Test-Driven/Agile Dev. & Projektet AVSATS

Robert Feldt, Chalmers, robert.feldt@chalmers.se

Agenda

- Test-Driven Utveckling: Vad vet vi?
- Agile Utveckling: Vad vet vi?
- Kort om Projekt AVSATS

Experiment på TDD

Table 1 Controlled and Quasi-Controlled Empirical Experiments on TDD

Investigator	A/I	Subjects	Software Quality	Developer Productivity
Janzen and Saiedian (2008)	I	teams of 1-3	TDD had better coverage and smaller modules	N/A
Janzen and Saiedian (2008)	A	1-2 teams of 3	TDD had better coverage, smaller methods and modules, and less complexity	N/A
Madeyski and Szala (2007)	A	1	N/A	TDD had 87-177% better productivity initially
Siniaalto and Abrahamsson (2007)	A	13	TDD improved coverage	N/A
Gupta and Jalote (2007)	A	22	Inconclusive	Improved overall productivity
George and Williams (2004)	I	24	TDD improved test coverage, possibly reduced cohesion	N/A
Geras et al (2004)	I	14	TDD had better quality	No impact
Kaufmann and Janzen (2003)	A	8	N/A	50% improvement
Erdogmus et al (2005)	A	35	No change	Improved productivity
Muller and Hagner (2002)	A	19	Less reliable, but better reuse	No change
Pančur et al (2003)	A	38	No change	No change

Experiment på TDD

Table 1 Controlled and Quasi-Controlled Empirical Experiments on TDD

Investigator	A/I	Subjects	Software Quality	Developer Productivity
Janzen and Saiedian (2008)	I	teams of 1-3	TDD had better coverage and smaller modules	N/A
Janzen and Saiedian (2008)	A	1-2 teams of 3	TDD had better coverage, smaller methods and modules, and less complexity	N/A
Madeyski and Szala (2007)	A	1	N/A	TDD had 87-177% better productivity initially
Siniaalto and Abrahamsson (2007)	A	13	TDD improved coverage	N/A
Gupta and Jalote (2007)	A	22	Inconclusive	Improved overall productivity
George and Williams (2004)	I	24	TDD improved test coverage, possibly reduced cohesion	N/A
Geras et al (2004)	I	14	TDD had better quality	No impact
Kaufmann and Janzen (2003)	A	8	N/A	50% improvement
Erdogmus et al (2005)	A	35	No change	Improved productivity
Muller and Hagner (2002)	A	19	Less reliable, but better reuse	No change
Pančur et al (2003)	A	38	No change	No change

Experiment på TDD

Table 1 Controlled and Quasi-Controlled Empirical Experiments on TDD

Investigator	A/I	Subjects	Software Quality	Developer Productivity
Janzen and Saiedian (2008)	I	teams of 1-3	TDD had better coverage and smaller modules	N/A
Janzen and Saiedian (2008)	A	1-2 teams of 3	TDD had better coverage, smaller methods and modules, and less complexity	N/A
Madeyski and Szala (2007)	A	1	N/A	TDD had 87-177% better productivity initially
Siniaalto and Abrahamsson (2007)	A	13	TDD improved coverage	N/A
Gupta and Jalote (2007)	A	22	Inconclusive	Improved overall productivity
George and Williams (2004)	I	24	TDD improved test coverage, possibly reduced cohesion	N/A
Geras et al (2004)	I	14	TDD had better quality	No impact
Kaufmann and Janzen (2003)	A	8	N/A	50% improvement
Erdogmus et al (2005)	A	35	No change	Improved productivity
Muller and Hagner (2002)	A	19	Less reliable, but better reuse	No change
Pančur et al (2003)	A	38	No change	No change

Empiriska studier på TDD

Table 2 Empirical Case Studies on TDD

Investigator	A/I	Subjects	Software Quality	Developer Productivity
Janzen and Saiedian (2008)	I	team of 3	TDD had better coverage and smaller methods and modules	N/A
Sanchez et al (2007)	I	9-17	30% reduction in defect density	Increased effort 19%
Damm and Lundberg (2006)	I	100	5-30% reduction in fault slip-through, 55% reduction in fault costs	Project cost increased by 5-6%
Maximilien and Williams (2003)	I	9	50% reduction in defect density	Minimal impact
Williams et al (2003)	I	9	40% reduction in defect density	No change
Bhat and Nagappan (2006)	A	11	2-4 times reduction in defect density	35% and 15% more time
Edwards (2004)	A	59	54% fewer defects	N/A

