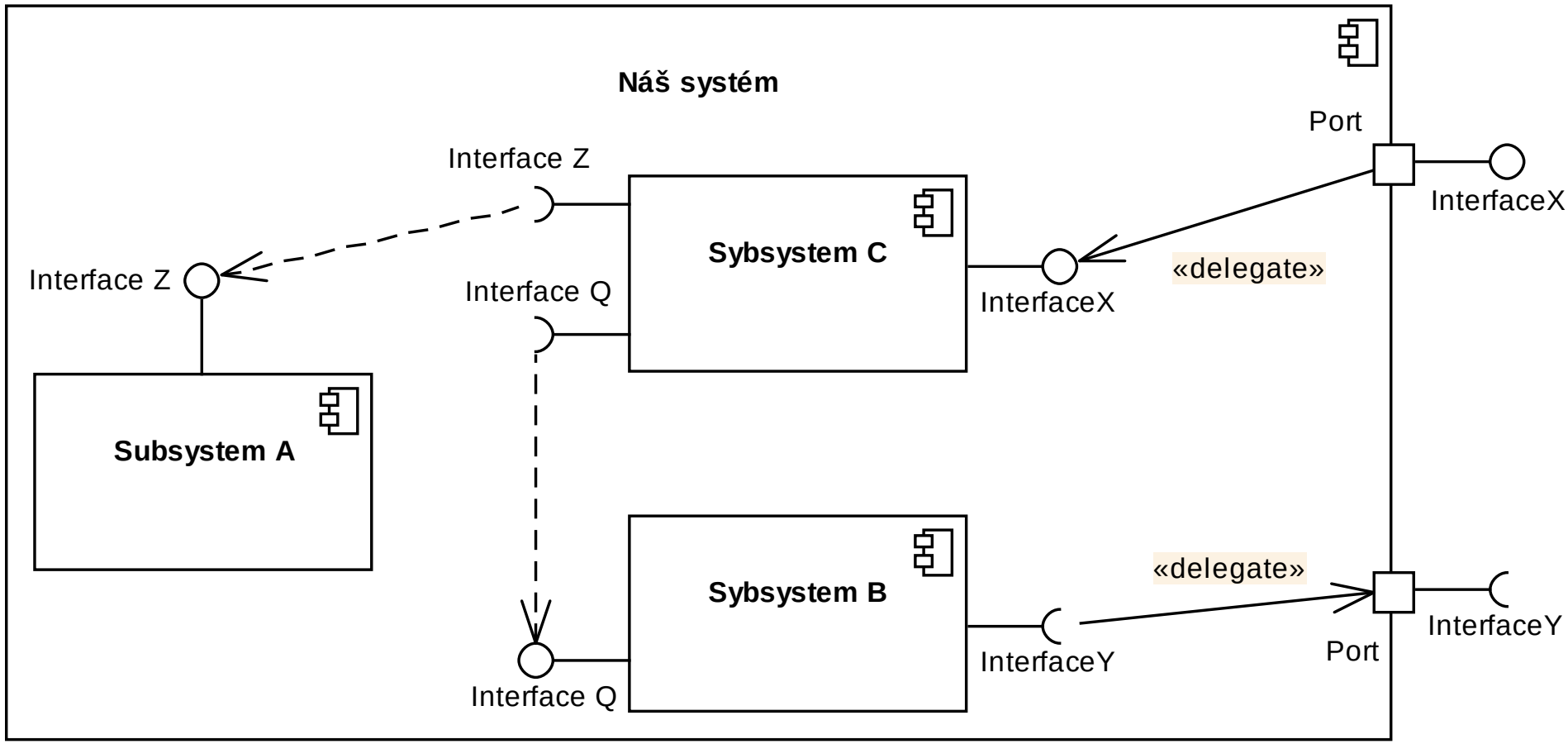
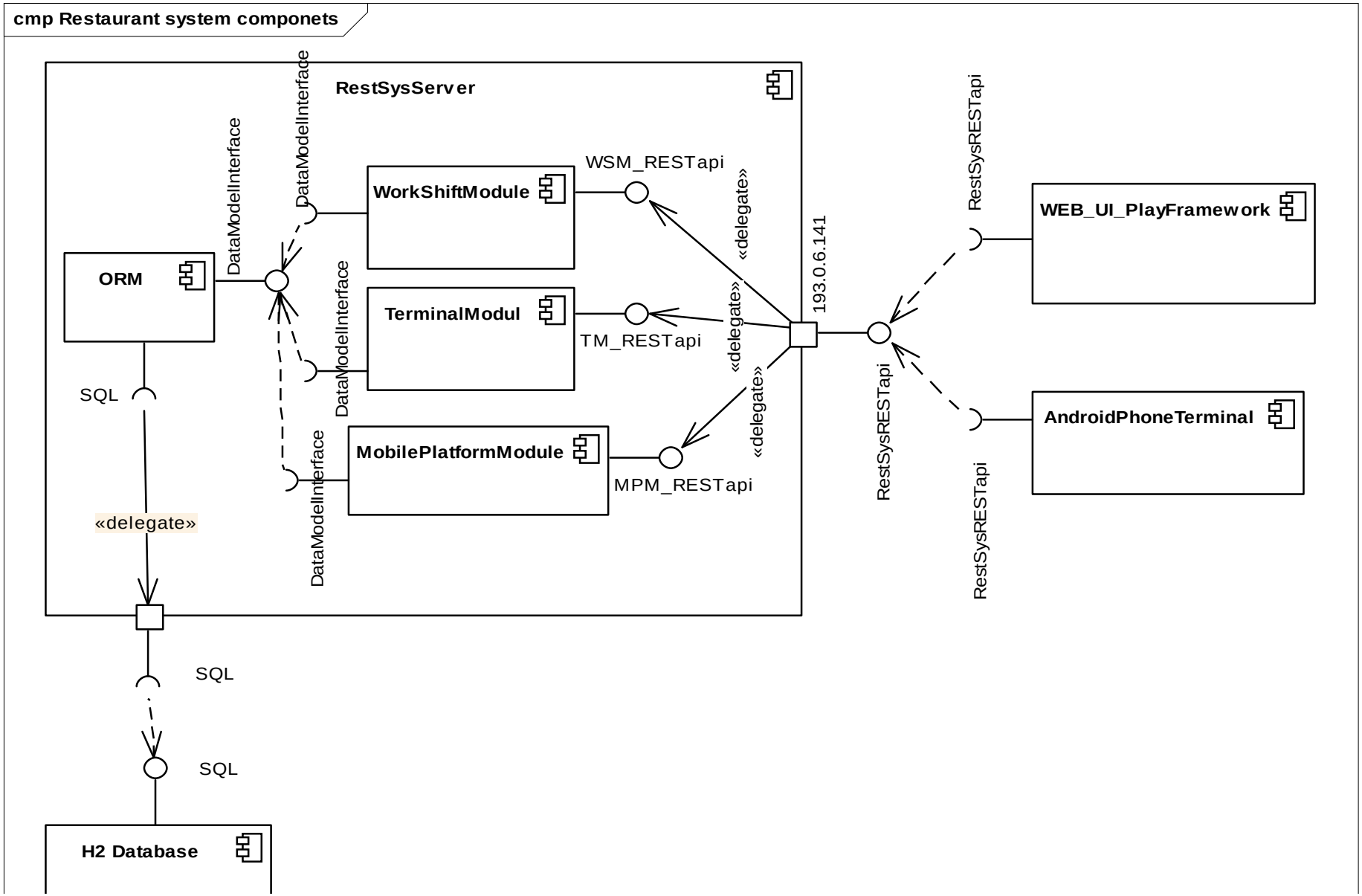
