TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Institut für Managementwissenschaften

Enterprise Information Systems
Sommersemester 2014

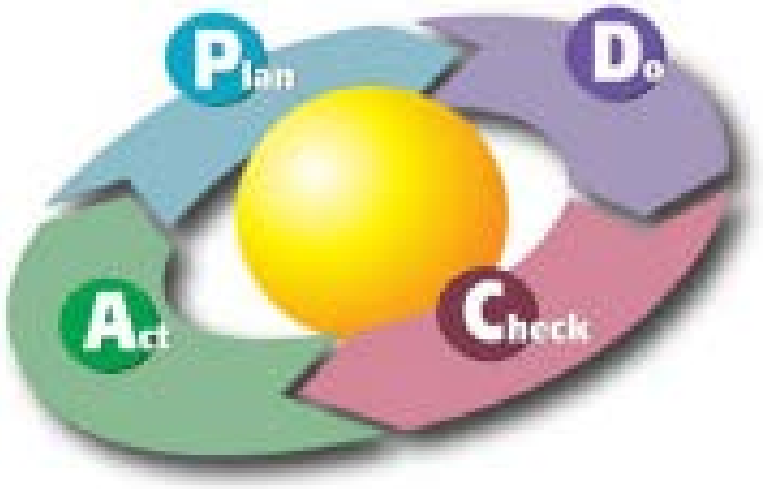
Übungsaufgabe

Reichl D. / Schwaiger W. / Stojkovic I.

Agenda

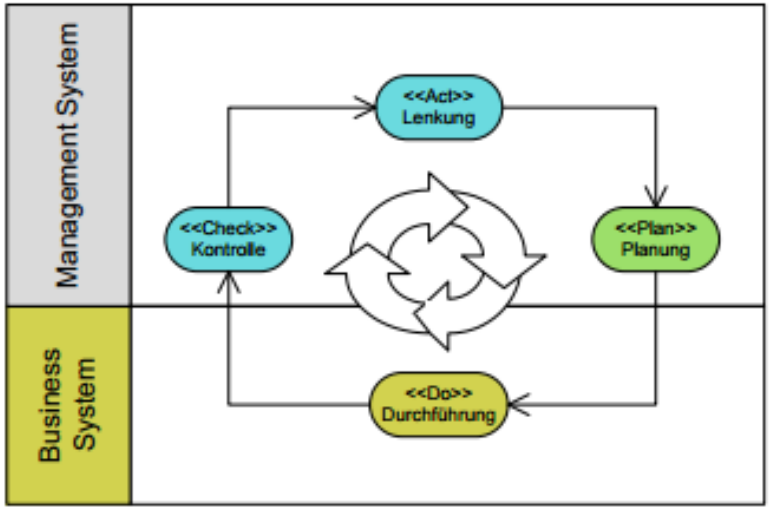
- Wiederholung PDCA + Beispiele
- CPPI Investment: Funktionsweise, Design und Implementierung
- PDCA-basierte MGT Infrastruktur
- Übungsaufgabe (Gruppenarbeit)

