**Słowo wstępne (17)**
**Przedmowa (19)**
**Wprowadzenie (33)**
**I: Narzędzia kompilujące (37)**
**1. Przygotowywanie projektu z wykorzystaniem Anta (41)**

- 1.1. Rola narzędzia Ant w procesie kompilacji (41)
- 1.2. Instalacja Anta (41)
- 1.3. Płynne wprowadzenie w świat Anta (44)
- 1.4. Kompilowanie kodu Javy za pomocą Anta (51)
- 1.5. Dostosowywanie skryptów kompilacji za pomocą właściwości (53)
- 1.6. Przeprowadzanie testów jednostkowych za pomocą Anta (57)
- 1.7. Generowanie dokumentacji za pomocą narzędzia Javadoc (75)
- 1.8. Pakowanie gotowej aplikacji (77)
- 1.9. Wdrażanie aplikacji (81)
- 1.10. Automatyczne przygotowywanie środowiska dla uruchamianych skryptów kompilacji (83)
- 1.11. Stosowanie zależności narzędzia Maven w Ancie wraz z zadaniami Mavena (85)
- 1.12. Stosowanie Anta w środowisku Eclipse (89)
- 1.13. Stosowanie Anta w środowisku NetBeans (89)
- 1.14. Modyfikowanie kodu XML-a za pomocą zadania XMLTask (90)
- 1.15. Konkluzja (95)

**2. Przygotowywanie projektu z wykorzystaniem Mavena 2 (97)**

- 2.1. Rola narzędzia Maven w procesie kompilacji (97)
- 2.2. Maven i Ant (98)
- 2.3. Instalacja Mavena (99)
- 2.4. Kompilacje deklaratywne i model obiektu projektu Mavena (101)
- 2.5. Zrozumieć cykl życia Mavena 2 (112)
- 2.6. Struktura katalogów Mavena (114)
- 2.7. Konfigurowanie Mavena pod kątem naszego środowiska (115)
- 2.8. Zarządzanie zależnościami w Mavenie 2 (118)
- 2.9. Poszukiwanie zależności za pośrednictwem witryny Maven Repository (126)
- 2.10. Dziedziczenie i agregacja projektów (127)
- 2.11. Tworzenie szablonu projektu za pomocą tzw. archetypów (131)
- 2.12. Kompilacja kodu (135)
- 2.13. Testowanie kodu (136)
- 2.14. Pakowanie i wdrażanie naszej aplikacji (138)
- 2.15. Wdrażanie aplikacji z wykorzystaniem narzędzia Cargo (140)
- 2.16. Stosowanie Mavena w środowisku Eclipse (144)
- 2.17. Stosowanie Mavena w środowisku NetBeans (147)
- 2.18. Dostosowywanie procesu kompilacji do specyficznych potrzeb projektu za pomocą własnych modułów rozszerzeń (147)
- 2.19. Konfigurowanie repozytorium korporacyjnego za pomocą narzędzia Archiva (154)
- 2.20. Konfigurowanie repozytorium korporacyjnego z wykorzystaniem narzędzia Artifactory (166)
- 2.21. Stosowanie narzędzia Ant w Mavenie (178)
- 2.22. Archetypy zaawansowane (183)

- 2.23. Stosowanie podzespołów (187)

**II: Narzędzia kontroli wersji (193)**
**3. Kontrola wersji z wykorzystaniem systemu CVS (195)**

- 3.1. Wprowadzenie do systemu CVS (195)
- 3.2. Konfigurowanie repozytorium systemu CVS (196)
- 3.3. Tworzenie nowego projektu w systemie CVS (196)
- 3.4. Wypożyczanie projektu (198)
- 3.5. Praca na plikach - aktualizowanie i zatwierdzanie plików z kodem źródłowym (200)
- 3.6. Blokowanie repozytorium (204)
- 3.7. Praca z mechanizmem zastępowania słów kluczowych (204)
- 3.8. Praca z plikami binarnymi (205)
- 3.9. Znaczniki systemu CVS (207)
- 3.10. Tworzenie odgałęzień w systemie CVS (208)
- 3.11. Scalanie zmian z odgałęzienia (210)
- 3.12. Przeglądanie historii zmian (211)
- 3.13. Wycofywanie zmian (213)
- 3.14. Stosowanie CVS-a w systemie Windows (214)