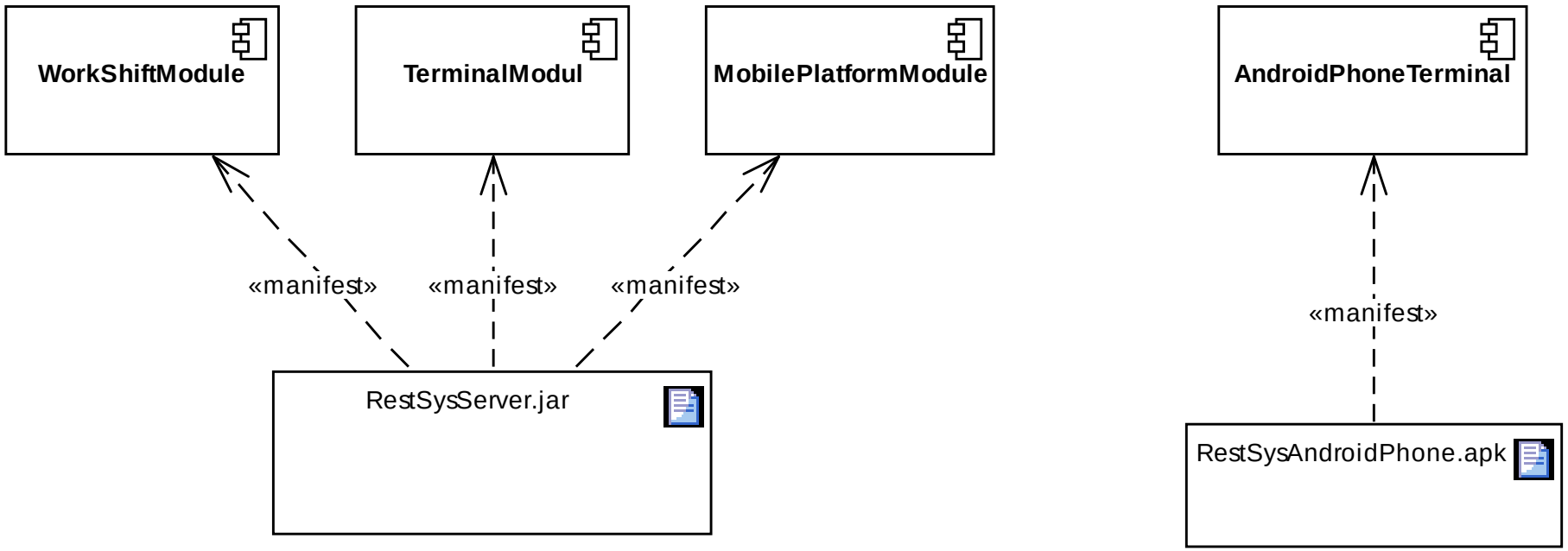


