Design Patterns:

Model - View - Controller

Software Systems - Programming - 5M3

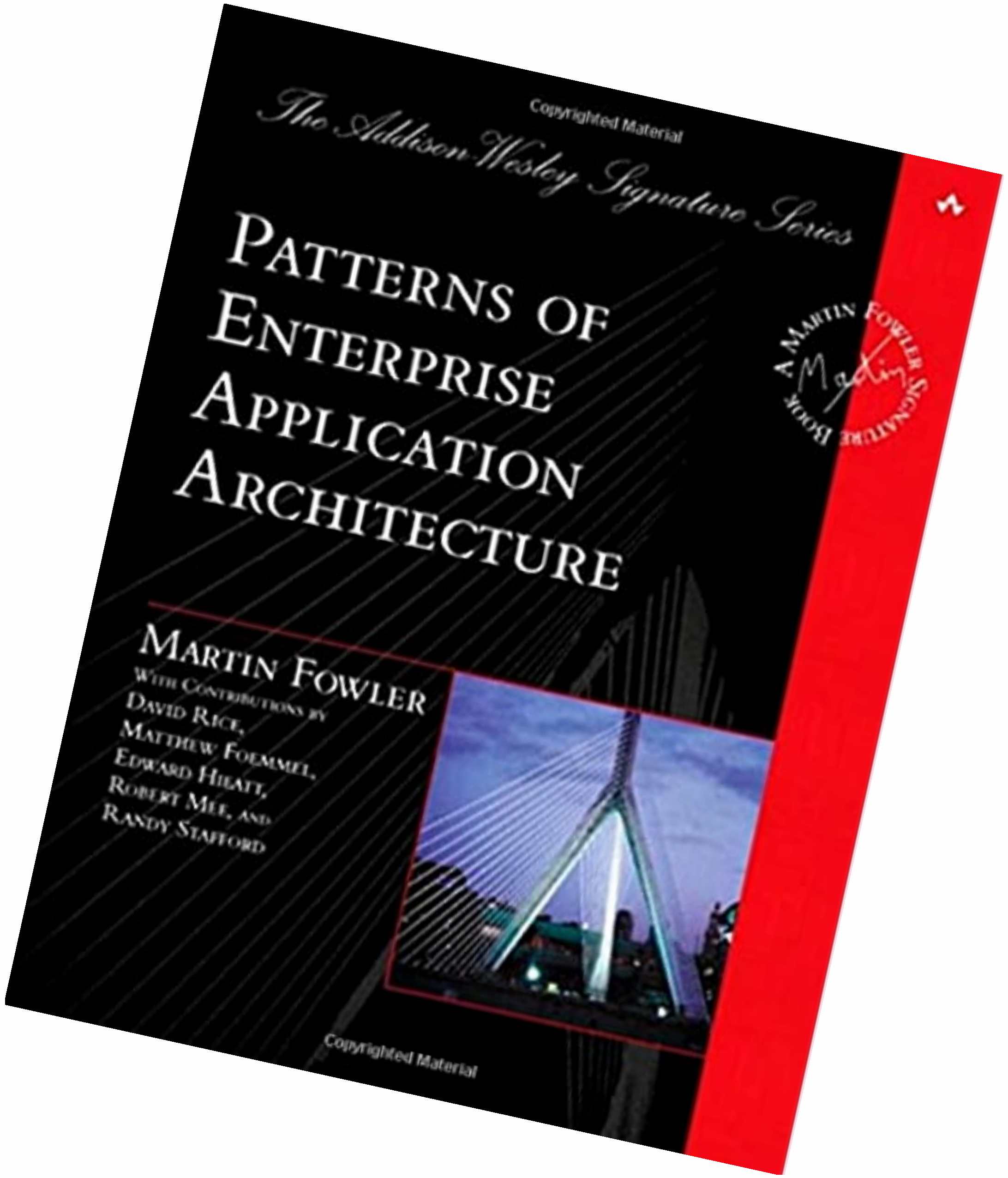
Dr. Vadim Zaytsev aka @grammarware, November/December 2020

