User Interface Component **Architecture**
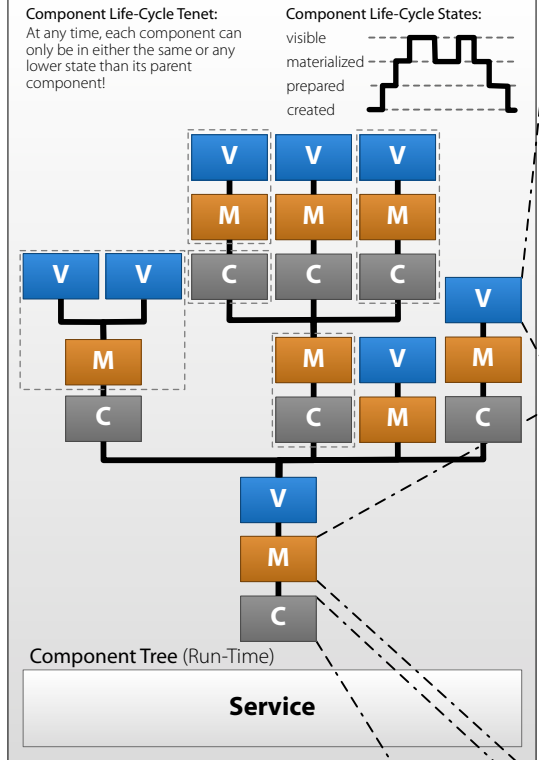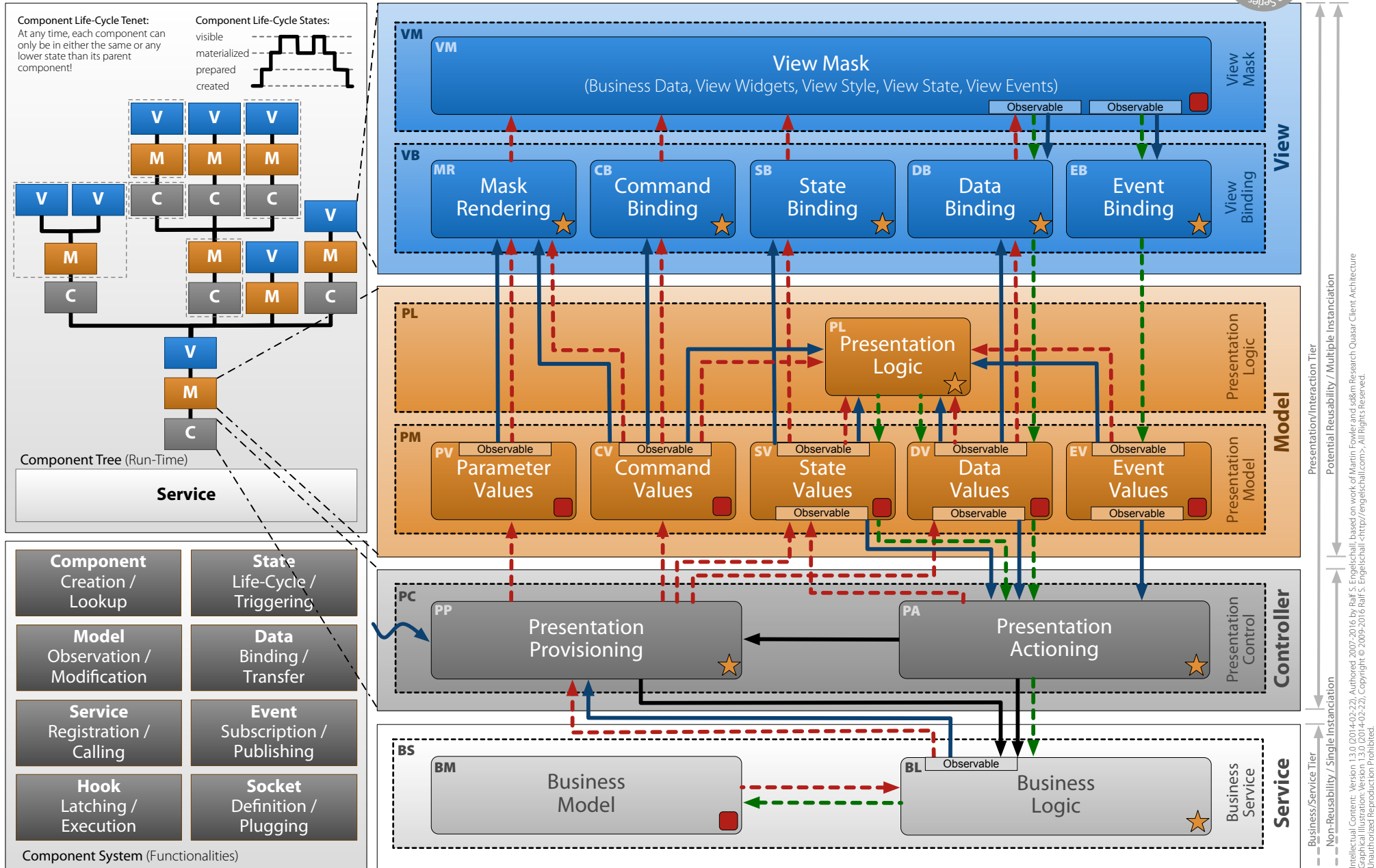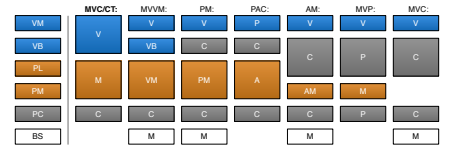
based on *Model-View-Controller / Component-Tree (MVC/CT)* pattern and **Component System (CS)**

