Declarative Deployment Model Normal Forms: **Interviews**

Information removed due to Double Blind Review

Purpose and Objectives

- Various kinds of declarative deployment technologies have been developed
  - For example, AWS Cloud Formation, Chef, TOSCA, Kubernetes, etc.
- **Most deployment technologies are extensible in terms of defining new types of components** that can be used in declarative deployment models
- However, defining new component types is a major challenge, as mistakes quickly lead to application failures, unnecessary system downtime, and severe security issues when the application needs to be managed later
  - e.g., if a component needs to be migrated to another cloud

The objectives of this interview are (1) to discuss problems in component type definitions that negatively impact management processes, and (2) to evaluate if the presented approach proposed by this paper solves these problems.

Terminology

- We define the term ***deployment technology*** as *a technology that is able to deploy and orchestrate software components as well as all required middleware and infrastructure components*
  - For example, this includes Terraform, AWS CloudFormation, Chef, but also technologies with other main purposes such as Kubernetes.

- ***Manually modelled management processes*** *are required for executing complex, application-specific management tasks. Such processes need to be manually modelled or implemented by developers in the form of workflows or scripts that execute the required management logic.*

- ***Automatically derived management processes*** *are offered by a deployment or management technology and can be simply started by administrators without the need to implement the processes themselves.*

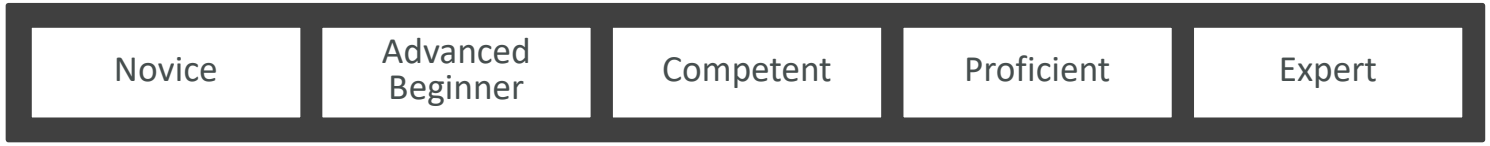
Publication

- Q1: Am I allowed to use the **anonymized** results of this interview in a scientific publication?

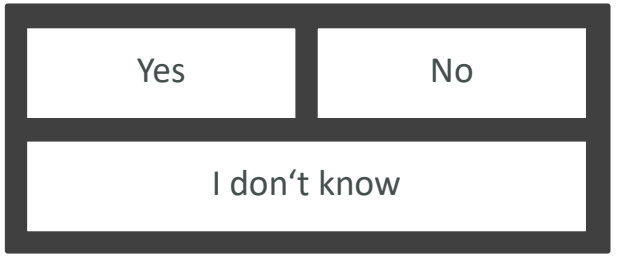
- Q2: Am I allowed to make the **anonymized** results of this interview available online on GitHub?

Personal Information

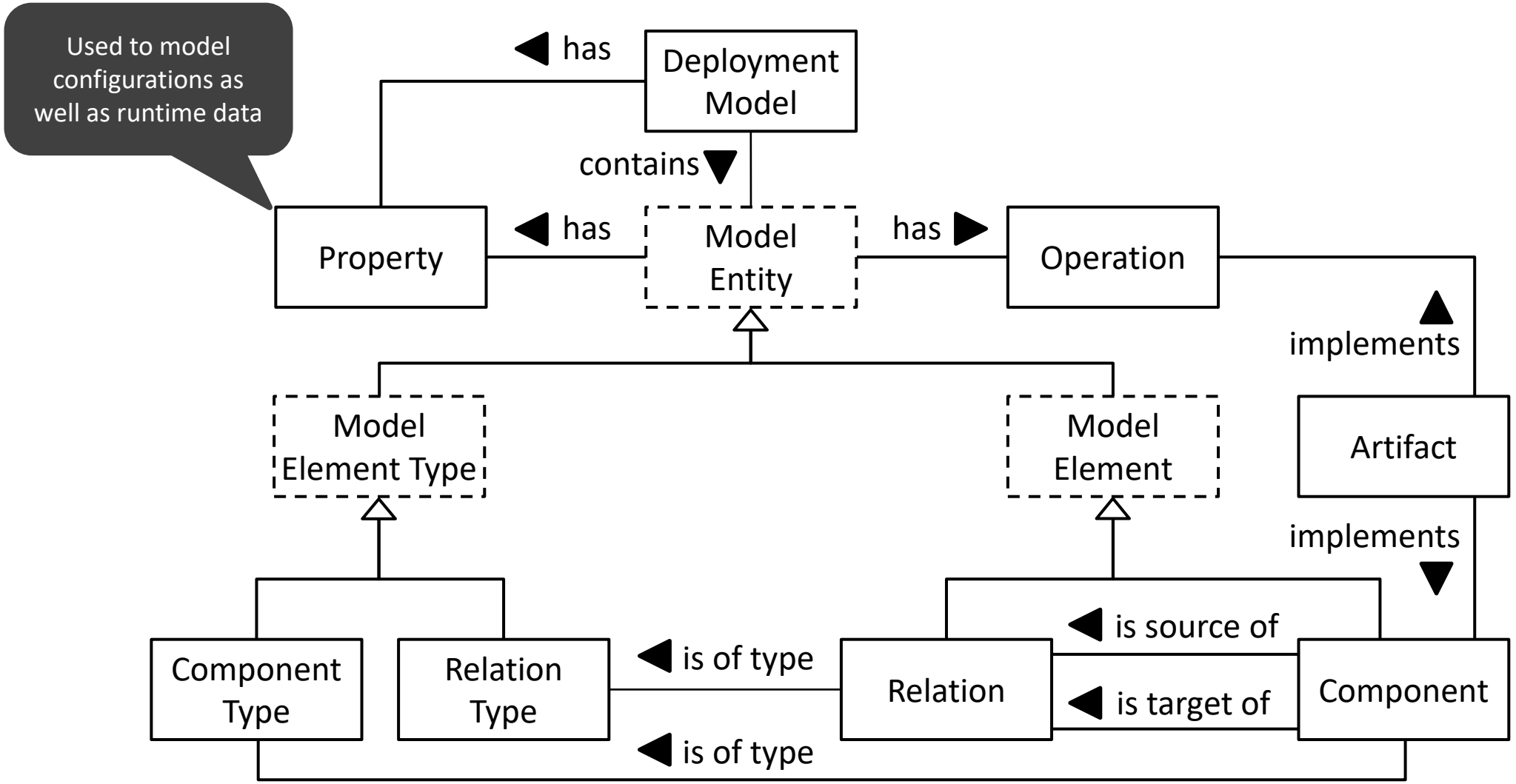
- Q3: How would you rate your skills in modeling executable declarative deployment models using the TOSCA standard?



- Q4: Would it be possible for you to create an executable TOSCA Service Template with a custom Node Type that provides a custom install script in less than 3 hours?

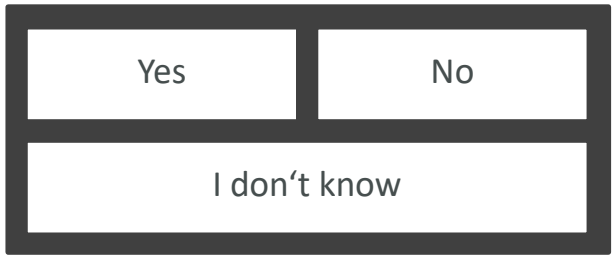


The Essential Deployment Metamodel

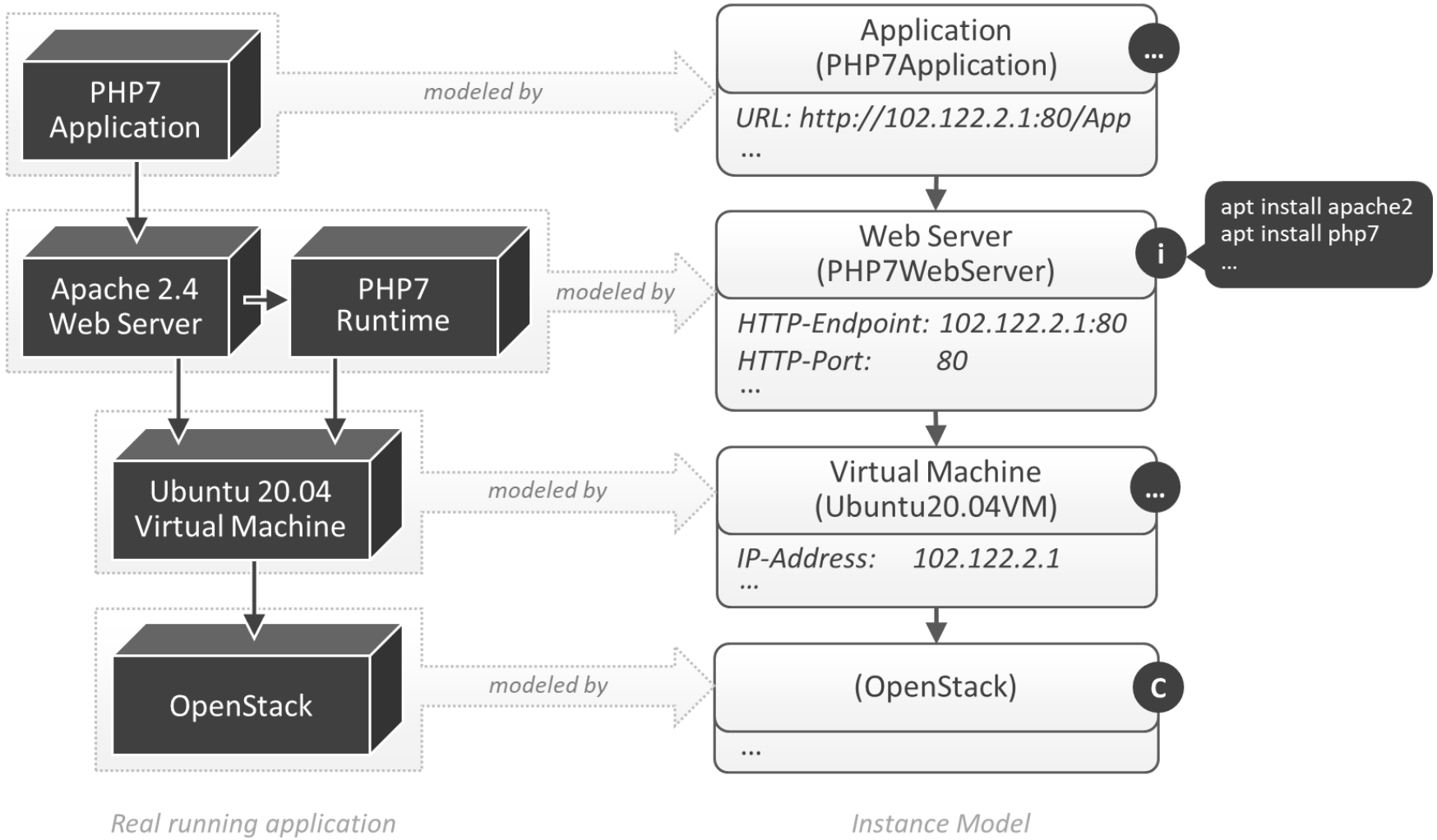


Opinion before knowing the approach

- Q5: Do you think there is a need for guidelines regarding the definition of custom or new component types that can be used in declarative deployment models?