Empiriska studier på TDD

Table 2 Empirical Case Studies on TDD

Investigator	A/I	Subjects	Software Quality	Developer Productivity
Janzen and Saiedian (2008)	I	team of 3	TDD had better coverage and smaller methods and modules	N/A
Sanchez et al (2007)	I	9-17	30% reduction in defect density	Increased effort 19%
Damm and Lundberg (2006)	I	100	5-30% reduction in fault slip-through, 55% reduction in fault costs	Project cost increased by 5-6%
Maximilien and Williams (2003)	I	9	50% reduction in defect density	Minimal impact
Williams et al (2003)	I	9	40% reduction in defect density	No change
Bhat and Nagappan (2006)	A	11	2-4 times reduction in defect density	35% and 15% more time
Edwards (2004)	A	59	54% fewer defects	N/A

Empiriska studier på TDD

Table 2 Empirical Case Studies on TDD

Investigator	A/I	Subjects	Software Quality	Developer Productivity
Janzen and Saiedian (2008)	I	team of 3	TDD had better coverage and smaller methods and modules	N/A
Sanchez et al (2007)	I	9-17	30% reduction in defect density	Increased effort 19%
Damm and Lundberg (2006)	I	100	5-30% reduction in fault slip-through, 55% reduction in fault costs	Project cost increased by 5-6%
Maximilien and Williams (2003)	I	9	50% reduction in defect density	Minimal impact
Williams et al (2003)	I	9	40% reduction in defect density	No change
Bhat and Nagappan (2006)	A	11	2-4 times reduction in defect density	35% and 15% more time
Edwards (2004)	A	59	54% fewer defects	N/A

Sammantaget: TDD

- Positivt:
  - 2-4 gånger färre defekter i 2 Microsoft projekt
  - 50% förbättrad kvalitet i TDD transition hos ett IBM team
  - Senaste studien: 40-90% färre pre-release defects
- Negativt:
  - TDD tar något mer tid (15-35%), ibland ingen ändrad kvalitet
  - I studentexperiment har det ibland gett sämre kvalitet
- Skillnader kan bero på att TDD är olika saker
  - Forskningen går nu mer in på att kvantifiera grad av TDD

Effectiveness of Unit Test Automation

- Team at Microsoft, 32 persons

Effectiveness of Unit Test Automation

- Team at Microsoft, 32 persons

Ad hoc & Individual Unit Testing

From

Effectiveness of Unit Test Automation

- Team at Microsoft, 32 persons

Ad hoc & Individual Unit Testing

From

Effectiveness of Unit Test Automation

- Team at Microsoft, 32 persons

Ad hoc & Individual Unit Testing

NUnit Autotesting by everyone

From **To**

Effectiveness of Unit Test Automation

- Team at Microsoft, 32 persons

Tests written every 2-3 days, **after** coding

Ad hoc & Individual Unit Testing

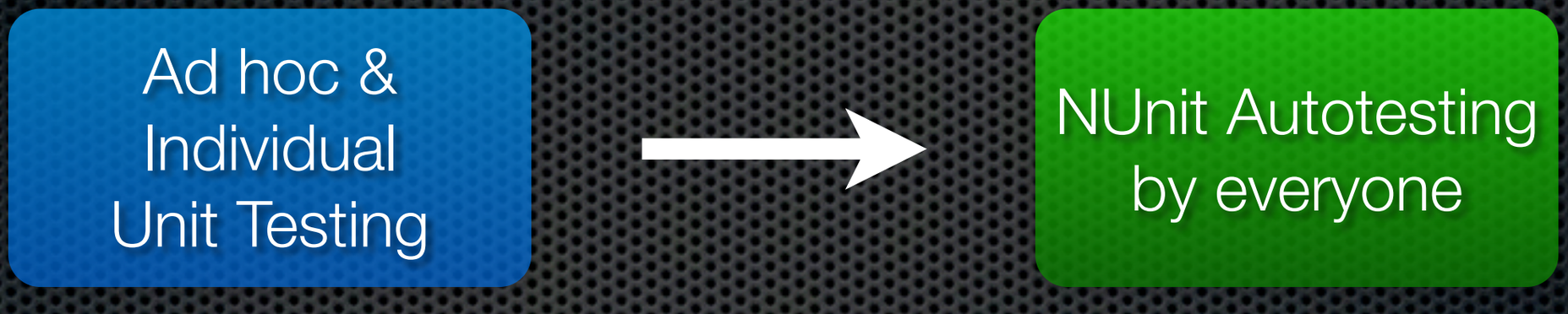
NUnit Autotesting by everyone

From **To**

Effectiveness of Unit Test Automation

- Team at Microsoft, 32 persons

Tests written every 2-3 days, **after** coding



From **To**

- Results (version 2 of product compared to version 1):
  - 21% fewer defects
  - 30% added development time
- “Would be even better if tests written iteratively”