UNIVERSITY OF TWENTE. | Formal Methods & Tools

Design Patterns

- Model–View–Controller

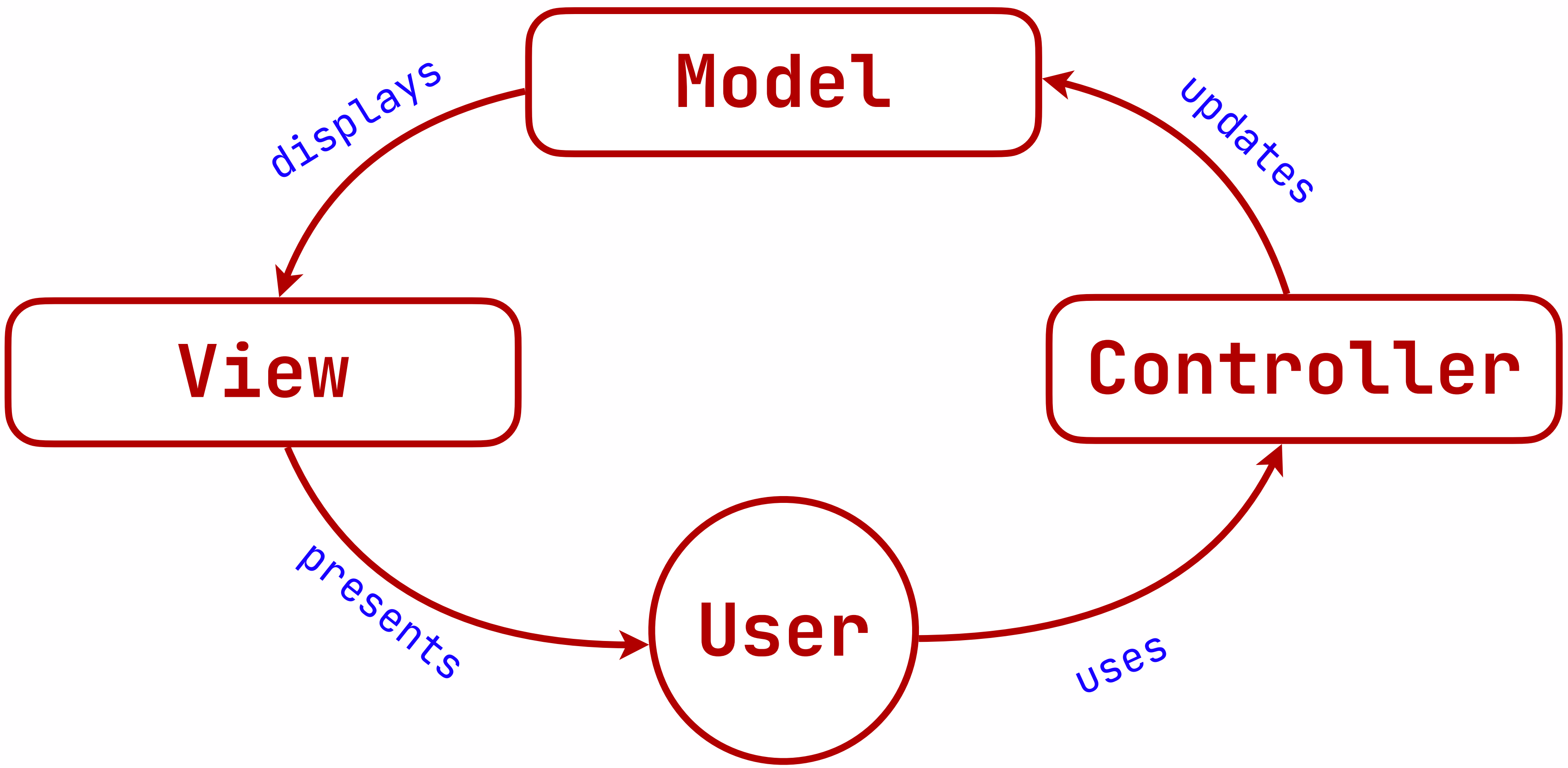
Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES



The Addison-Wesley Signature Series
Patterns of Enterprise Application Architecture
Martin Fowler
With contributions by David Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, and Randy Stafford
A Martin Fowler Signature Book