**4. Kontrola wersji z wykorzystaniem systemu Subversion (217)**

- 4.1. Wprowadzenie do systemu Subversion (217)
- 4.2. Instalacja systemu Subversion (221)
- 4.3. Typy repozytoriów systemu Subversion (221)
- 4.4. Konfigurowanie repozytorium systemu Subversion (223)
- 4.5. Tworzenie nowego projektu w systemie Subversion (225)
- 4.6. Wypożyczanie kopii roboczej (227)
- 4.7. Importowanie istniejących plików do repozytorium systemu Subversion (228)
- 4.8. Zrozumieć adresy URL repozytorium systemu Subversion (230)
- 4.9. Praca z plikami (231)
- 4.10. Sprawdzanie bieżącej sytuacji - polecenie status (235)
- 4.11. Rozwiązywanie konfliktów (237)
- 4.12. Stosowanie znaczników, odgałęzień i operacji scalania (239)
- 4.13. Przywracanie poprzedniej rewizji (243)
- 4.14. Blokowanie dostępu do plików binarnych (244)
- 4.15. Zdejmowanie i przechwytywanie blokad (246)
- 4.16. Udostępnianie zablokowanych plików tylko do odczytu za pomocą właściwości svn:needs-lock (248)
- 4.17. Stosowanie właściwości (249)
- 4.18. Historia zmian w systemie Subversion - rejestrowanie zdarzeń i określanie odpowiedzialności za zmiany (252)
- 4.19. Konfigurowanie serwera systemu Subversion z wykorzystaniem serwera svnserve (253)
- 4.20. Konfigurowanie bezpiecznego serwera svnserve (257)
- 4.21. Konfigurowanie serwera Subversion z obsługą protokołu WebDAV/DeltaV (258)
- 4.22. Konfigurowanie bezpiecznego serwera WebDAV/DeltaV (263)

- 4.23. Dostosowywanie działania systemu Subversion za pomocą skryptów przechwytujących (264)
- 4.24. Instalacja systemu Subversion w formie usługi systemu operacyjnego Windows (266)
- 4.25. Sporządzanie kopii zapasowej i przywracanie repozytorium systemu Subversion (268)
- 4.26. Stosowanie systemu Subversion w środowisku Eclipse (268)
- 4.27. Stosowanie systemu Subversion w środowisku NetBeans (275)
- 4.28. Stosowanie systemu Subversion w systemie operacyjnym Windows (281)
- 4.29. Śledzenie usterek i kontrola zmian (287)
- 4.30. Stosowanie systemu Subversion w Ancie (290)
- 4.31. Konkluzja (292)

**III: Ciągła integracja (293)**

**5. Konfigurowanie serwera ciągłej integracji za pomocą narzędzia Continuum (297)**

- 5.1. Wprowadzenie do narzędzia Continuum (297)
- 5.2. Instalacja serwera narzędzia Continuum (297)
- 5.3. Ręczne uruchamianie i zatrzymywanie serwera (301)
- 5.4. Sprawdzanie stanu serwera (302)
- 5.5. Uruchamianie serwera narzędzia Continuum w trybie ze szczegółowymi komunikatami (302)
- 5.6. Dodawanie grupy projektów (303)
- 5.7. Dodawanie projektu Mavena (303)
- 5.8. Dodawanie projektu Anta (306)
- 5.9. Dodawanie projektu kompilowanego za pomocą skryptu powłoki (307)
- 5.10. Zarządzanie kompilacjami projektu (307)
- 5.11. Zarządzanie użytkownikami (309)
- 5.12. Konfigurowanie mechanizmów powiadomień (311)
- 5.13. Konfigurowanie planowanych kompilacji (311)
- 5.14. Diagnozowanie procesu kompilacji (314)
- 5.15. Konfigurowanie serwera poczty elektronicznej narzędzia Continuum (314)
- 5.16. Konfigurowanie portów witryny internetowej serwera Continuum (315)
- 5.17. Automatyczne generowanie witryny Mavena za pomocą narzędzia Continuum (316)
- 5.18. Konfigurowanie zadania ręcznej kompilacji (317)
- 5.19. Konkluzja (319)