Traditionell vs Agil Utveckling

Table 2
Main differences between traditional development and agile development [47]

	Traditional development	Agile development
Fundamental assumption	Systems are fully specifiable, predictable, and are built through meticulous and extensive planning	High-quality adaptive software is developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change
Management style	Command and control	Leadership and collaboration
Knowledge management	Explicit	Tacit
Communication	Formal	Informal
Development model	Life-cycle model (waterfall, spiral or some variation)	The evolutionary-delivery model
Desired organizational form/structure	Mechanistic (bureaucratic with high formalization), aimed at large organizations	Organic (flexible and participative encouraging cooperative social action), aimed at small and medium-sized organizations
Quality control	Heavy planning and strict control. Late, heavy testing	Continuous control of requirements, design and solutions. Continuous testing

Traditionell vs Agil Utveckling

Table 2
Main differences between traditional development and agile development [47]

	Traditional development	Agile development
Fundamental assumption	Systems are fully specifiable, predictable, and are built through meticulous and extensive planning	High-quality adaptive software is developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change
Management style	Command and control	Leadership and collaboration
Knowledge management	Explicit	Tacit
Communication	Formal	Informal
Development model	Life-cycle model (waterfall, spiral or some variation)	The evolutionary-delivery model
Desired organizational form/structure	Mechanistic (bureaucratic with high formalization), aimed at large organizations	Organic (flexible and participative encouraging cooperative social action), aimed at small and medium-sized organizations
Quality control	Heavy planning and strict control. Late, heavy testing	Continuous control of requirements, design and solutions. Continuous testing

Stelt!

Oflexibelt!

Tråkigt!

Traditionell vs Agil Utveckling

Table 2
Main differences between traditional development and agile development [47]

	Traditional development	Agile development
Fundamental assumption	Systems are fully specifiable, predictable, and are built through meticulous and extensive planning	High-quality adaptive software is developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change
Management style	Command and control	Leadership and collaboration
Knowledge management	Explicit	Tacit
Communication	Formal	Informal
Development model	Life-cycle model (waterfall, spiral or some variation)	The evolutionary-delivery model
Desired organizational form/structure	Mechanistic (bureaucratic with high formalization), aimed at large organizations	Organic (flexible and participative encouraging cooperative social action), aimed at small and medium-sized organizations
Quality control	Heavy planning and strict control. Late, heavy testing	Continuous control of requirements, design and solutions. Continuous testing

Stelt!

Oflexibelt!

Tråkigt!

Ej stora projekt!

Inget nytt!

Ingen använder!

Används Agile?

- 2005 sade (av ett stort antal USA+EU företag):
  - 14% att de använde agila utvecklingsmetoder
  - 49% att de ville använda dem
- Agile conference har fler besökare än de flesta traditionella programvarukonferenser
  - Viss avmattning 2008-2009 (downturn?)

Hur påverkas produktiviteten?

Table 14
Comparisons of productivity

Study	$\text{Productivity}_{\text{TRAD}}$	$\text{Productivity}_{\text{AGILE}}$	Productivity gain (%)
S7	3 LOC/h[a]	13.1 LOC/h	337
S10	3.8 LOC/h	5.4 LOC/h	42
S14	300 LOC/month	440 LOC/month	46
S32	157 LOC/ engineer[b]	88 LOC/engineer	−44

[a] V-model.

[b] Comparisons were made between two one-semester courses; however, the actual hours worked by the members of the teams were not measured.

Hur påverkas produktiviteten?

Table 14
Comparisons of productivity

Study	Productivity TRAD	Productivity AGILE	Productivity gain (%)
S7	3 LOC/h[a]	13.1 LOC/h	337
S10	3.8 LOC/h	5.4 LOC/h	42
S14	300 LOC/month	440 LOC/month	46
S32	157 LOC/ engineer[b]	88 LOC/engineer	−44

Trad.

Agile

[a] V-model.

[b] Comparisons were made between two one-semester courses; however, the actual hours worked by the members of the teams were not measured.