Diagram komponent



Artifact - Artefakt

- je obvykle fyzickým projevem jedné či více komponent



Artifact - Artefakt

- standard UML má předdefinované stereotypy:

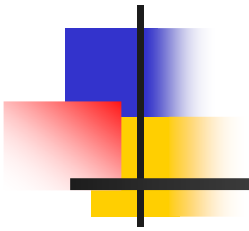
artifact stereotype	Význam stereotypu
«file»	fyzický soubor
«deployment spec»	specifikace nasazení (e.g. web.xml in J2EE)
«document»	dokument
«executable»	spustitelný soubor
«library»	statická nebo dynamicky link. knihovna (DLL) nebo Java Archive (JAR)
«script»	skript proveditelný pomocí interpretru
«source»	zkompilovatelný zdrojový soubor

- možno nadefinovat si vlastní stereotyp dle potřeby

Diagram komponent

Více a přesnější informací naleznete:

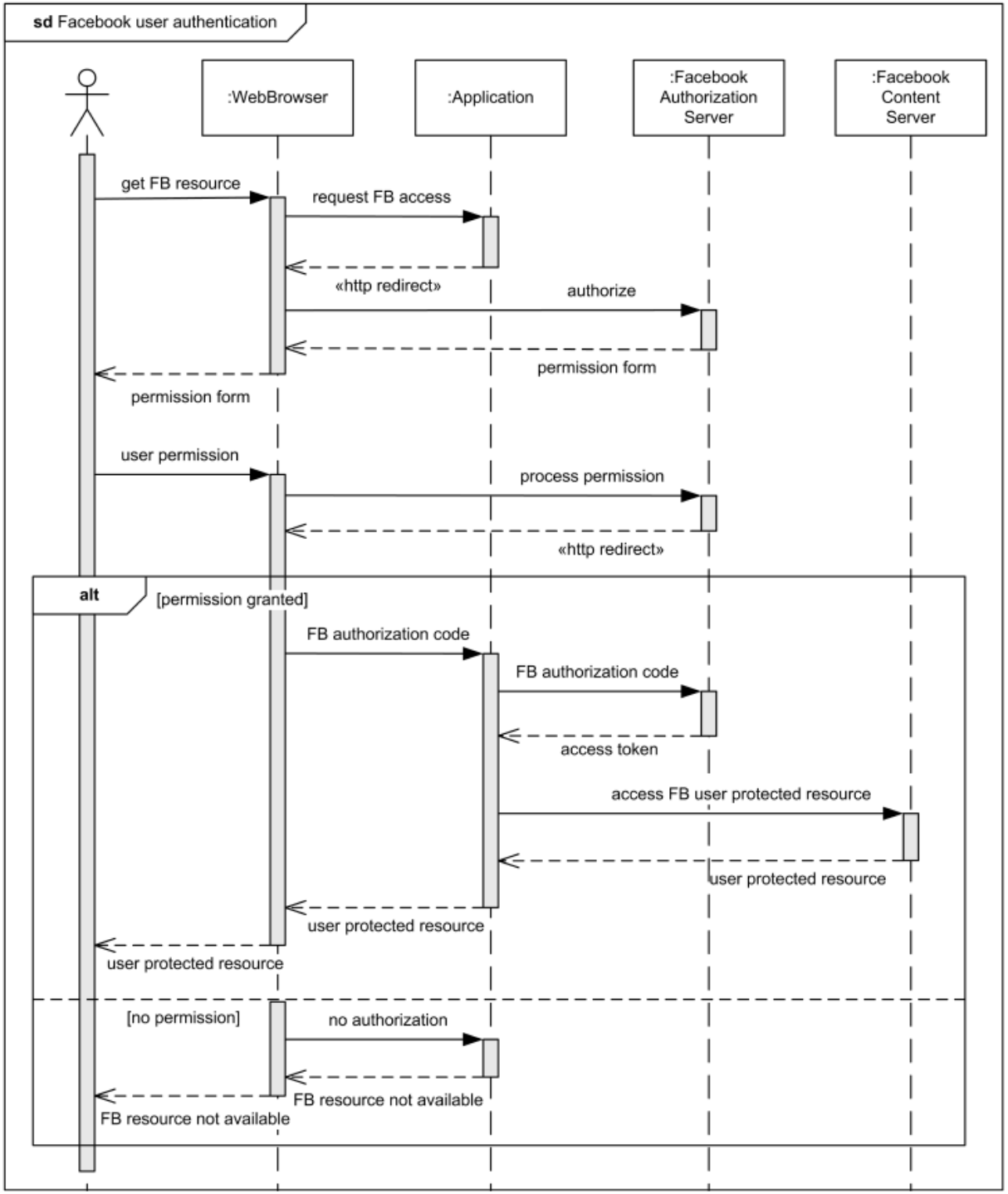
- V knize *UML2 a unifikovaný proces vývoje aplikací* (kapitola 19)
- http://www.uml-diagrams.org/component-diagrams.html
- http://ocup.ocup.cz/search/label/komponenty