**6. Konfigurowanie serwera ciągłej integracji za pomocą narzędzia CruiseControl (321)**

- 6.1. Wprowadzenie do narzędzia CruiseControl (321)
- 6.2. Instalacja narzędzia CruiseControl (322)
- 6.3. Konfigurowanie projektu Anta (323)
- 6.4. Powiadamianie członków zespołu za pomocą mechanizmów publikujących (329)
- 6.5. Konfigurowanie projektu Mavena 2 w narzędziu CruiseControl (336)
- 6.6. Panel administracyjny narzędzia CruiseControl (338)
- 6.7. Dodatkowe narzędzia (339)
- 6.8. Konkluzja (340)

**7. LuntBuild - serwer ciągłej integracji z interfejsem WWW (341)**

- 7.1. Wprowadzenie do narzędzia LuntBuild (341)
- 7.2. Instalowanie narzędzia LuntBuild (341)
- 7.3. Konfigurowanie serwera LuntBuild (343)
- 7.4. Dodawanie projektu (345)
- 7.5. Wykorzystywanie zmiennych projektowych do numerowania wersji (352)
- 7.6. Diagnostyka wyników kompilacji (353)
- 7.7. Stosowanie narzędzia LuntBuild w środowisku Eclipse (355)
- 7.8. Raportowanie w systemie LuntBuild o pokryciu testami z wykorzystaniem narzędzia Cobertura (359)
- 7.9. Integrowanie narzędzia LuntBuild z Mavenem (365)
- 7.10. Konkluzja (370)

**8. Ciągła integracja z wykorzystaniem narzędzia Hudson (371)**

- 8.1. Wprowadzenie do narzędzia Hudson (371)
- 8.2. Instalacja narzędzia Hudson (371)
- 8.3. Zarządzanie katalogiem domowym Hudsona (372)
- 8.4. Instalacja aktualizacji (373)
- 8.5. Konfigurowanie Hudsona (374)
- 8.6. Dodawanie nowego zadania kompilacji (376)
- 8.7. Organizowanie zadań (381)
- 8.8. Monitorowanie kompilacji (382)
- 8.9. Przeglądanie i awansowanie wybranych kompilacji (383)
- 8.10. Zarządzanie użytkownikami (385)
- 8.11. Uwierzytelnianie i bezpieczeństwo (386)
- 8.12. Przeglądanie zmian (386)
- 8.13. Moduły rozszerzeń Hudsona (387)
- 8.14. Śledzenie wyników testów (388)
- 8.15. Śledzenie mierników kodu źródłowego (388)
- 8.16. Raportowanie o pokryciu kodu (390)

**9. Konfigurowanie platformy natychmiastowej komunikacji za pomocą serwera Openfire (393)**

- 9.1. Natychmiastowa komunikacja w projekcie informatycznym (393)
- 9.2. Instalacja serwera Openfire (394)
- 9.3. Konfigurowanie użytkowników i kont użytkowników serwera Openfire (394)
- 9.4. Uwierzytelnianie użytkowników z wykorzystaniem zewnętrznej bazy danych (396)
- 9.5. Uwierzytelnianie użytkowników na serwerze POP3 (397)
- 9.6. Organizowanie wirtualnych spotkań zespołu z wykorzystaniem czatu grupowego (398)
- 9.7. Rozszerzanie funkcjonalności serwera Openfire za pomocą modułów rozszerzeń (400)
- 9.8. Stosowanie serwera Openfire z systemem Continuum (400)
- 9.9. Stosowanie serwera Openfire z systemem CruiseControl (401)
- 9.10. Stosowanie serwera Openfire z narzędziem LuntBuild (402)
- 9.11. Wysyłanie komunikatów Jabbera z poziomu aplikacji Javy za pośrednictwem interfejsu API Smack (402)
- 9.12. Wykrywanie obecności interfejsu API Smack (405)

- 9.13. Otrzymywanie wiadomości z wykorzystaniem interfejsu API Smack (405)