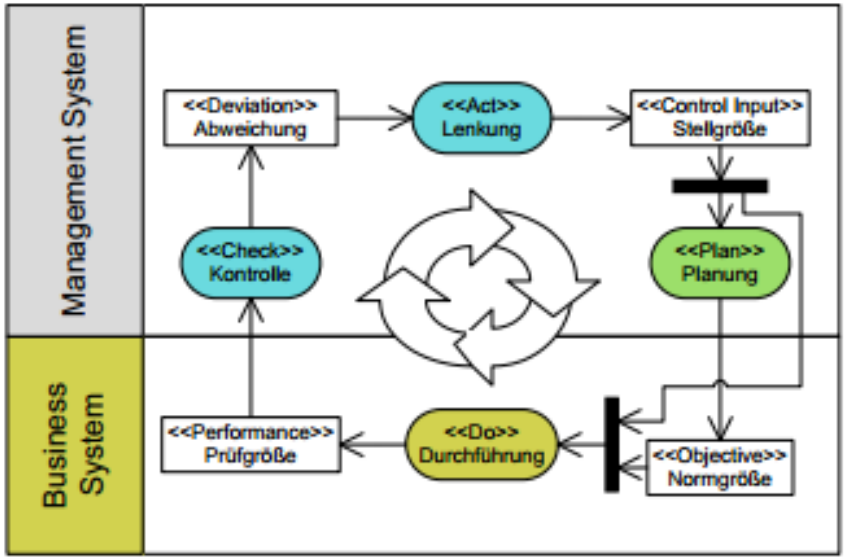
Wiederholung PDCA + Beispiele



Wiederholung PDCA + Beispiele

Modellierung im PDCA-Diagramm



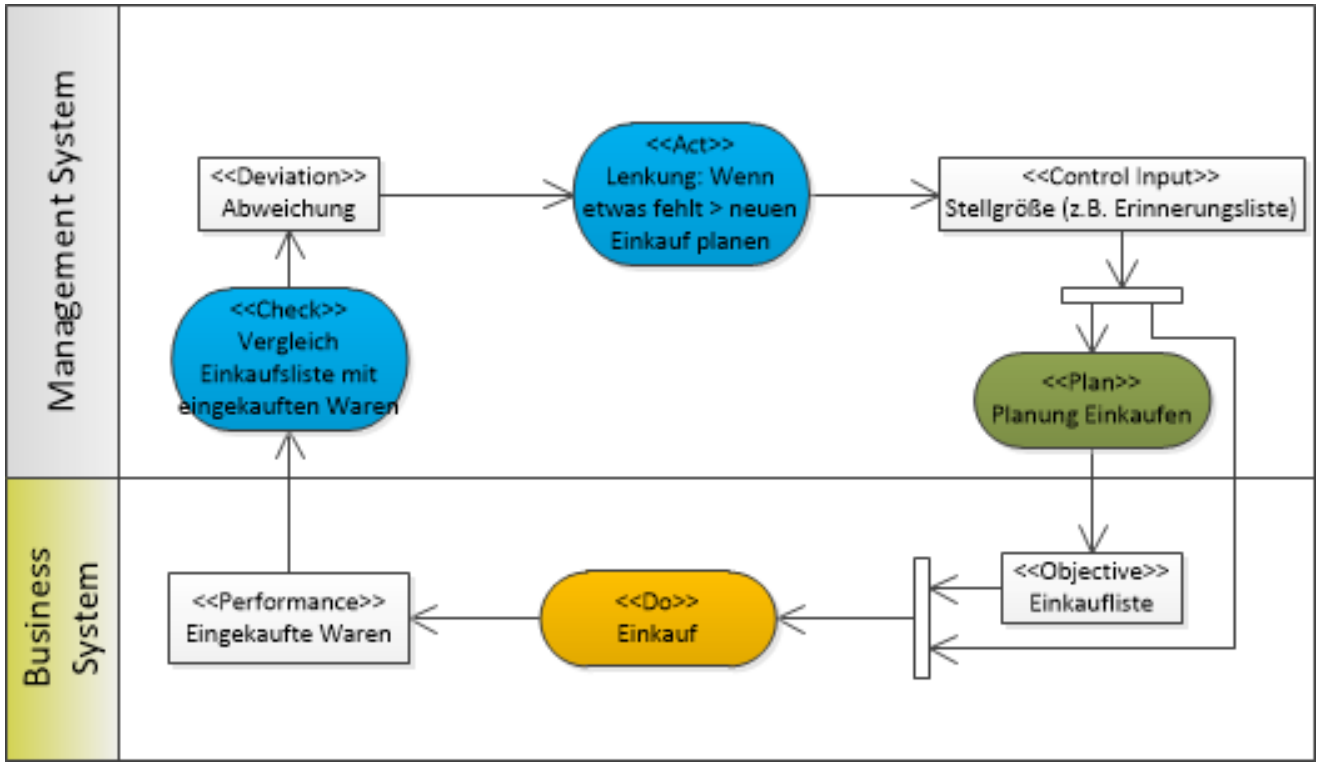


Wiederholung PDCA + Beispiele

Beispiel - Einkaufen

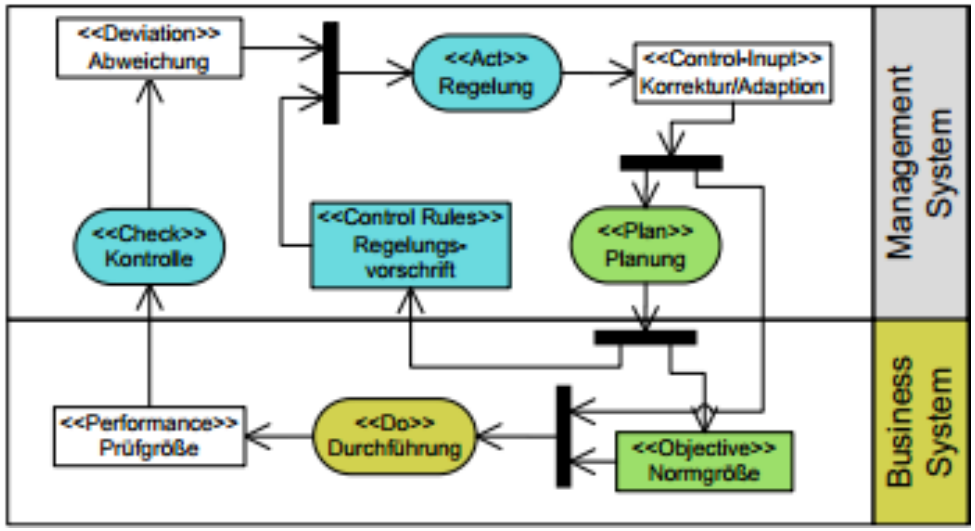
Wiederholung PDCA + Beispiele

Beispiel - Einkaufen



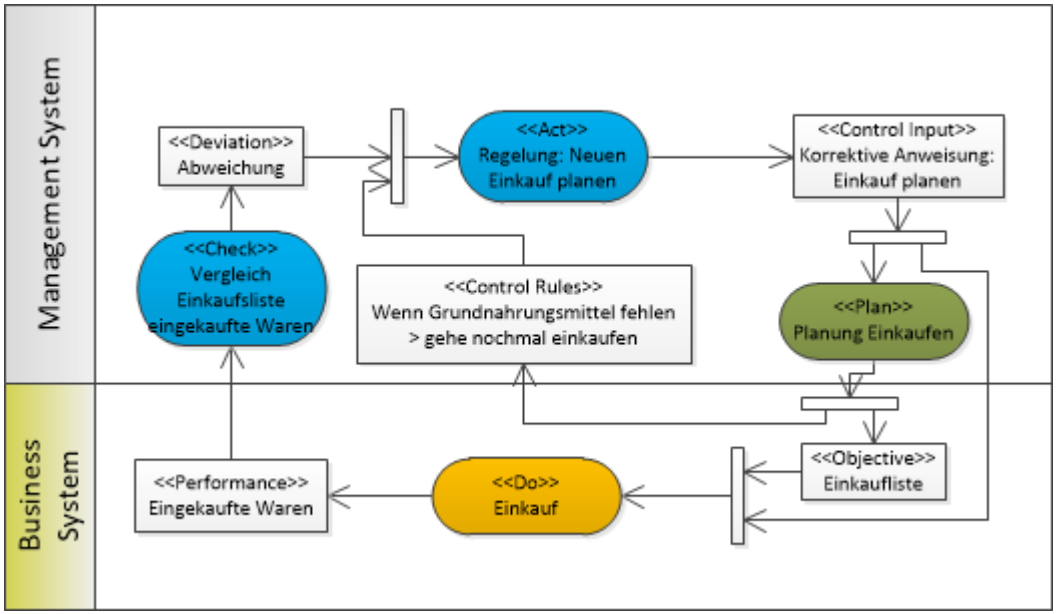
Closed Loop MGT

- Inkludiert eine Regelungsvorschrift, welche angibt, welche konkreten Anpassungen mit den Abweichungen verbunden sind



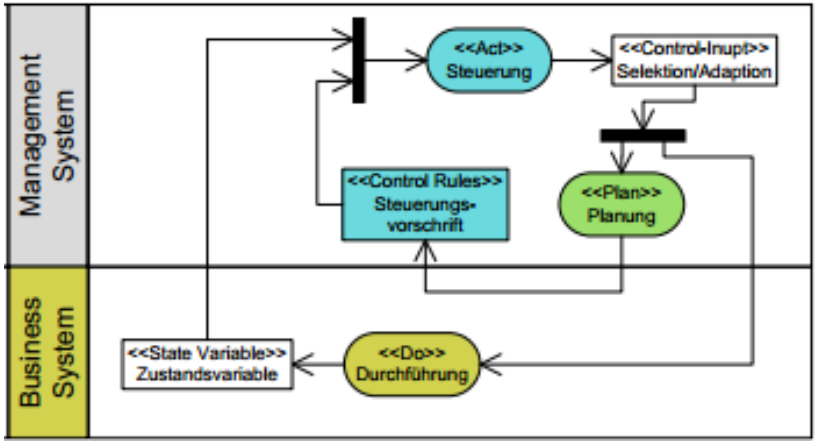
Closed Loop MGT beim Einkaufen

- Regelungsvorschrift: Wenn Grundnahrungsmittel fehlen > gehe nochmal einkaufen



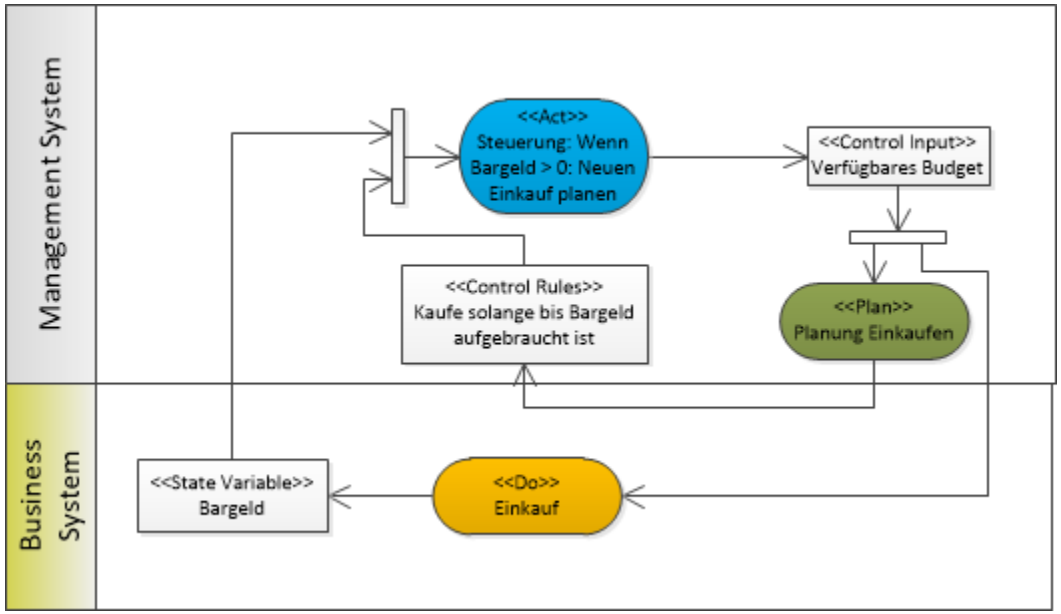
Open Loop MGT

- Inkludiert eine Steuerungsvorschrift, welche angibt, welche konkrete Anpassungen mit den verschiedenen Ausprägungen der Zustandsvariable verbunden sind. Die Steuerung hat keine Kontrolle (Check).