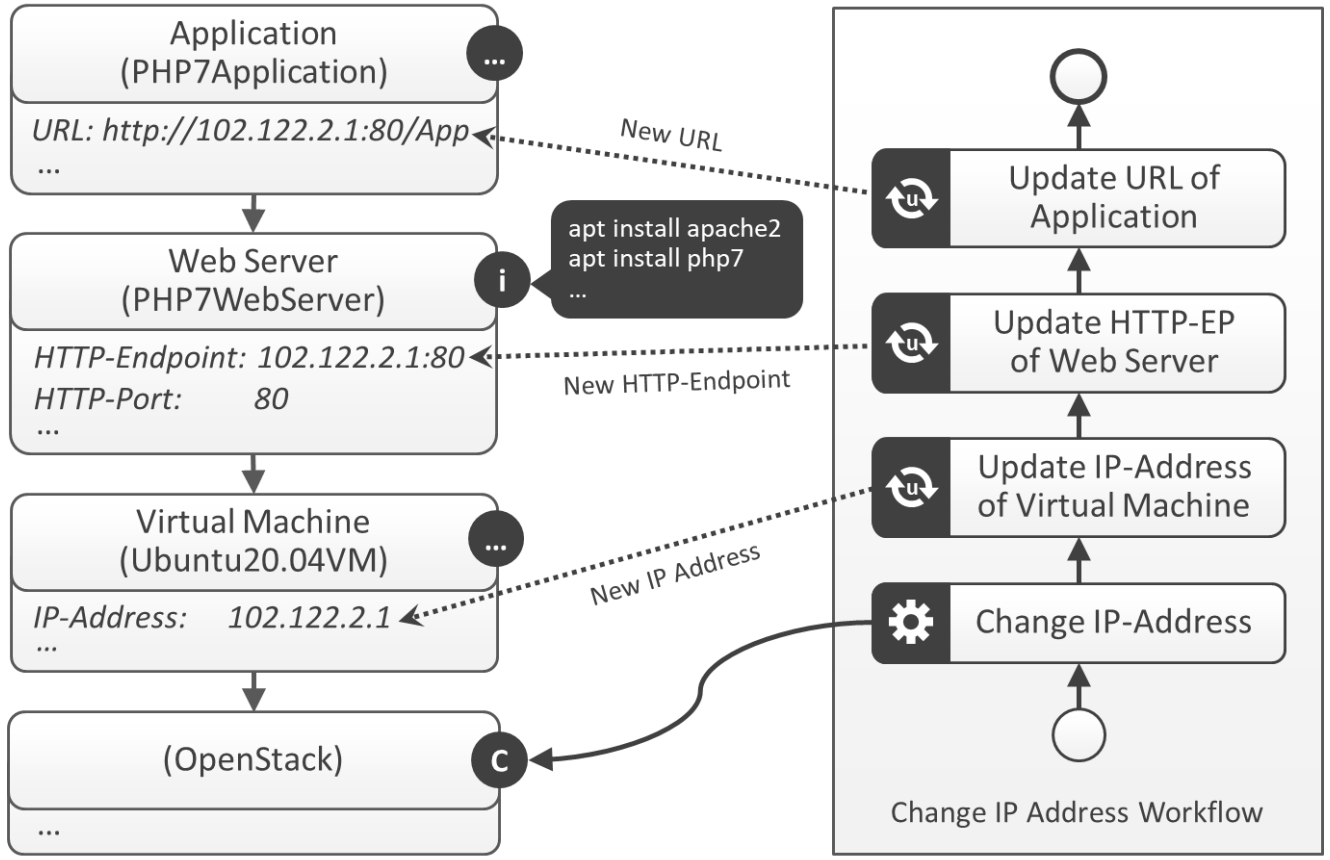


Running Example


PHP7 Application
modeled by
Application (PHP7Application)
...
URL: http://102.122.2.1:80/App
...
Apache 2.4 Web Server
PHP7 Runtime
modeled by
Web Server (PHP7WebServer)
i
apt install apache2
apt install php7
...
HTTP-Endpoint: 102.122.2.1:80
HTTP-Port: 80
...
Ubuntu 20.04 Virtual Machine
modeled by
Virtual Machine (Ubuntu20.04VM)
...
IP-Address: 102.122.2.1
...
OpenStack
modeled by
(OpenStack)
C
...
Real running application
Instance Model

Case Study 1

In the following, we consider this case study: a **manually modelled or implemented management process** needs to update the public IP address of the running virtual machine used in the running example.

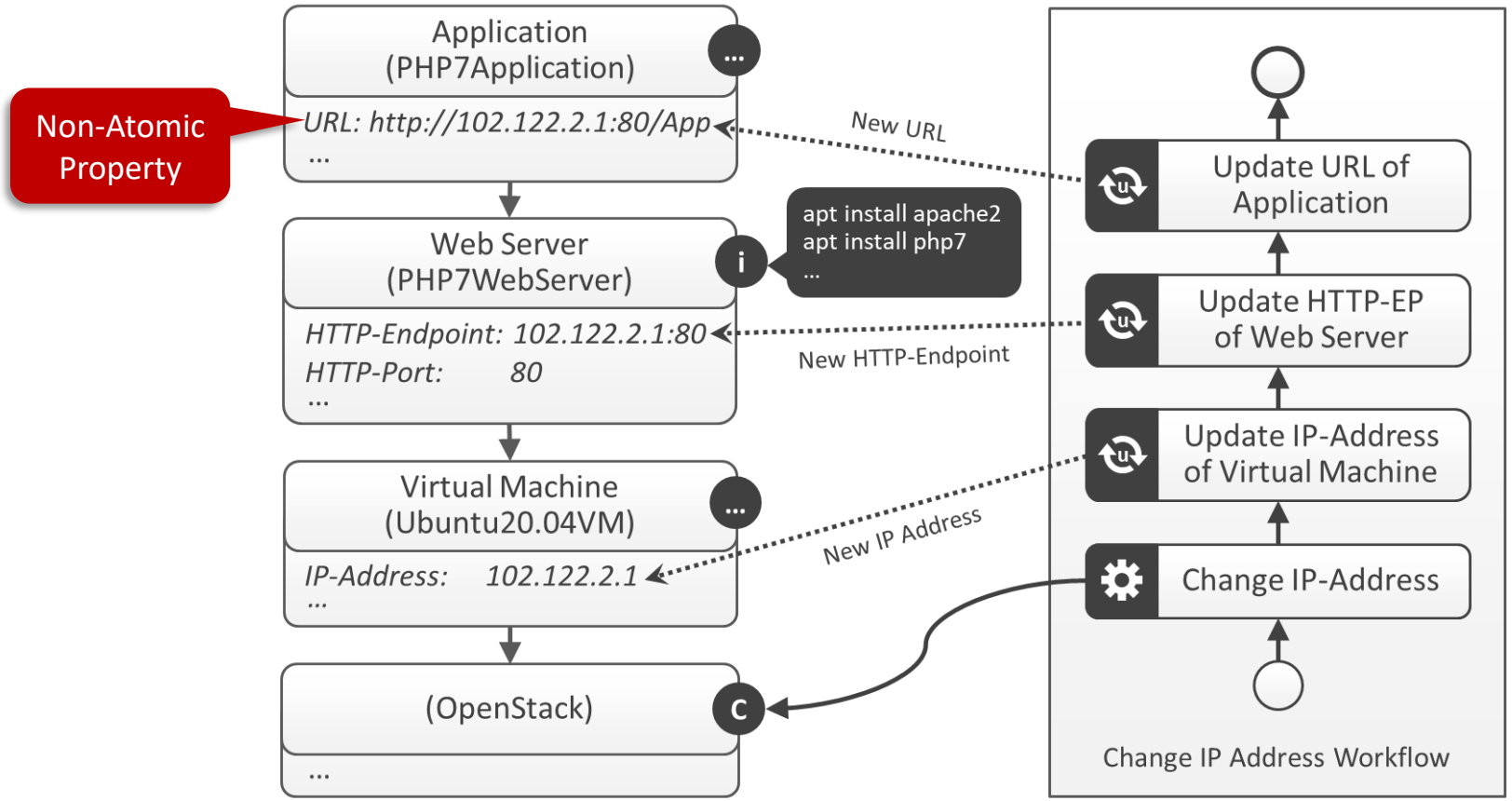


Problem 1: Non-Atomic Property

Q6: To what extent do you agree or disagree that defining non-atomic properties of component types increases the risk of erroneous data in instance models caused by manually created management processes?

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

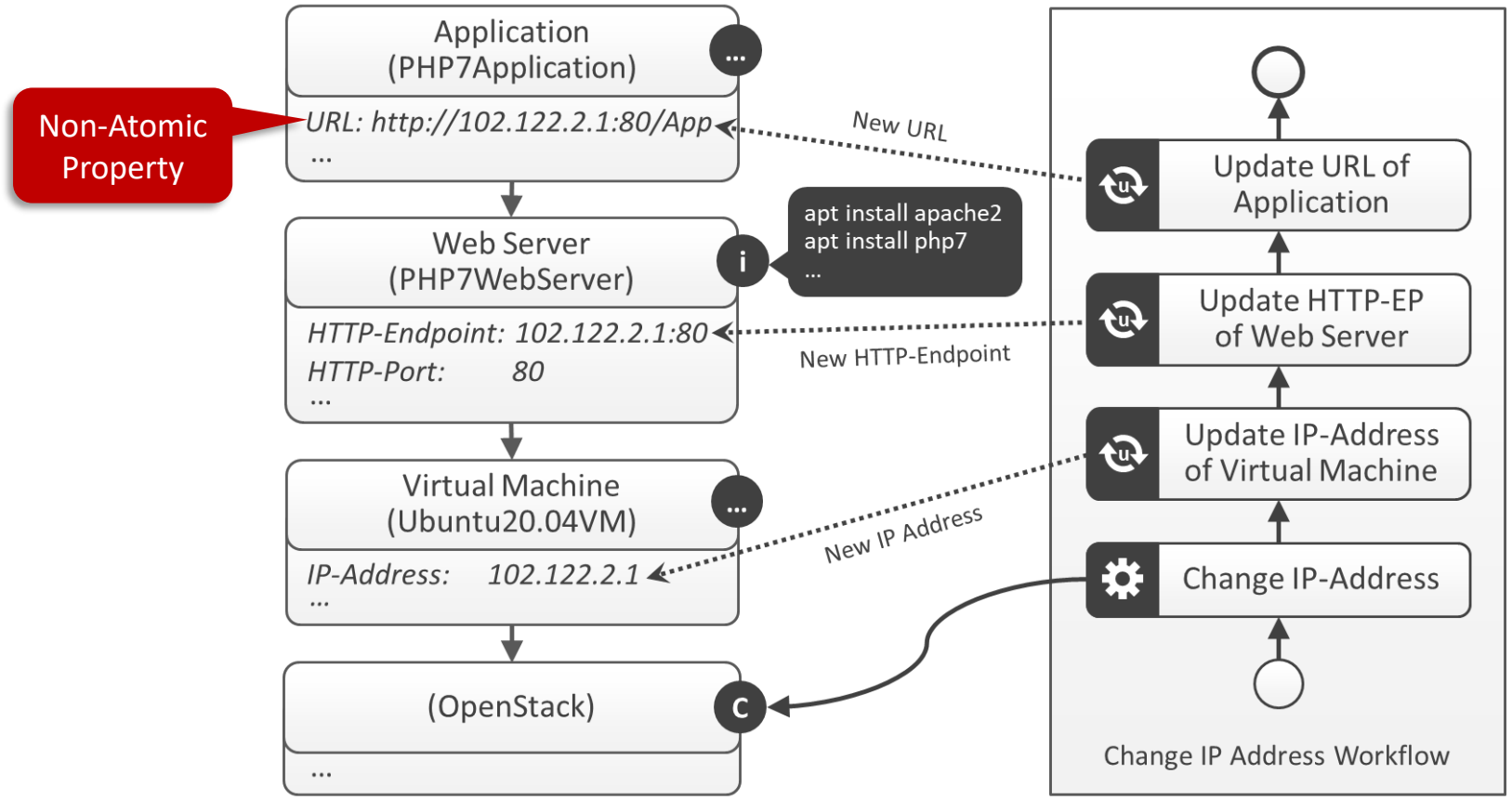
I don‘t know



Q7: In your opinion, how likely is it that an inexperienced developer who defines a new component type will define a non-atomic property, e.g., an absolute URL of a web application?