**IV: Testy jednostkowe (407)**
**10. Testowanie kodu z wykorzystaniem frameworku JUnit (409)**

- 10.1. Frameworki JUnit 3.8 i JUnit 4 (409)
- 10.2. Testowanie jednostkowe z wykorzystaniem frameworku JUnit 4 (410)
- 10.3. Konfigurowanie i optymalizacja przypadków testów jednostkowych (412)
- 10.4. Proste testy wydajności z wykorzystaniem limitów czasowych (414)
- 10.5. Prosta weryfikacja występowania wyjątków (415)
- 10.6. Stosowanie testów sparametryzowanych (415)
- 10.7. Stosowanie metody assertThat() i biblioteki Hamcrest (418)
- 10.8. Teorie we frameworku JUnit 4 (421)
- 10.9. Stosowanie frameworku JUnit 4 w projektach Mavena 2 (423)
- 10.10. Stosowanie frameworku JUnit 4 w projektach Anta (423)
- 10.11. Selektywne wykonywanie testów frameworku JUnit 4 w Ancie (426)
- 10.12. Testy integracyjne (428)
- 10.13. Korzystanie z frameworku JUnit 4 w środowisku Eclipse (429)

**11. Testowanie nowej generacji z wykorzystaniem frameworku TestNG (433)**

- 11.1. Wprowadzenie do frameworku TestNG (433)
- 11.2. Tworzenie prostych testów jednostkowych za pomocą frameworku TestNG (433)
- 11.3. Definiowanie pakietów testów frameworku TestNG (435)
- 11.4. Moduł rozszerzenia frameworku TestNG dla środowiska Eclipse (437)
- 11.5. Stosowanie frameworku TestNG w Ancie (440)
- 11.6. Korzystanie z frameworku TestNG w Mavenie 2 (443)
- 11.7. Zarządzanie cyklem życia testów (444)
- 11.8. Stosowanie grup testów (449)
- 11.9. Zarządzanie zależnościami (451)
- 11.10. Testowanie równoległe (454)
- 11.11. Parametry testów i testowanie sterowane danymi (455)
- 11.12. Weryfikacja wyjątków (456)
- 11.13. Obsługa błędów częściowych (456)
- 11.14. Ponowne wykonywanie testów zakończonych niepowodzeniem (457)

**12. Maksymalizacja pokrycia testami za pomocą narzędzia Cobertura (459)**

- 12.1. Pokrycie testami (459)
- 12.2. Uruchamianie narzędzia Cobertura za pośrednictwem Anta (460)
- 12.3. Weryfikacja pokrycia kodu testami frameworku TestNG (463)
- 12.4. Interpretacja raportu narzędzia Cobertura (465)
- 12.5. Wymuszanie dużego pokrycia kodu (467)
- 12.6. Generowanie raportów narzędzia Cobertura w Mavenie (469)
- 12.7. Integracja testów pokrycia kodu z procesem kompilacji Mavena (471)
- 12.8. Badanie pokrycia kodu w środowisku Eclipse (473)
- 12.9. Konkluzja (475)

**V: Testy integracyjne, funkcjonalne, obciążeniowe i wydajnościowe (477)**

**13. Testowanie aplikacji frameworku Struts z wykorzystaniem frameworku StrutsTestCase (481)**


- 13.1. Wprowadzenie (481)
- 13.2. Testowanie aplikacji frameworku Struts (482)
- 13.3. Wprowadzenie do frameworku StrutsTestCase (483)
- 13.4. Testy obiektów zastępczych z wykorzystaniem frameworku StrutsTestCase (483)
- 13.5. Testowanie mechanizmów obsługi błędów w aplikacji frameworku Struts (488)
- 13.6. Dostosowywanie środowiska testowego (489)
- 13.7. Testy wydajnościowe pierwszego stopnia (489)
- 13.8. Konkluzja (490)


**14. Testy integracyjne baz danych z wykorzystaniem frameworku DbUnit (491)**


- 14.1. Wprowadzenie (491)
- 14.2. Przegląd (491)
- 14.3. Struktura frameworku DbUnit (493)
- 14.4. Przykładowa aplikacja (497)
- 14.5. Wypełnianie bazy danych (498)
- 14.6. Weryfikacja bazy danych (506)
- 14.7. Zastępowanie wartości (510)
- 14.8. Alternatywne formaty zbiorów danych (516)
- 14.9. Obsługa niestandardowych testów danych (520)
- 14.10. Pozostałe zastosowania (524)


**15. Testy wydajnościowe z wykorzystaniem frameworku JUnitPerf (533)**


- 15.1. Wprowadzenie do frameworku JUnitPerf (533)
- 15.2. Badanie wydajności za pomocą klasy TimedTest (534)
- 15.3. Symulowanie obciążenia za pomocą klasy LoadTest (536)
- 15.4. Przeprowadzanie testów wydajnościowych, które nie gwarantują bezpieczeństwa przetwarzania wielowątkowego (539)
- 15.5. Oddzielanie testów wydajnościowych od testów jednostkowych w Ancie (540)
- 15.6. Oddzielanie testów wydajnościowych od testów jednostkowych w Mavenie (541)