Hur påverkas produktiviteten?

Table 14
Comparisons of productivity

Study	Productivity TRAD	Productivity AGILE	Productivity gain (%)
S7	3 LOC/h[a]	13.1 LOC/h	
S10	3.8 LOC/h	5.4 LOC/h	42
S14	300 LOC/month	440 LOC/month	46
S32	157 LOC/ engineer[b]	88 LOC/engineer	−44

Trad.

Agile

More code but same func!

[a] V-model.

[b] Comparisons were made between two one-semester courses; however, the actual hours worked by the members of the teams were not measured.

Sammantaget: Agile

- Positivt:
  - Utvecklarna mer nöjda med sitt jobb
  - Ökad produktivitet kan uppnås
  - Kunderna kan bli mer nöjda med produkterna
- Negativt:
  - “On-site customer” ej praktiskt
  - Svårt att introducera i komplexa projekt
- Finns dock inte så mycket forskning och mest på XP

“Agile” Kravhantering i praktiken

- Intervjuer med 54 personer i 16 företag
  - Använde XP eller SCRUM, helt eller delvis
- Frågor:
  - Hur jobbar “agile” utvecklare med kravhantering?
  - Vilka fördelar och nackdelar ger detta?

Vad använder de?

Agile requirements-engineering practices in 16 organizations

	Practice						
Adoption level	**Face-to-face communication**	**Iterative RE**	**Extreme prioritization**	**Constant planning**	**Prototyping**	**Test-driven development**	**Reviews & tests**
High	8	9	10	8	8	5	11
Medium	8	5	6	6	3	1	4
Low	0	2	0	2	0	0	1
None	0	0	0	0	5	10	0

Fördelar/Nackdelar

Prototyper

Test-driven Utveckling

Granskningar & Acceptanstester

Fördelar/Nackdelar

Prototyper

Snabbare feedback

Test-driven Utveckling

Granskningar & Acceptanstester

Fördelar/Nackdelar

Prototyper

Snabbare feedback

Orealistisk förväntan på utv.tid

Test-driven Utveckling

Granskningar & Acceptanstester

Fördelar/Nackdelar

Prototyper

Snabbare feedback

Orealistisk förväntan på utv.tid

Test-driven Utveckling

Tester fångar krav

Granskningar & Acceptanstester

Fördelar/Nackdelar

Prototyper

- Snabbare feedback
- Orealistisk förväntan på utv.tid

Test-driven Utveckling

- Tester fångar krav
- Frihet att experimentera

Granskningar & Acceptanstester

Fördelar/Nackdelar

Prototyper

Snabbare feedback

Orealistisk förväntan på utv.tid

Test-driven Utveckling

Tester fångar krav

Kräver nära kundkontakt

Frihet att experimentera

Granskningar & Acceptanstester

Fördelar/Nackdelar

Prototyper

Snabbare feedback

Orealistisk förväntan på utv.tid

Test-driven Utveckling

Tester fångar krav

Kräver nära kundkontakt

Frihet att experimentera

Utvecklarna spjärnar emot

Granskningar & Acceptanstester

Fördelar/Nackdelar

Prototyper

Snabbare feedback

Orealistisk förväntan på utv.tid

Test-driven Utveckling

Tester fångar krav

Kräver nära kundkontakt

Frihet att experimentera

Utvecklarna spjärnar emot

Granskningar & Acceptanstester

Statusrapport till kunder

Fördelar/Nackdelar

Prototyper

- Snabbare feedback
- Orealistisk förväntan på utv.tid

Test-driven Utveckling

- Tester fångar krav
- Kräver nära kundkontakt
- Frihet att experimentera
- Utvecklarna spjärnar emot