Sekvenční diagramy

Martin Komárek

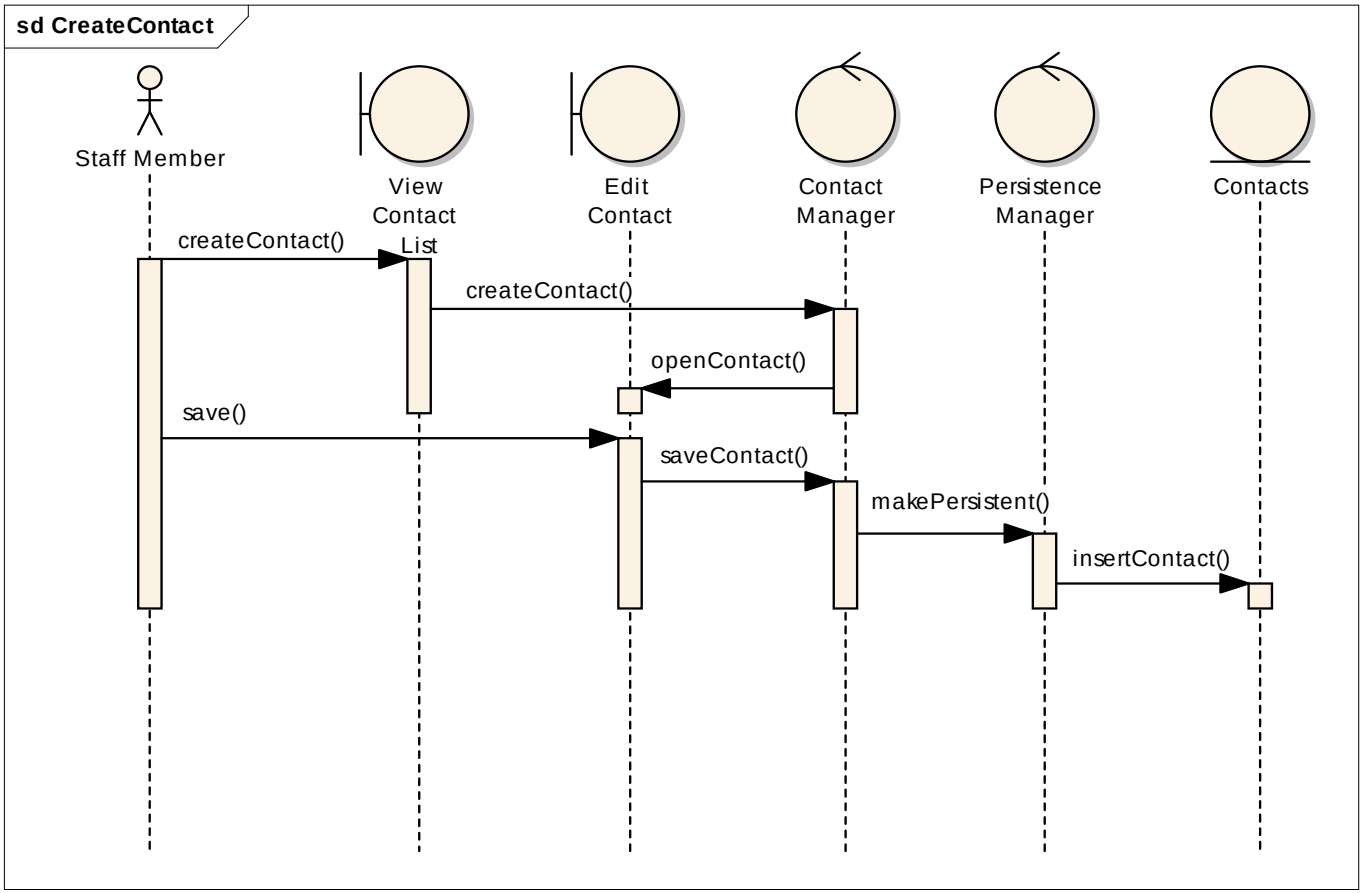
Zdroj: Slides - Arlow J.



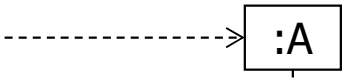
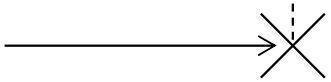
Sekvenční diagram