**16. Wykonywanie testów obciążeniowych i wydajnościowych za pomocą narzędzia JMeter (543)**


- 16.1. Wprowadzenie (543)
- 16.2. Instalacja narzędzia JMeter (544)
- 16.3. Testowanie prostej aplikacji internetowej (544)
- 16.4. Projektowanie struktury naszego przypadku testowego (550)
- 16.5. Rejestrowanie i wyświetlanie wyników testu (553)
- 16.6. Rejestrowanie przypadku testowego za pomocą serwera proxy narzędzia JMeter (556)
- 16.7. Testowanie z wykorzystaniem zmiennych (558)
- 16.8. Testowanie na wielu komputerach (560)

**17. Testowanie usług sieciowych za pomocą narzędzia SoapUI (563)**


- 17.1. Wprowadzenie (563)
- 17.2. Wprowadzenie do narzędzia SoapUI (563)
- 17.3. Instalacja narzędzia SoapUI (565)
- 17.4. Instalacja lokalnej usługi sieciowej (565)
- 17.5. Testowanie usług sieciowych za pomocą narzędzia SoapUI (567)
- 17.6. Przeprowadzanie testów obciążeniowych za pomocą narzędzia SoapUI (573)
- 17.7. Uruchamianie narzędzia SoapUI z poziomu wiersza poleceń (576)
- 17.8. Uruchamianie narzędzia SoapUI za pośrednictwem Anta (578)
- 17.9. Uruchamianie narzędzia SoapUI za pośrednictwem Mavena (579)
- 17.10. Testy ciągłe (580)
- 17.11. Konkluzja (581)


**18. Profilowanie i monitorowanie aplikacji Javy za pomocą narzędzi pakietu Sun JDK (583)**


- 18.1. Narzędzia profilujące i monitorujące pakietu Sun JDK (583)
- 18.2. Nawiązywanie połączenia z aplikacją Javy i monitorowanie jej działania za pomocą narzędzia JConsole (583)
- 18.3. Monitorowanie zdalnej aplikacji na serwerze Tomcat za pomocą narzędzia JConsole (587)
- 18.4. Wykrywanie i identyfikacja wycieków pamięci za pomocą narzędzi pakietu JDK (588)
- 18.5. Diagnozowanie wycieków pamięci z wykorzystaniem zrzutów sterty oraz narzędzi jmap i jhat (593)
- 18.6. Wykrywanie zakleszczeń (595)


**19. Profilowanie aplikacji Javy w środowisku Eclipse (599)**


- 19.1. Profilowanie aplikacji z poziomu środowiska IDE (599)
- 19.2. Platforma TPTP środowiska Eclipse (599)
- 19.3. Instalacja platformy TPTP (601)
- 19.4. Platformy TPTP i Java 6 (601)
- 19.5. Podstawowe techniki profilowania z wykorzystaniem platformy TPTP (602)
- 19.6. Ocena użycia pamięci na podstawie wyników podstawowej analizy pamięci (607)
- 19.7. Analiza czasu wykonywania (609)
- 19.8. Wyświetlanie statystyk pokrycia (610)
- 19.9. Stosowanie filtrów zawężających uzyskiwane wyniki (611)
- 19.10. Profilowanie aplikacji internetowej (613)
- 19.11. Konkluzja (613)


**20. Testowanie interfejsów użytkownika (615)**


- 20.1. Wprowadzenie (615)
- 20.2. Testowanie aplikacji internetowej za pomocą narzędzia Selenium (615)
- 20.3. Testowanie graficznych interfejsów Swinga za pomocą narzędzia FEST (642)
- 20.4. Konkluzja (651)