Open Loop MGT beim Einkaufen

- Steuerungsvorschrift: Kaufe solange bis Bargeld aufgebraucht ist



Single & Double Loop

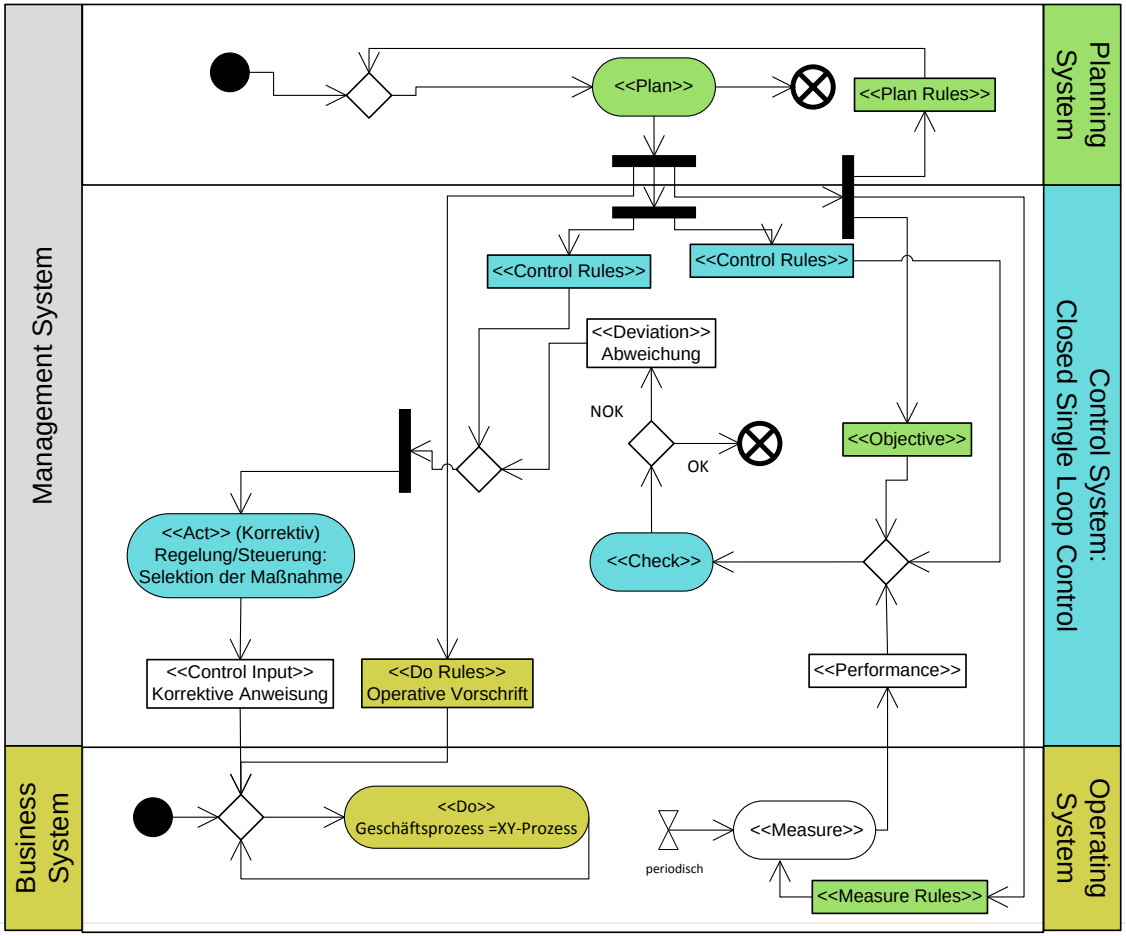
- Single Loop: Es wirken korrektive Anpassungen in den Geschäftsprozess zurück
- Double Loop: Es wirken korrektive Anpassungen in den Geschäftsprozess zurück als auch adaptive Anpassungen des MGT-Prozesses, indem über die Planung die Ziele, die Steuerungsvorschrift bzw. die operative Vorschrift angepasst werden.

Weiteres Beispiel

- Überlegen Sie wie ein Open Loop MGT Prozess und ein Closed Loop MGT Prozess für „Koffer packen“ ausschauen könnte.
- Was könnten mögliche Regel- bzw. Steuervorschriften darstellen?
- Was könnten adaptive (Single Loop) also auch korrektive Anpassungen (Double Loop) darstellen?

Wiederholung PDCA + Beispiele

Beispiel – Kofferpacken (Live mit Visio)



Agenda

- Wiederholung PDCA + Beispiele
- **CPPI Investment: Funktionsweise, Design und Implementierung**
- PDCA-basierte MGT Infrastruktur
- Übungsaufgabe (Gruppenarbeit)

CPPI: Funktionsweise, Design und Implementierung

Constant Proportion Portfolio Insurance

Die **Constant Proportion Portfolio Insurance** (CPPI) ist eine dynamische Portfolio-Absicherungsstrategie.

Grundkonzept [Bearbeiten]

http://de.wikipedia.org/wiki/Constant_Proportion_Portfolio_Insurance

Das Ziel der Portfolioabsicherungsstrategien ist es, das Verlustrisiko im Falle sinkender Kurse an den Wertpapiermärkten zu begrenzen und gleichzeitig eine Partizipation an steigenden Wertpapiermärkten zu ermöglichen. Zu den Instrumenten der CPPI-Strategie gehören risikobehaftete Finanzanlagen (Aktien) und risikolose Festzinsanlagen (Geldmarktfondszertifikate). Da dies eine dynamische Strategie ist, findet während des gesamten Betrachtungszeitraums eine permanente Portfolioumschichtung zwischen risikobehafteten und risikolosen Anlagen statt. Bei der CPPI-Strategie wird von einem ex-ante festgelegten Portfoliomindestwert ausgegangen.

Um komplexe Sachverhalte des Grundkonzeptes der CPPI-Strategie möglichst einfach und verständlich darzustellen, wird von folgenden Annahmen ausgegangen:

- Auf dem Markt wird mit dividendenlosen Aktien gehandelt.
- Es besteht keine Möglichkeit Fremdkapital aufzunehmen.
- Es sind keine Leerverkäufe zugelassen.
- Es ist möglich jede beliebige Anzahl der Wertpapiere zu kaufen (beliebige Teilbarkeit).
- Auf dem Markt herrscht während der ganzen Anlageperiode ein konstanter Zins.

CPPI-Modellierung: B&M-, IS- und IT-Perspektive

a) CPPI-Modell: Mathematische Modellierung
b) CPPI-Modell: Beispielshafte Umsetzung
c) CPPI-Modell: Semantische Modellierung mit dem MGT-Aktivitätsdiagramm
d) CPPI-Modell: Abstrakte Implementierung in der Management Infrastruktur
e) CPPI-Modell: Konkrete Implementierung in der Management Infrastruktur

CPPI-Modell: Mathematische Modellierung (1/2)

1. Setzen des Mindestportfoliowerts für das Ende des Betrachtungshorizonts: F_T
2. Berechnung der Wertuntergrenze über den Betrachtungshorizont: $F_t = \frac{F_T}{(1+R_t)^{T_{0,t}}}$
3. Berechnung des Sicherheitspolsters: $C_t = \max(W_t - F_t; 0)$
4. Berechnung des Aktieninvestments: $X_{r,t} = \min(m \cdot C_t; b \cdot W_t)$
5. Berechnung des risikolosen Investments: $X_{f,t} = W_t - X_{r,t}$
6. Warten bis zum Ende der Periode
7. Berechnung der Aktienrendite: $TSR_t = \frac{S_t}{S_{t-1}} - 1$
8. Berechnung des Portfoliowerts: $W_t = X_{r,t-1} \cdot (1 + TSR_t) + X_{f,t-1} \cdot (1 + R_0)^{1/365}$
9. Gehe zu 3.