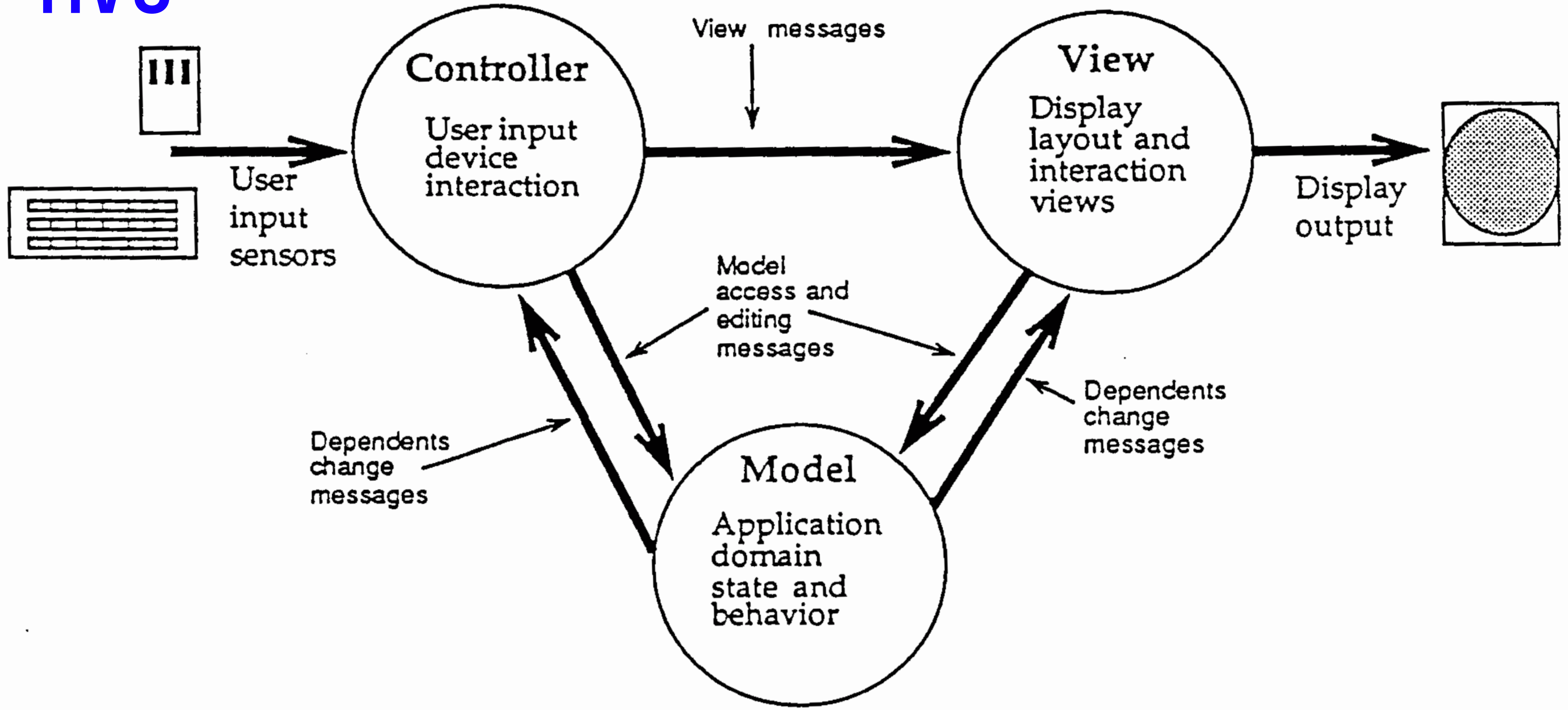
Design Patterns (1994), _Patterns of EAA_ (2002)

