Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
all	all	all	use of the word synchronization in the concurrency patterns might confuse readers: why aren't these patterns in the synchronization chapter then?	replace synchronization with coordination		
Front cover	8	1	(447)provides	(447) provides	FB 24.10.00	
XIII	1	5	Programming Design	Program Design	FB 24.10.00	2
XIII	1	9	[Bat97]	[Bat79]	FB 25.07.00	2
XIV	7	1	Distributed	Networked	FB 24.10.00	2
XVI	3	1	Distributed	Networked	FB 24.10.00	2
XVIII	4	2	Federico	Frederico	FB 24.10.00	2
XVIII / XIX				add Kobi Cohen-Arazi, Robert Crell, Mike Curtis, Christopher Kohlhoff, Patrick Rabau, Eric Samuelsson, Stefan Scherer, and Johnny Willemsen to list of generally acknowledged people. This causes pagination to break.	FB 24.10.00	2
XIX	1	2		add Chung-Horng Lung	FB 06.05.01	
XX	2	1	Very special thanks go to Steve Rickaby, our copy editor,	Very special thanks go to Steve Rickaby, of Wordmongers Ltd, our copy editor,	FB 31.07.00	2
XXI	1	8	design patterns	patterns and pattern languages	FB 18.09.00	2
XXIII	3	1	Principal Senior	Senior Principal	FB 30.10.00	2
XXV	7	1	Louis Carroll	Lewis Carroll	FB 29.01.01	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
2	7	2+7	concurrent and networked	complex concurrent and networked (Note that this breaks pagination)		
2	6	4	common-off-the-shelf	commercial-off-the-shelf	FB 06.05.01	
3	3	3	have focused on	have focused largely on	FB 18.09.00	2
3	5	2	small	smaller	FB 18.09.00	2
3	5	3	available to	enjoyed by	FB 18.09.00	2
5	1	2-4	In today's competitive, time-to-market-driven environments, however, this often yields non-optimal *ad hoc* solutions.	In today's competitive, time-to-market-driven business environments, however, this process often yields non-optimal *ad hoc* solutions. (Note that this breaks pagination heavily)		
7	1	1	common	general	FB 19.09.00	2
11	5	8	completely	remove word	FB 24.10.00	2
14	7	1-2	At the next layer is an *event demultiplexer*, such as `select()` [Ste98], which waits for events to arrive ...	At the next layer is an *event demultiplexer*, which uses functions like `select()` [Ste98] to wait for events to arrive ...	FB 24.10.00	2
15	2	4	The events that are exchanged between peers in this architecture play four different roles [Bl91]	move sentence down below the diagram	FB 24.10.00	2
15	4	5	often waits	can wait	FB 24.10.00	2
20	2	7	all waiting threads will compete	the waiting threads can compete	FB 24.10.00	2
28	8	2	many	multiple	FB 24.10.00	2
29	diagram		mutex_lock (mutex) mutex_unlock (mutex)	mutex_lock (&mutex) mutex_unlock (&mutex)	FB 16.08.00	2
29	diagram		release	release()	FB 16.08.00	2
30	3	5	from	of	FB 24.10.00	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
30	4	2	When an event arrives	When one or more events arrive	FB 24.10.00	2
30	4	4	this event to a designated event handler	the events to designated event handlers	FB 24.10.00	2
31	5	1	which uses flow control	which is a transport protocol that uses flow control	FB 24.10.00	2
31	6	5	must not block while waiting for connection flow control to abate so it can finish sending a file to a client	must not block waiting for connection flow control to abate while sending a file to a client	FB 24.10.00	2
33	diagram		1..*	2	FB 24.10.00	2
33	diagram		--	notify_all()	FB 16.08.00	2
34	6	7	--	add: in the JAWS server.	FB 24.10.00	2
35	1	1	Handler	Handlers	FB 24.10.00	2
35/36	3	3	accept().	accept() on the same handle. Note that this causes a change in the page break	FB 24.10.00	2
35/36	4	5	waiting their turn	waiting on a synchronizer for their turn	FB 24.10.00	2
47	2	9	server, which	server application, which	FB 24.10.00	2
47	diagram		upper arrow to database	remove	FB 24.10.00	2
48	diagram		AF_INET	PF_INET	FB 08.01.01	2
48	diagram		Comment on 32 bit integer is after the #ifdef	Move in front of the #ifdef	FB 29.01.01	2
48	diagram		ace condition	race condition	FB 29.01.01	2
49	1	5	the code	the application code	FB 24.10.00	2
49	3	last	object oriented	object-oriented	FB 24.10.00	2
50	1	1	However, developers	Remove: however	FB 24.10.00	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
52	diagram		functionA in methodN	methodN should call functionC	FB 24.10.00	2
54	1	1	server carefully	remove: carefully	FB 24.10.00	2
61	diagram		`u_long addr)`	`ulong addr = 0)`	FB 06.05.01	
61	diagram		`return addr_.sin_port;`	`return ntohs (addr_.sin_port);`	FB 08.01.01	2
61	diagram		`return addr_.sin_addr. s_addr;`	`return ntohl (addr_.sin_addr. s_addr);`	FB 08.01.01	2
61	3	4/5	ntons() ... ntonl()	htons() ... htonl()	FB 08.01.01	2
61	diagram		AF_INET	PF_INET	FB 08.01.01	2
65	diagram		`<error_>`	`<status_>`	FB 08.01.01	2
67	5	6		full stop is missing	FB 06.02.01	2