CPPI-Modell: Mathematische Modellierung (2/2)

- Legende

F_T	Floor Value (Mindestportfoliowert) für Zeitpunkt T
F_t	Floor-Barwert im Zeitpunkt t
S_t	Stock Price (Aktienkurs) im Zeitpunkt t
C_t	Cushion (Sicherheitspolster) im Zeitpunkt t
$X_{r,t}$	Risky Exposure (Aktieninvestment) im Zeitpunkt t
$X_{r,t}$	Riskless Exposure (risikoloses Investment) im Zeitpunkt t
W_t	Wealth (Portfoliowert) im Zeitpunkt t
TSR_t	Total Shareholder Return (Aktienrendite) vom Zeitpunkt t-1 bis t
R_0	Risikoloser Abzinsungsatz (p.a.)
T_{tT}	Fristigkeit (Zeitraum) von t bis T in Jahren
m	Multiplier (Multiplikator)
b	Maximum Risky Fraction (maximaler Aktienanteil)

CPPI-Modell: Beispielshafte Umsetzung

a) Festlegung der Modellparameter

F_T	100
R_0	5%
$T_{0,T}$	1
W_0	100
m	2
b	90%
d	365

b) Durchführung der Berechnungen

t	$T_{t,T}$	F_t	C_t	$X_{r,t}$	$X_{f,t}$	S_t	TSR_t	W_t
0	1,0000	95,24	4,76	9,52	90,48	100		
1	0,9973	95,25	5,24	10,47	90,01	105	5,00%	100,49
2	0,9945	95,26	5,04	10,07	90,23	103	-1,90%	100,30
3	0,9918	95,28	4,35	8,70	90,92	96	-6,80%	99,63
4	0,9890	95,29	4,08	8,16	91,21	93	-3,13%	99,37
5	0,9863	95,30	4,43	8,86	90,87	97	4,30%	99,73
6	0,9836	95,31	4,98	9,95	90,34	103	6,19%	100,29
7	0,9808	95,33	5,46	10,92	89,87	108	4,85%	100,79
8	0,9781	95,34	5,86	11,73	89,48	112	3,70%	101,20
9	0,9753	95,35	4,61	9,21	90,75	100	-10,71%	99,96
10	0,9726	95,37	4,97	9,95	90,39	104	4,00%	100,34

Semantische Modellierung mit dem MGT-Aktivitätsdiagramm (1/2)

- Betrachtung des CPPI-Modells als PDCA-Managementprozess mit

a) Check Rules: Vergleich von Portfoliowert und Wertuntergrenze

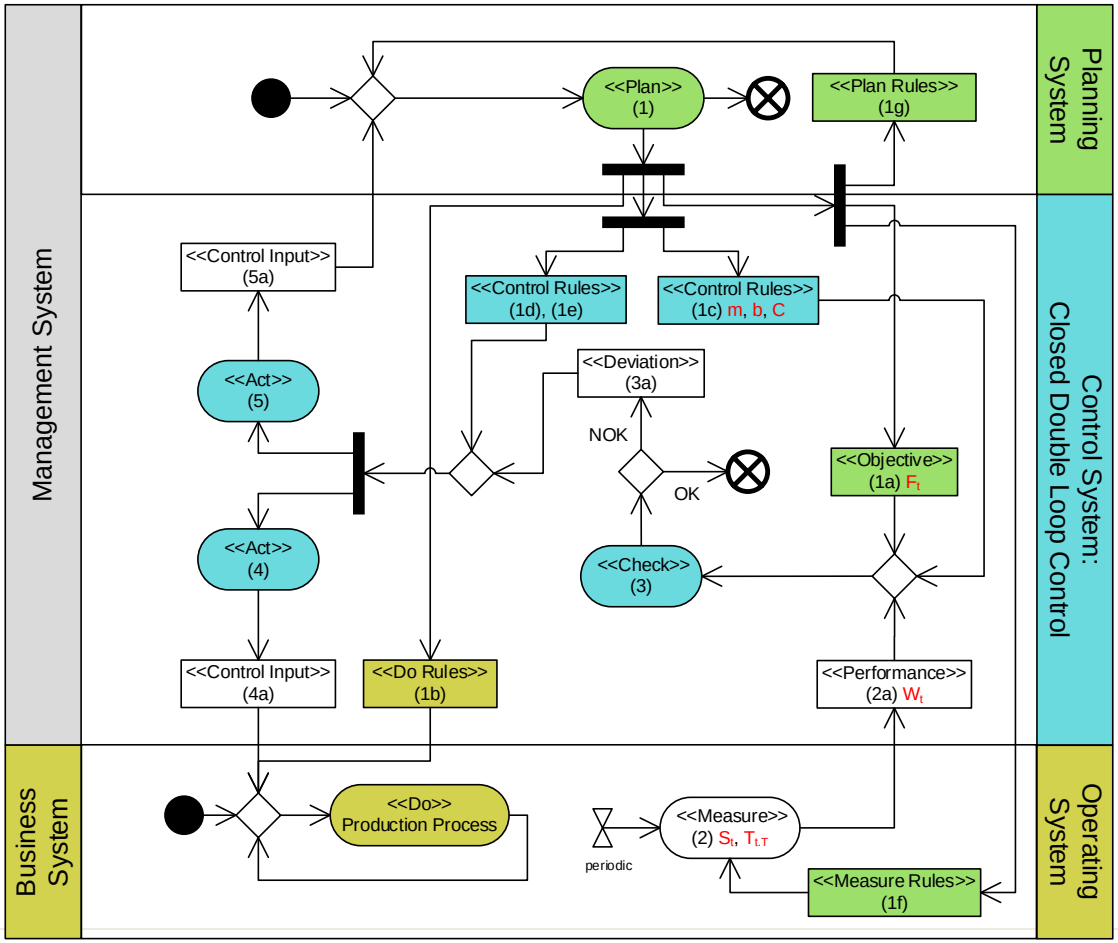
$$C_t = \max(\underbrace{W_t - F_t}_{\textit{Check Rule}};0)$$

b) Act Rules: Anpassung der Portfoliozusammenstellung

$$X_{r,t} = \underbrace{\min(m \cdot C_t; b \cdot W_t)}_{\textit{Act Rule}}$$

$$X_{f,t} = \underbrace{W_t - X_{r,t}}_{\textit{Act Rule}}$$

Semantische Modellierung mit dem MGT-Aktivitätsdiagramm (2/2)



Agenda

- Wiederholung PDCA + Beispiele
- CPPI Investment: Funktionsweise, Design und Implementierung
- **PDCA-basierte MGT Infrastruktur**
- Übungsaufgabe (Gruppenarbeit)

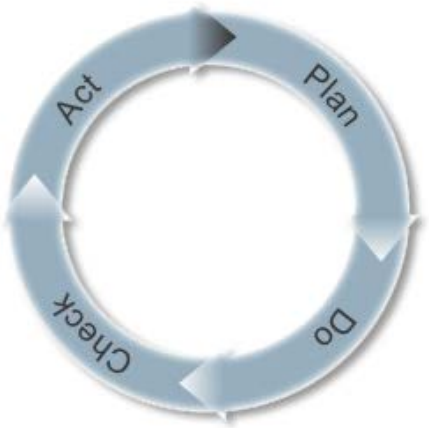
PDCA-basierte MGT Infrastruktur

Framework for Design and Implementation of MGT Processes

- Meta MGT process
- PDCA Foundation
- Support for designer in design process
- Support for programmer in development process