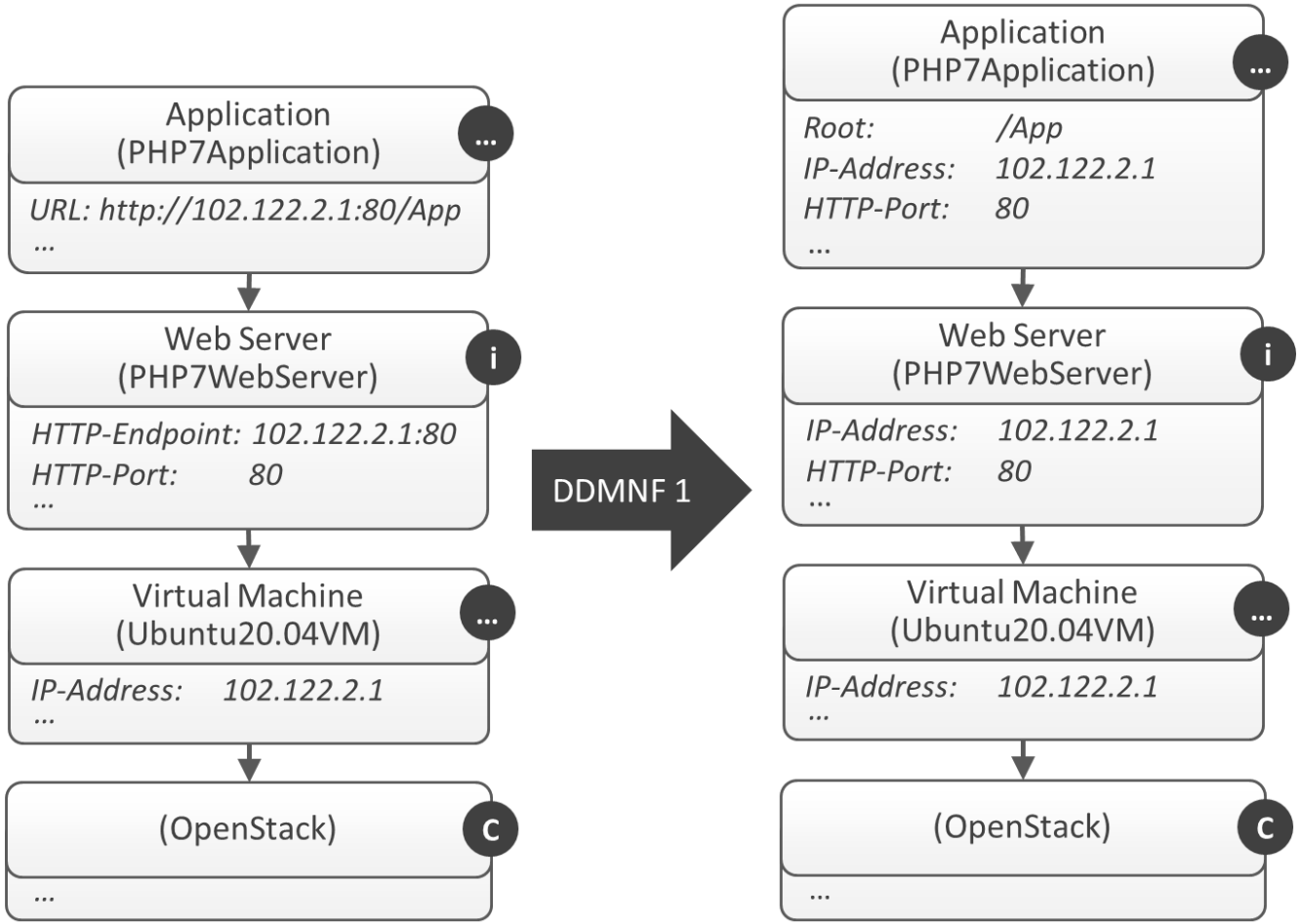
Very Unlikely	Unlikely	Neither likely nor unlikely	Likely	Very likely

I don't know



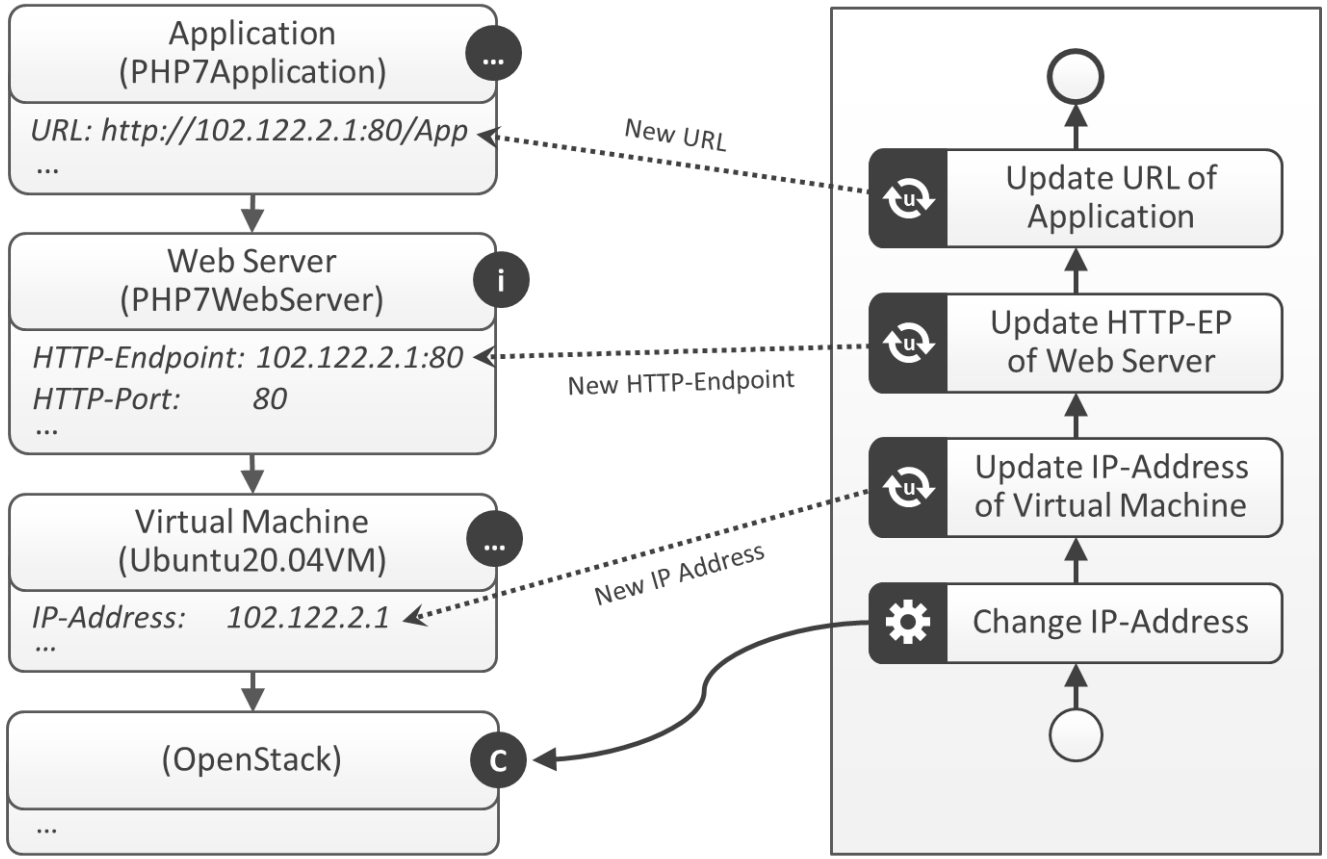
Declarative Deployment Model Normal Form 1 (DDMNF1)

Declarative Deployment Model Normal Form 1 (DDMNF1)



Values of properties must be atomic, i.e., multiple different technical or conceptual information must not be mixed in any form in a single property, but must be modelled as individual properties.

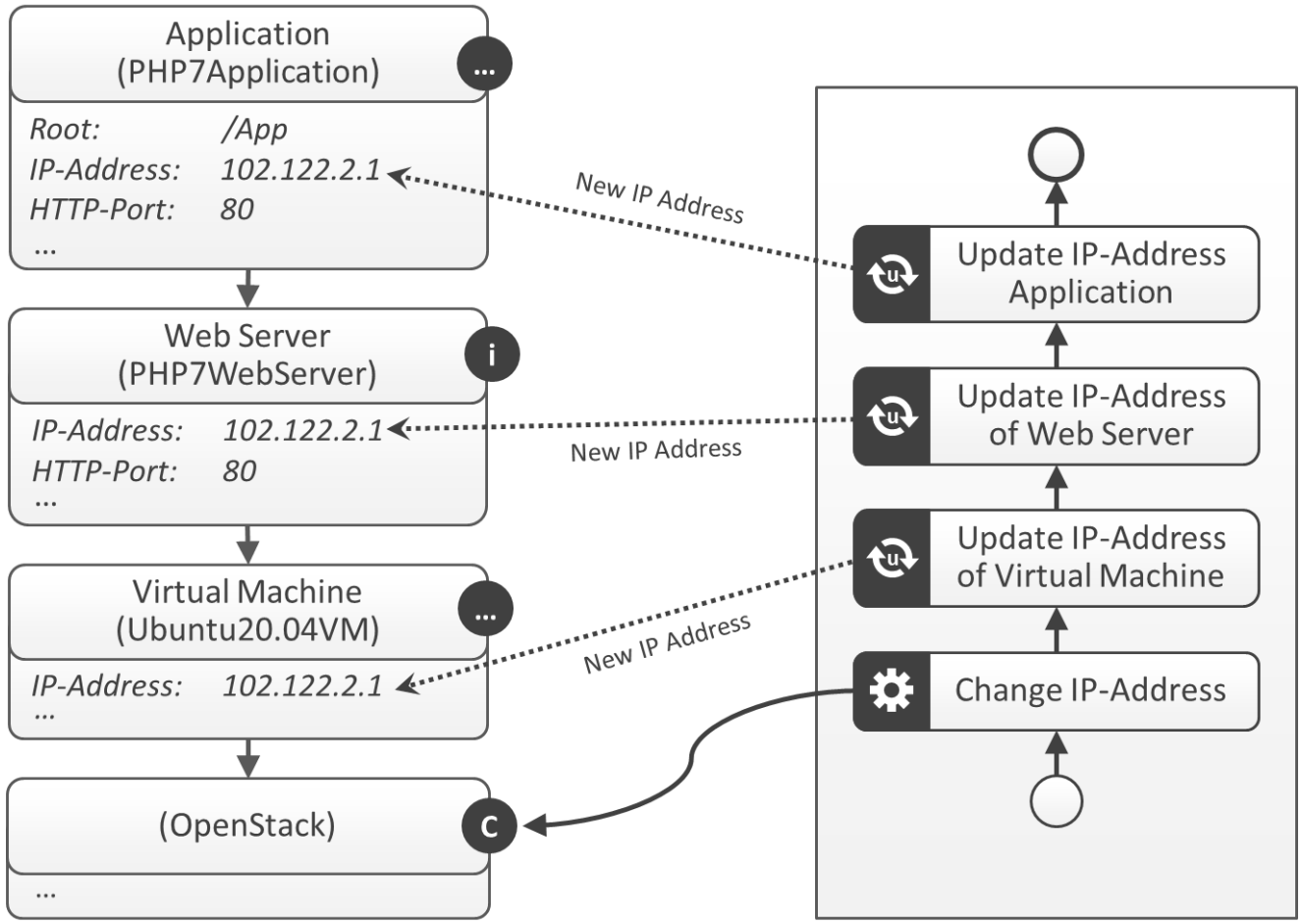
Management process that changes the IP address of the virtual machine **before** applying DDMNF1