68	right column		--	add: // Lock to protect request_count static Thread_Mutex lock;	FB 24.10.00	2
68	right column		--	adjustment of code lines	FB 29.01.01	2
69	diagram		AF_INET	PF_INET	FB 08.01.01	2
69	diagram		`thr_mgr.spawn`	`Thread_Manager::i nstance()->spawn`	FB 08.01.01	2
69	diagram		port	LOGGING_PORT	FB 30.01.01	2
70	2	4	`ACE_ Thread_Mutex`	`ACE_Thread_Mutex`	FB 29.01.01	2
74	3	2		add Patrick Rabau and Mike Curtis	FB 24.01.01	2
74	3	1	Ralph Johnson, Bob Hanmer	Bob Hanmer, Ralph Johnson	FB 06.05.01	
74	3	2		add Chung-Horng Lung	FB 06.05.01	
76	3	4	the	a	FB + DS 22.01.01	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
76	6	5	with high availability requirements	that cannot tolerate down-time	FB + DS 22.01.01	2
77	7	2	component components	components	FB 30.11.00	2
78	1	2	unnecessarily	move to the end of the sentence	FB + DS 22.01.01	2
79	3	6	, such as a DLL	move behind ‘form’	FB + DS 22.01.01	2
85	diagram		`<script_name>`	`<script_name> file`	FB + DS 22.01.01	2
88	5	9	into	add ‘and out of ‘	FB + DS 22.01.01	2
90	8	7	for determining	that determine	FB + DS 22.01.01	2
90	8/9			move last sentence from bulleted paragraph 8 to paragraph 9	FB 08.02.01	2
92	diagram		code arrangement and event type name		FB + DS 22.01.01	2
93	1	2	by a component configurator	remove	FB + DS 22.01.01	2
93	1	3		add comma before ‘such’	FB + DS 22.01.01	2
93	2	5	are	can be	FB + DS 22.01.01	2
93	5	5-7	because each component can be isolated from accidental corruption via operating system and hardware protection mechanisms	because operating system and hardware protection mechanisms can isolate each component from ac-cidental corruption	FB + DS 22.01.01	2
95	2	5	information passed	information to be passed	FB 14.02.01	2
101	2	6	`Time_ Server`	`Time_Server`	FB + DS 24.01.01	2
101	2	2	collocate ... in	distribute ... across	FB 06.05.01	

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
103	3	4	the *Implementation* step	*Implementation* activity 1 (83)	FB + DS 24.01.01	2
104	4	4	the *Implementation* step	*Implementation* activity 1 (83)	FB + DS 24.01.01	2
106	5	4	types of system	system types	FB 19.02.01	2
107	6	1	credit	credits	FB 19.02.01	2
112	5	5	[GHJV95]	[GoF95]	FB 07.05.01	
116	diagram		attach(), callback()	register(), dipatch()	FB 07.05.01	
120	table §3	1	marshaled	demarshaled	FB 07.05.01	
122	5	4	simplicity	uniformity	FB + DS 24.01.01	2
124	5	3	correspond to	play	FB + DS 24.01.01	2
124	5	2	A dispatcher corresponds to the Observer pattern's [GoF95] *subject* role	A dispatcher plays the *subject* role in the Observer pattern	FB + DS 24.01.01	2
128	diagram		`interceptor.clone()`	`interceptor_.clone()`	FB 07.05.01	
131	3	3	tokens	token	FB 07.05.01	
137	3	6/7	are complex to implement, use and optimize	are hard to implement, use, and optimize	FB + DS 24.01.01	2
148	5	5	extension	delete this word	FB 07.05.01	
150	6	3	EJB	Enterprise JavaBeans	FB + DS 24.01.01	2
150	5	2	specifying	defining	FB + DS 24.01.01	2
153	4	7	implementation activity 3	implementation activity 3.3 (153)	FB + DS 24.01.01	2
154	3	3	implementation activity 3	implementation activity 3.3	FB + DS 24.01.01	2
158	1	6	provide	prevent	FB 06.02.01	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
158	5	4	types	type	FB + DS 24.01.01	2
158	5	3	. These	, which	FB + DS 24.01.01	2
165	diagram		// Ask the factory finder for comp. factories	// Get references to the component factories	FB 07.05.01	
166	diagram		// Register components	// Register component factories	FB 07.05.01	
166	diagram		// access	// Access	FB 07.05.01	
170	5	6	can to create	can create	FB 14.02.01	2
171	4	5	Enterprise JavaBeans	Enterprise JavaBeans (EJB)	FB + DS 24.01.01	2
171	4	8	subset Enterprise JavaBeans (EJB)	subset of EJB	FB + DS 24.01.01	2
172	1	7	transparent	transparently	FB 14.02.01	2
176	4	3/4	scale to support ... well	scale up to support	FB 18.09.00	2
177	2	2	and and	and	FB 07.05.01	
177	7	7	on top of Reactor	top of a reactor or proactor	FB 18.09.00	2
179	diagram		topmost occurrence of “logging records”	move this down so that it does not chop the network line	FB 18.09.00	2
180	6	3	is	is	FB 07.05.01	
185	diagram		handle_event call from reactor to event handler: dashed arrow	solid arrow	FB 09.11.00	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
187			Good question! The reason is because the same Event_Handler can be registered multiple times for different HANDLEs (see the second register_handler() API on the bottom of page 189). In such a case, the Event_Handler does not actually own the HANDLE, so it must be passed in. Frank, I don't know if we have enough space to add this bit of insight to the comment on page 187, but it would be nice to consider!			