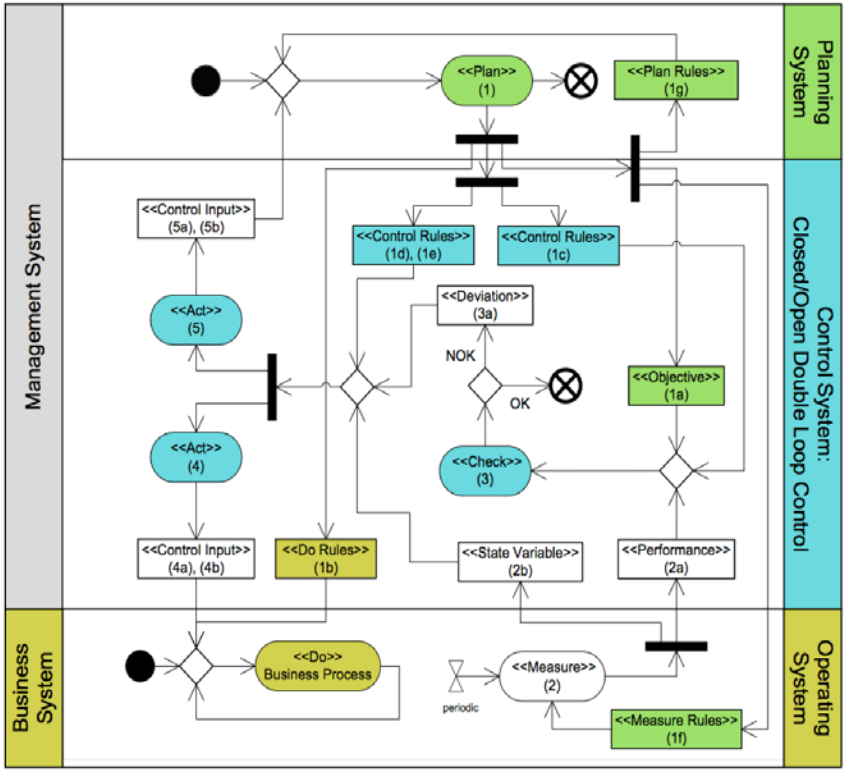
PDCA Foundation (1/2)

- PDCA-Cycle = international standard for modelling management processes
- cybernetic feedback mechanism
- four stages Plan, Do, Check, Act



PDCA Foundation (2/2)

- Management Activity Diagram: base for PDCA-based management processes

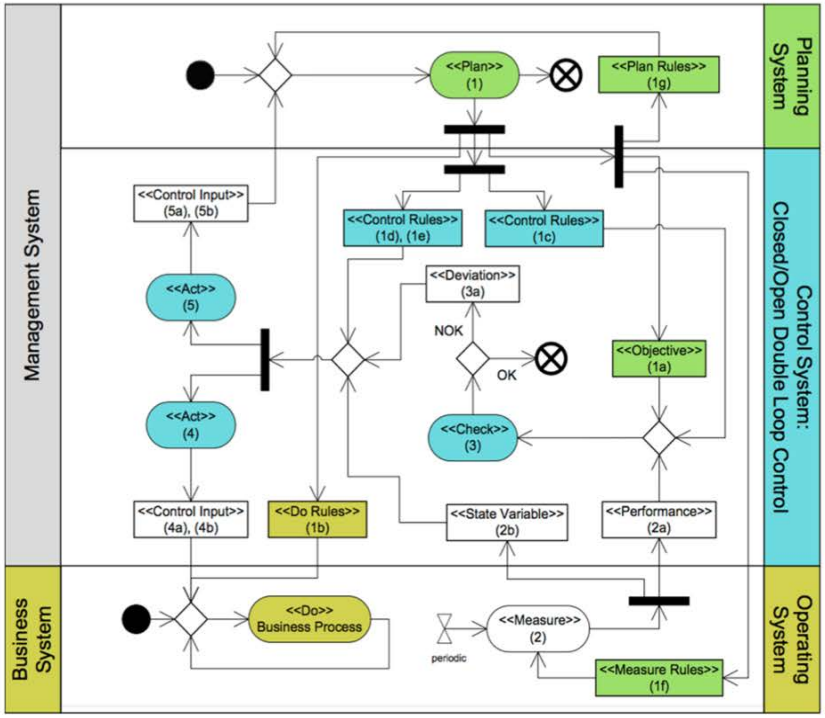


Support for Design of PDCA-based MGT Processes (1/5)

- User provides the designer with information about what is needed for one concrete MGT process
- Based on the process type designer chooses one of eight configurations from our framework
- The framework selects appropriate process skeleton represented as UML activity diagram
- Translation of generic elements into concrete elements
- Description of the contents of concrete elements

Support for Design of PDCA-based MGT Processes (2/5)

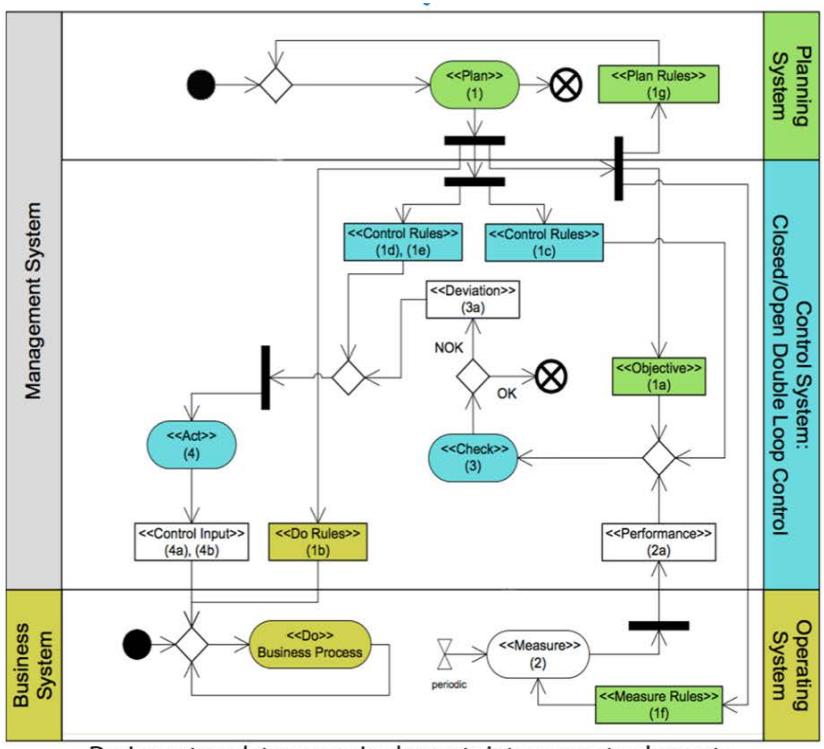
Framework is based on meta generic management process and offers eight configurations that describe different management processes



Designer chooses one of eight configurations from generic management process framework