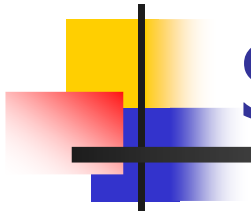
Zdroj: uml-diagrams.org

Sekvenční diagram – použití stereotypů



Zprávy

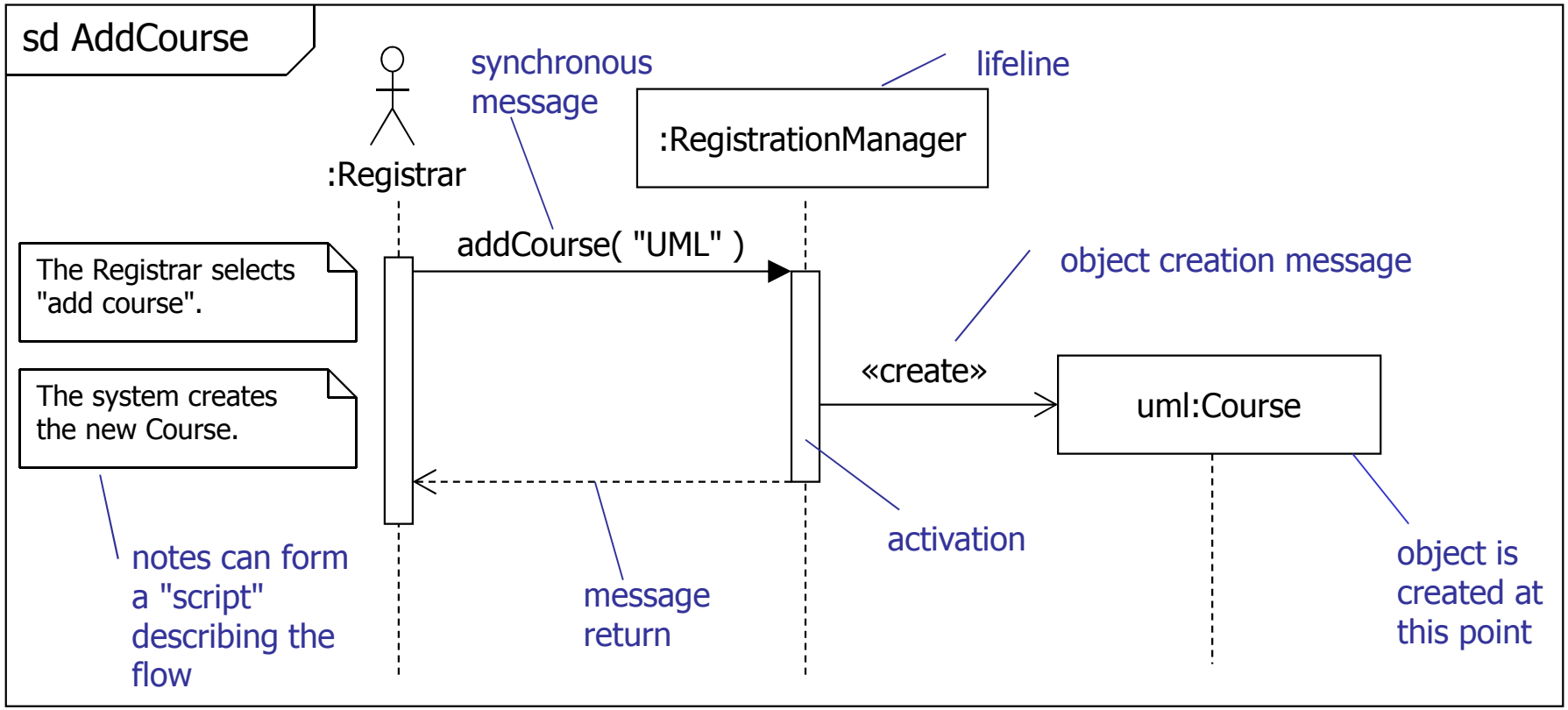
sender → receiver/ target	typ	sémantika
	synchronní	odesílatel čeká na odpověď
	asynchronní	odesílatel nečeká na odpověď
	návrat	returning from a synchronous operation call návrat ze synchronního volání
	konstrukce	odesílatel vyváří cíl
	destrukce	odesílatel destruuje cíl