**VI: Narzędzia pomiaru jakości (653)**

**21. Wykrywanie i wymuszanie standardów kodowania za pomocą narzędzia Checkstyle (657)**


- 21.1. Wymuszanie standardów kodowania za pomocą narzędzia Checkstyle (657)
- 21.2. Stosowanie narzędzia Checkstyle w środowisku Eclipse (659)
- 21.3. Modyfikowanie reguł narzędzia Checkstyle w środowisku Eclipse (663)
- 21.4. Dostosowywanie reguł narzędzia Checkstyle z wykorzystaniem plików konfiguracyjnych w formacie XML (665)
- 21.5. Dostosowywanie pracy narzędzia Checkstyle - reguły, bez których możemy sobie poradzić, i kilka reguł, z których warto korzystać (667)
- 21.6. Stosowanie narzędzia Checkstyle do definiowania reguł dla nagłówków w kodzie źródłowym (671)
- 21.7. Wstrzymywanie testów narzędzia Checkstyle (672)
- 21.8. Korzystanie z narzędzia Checkstyle w Ancie (673)
- 21.9. Korzystanie z narzędzia Checkstyle w Mavenie (674)


**22. Wstępne wykrywanie błędów za pomocą narzędzia PMD (677)**


- 22.1. Narzędzie PMD i statyczna analiza kodu (677)
- 22.2. Korzystanie z narzędzia PMD w środowisku Eclipse (677)
- 22.3. Konfiguracja reguł narzędzia PMD w środowisku Eclipse (680)
- 22.4. Więcej o zbiorach reguł narzędzia PMD (681)
- 22.5. Pisanie własnych zbiorów reguł narzędzia (684)
- 22.6. Generowanie raportu narzędzia PMD w środowisku Eclipse (685)
- 22.7. Wstrzymywanie reguł narzędzia PMD (686)
- 22.8. Wykrywanie praktyki "wytnij i wklej" za pomocą narzędzia CPD (687)
- 22.9. Stosowanie narzędzia PMD w Ancie (688)
- 22.10. Stosowanie narzędzia PMD w Mavenie (691)


**23. Wstępne wykrywanie błędów za pomocą narzędzia FindBugs (693)**


- 23.1. FindBugs jako wyspecjalizowany zabójca błędów (693)
- 23.2. Stosowanie narzędzia FindBugs w środowisku Eclipse (695)
- 23.3. Wybiórcze zawieszanie stosowania reguł za pomocą filtrów narzędzia FindBugs (697)
- 23.4. Stosowanie adnotacji narzędzia FindBugs (698)
- 23.5. Korzystanie z narzędzia FindBugs w Ancie (700)
- 23.6. Korzystanie z narzędzia FindBugs w Mavenie (702)
- 23.7. Konkluzja (704)


**24. Analiza wyników - półautomatyczne przeglądy kodu za pomocą narzędzia Jupiter (705)**


- 24.1. Wprowadzenie do Jupitera - narzędzia do przeglądania kodu w środowisku Eclipse (705)
- 24.2. Instalacja narzędzia Jupiter w środowisku Eclipse (706)
- 24.3. Zrozumieć proces przeglądów kodu narzędzia Jupiter (706)
- 24.4. Prowadzenie przeglądów własnego kodu (708)
- 24.5. Konfiguracja (709)

- 24.6. Ustawianie domyślnych wartości konfiguracyjnych (713)
- 24.7. Przeglądy indywidualne (714)
- 24.8. Przeglądy zespołowe (716)
- 24.9. Faza wprowadzania poprawek (719)
- 24.10. Wewnętrzne działania Jupitera (719)
- 24.11. Konkluzja (721)

**25. Koncentrujmy się na tym, co naprawdę ważne - narzędzie Mylyn (723)**

- 25.1. Wprowadzenie do narzędzia Mylyn (723)
- 25.2. Instalacja rozszerzenia Mylyn (724)
- 25.3. Śledzenie zadań i problemów (725)
- 25.4. Korzystanie z repozytoriów zadań (727)
- 25.5. Koncentrowanie się na wybranych zadaniach z wykorzystaniem mechanizmów zarządzania kontekstami (731)
- 25.6. Korzystanie ze zbiorów zmian środowiska Eclipse (734)
- 25.7. Współdzielenie kontekstu z pozostałymi programistami (736)
- 25.8. Konkluzja (737)

**26. Monitorowanie statystyk kompilacji (739)**

- 26.1. Wprowadzenie (739)
- 26.2. Narzędzie QALab (739)
- 26.3. Mierzenie ilości kodu źródłowego za pomocą modułu rozszerzenia StatSCM (747)
- 26.4. Statystyki narzędzia StatSVN w Ancie (748)