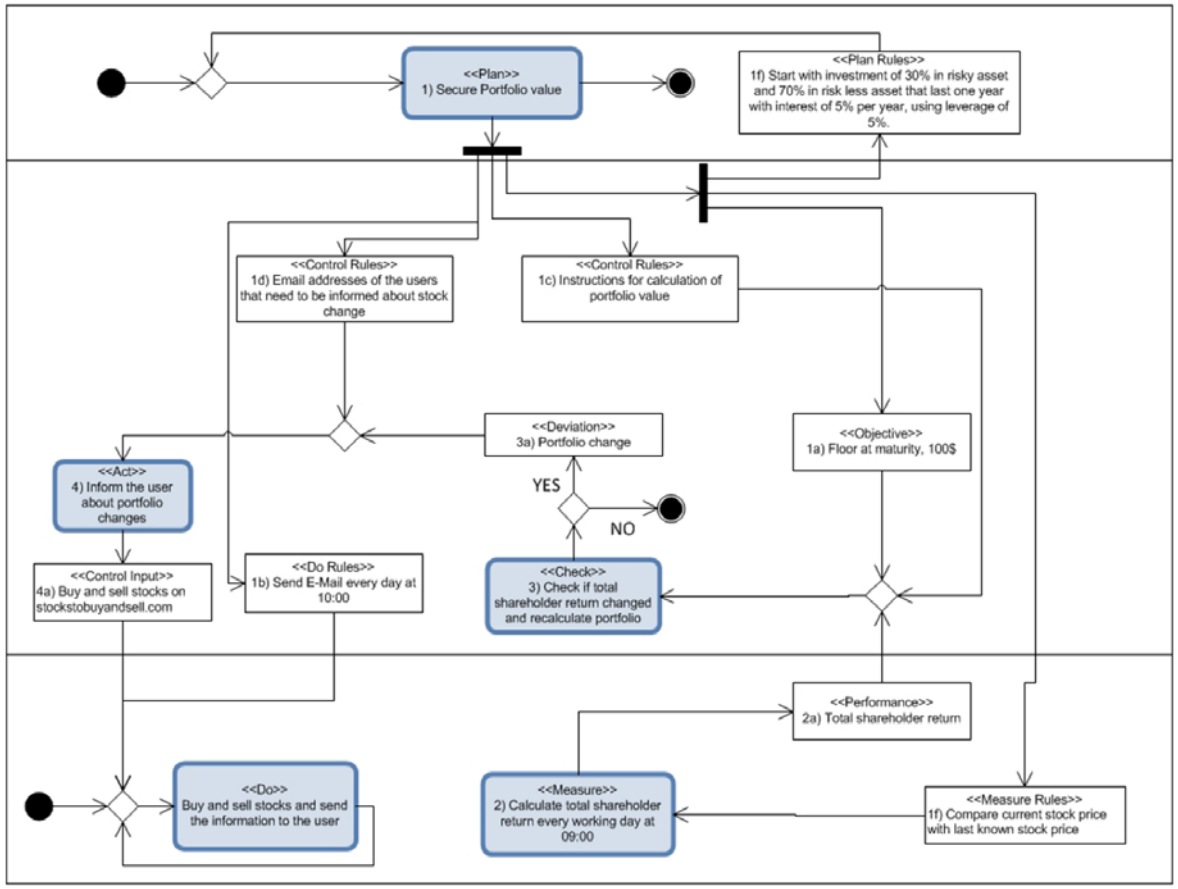
Support for Design of PDCA-based MGT Processes (3/5)

Framework decides which elements has to be concretized and provides the designer with skeleton for pdca based management process

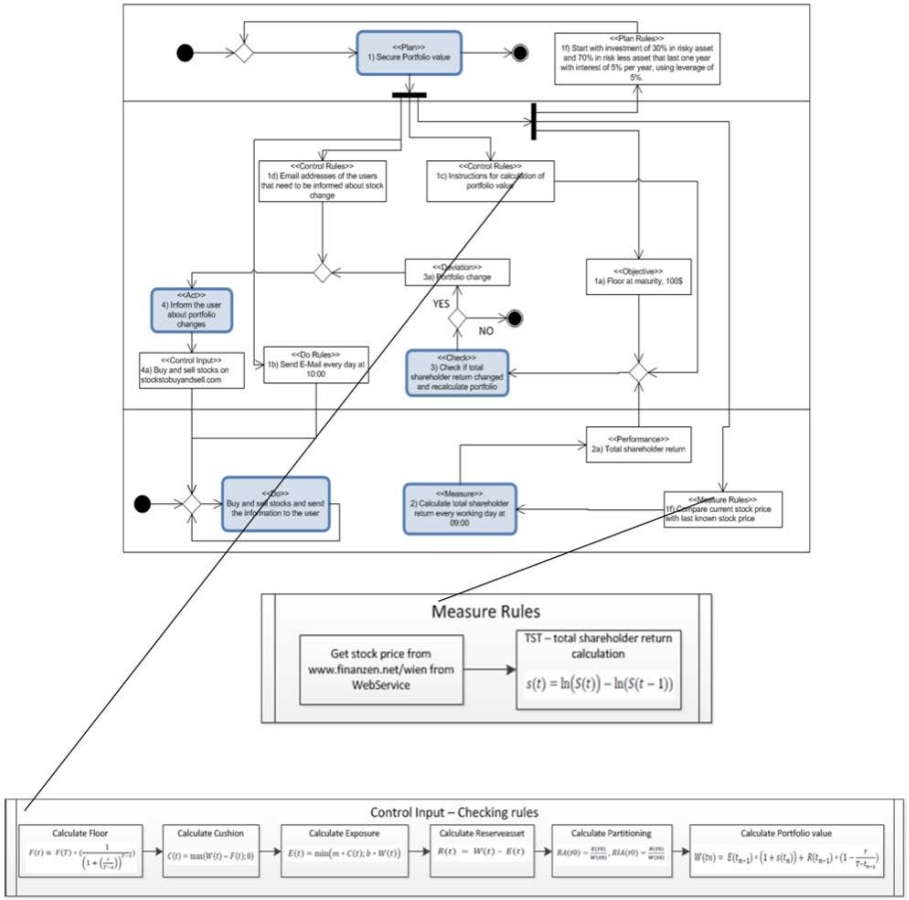


Designer translates generic elements into concrete elements

Support for Design of PDCA-based MGT Processes (4/5)



Support for Design of PDCA-based MGT Processes (5/5)

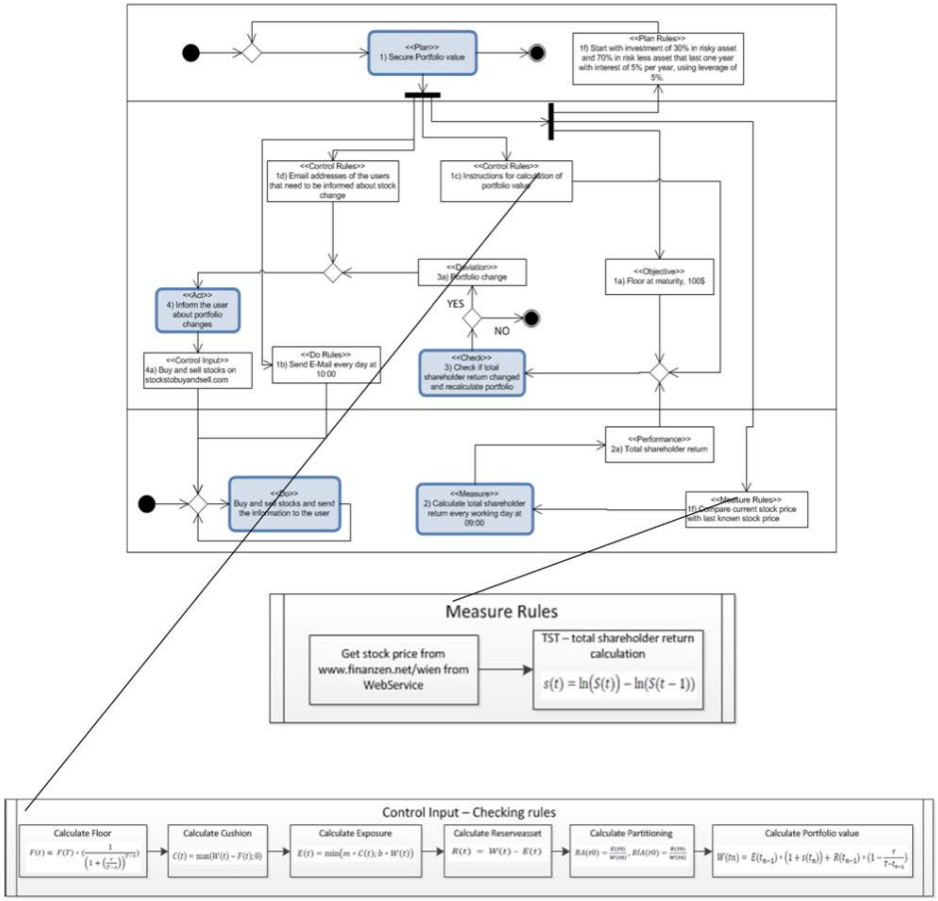


Support for Implementation of PDCA based MGT processes (1/12)