Instance Model

Change IP Address Process

Management process that changes the IP address of the virtual machine **after** applying DDMNF1



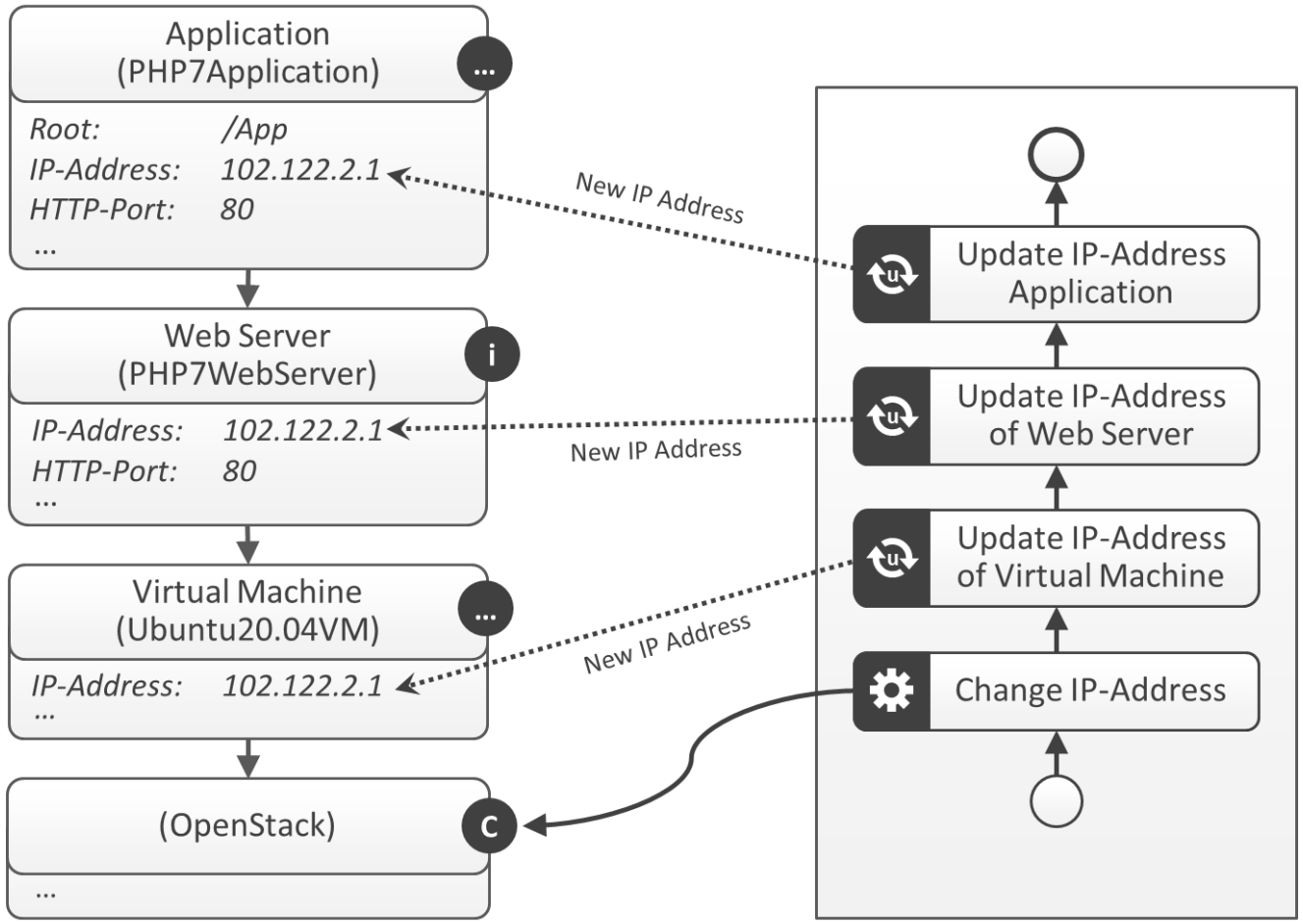
Instance Model

Change IP Address Process

Q8: To what extent do you agree or disagree that applying DDMNF1 decreases the risk of erroneous data in instance models caused by manually created management processes?

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

I don‘t know



Instance Model

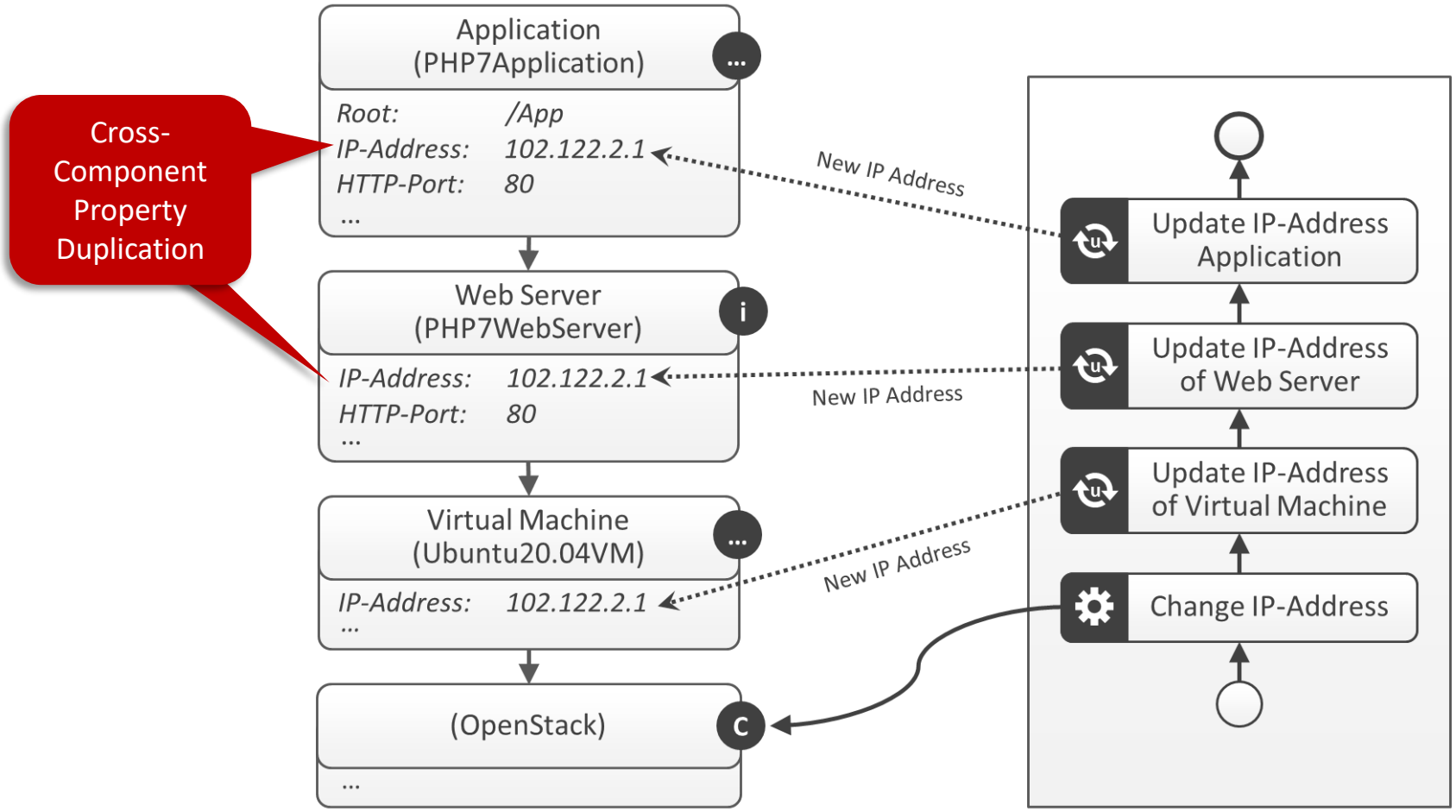
Change IP Address Process

Problem 2: Cross-Component Property Duplication

Q9: To what extent do you agree or disagree that duplicating properties that technically or conceptually belong to exactly one component type to other component types increases the risk of inconsistent data in instance models caused by manually created management processes?