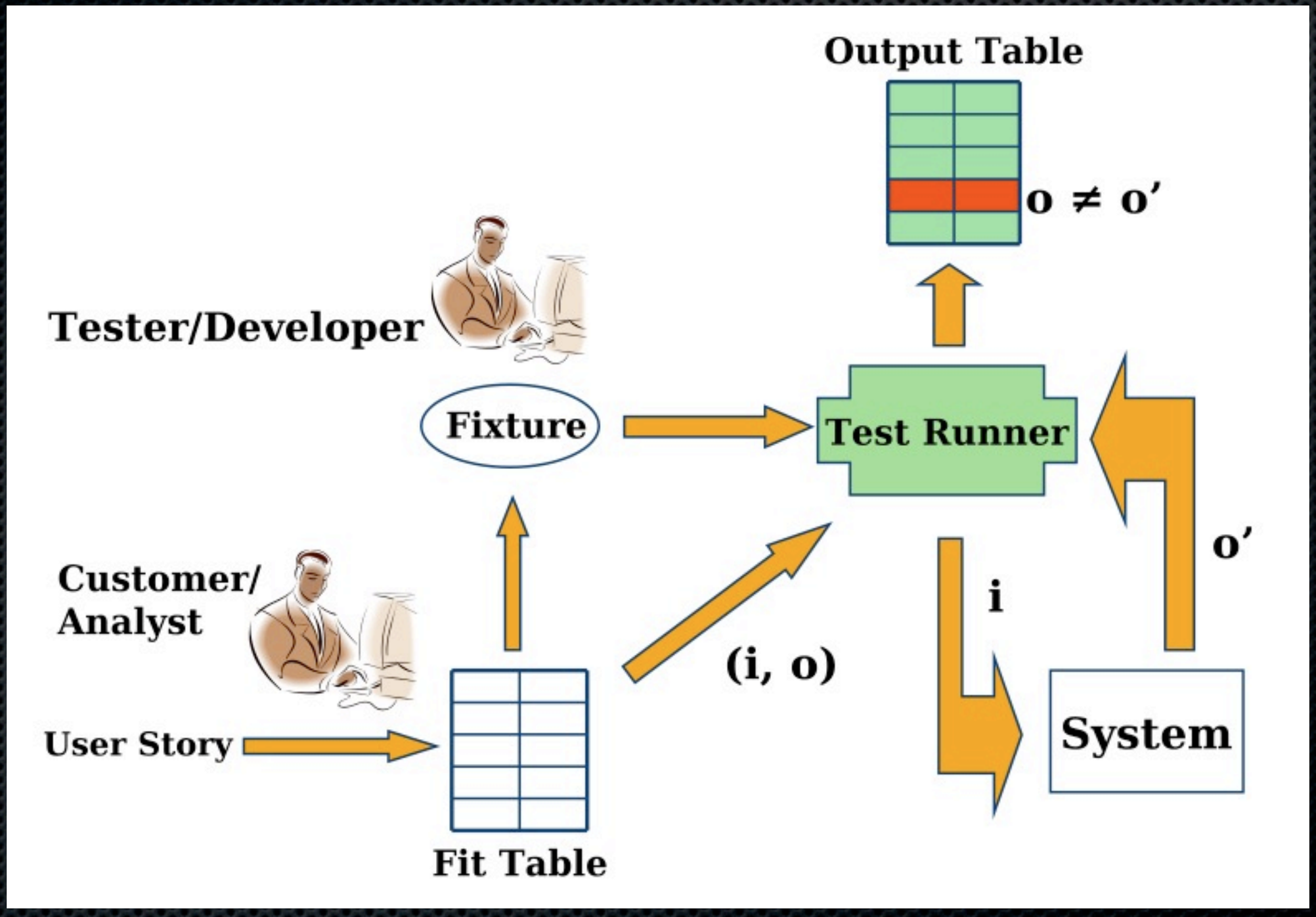
Granskningar & Acceptanstester

- Statusrapport till kunder
- Svårt ta fram acceptanstester

Acceptanstester förtydligar krav

- Studie på två italienska universitet
- Mål: Utvärdera hur FIT tabeller påverkar förståelse av krav
- Jämför:
  - Grupp 1: Traditionell krav i text
  - Grupp 2: Krav i text + FIT tabeller

Acceptanstestning med FIT



Acceptanstester förtydligar krav

- Resultat:
  - FIT tabeller gav 400% större chans att besvara frågor om kraven korrekt
  - Det tog lika lång tid för båda grupperna
- Men:
  - FIT tabellerna passar inte alla krav

Acceptanstester förtydligar krav

- Resultat:
  - FIT tabeller gav 400% större chans att besvara frågor om kraven korrekt
  - Det tog lika lång tid för båda grupperna
- Men:
  - FIT tabellerna passar inte alla krav

Exempel på konkreta indata hjälper kravförståelse

Acceptanstester förtydligar krav

- Resultat:
  - FIT tabeller gav 400% större chans att besvara frågor om kraven korrekt
  - Det tog lika lång tid för båda grupperna
- Men:
  - FIT tabellerna passar inte alla krav

Exempel på konkreta indata hjälper kravförståelse

Information av flera typer ger bättre förståelse