- Designer provides the programmer with concrete diagram and additional descriptions for each of the complex elements of the diagram
- Framework provides the programmer with generic implementation
- Programmer derives generic elements into concrete objects and implements the methods

PDCA-basierte MGT Infrastruktur

Support for Implementation of PDCA based MGT processes (2/12)

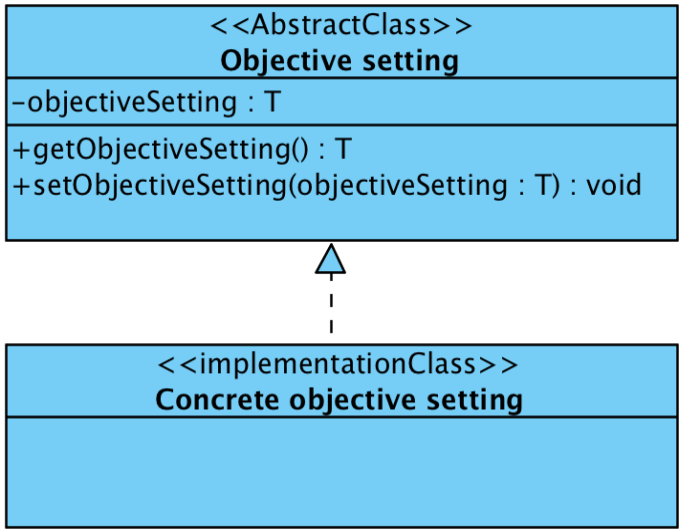


Support for Implementation of PDCA based MGT processes (3/12)

- Designer provides the programmer with concrete diagram and additional descriptions for each of the complex elements of the diagram
- Framework provides the programmer with generic implementation
- Programmer derives generic elements into concrete objects and implements the methods

Support for Implementation of PDCA based MGT processes (4/12)

- The programmer uses generic implementation and derives concrete objects



Support for Implementation of PDCA based MGT processes (5/12)

- The programmer derives generic objects and implements each element of a diagram

```
public class RiskMGTCheckProcess extends CheckProcess<BigDecimal> {

  private final static Log log = Logging.getLog(RiskMGTCheckProcess.class);

  private PDCAServiceLocal pdcaService;

  @SuppressWarnings("unchecked")
  public RiskMGTCheckProcess(RiskMGTCheckRules checkingRules, RiskMGTObjectiveSettir
    super();
    try {
      this.checkingRules = checkingRules;
      this.objectiveSetting = objective;
      this.performanceValue = performanceValue;
      pdcaService = (PDCAServiceLocal) PDCAHelper.getService(PDCAService.SVC_NAME);
    } catch (ServiceNotFoundException e) {
      e.printStackTrace();
    }
  }
```

Support for Implementation of PDCA based MGT processes (6/12)

- Generic implementation
- Generic programming
- Java Generics

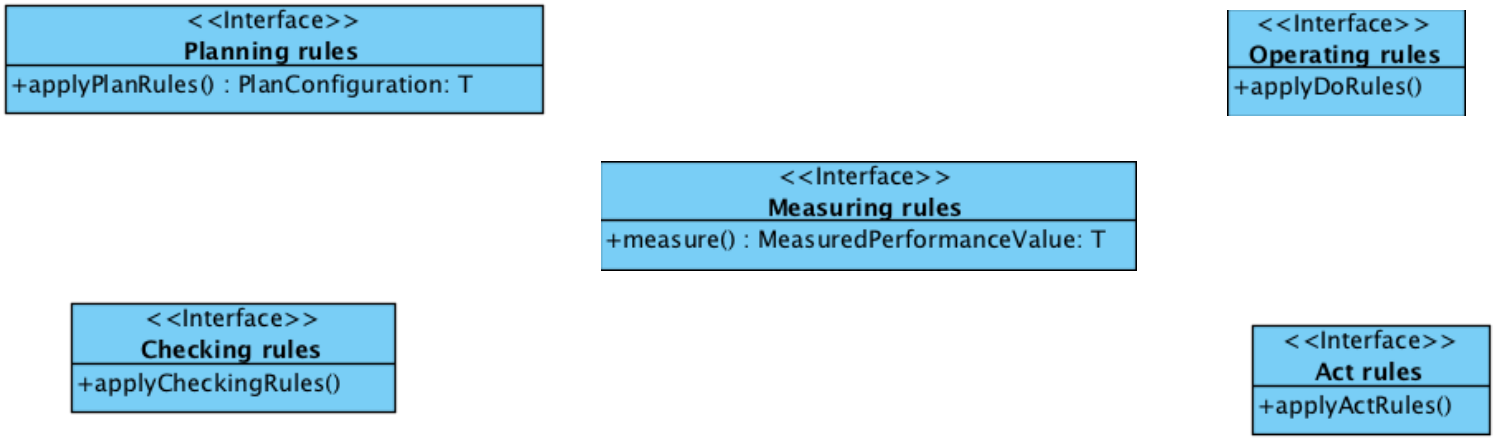
```
@MappedSuperclass
public abstract class PlanDeviation<T> {
  protected T value;

  public PlanDeviation(T value) {
    super();
    this.value = value;
  }

  public T getValue() {
    return value;
  }
}
```

```
@MappedSuperclass
public abstract class CorrectiveActOutput<T> {
  protected T value;

  public CorrectiveActOutput(T value) {
    super();
    this.value = value;
  }

  public T getValue() {
    return value;
  }
}
```

```
@MappedSuperclass
public abstract class MeasuredPerformanceValue<T> {

  T value;

  public T getValue() {
    return value;
  }

  public MeasuredPerformanceValue(T value) {
    super();
    this.value = value;
  }
}
```