Component Life-Cycle Tenet:
At any time, each component can only be in either the same or any lower state than its parent component!

Component Life-Cycle States:
visible
materialized
prepared
created

Component Tree (Run-Time)

V M C

Service

Component System (Functionalities)

Component Creation / Lookup	**State** Life-Cycle / Triggering
Model Observation / Modification	**Data** Binding / Transfer
Service Registration / Calling	**Event** Subscription / Publishing
Hook Latching / Execution	**Socket** Definition / Plugging

View

VM — **View Mask**
(Business Data, View Widgets, View Style, View State, View Events)
Observable Observable

View Mask

VB — *View Binding*

MR Mask Rendering	**CB** Command Binding	**SB** State Binding	**DB** Data Binding	**EB** Event Binding

Model

PL — Presentation Logic
PL Presentation Logic

PM — *Presentation Model*

PV Parameter Values Observable	**CV** Command Values Observable	**SV** State Values Observable	**DV** Data Values Observable	**EV** Event Values Observable

Controller

PC — *Presentation Control*

PP Presentation Provisioning

PA Presentation Actioning

Service

BS — *Business Service*

BM Business Model

BL Business Logic — Observable

Presentation/Interaction Tier
Potential Reusability / Multiple Instanciation

Business/Service Tier
Non-Reusability / Single Instanciation

The *User Interface Component Architecture* is based on the *Model-View-Controller / Component-Tree (MVC/CT)* pattern from *Ralf S. Engelschall*. The MVC/CT pattern consists of a **Tree** of structured **Components**, forming the presentation tier of an application. Each component plays either the **Model**, **View** or **Controller** roles. The **Model** and **View** roles can be generic and reusable within the **Component Tree**. The **Controller** role is always application-specific and not reusable. These 3 roles are mapped onto the 5 logical UI separation layers **View Mask, View Binding, Presentation Logic, Presentation Model** and **Presentation Controller.** All **Components** are managed under run-time by a **Component System (CS).**

MVC/CT:	MVVM:	PM:	PAC:	AM:	MVP:	MVC:
VM		V	P	V	V	V
VB	VB	C	C			
PL				C	C	C
PM	VM	PM	A	AM	M	
PC	C	C	C	C	P	C
BS	M	M		M	M	M

Legend:
- component
- layer
- aspect
- active/code
- passive/data
- Control Flow (Direct Call)
- Control Flow (Callback)
- Data Flow (upward)
- Data Flow (downward)