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

I don‘t know



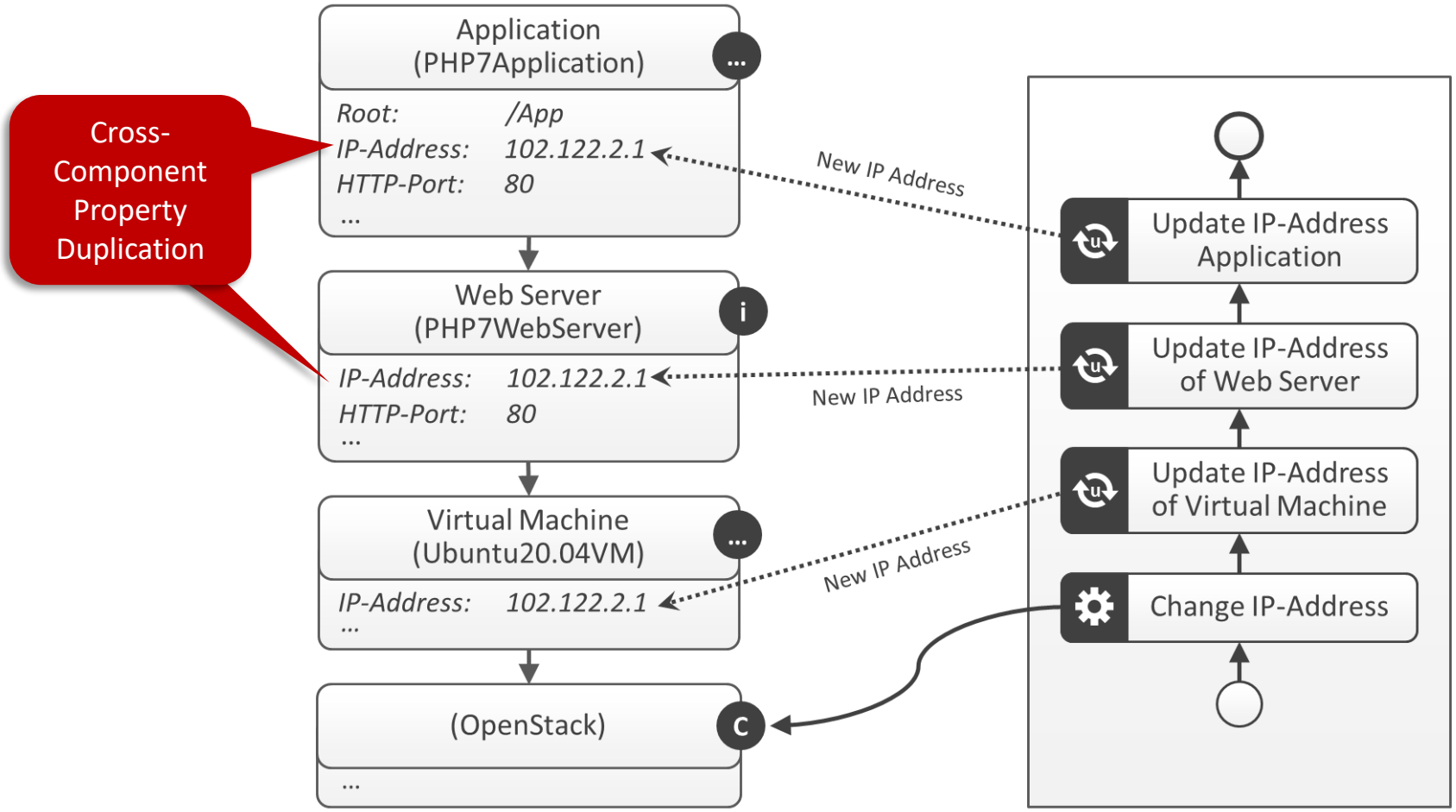
Instance Model

Change IP Address Process

Q10: In your opinion, how likely is it that an inexperienced developer who defines a new component type will duplicate a property from another type, e.g., an IP address of a virtual machine?

Very Unlikely	Unlikely	Neither likely nor unlikely	Likely	Very likely

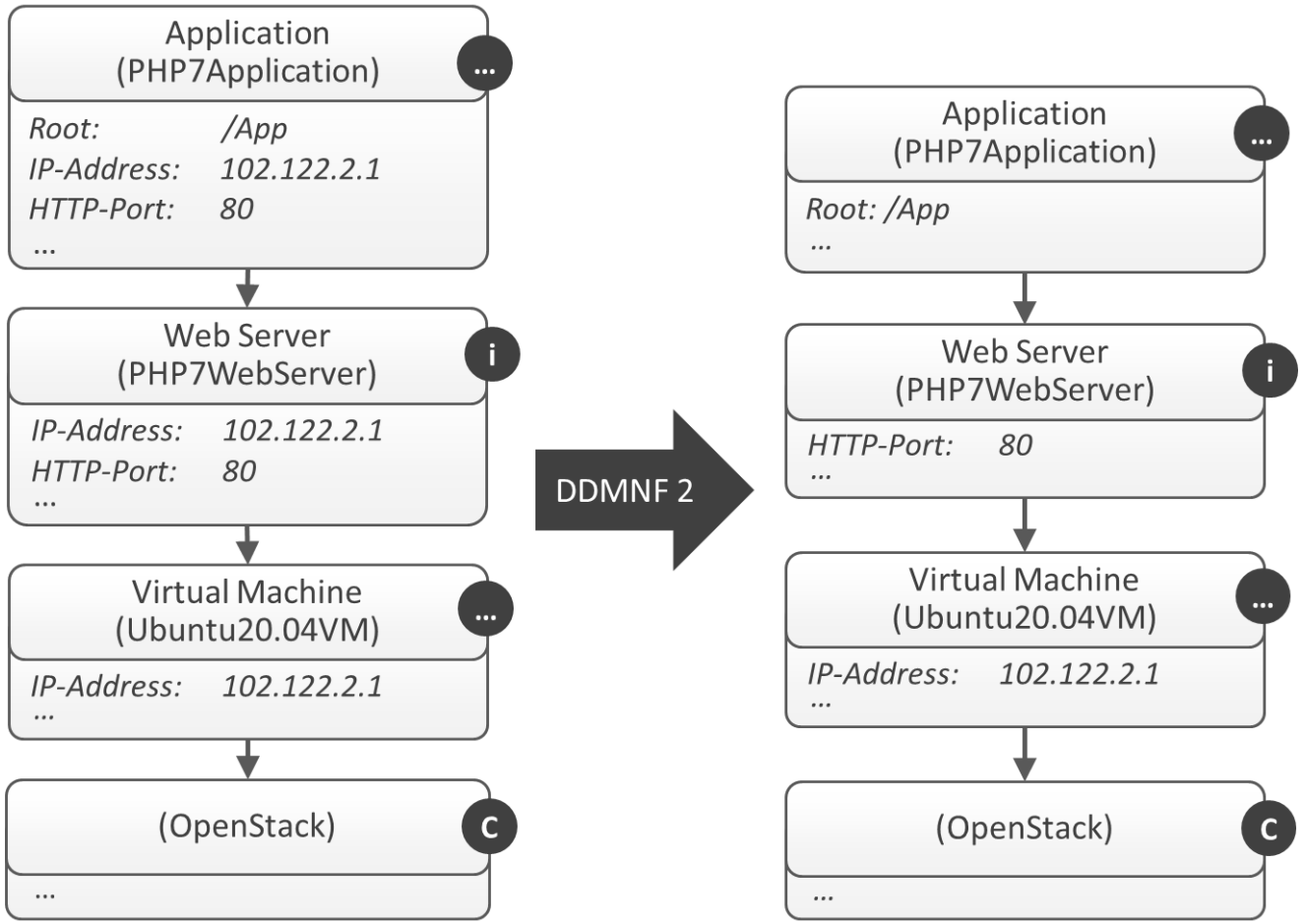
I don't know



Instance Model *Change IP Address Process*

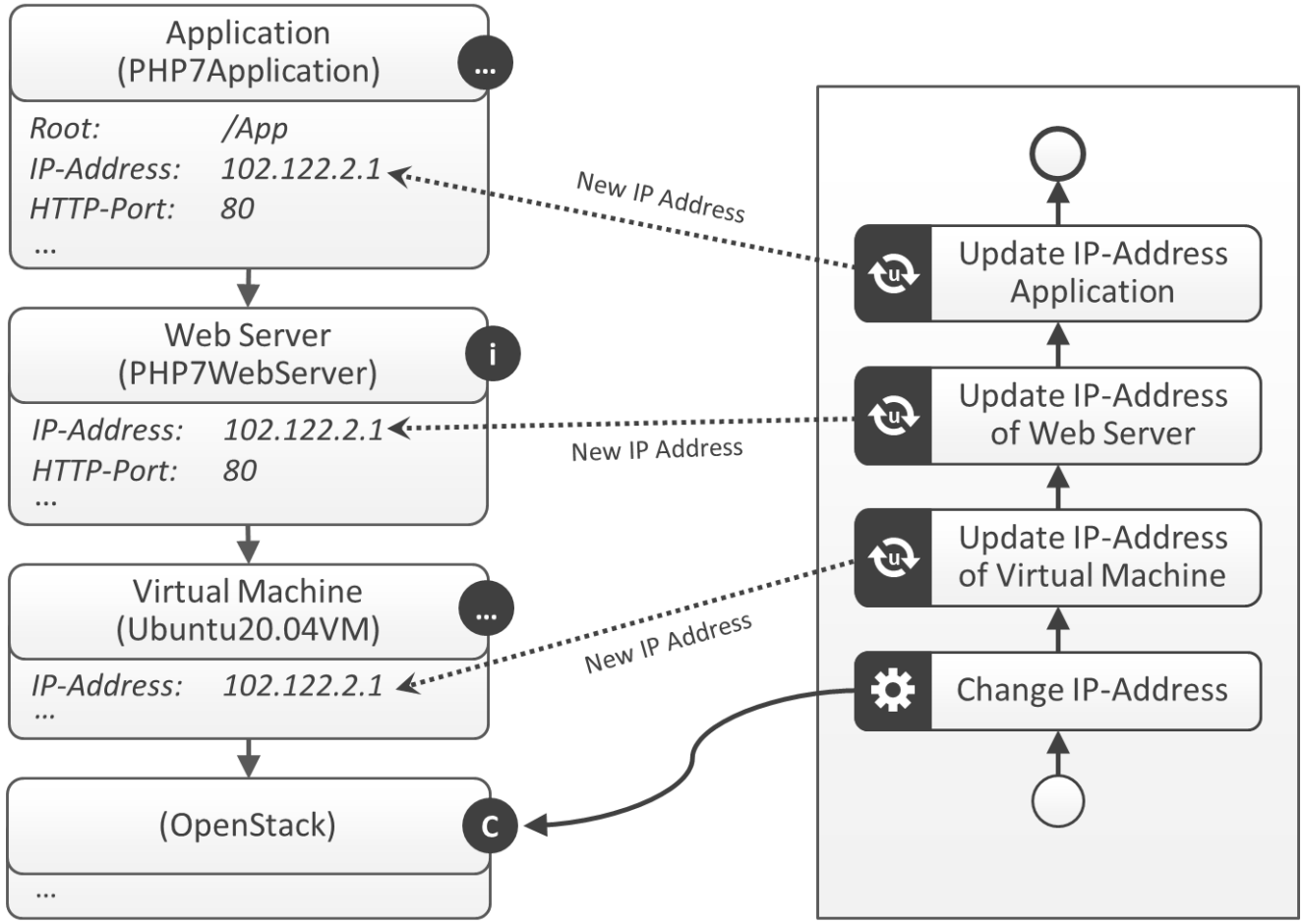
Declarative Deployment Model Normal Form 2 (DDMNF2)

Declarative Deployment Model Normal Form 2 (DDMNF2)



A property that technically or conceptually belongs to exactly one component type may only be contained in the properties of this component type and must not be contained in the properties of any other component type.