MVC

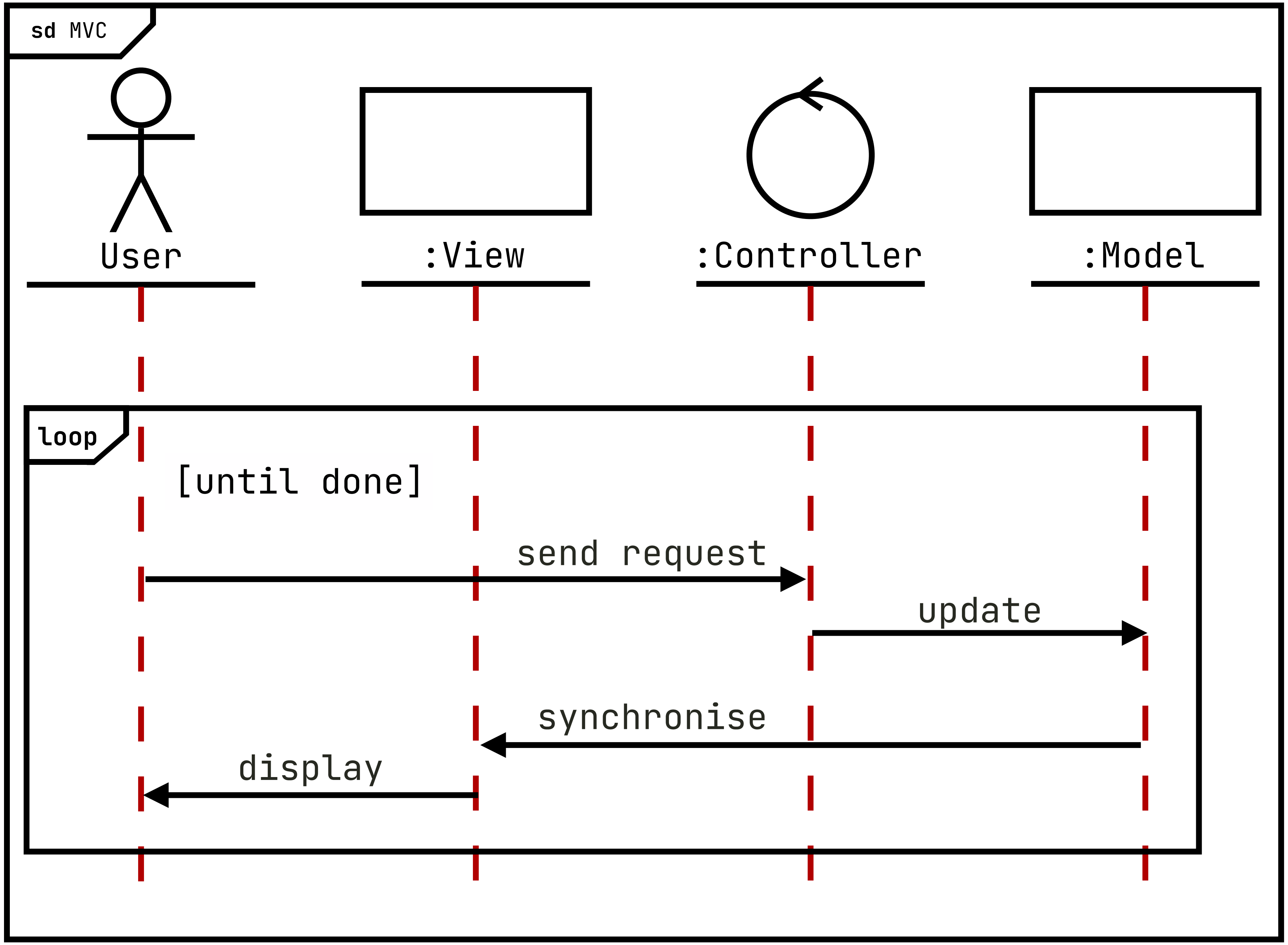


https://en.wikipedia.org/wiki/Model-view-controller

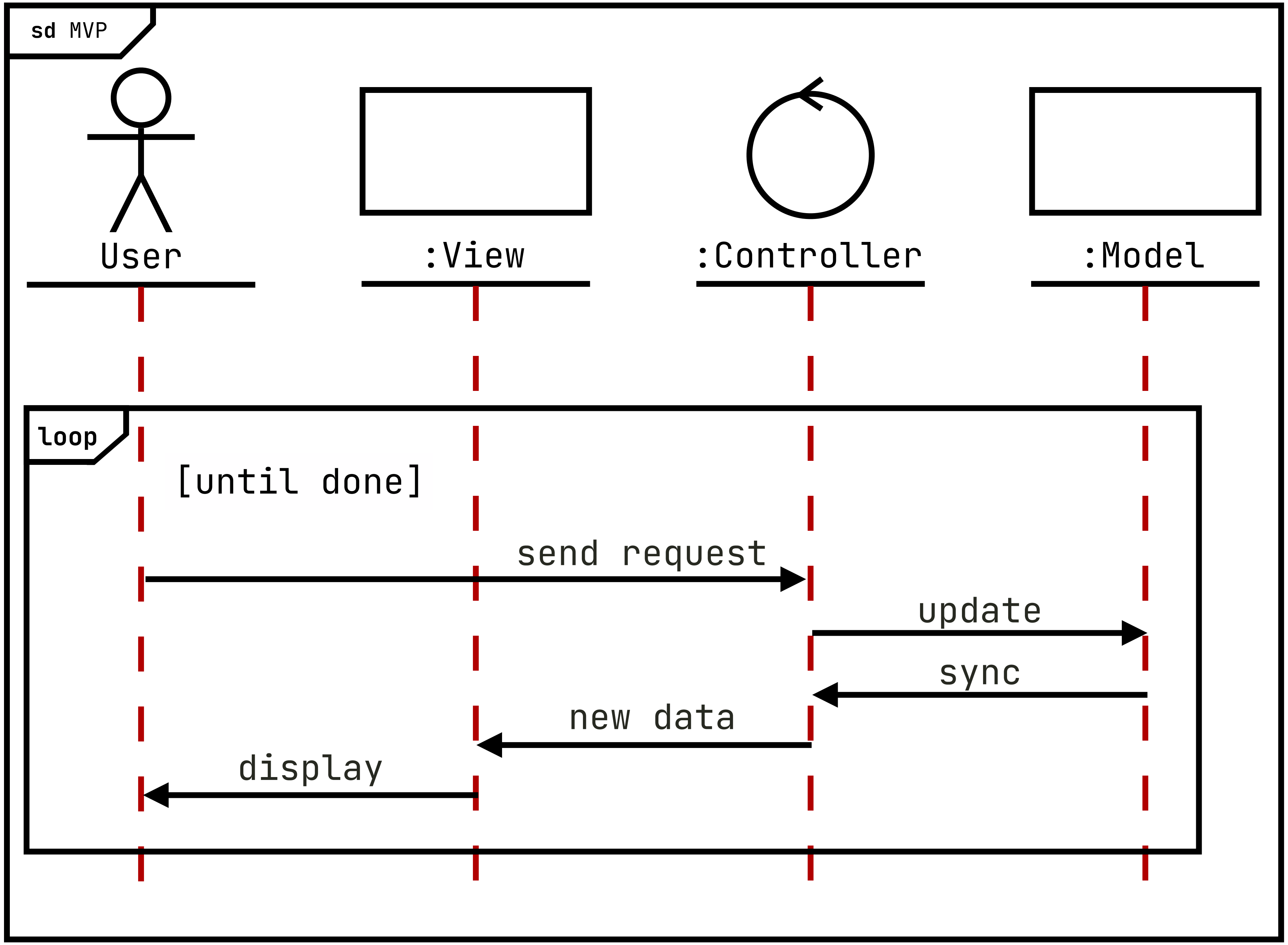
MVC


Controller
User input device interaction
View messages
View
Display layout and interaction views
Display output
User input sensors
Model access and editing messages
Dependents change messages
Dependents change messages
Model
Application domain state and behavior

Krasner, Pope, *A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System*, 1988


sd MVC
User
:View
:Controller
:Model
loop
[until done]
send request
update
synchronise
display

cf. L4T2


sd MVP
User
:View
:Controller
:Model
loop
[until done]
send request
update
sync
new data
display

cf. L4T2

Model

- Data + core logic
- Domain-specific simulation/implementation
- Can have multiple views and controllers
- Usually passive
- Examples:
  - `int`
  - `HashMap<String,Integer>`
  - `Library`

View

- Interface
- Usually graphical
- Visualisation of a model
- Can be hierarchical (superviews, subviews)
- Contains auto-updated bindings to one model
- Triggers events in the controller
- Example:
    - Swing `JTextField`

Controller

- Handler of user actions
- Reacts to input devices acting on a view
  - e.g., mouse click on a button
- Quite often implemented as a Listener
- Becomes more automated as frameworks progress
  - view-model bindings can be automated