**VII: Narzędzia do zarządzania problemami (751)**
**27. Bugzilla (753)**

- 27.1. Wprowadzenie do narzędzia Bugzilla (753)
- 27.2. Instalacja narzędzia Bugzilla (753)
- 27.3. Konfigurowanie środowiska narzędzia Bugzilla (757)
- 27.4. Zarządzanie kontami użytkowników (758)
- 27.5. Ograniczanie dostępu do bazy danych z wykorzystaniem grup użytkowników (760)
- 27.6. Konfigurowanie produktu (762)
- 27.7. Śledzenie postępu z wykorzystaniem tzw. kamieni milowych (764)
- 27.8. Zarządzanie grupami produktów z wykorzystaniem klasyfikacji (764)
- 27.9. Przeszukiwanie błędów (765)
- 27.10. Tworzenie nowego błędu (767)
- 27.11. Cykl życia błędu reprezentowanego w systemie Bugzilla (768)
- 27.12. Tworzenie harmonogramu rozsyłania powiadomień (pojękiwania) (770)
- 27.13. Dostosowywanie pól systemu Bugzilla do potrzeb konkretnego projektu (771)
- 27.14. Konkluzja (772)

**28. Trac - lekkie zarządzanie projektami (773)**

- 28.1. Wprowadzenie do narzędzia Trac (773)
- 28.2. Instalacja narzędzia Trac (774)

- 28.3. Definiowanie projektu narzędzia Trac (776)
- 28.4. Uruchamianie narzędzia Trac w formie autonomicznego serwera (778)
- 28.5 Konfiguracja polecenia tracd jako usługi systemu Windows (779)
- 28.6. Instalacja narzędzia Trac na serwerze Apache (780)
- 28.7. Administrowanie witryną internetową Traca (781)
- 28.8. Zarządzanie kontami użytkowników (783)
- 28.9. Dostosowywanie witryny internetowej narzędzia Trac - korzystanie z funkcji witryn typu wiki (786)
- 28.10. Stosowanie systemu zarządzania biletami Traca (790)
- 28.11. Aktualizowanie błędów reprezentowanych w narzędziu Trac na podstawie zawartości repozytorium systemu Subversion (794)
- 28.12. Modyfikowanie pól biletów Traca (795)
- 28.13. Konfigurowanie powiadomień wysyłanych pocztą elektroniczną (797)
- 28.14. Raportowanie z wykorzystaniem zapytań i raportów Traca (797)
- 28.15. Zarządzanie postępami prac za pomocą map drogowych i diagramów linii czasu (800)
- 28.16. Przeglądanie repozytorium z kodem źródłowym (802)
- 28.17. Stosowanie kanałów RSS i formatu iCalendar (802)
- 28.18. Dostosowywanie stron witryny wiki za pomocą skryptów Pythona (805)
- 28.19. Konkluzja (806)

**VIII: Narzędzia do dokumentacji technicznej (807)**
**29. Komunikacja w ramach zespołu projektowego za pośrednictwem witryny Mavena 2 (809)**

- 29.1. Witryna internetowa Mavena 2 jako narzędzie komunikacyjne (809)
- 29.2. Konfigurowanie mechanizmu generowania witryny o projekcie Mavena (810)
- 29.3. Włączanie do witryny Mavena raportów generowanych przez inne narzędzia (815)
- 29.4. Tworzenie dedykowanego projektu witryny Mavena (819)
- 29.5. Definiowanie szkicu witryny (821)
- 29.6. Architektura mechanizmu generującego witryny Mavena (822)
- 29.7. Stosowanie fragmentów kodu (826)
- 29.8. Modyfikowanie wyglądu i sposobu obsługi witryny Mavena (827)
- 29.9. Udostępnianie witryny (830)

**30. Automatyczne generowanie dokumentacji technicznej (833)**

- 30.1. Wprowadzenie (833)
- 30.2. Wizualizacja struktury bazy danych za pomocą narzędzia SchemaSpy (833)
- 30.3. Generowanie dokumentacji kodu źródłowego za pomocą Doxygena (841)
- 30.4. Umieszczanie diagramów notacji UML w dokumentacji narzędzia Javadoc z wykorzystaniem narzędzia UmlGraph (850)
- 30.5. Konkluzja (854)

**Bibliografia (855)**
**Skorowidz (857)**