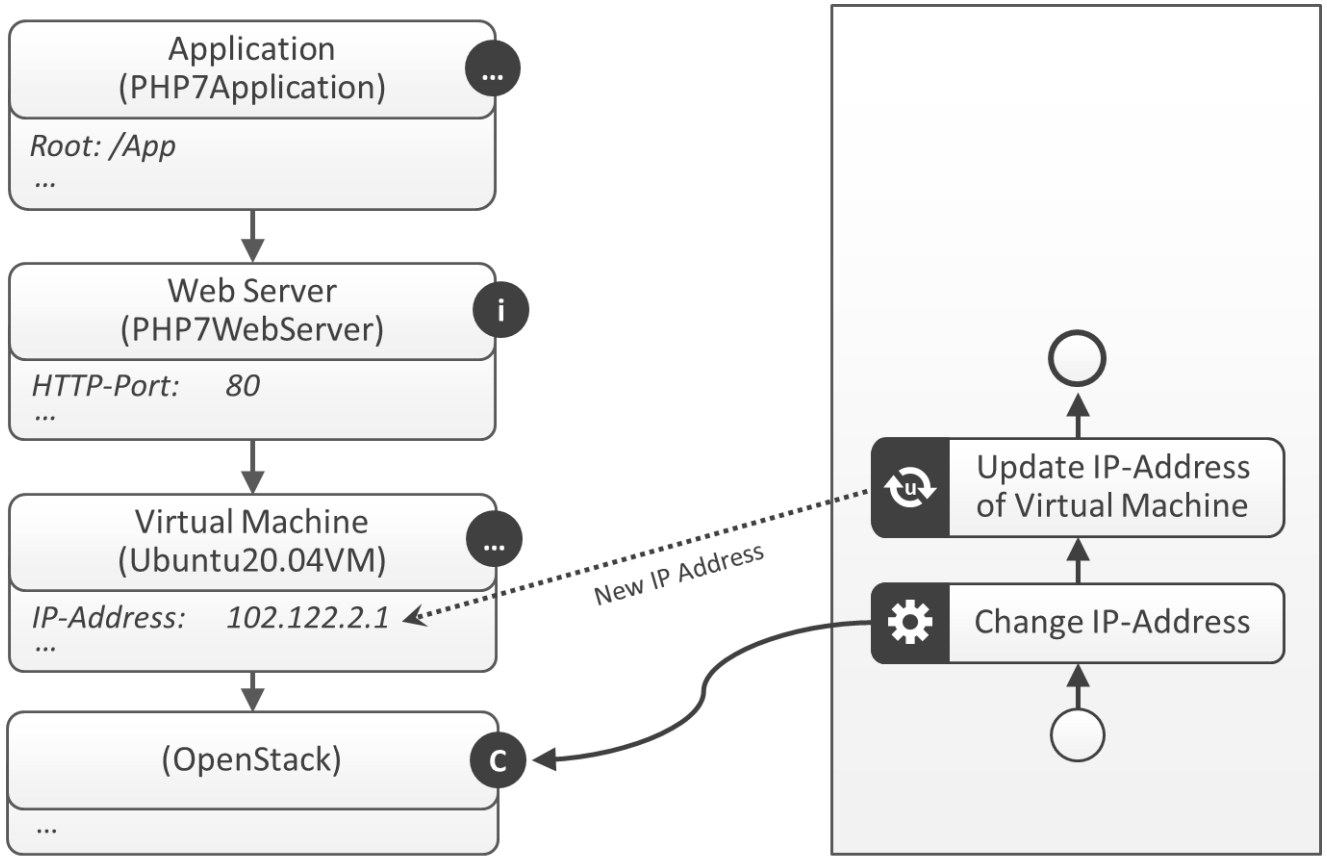
Management process that changes the IP address of the virtual machine **before** applying DDMNF2



Instance Model

Change IP Address Process

Management process that changes the IP address of the virtual machine **after** applying DDMNF2



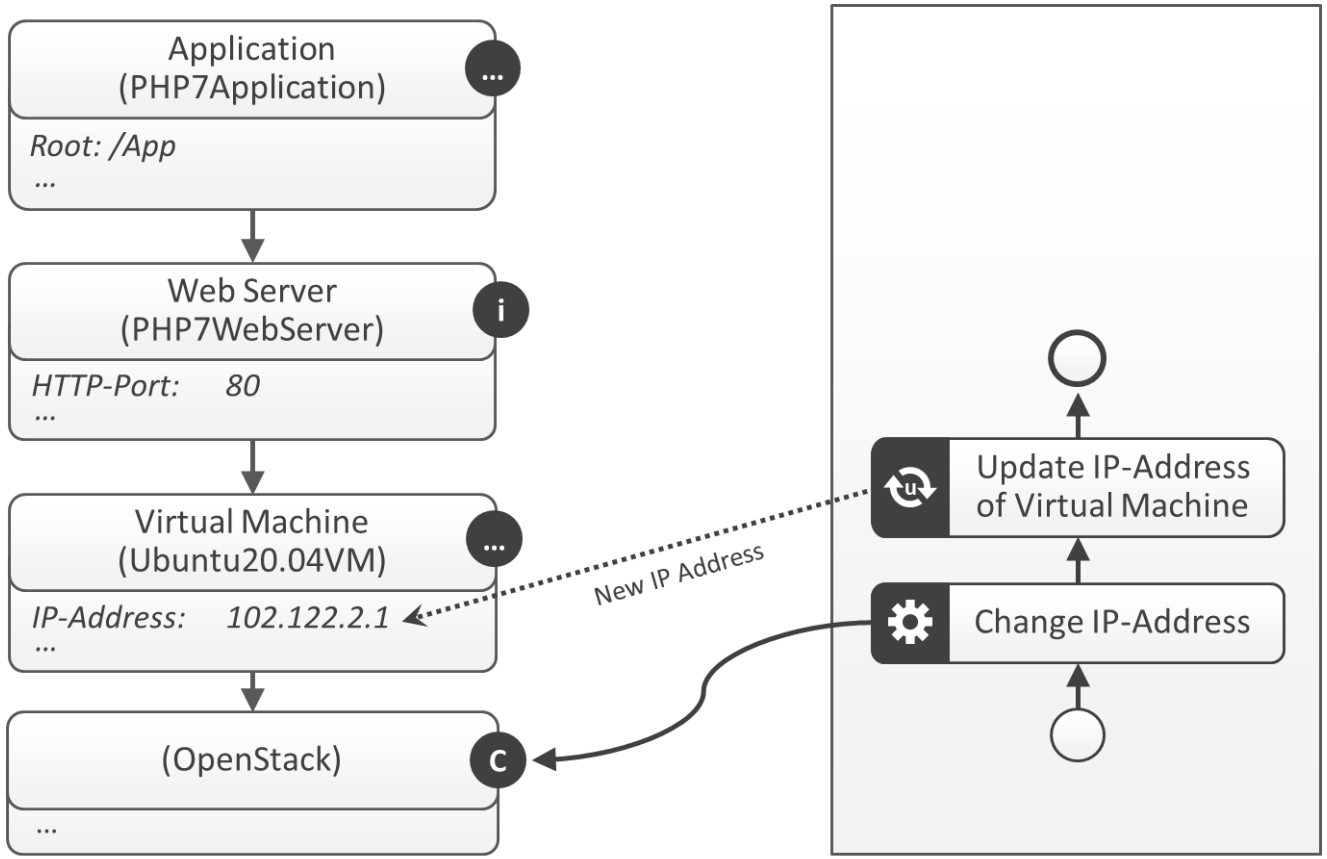
Instance Model

Change IP Address Process

Q11: To what extent do you agree or disagree that applying DDMNF2 decreases the risk of inconsistent data in instance models caused by manually created management processes?

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

I don‘t know

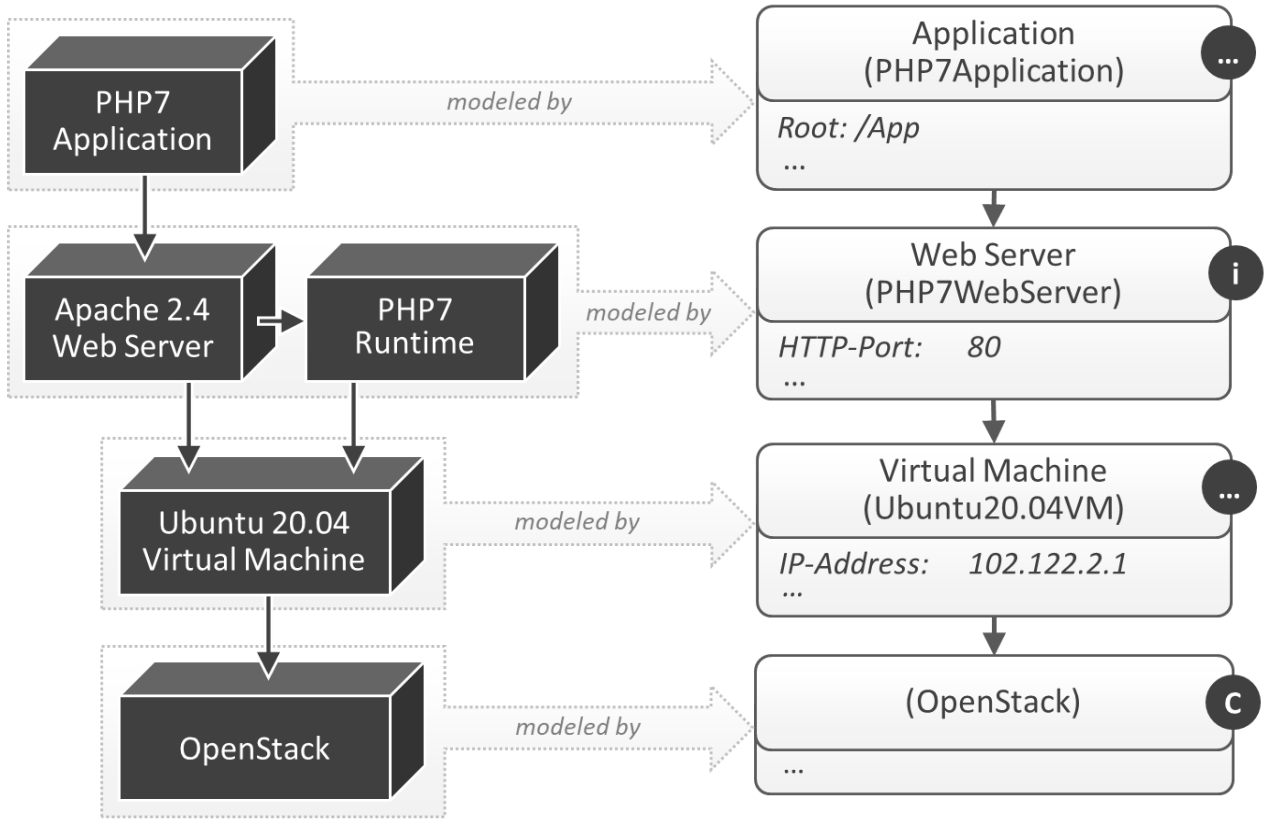


Instance Model

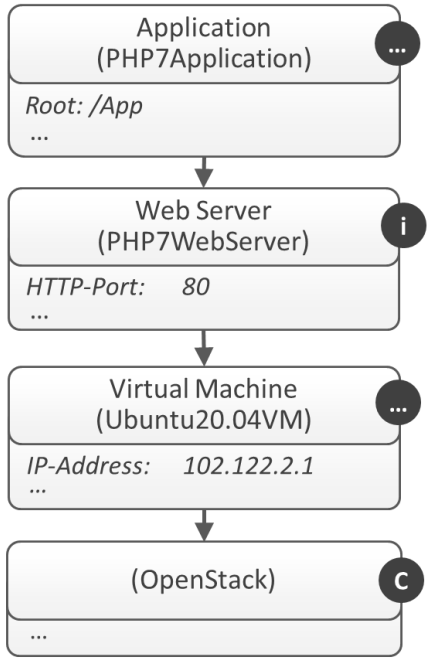
Change IP Address Process

Case Study 2

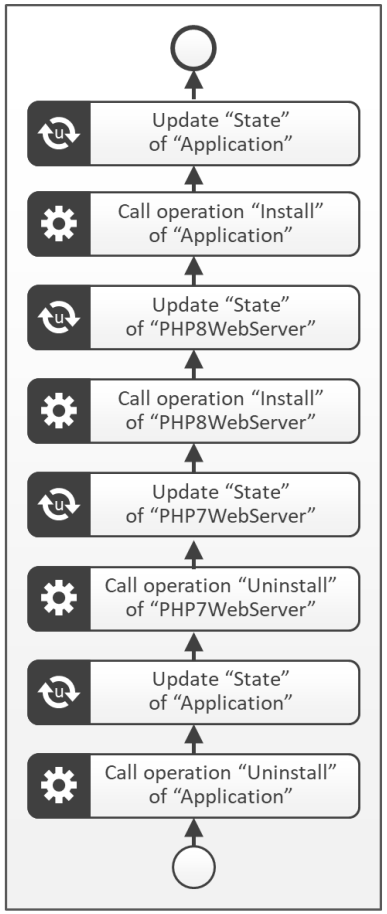
We remember…


PHP7 Application
modeled by
Application (PHP7Application)
Root: /App
...
Apache 2.4 Web Server
PHP7 Runtime
modeled by
Web Server (PHP7WebServer)
HTTP-Port: 80
...
Ubuntu 20.04 Virtual Machine
modeled by
Virtual Machine (Ubuntu20.04VM)
IP-Address: 102.122.2.1
...
OpenStack
modeled by
(OpenStack)
...
Real running application
Instance Model

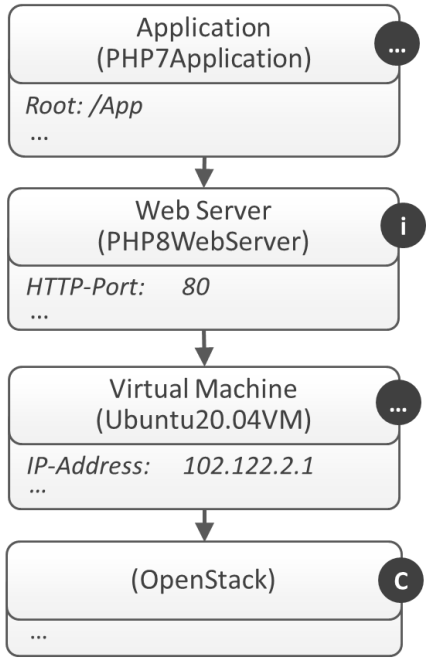
In the following, we consider this second case study: a **manually modelled or implemented management process** needs to update the PHP runtime in the running example from PHP7 to PHP8.



