.NETcampus 2010
Meet | Learn | Share

UGIdotNET

Architecting Web Applications

Andrea Saltarello
Solution Architect @ Managed Designs
UGIdotNET
http://blogs.ugidotnet.org/pape
http://twitter.com/andysal74

[Academic Tour]
Italy 2010
imagine cup

.NETcampus

Sponsor Platinum

Architettura: ISO/IEC 42010 fact sheet

Titolo: **IEEE Recommended Practice for Architectural Description of Software-Intensive Systems**

Scope: This recommended practice addresses the architectural description of software-intensive systems. A *software-intensive system is any system where software contributes essential influences to the design, construction, deployment, and evolution of the system as a whole*.

architect: The person, team, or organization responsible for systems architecture.

architecting: The activities of defining, documenting, maintaining, improving, and certifying properimplementation of an architecture.

architecture: The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.

Architettura: rappresentazione

L'architettura si rappresenta mediante le *view*, che sono la rappresentazione del sistema osservato da un certo *view point*.

Tutto fa brodo, ma ISO 19501 (UML) offre alcuni view point "significativi" espressi con una notazione *standard*

La favola del 3-tier

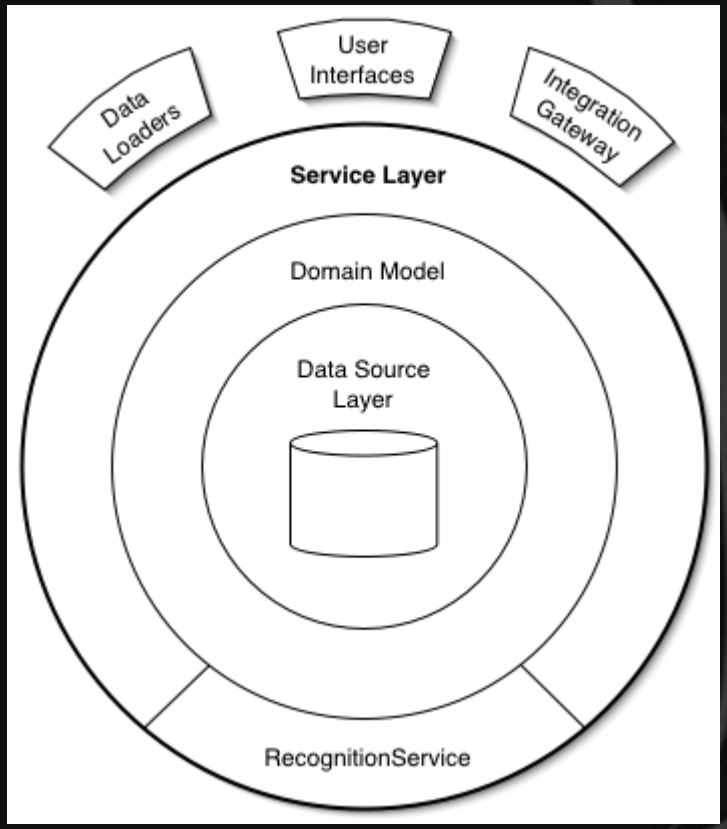
C'era una volta una architettura 3-tier:

- Presentation
- Business
- Data

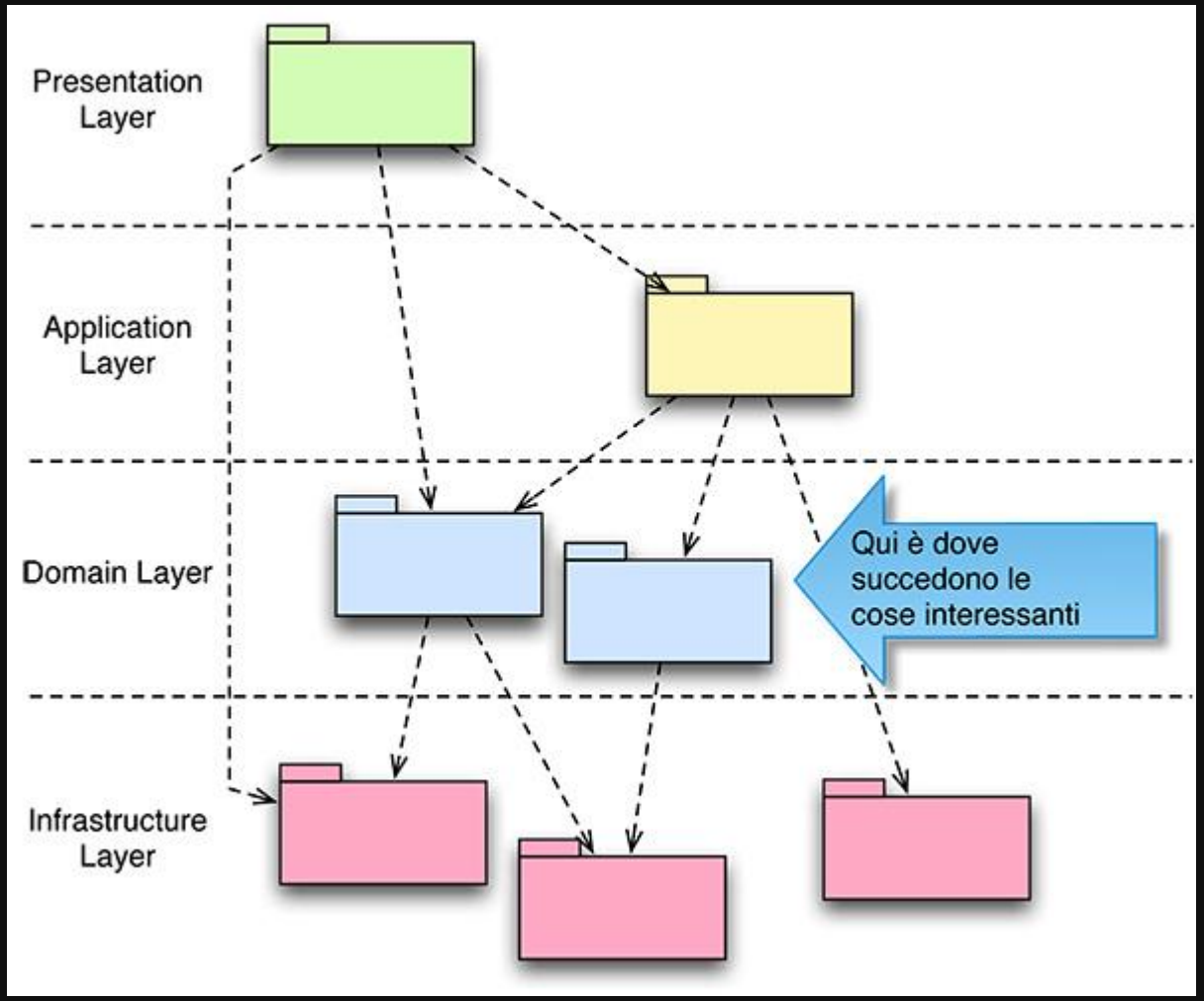
“Service Layer” [P of EAA, 133]

Defines an application's boundary with a layer of services that establishes a set of available operations and coordinates the application's response in each operation.

A Service Layer defines an application's boundary [Cockburn PloP] and its set of available operations from the perspective of interfacing client layers. It encapsulates the application's business logic, controlling transactions and coordinating responses in the implementation of its operations



DDD “Architecture”



Service Layer vs. Domain Model

Application Layer: Defines the jobs the software is supposed to do and directs the expressive domain objects to work out problems. The tasks this layer is responsible for are meaningful to the business or necessary for interaction with the application layers of other systems. This layer is kept thin. It does not contain business rules or knowledge, but only coordinates tasks and delegates work to collaborations of domain objects in the next layer down.

Domain Layer (or Model Layer): Responsible for representing concepts of the business, information about the business situation, and business rules. State that reflects the business situation is controlled and used here, even though the technical details of storing it are delegated to the infrastructure. This layer is the heart of business software. [Evans, DDD]

DDD Key Concepts

- Il Dominio è composto da **Entità** (identità e stato) e **Valori** (solo stato)
- La persistenza di entità e valori è gestita da **Repository**
- Le entità contengono la domain logic
- L'application logic è implementata sotto forma di **Servizi**
- Le istanze delle entità di dominio sono costruite da **Factory** (pattern **Builder** [GoF])
- Entità e Valori a runtime formano dei grafi di oggetti. I grafi dotati di "dignità propria" sono chiamati **Aggregate** e il sistema (leggi: il/i Repository) si "impegna" a gestirli correttamente
- GUI?!?!?! Cos'è la GUI?!?!?!? ☺