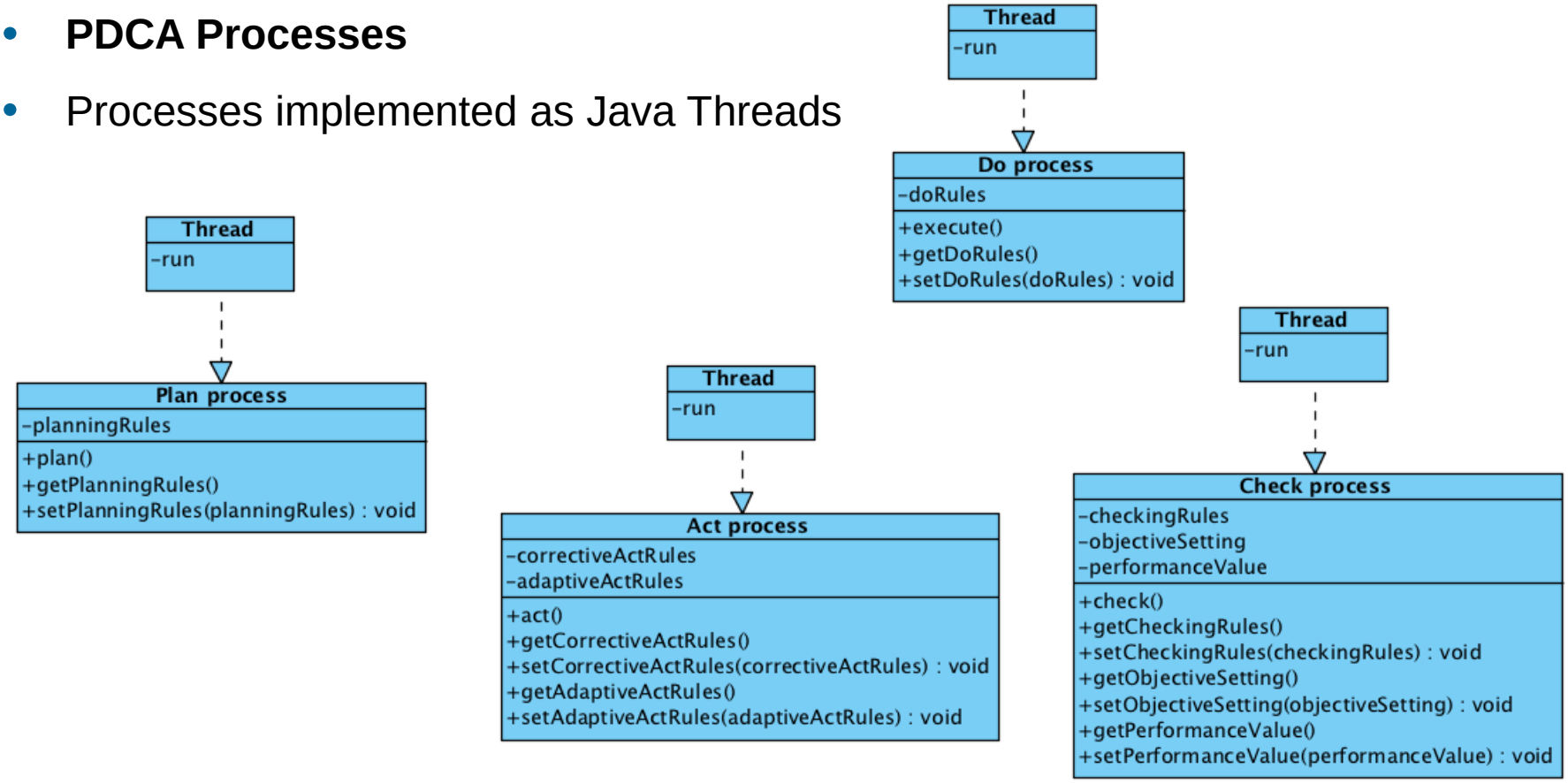
Support for Implementation of PDCA based MGT processes (7/12)

- **PDCA Rules**
- Rules described as interfaces and abstract classes



Support for Implementation of PDCA based MGT processes (8/12)

- **PDCA Processes**
- Processes implemented as Java Threads



Support for Implementation of PDCA based MGT processes (9/12)

- **Software Engineering principles**: Programming to the interface

```
public interface PlanCheckRules<T> {
  PlanDeviation<T> getCheckResult(PlanObjectiveSetting<T> objective,
      MeasuredPerformanceValue<T> performanceMeasureValue);
}
```

Support for Implementation of PDCA based MGT processes (10/12)

- **Software Engineering principles**: loose coupling

```
public abstract class DOProcess implements Runnable {

  protected DoRules doRules;

  public abstract void operate();

}
```

Support for Implementation of PDCA based MGT processes (11/12)

- **Software Engineering principles**: high cohesion

```
public abstract class CheckProcess<T> implements Runnable {

  public CheckingRules checkingRules;
  public ObjectiveSetting<T> objectiveSetting;
  public MeasuredPerformanceValue<T> performanceValue;

  public abstract Deviation<T> getCheckResult(ObjectiveSetting<T> objective, Measur

  public void setPerformanceValue(MeasuredPerformanceValue<T> performanceValue) {
    this.performanceValue = performanceValue;
  }
}
```

Support for Implementation of PDCA based MGT processes (12/12)

- **Software Engineering principles**: Design patterns – MVC, Strategy, Facade

```
public class RiskMGTMonteCarloStrategy implements RiskMGTRiskChanging {

  private final static Log log = Logging.getLog(RiskMGTMonteCarloStrategy.class);

  private RandomDataGenerator randomDataGenerator;
  public BigDecimal maxRiskValue;

  public RiskMGTMonteCarloStrategy(BigDecimal maxRiskValue) {
    super();
    this.maxRiskValue = maxRiskValue;
    this.randomDataGenerator = new RandomDataGenerator();
  }

  public BigDecimal getActualRisk() {
    long N = 20;
    double x = 0.0;
    for (int i = 0; i < N; i++) {
      x = randomDataGenerator.nextUniform(0.0, maxRiskValue.doubleValue());
    }
    log.info("Actual risk: "+x);
    return new BigDecimal(x);
  }

}
```

Agenda

- Wiederholung PDCA + Beispiele
- CPPI Investment: Funktionsweise, Design und Implementierung
- PDCA-basierte MGT Infrastruktur
- **Übungsaufgabe (Gruppenarbeit)**

Übungsaufgabe (Gruppenarbeit)

Übungsaufgabe

Übungsaufgabe (Gruppenarbeit)

Organisatorisches

- Übung ist in 3-Gruppen zu lösen!
- Max. 30 Punkte sind zu erreichen
- Plagiate werden mit 0 Punkte bewertet
- Deadline: 22.05.2014
- TUWEL – Gruppanmeldung hat schon gestartet!
- Bis spätestens 9. Mai – 23:55 zu einer Gruppe anmelden!

Übungsaufgabe (Gruppenarbeit)

Aufgabenstellung (1/2)

- Sie haben die Aufgabe sich mit der CPPI-Strategie auseinanderzusetzen und diese zu implementieren. Dafür werden Sie die 3 verschiedenen Sichtweisen (B&M Domain, IS Domain, IT Domain) annehmen.
- Es wird Ihnen ein Code-Konstrukt zur Verfügung gestellt, welches Sie unbedingt verwenden müssen(!).
- Die Aufgabe gliedert sich in 3 Teile.

Übungsaufgabe (Gruppenarbeit)

Aufgabenstellung (2/2)

1) Beschreibung der CPPI-Strategie mit eigenen Worten. Identifizieren und beschreiben Sie dabei die beteiligten Aktivitäten und Datenflüsse. (Nutzer-Perspektive)
2) Ordnen Sie die identifizierten Aktivitäten einer PDCA-Management-Infrastruktur (Aktivitäten und Informationsobjekte) zu. Begründete Entscheidung für eine konkrete Management-Infrastruktur, welche zur Implementierung der CPPI-Strategie verwendet wird. (Designer-Perspektive)
3) Implementieren Sie die CPPI-Strategie. Verwenden Sie dafür die vorgesehenen abstrakten Klassen. Dokumentieren Sie welche Klassen und Methoden konkret welchen CPPI-Aktivitäten entsprechen. Verwenden Sie dafür Code-Snippets für eine detaillierte Beschreibung. (Programmierer-Perspektive)

Übungsaufgabe (Gruppenarbeit)

Abgabe

- Die Abgabe hat in Form einer ZIP-Datei in TUWEL hochgeladen zu werden.
- 5-7 Seiten
- Die ZIP-Datei beinhaltet dabei:
  - Abgabedokument in PDF-Format (Vorlage finden Sie im Java Projekt im Ordner „resources“)
  - Source-Code des Codes
- Es sind keine Third-Party-Libraries (außer die mitgelieferten) erlaubt. Zur Bewertung der Abgabe muss es möglich sein die Sourcefiles in Java 1.7 ohne weiteres zu kompilieren und auszuführen.

Übungsaufgabe (Gruppenarbeit)

Hinweise

- Halten Sie sich an den vorgegebenen Code! Für die Auswertung werden auch automatisierte Tests verwendet. Können diese nicht ausgeführt werden wird der entsprechende Teil mit 0 Punkten bewertet
- Fangen Sie rechtzeitig an!
- Verwenden Sie den Logger!
- Es ist nicht notwendig eine GUI zu erstellen!
- Verwenden Sie für Dezimalzahlen „java.math.BigDecimal“

Übungsaufgabe (Gruppenarbeit)

Fragen?

- TUWEL-Forum
- David Reichl david.reichl@tuwien.ac.at

VIEL ERFOLG!