Sekvenční diagramy

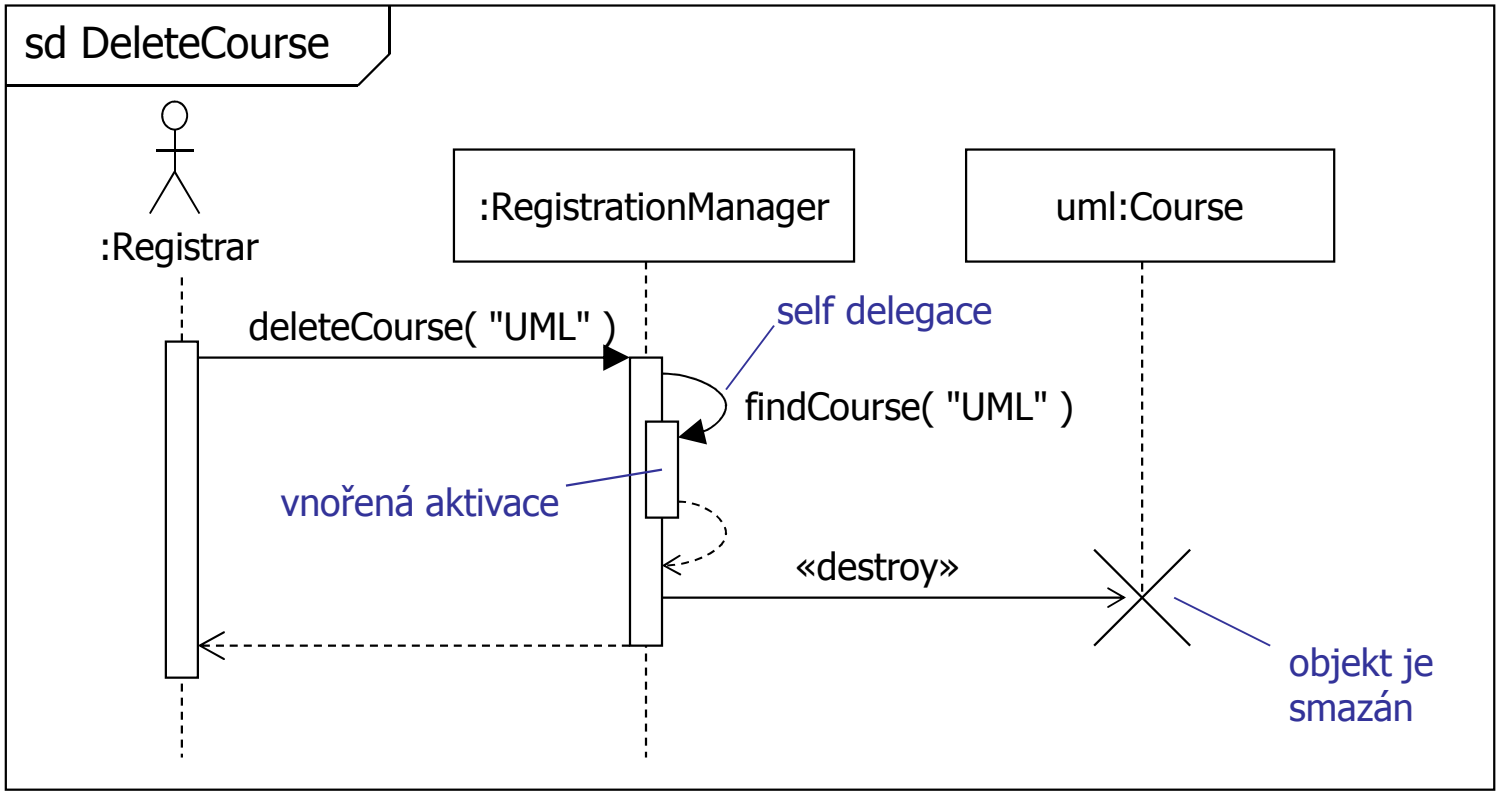
- kladou důraz na časové hledisko
- ukazují
  - kdo (instance tříd) se podílí na realizaci
  - jaké zprávy a s jakými parametry si objekty vyměňují
  - v jakém pořadí