Instance Model



Update PHP Version Process

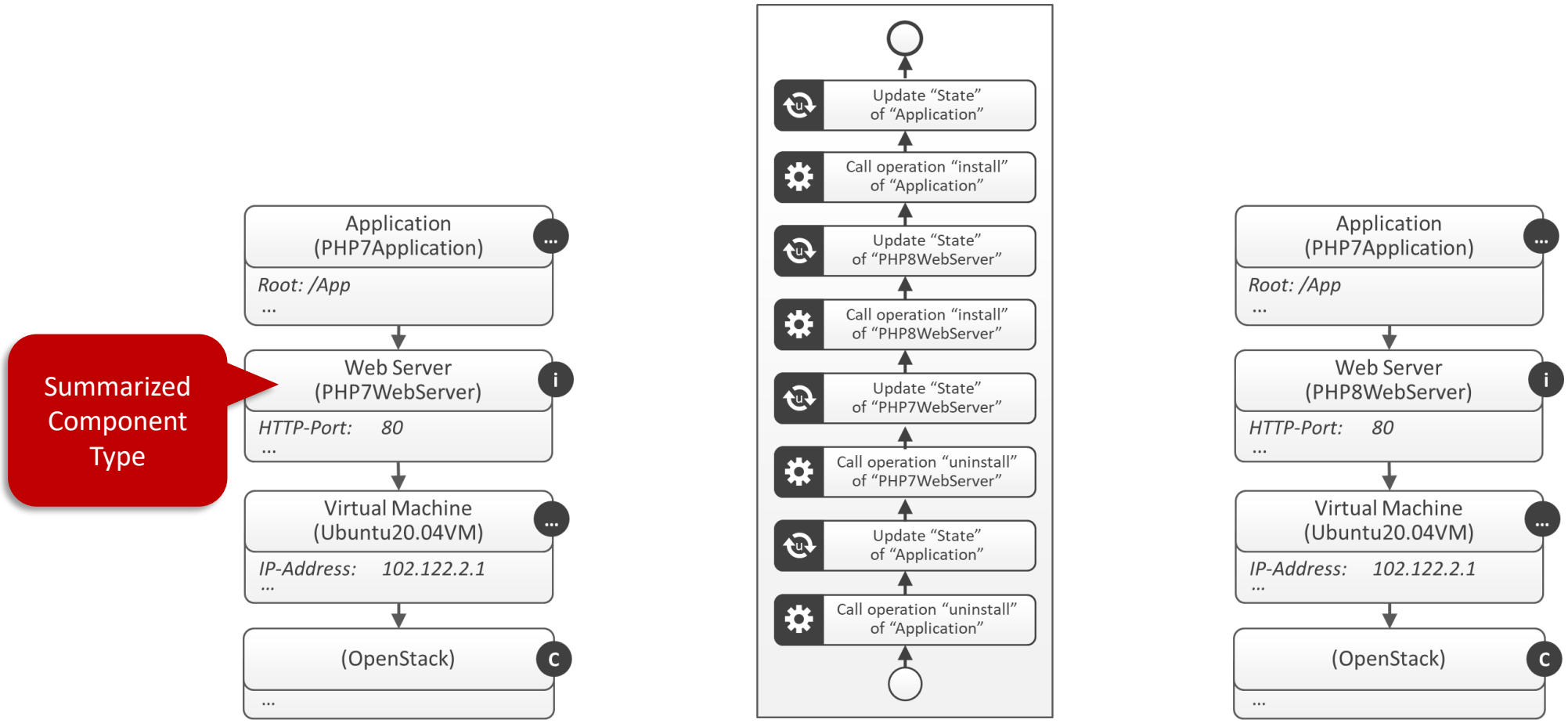


Resulting Instance Model

Problem 3: Summarized Component Types

Q12: To what extent do you agree or disagree that combining multiple standalone components into a single component type increases the risk of inefficient manually created management processes?

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree	I don‘t know



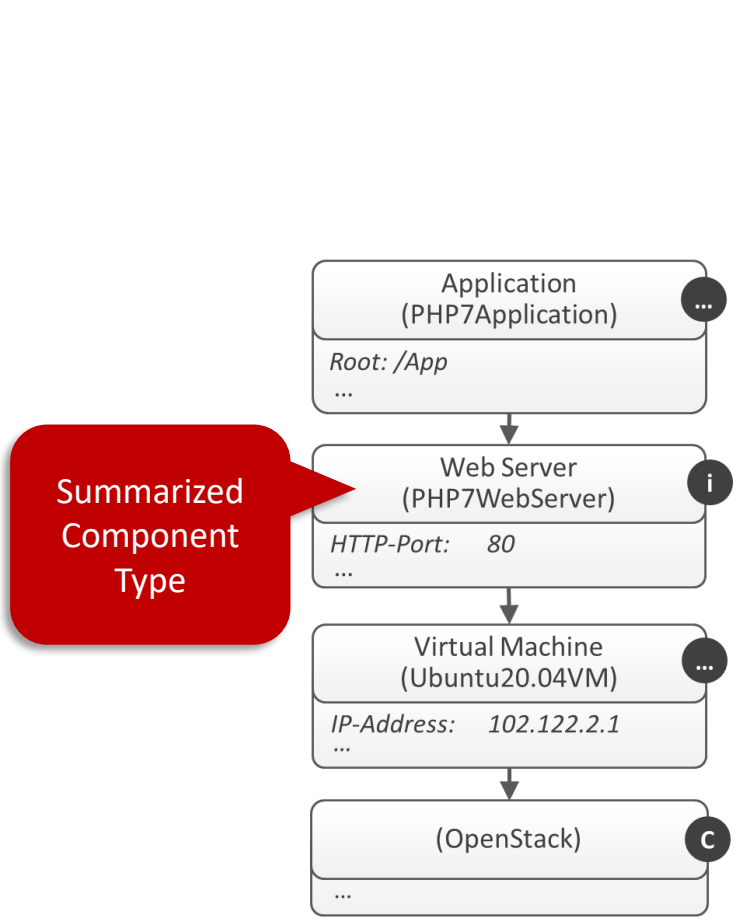
Instance Model

Update PHP Version Process before applying normal form 3

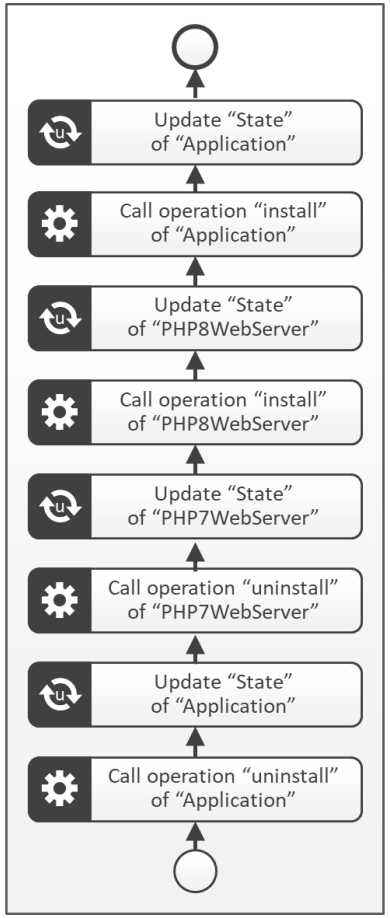
Resulting Instance Model

Q13: In your opinion, how likely is it that an inexperienced developer combines multiple components that can run standalone and that have own lifecycles into a single component type?

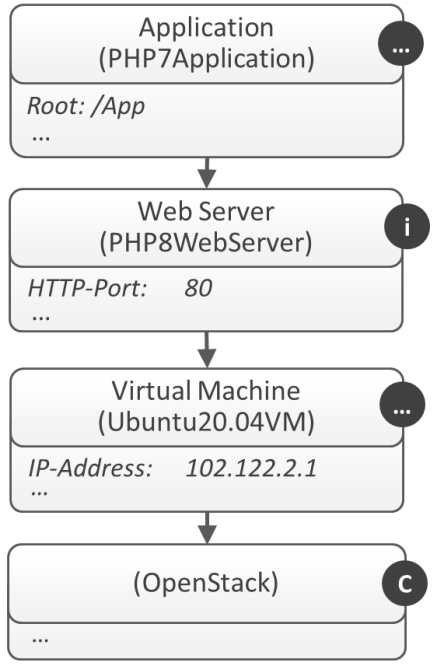
Very Unlikely	Unlikely	Neither likely nor unlikely	Likely	Very likely	I don‘t know