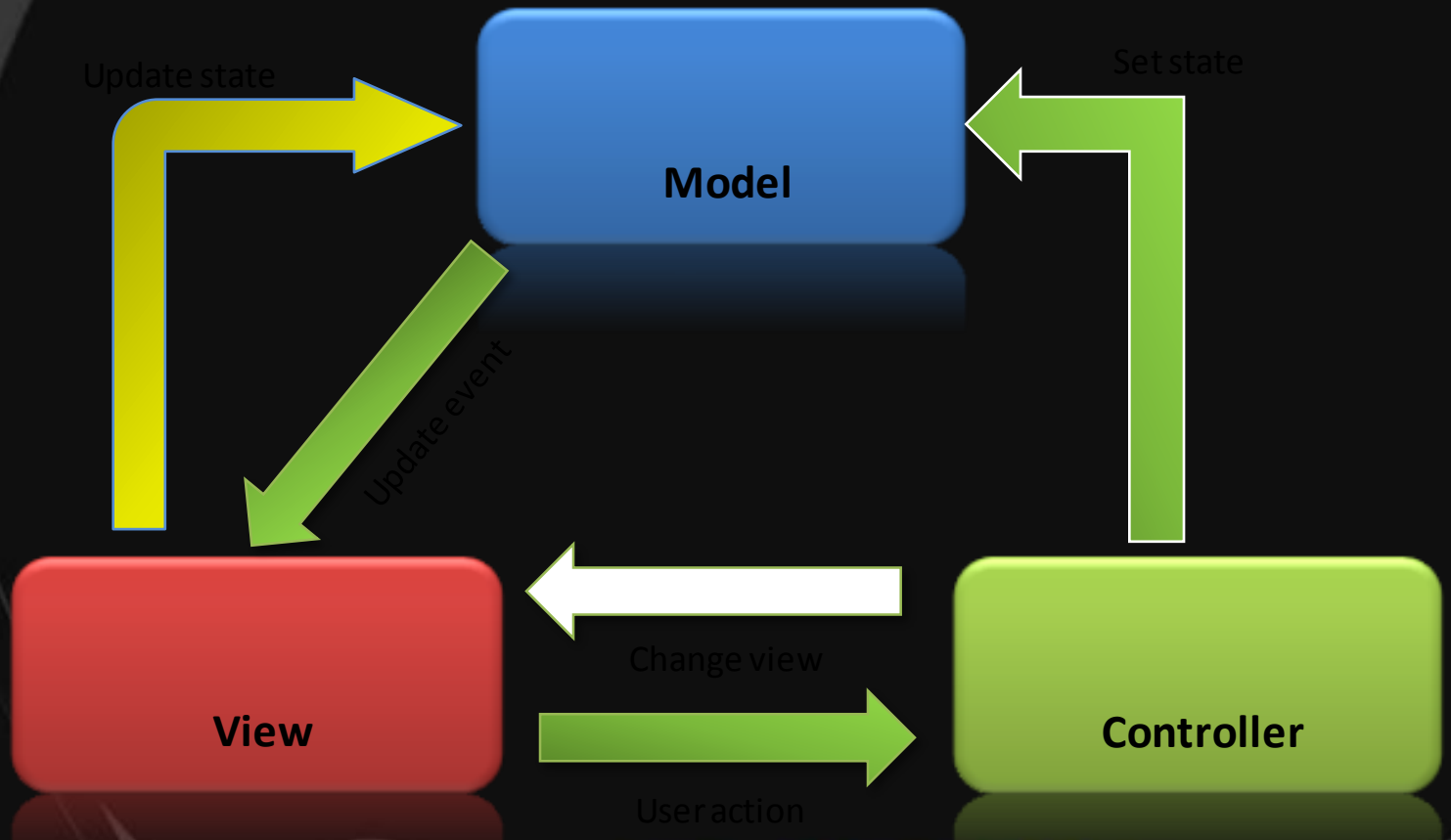
Model View Controller

Formulato nel 1979 da **Trygve Reenskaug**, membro del team Smalltalk @ **Xerox PARC**

La prima motivazione di design è:
Separare gli aspetti di presentazione da quelli di dominio.

E' la "radice concettuale" di tutti gli attuali pattern:
Model View Presenter, Presenter-First, Passive View, Presentation Abstraction Control, HMVC...

MVC - Model View Controller



MVC vs. WWW

MVC presuppone una relazione di tipo **Observer** tra Modello e View, ma:

- Observer è “stateful”
- HTTP è stateless

In pratica, sul web si usa una “revisione” di MVC chiamata Model 2 (implementata da tutti i toolkit, es: ASP.NET MVC e Castle.Monorail)

Bibliografia “aulica”

[ISO/IEC 9126] Software engineering — Product quality

[ISO/IEC 42010] Systems and software engineering — Recommended practice for architectural description of software-intensive systems

[ISO/IEC 19501] Unified Modeling Language (UML) Version 1.4.2

Bibliografia “spicciola”

[DDD] **Domain Driven Design**, Eric Evans, Addison-Wesley

[NAAE] **Microsoft .NET: Architecting Applications for the Enterprise**, Andrea Saltarello & Dino Esposito, Microsoft Press

[P of EAA] **Pattern of Enterprise Application Architecture**, Martin Fowler, Addison-Wesley

Thanks & links a materiale

http://blogs.ugidotnet.org/pape

http://nsk.codeplex.com

.NETcamPUS 2010

| Meet | Learn | Share |

.NETcamPUS 2010

| Meet | Learn | Share |