  - then the model auto-updates the view

Model/View Separation

- Completely different domains!
  - domain model, scenario, script
  - user actions, buttons, widgets
- Devs can specialise
- Multiple views are a necessity
- Testable design is modular

Fowler et al, *Patterns of EAA* (2002)

Model/Controller Separation

- Naïve: model is data, controller is code
- The separation is between
  - business logic
  - user interaction logic
- Requires different expertise
- UX

View/Controller Separation

- Also different domains:
  - events, bindings, hiding/propagation, editability
  - buttons, editfields, comboboxes, dropdown lists
- As complexity of both grows
  - separation is different

Fowler et al, *Patterns of EAA* (2002)

Views and Controllers

wave to activate

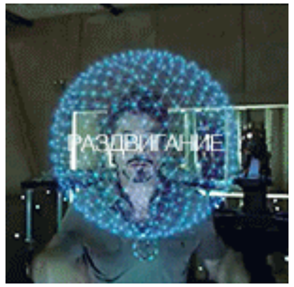
push to move

turn to rotate

swipe to dismiss

touch to select

spread to scale



raise hand to shoot

MAKE IT SO
Interaction Design Lessons from Science Fiction
by NATHAN SHEDROFF & CHRISTOPHER NOESSEL
foreword by Bruce Sterling
Rosenfeld

Alternatives to MVC

- Page Controller
- Front Controller
- Application Controller
- Template View
- Transform View
- Model–View–Presenter
- . . .

Conclusion

- Model is the core abstraction
- View is how the user sees it
- Controller is how user actions are interpreted
- Dependencies can differ
- Often
  - misunderstood
  - misinterpreted
  - misimplemented