Instance Model



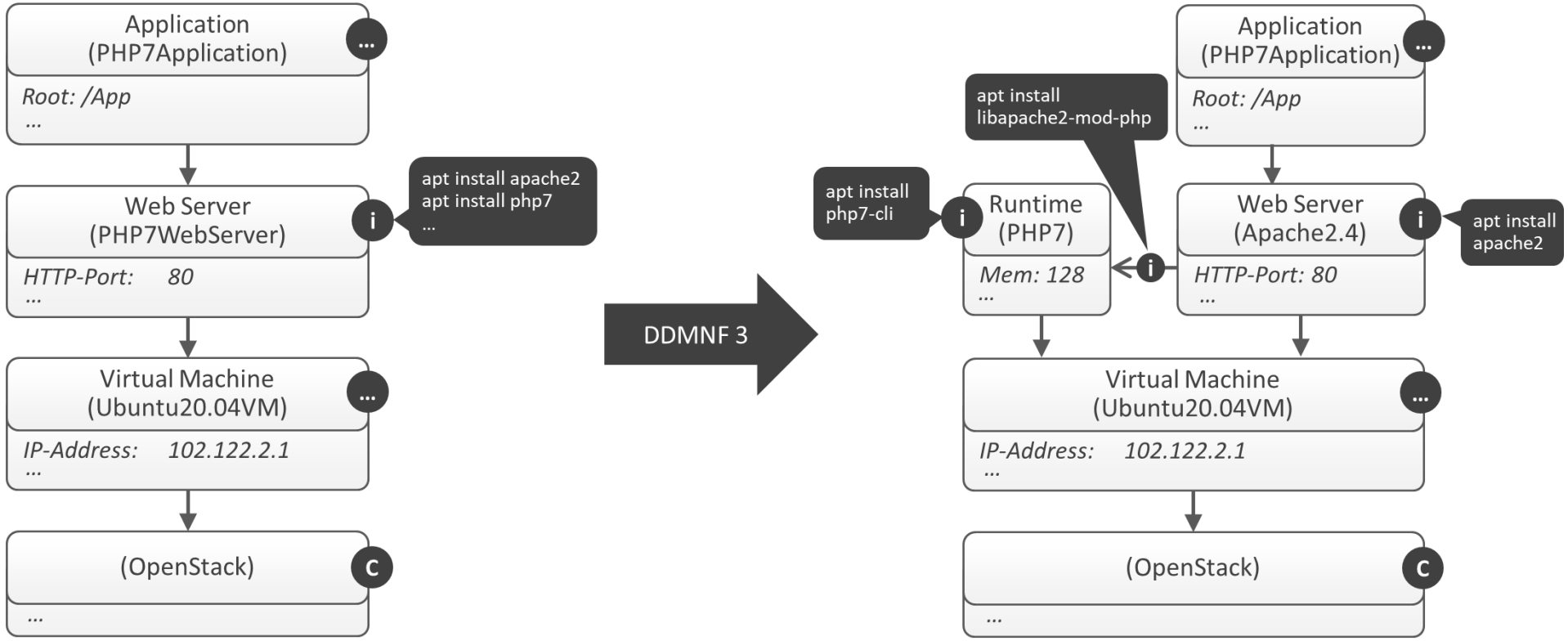
Update PHP Version Process before applying normal form 3



Resulting Instance Model

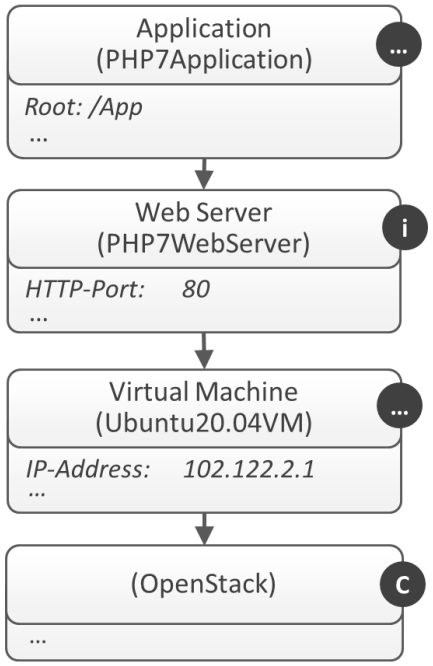
Declarative Deployment Model Normal Form 3 (DDMNF3)

Declarative Deployment Model Normal Form 3 (DDMNF3)

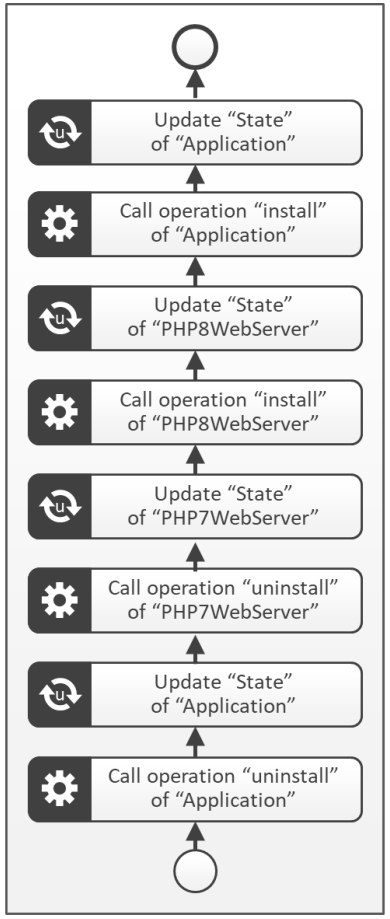


Components that can run standalone and that have own lifecycles must be modelled as separate component types and must not be combined into a single component type.

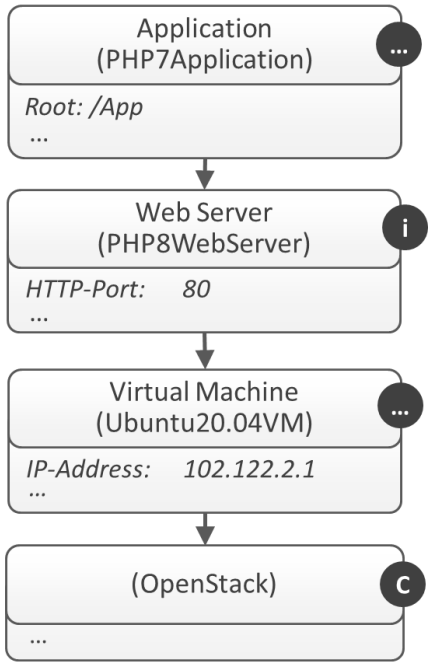
Management process that updates the PHP runtime from PHP7 to PHP8 **before** applying DDMNF3



Instance Model

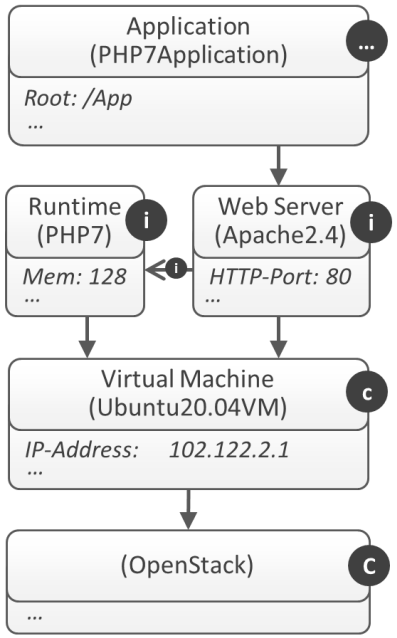


Update PHP Version Process before applying normal form 3

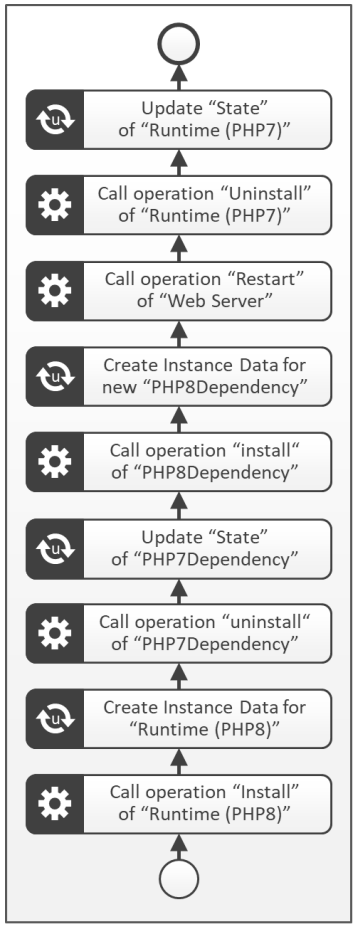


Resulting Instance Model

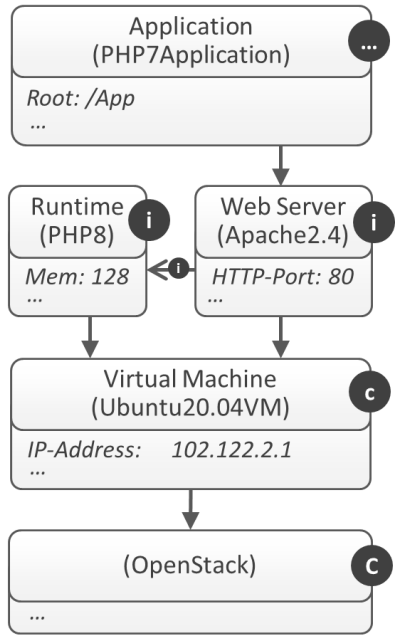
Management process that updates the PHP runtime from PHP7 to PHP8 **after** applying DDMNF3



Normalized Instance Model



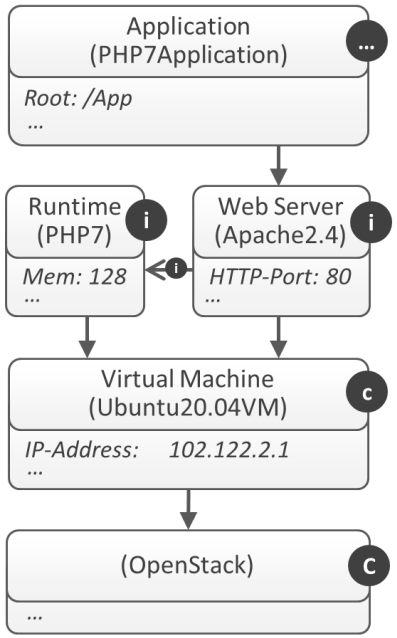
Update PHP Version Process after applying normal form 3



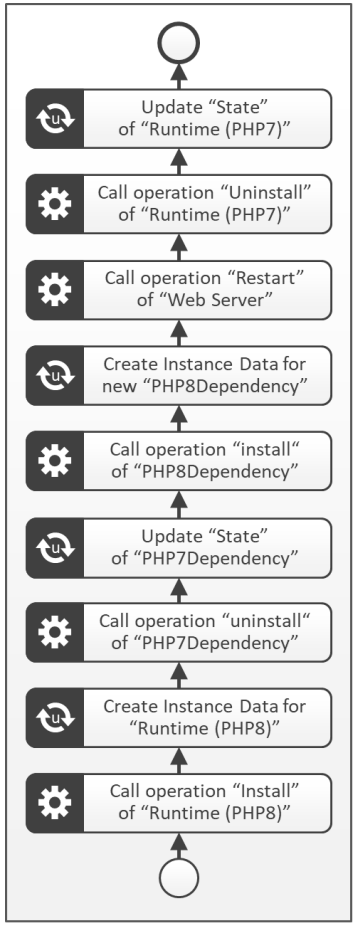
Normalized Instance Model

Q14: To what extent do you agree or disagree that applying DDMNF3 decreases the risk of inefficient manually created management processes?

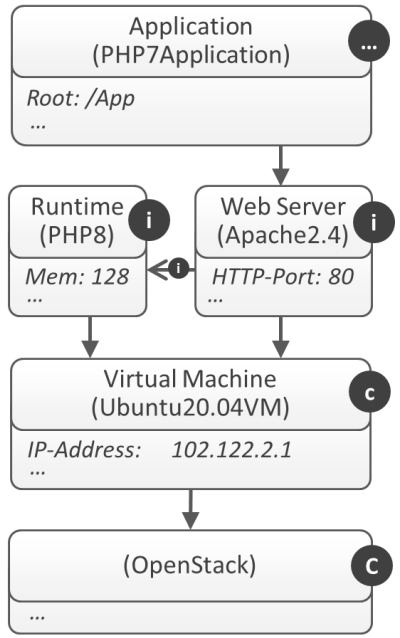
Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree	I don‘t know



Normalized Instance Model



Update PHP Version Process after applying normal form 3



Normalized Instance Model

That‘s it ☺