Syntaxe sekvenčních diagramů

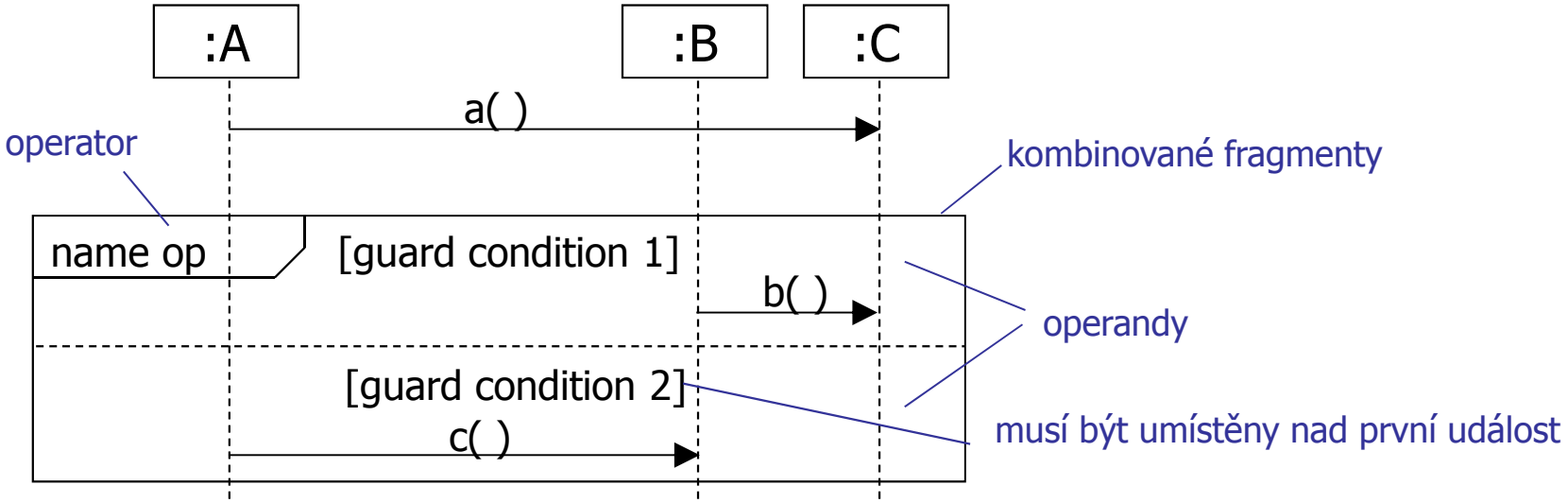


- aktivace nejsou povinné

Smazání and self-delegace



Kombinované fragmenty



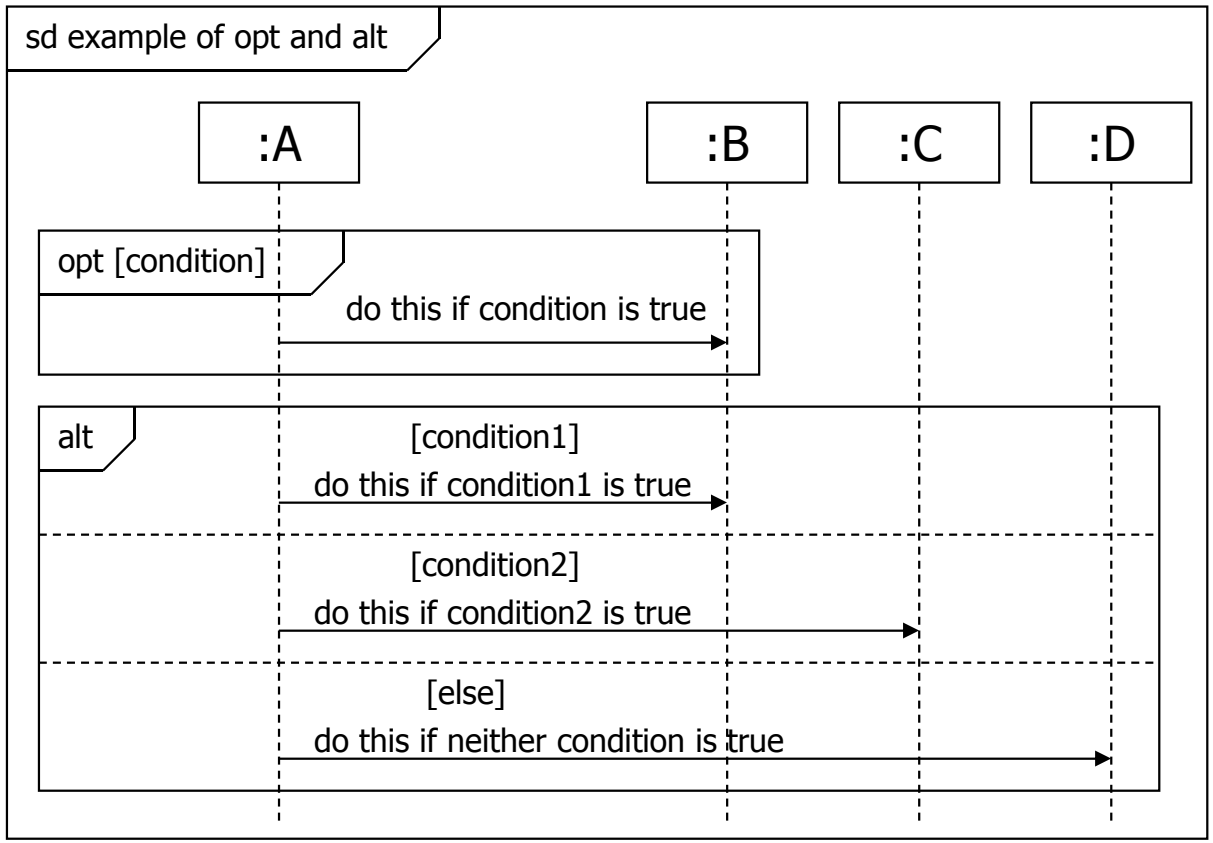
- Operator říká jak (režim) budou operandy spouštěny
- *Guard condition* - podmínka spuštění daného operandu/ů

Běžné operátory

operator	long name	sémantika
opt	Option	if ...then...
alt	Alternatives	switch...case...
loop	Loop	loop min, max [podmínka] Iteruje minimálně *min times,* pokračuje do *max times* dokud podmínka splněna
break	Break	zbytek je přeskočen
ref	Reference	odkaz na jinou interakci= volání procedury

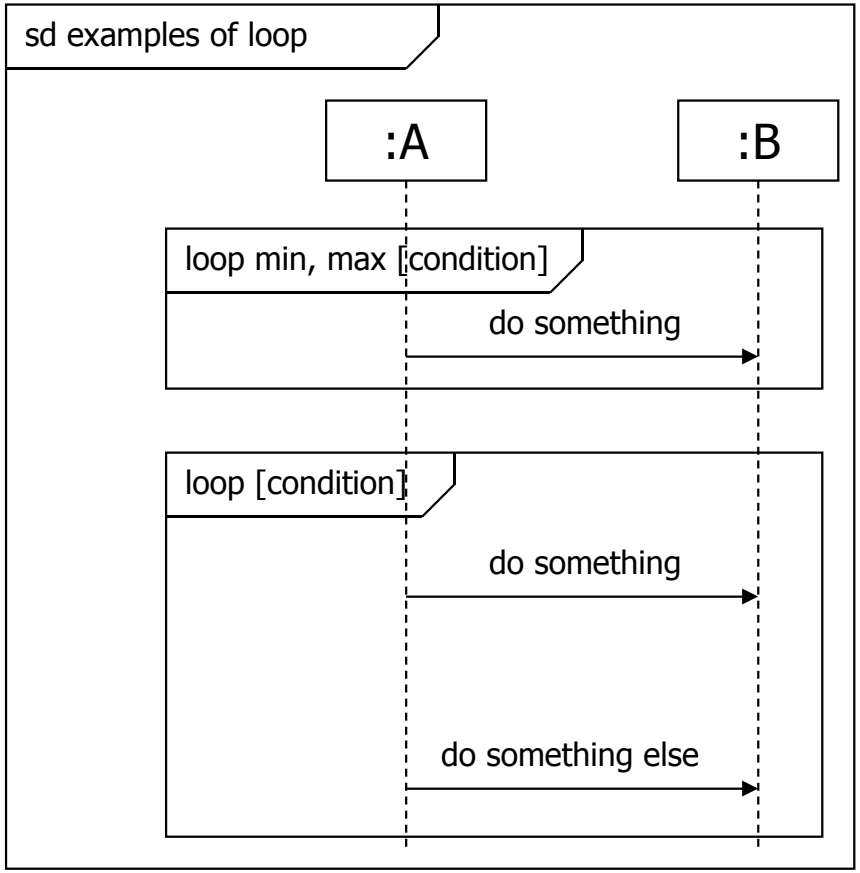
Větvení pomocí opt a alt

- opt :
  - pouze 1 operand
- alt :
  - 2 a více operandů

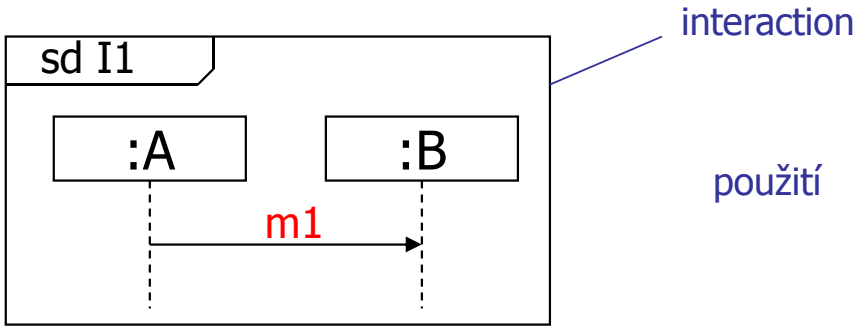


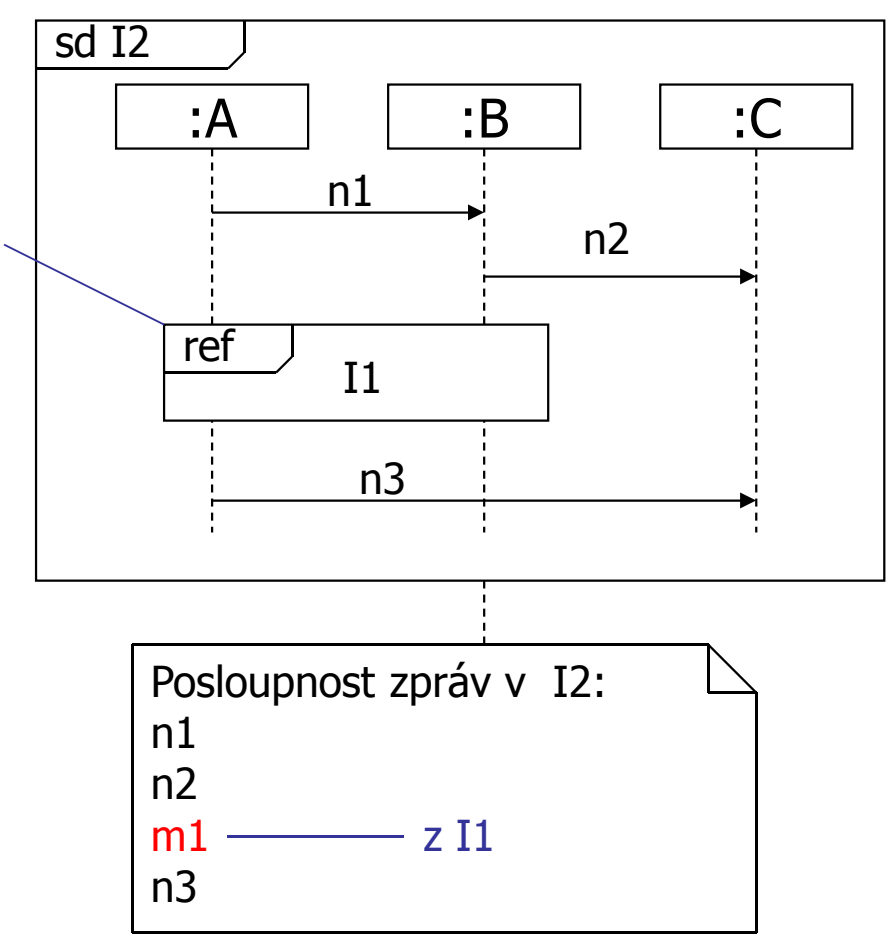
Iterace s loop

- loop sémantika:
  - dělej min krát, potom dělej (max – min) krát dokud podmínka platí



Volání jiné interakce





- Použité lifelines musí být i ve volajícím sd
- Volání protíná lifelines, které používá