192	3	4	reading for writing	ready for writing	FB 07.05.01	
194	code		demux_table	demux_table_	FB 29.01.01	2
195	code		demux_table	demux_table_	FB 29.01.01	2
201	4	6	is then	then	FB 07.05.01	
202	2	4	add sentence	When the `CLOSE_EVENT` flag is passed to the `handle_event()` method the reactor will automatically remove the handler from its internal demultiplexing table after the method returns.	FB 08.01.01	2
202	2	3/4	`EVENT flag`	`CLOSE_EVENT flag`	FB 08.01.01	2
211	4	all	generalize real-world known use to having a receptionist handling and dispatching multiple phone lines.			

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
218	4	6	generates	initiates	FB 07.05.01	
226	diagram		event	completion event	FB 24.10.00	2
226	diagram			add handler parameter to first call from initiator to AOP.	FB 24.10.00	2
229	4	4	concrete event handler	concrete completion handler	FB 07.05.01	
230	2	7	concrete event handler	concrete completion handler	FB 07.05.01	
247	4	4	handle_events()	handle_event()	FB 07.05.01	
267	diagram		process_result	handle_event	FB 24.10.00	2
234	diagram		Async_Stream::read	Async_Stream::async_read	FB 14.02.01	2
243	4	4	6.2	5.2	FB 07.05.01	
245	2	3		add (2) after sentence 1.	FB 14.02.01	2
250	1	3	the waits	then waits	FB 07.05.01	
251	diagram		more synchronous	more asynchronous	FB 07.05.01	
258	3	6	allot	allow	FB 07.05.01	
265	CRC-card Initiator			add Completion Handler to Collaborator list	FB 07.05.01	
284	1	1	server	service	FB 07.05.01	
284	4	8	initiator	service	FB 07.05.01	
295/ 297	diagram			add a handle to the diagram to better show how synchronous and asynchronous connect works		
313	diagram		the dashed rectangle for templates is only for unbounded template classes, but not for concrete instances.	remove the dashed rectangle and bind the classes to the parameters in the ‘ordibary’ class box	FB 16.01.01	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
316	diagram		the dashed rectangle for templates is only for unbounded template classes, but not for concrete instances.	remove the dashed rectangle and bind the classes to the parameters in the ‘ordibary’ class box	FB 16.01.01	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
322	--	--	to be added as another see also	The intent of the Acceptor Connector pattern is similar to the Configuration pattern \cite{Magee:95}. The Configuration pattern decouples structural issues related to configuring services in distributed applications from the execution of the services themselves. This pattern has been used in frameworks for configuring distributed systems, such as Regis \cite{Magee:94}, to support the construction of a distributed system from a set of components. In a similar way, the Acceptor-Connector pattern decouples service initialization from service processing. The primary difference is that the Configuration pattern focuses more on the active composition of a chain of related services, whereas the Acceptor-Connector pattern focuses on the passive initialization of a service handler at a particular endpoint. In addition, the Acceptor-Connector pattern also focuses on decoupling service behavior from the service's concurrency strategies.		
322				add a new credits section for Eric Samuelsson's help with the class diagrams	FB 16.01.01	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
334	7	2	likely to occur	likely occur	FB 29.01.01	2
362	--	--	--	In Java, once the spec got changed, you can make double-checked locking work by using a volatile field for the field that is double-checked.		
374	diagram		ActivationQueue, enqueue(), dequeue()	ActivationList, insert(), remove()	FB 27.11.00	2
394	3		Siemens MedCom	Siemens Syngo	FB 19.09.00	2
404	diagram		1..*	*	FB 24.10.00	2
404	diagram		--	notify_all()	FB 16.08.00	2
436	diagram		sowakeup()	sbwakeup()	FB 29.01.01	2
443	3	3	within	with	FB 07.05.01	
455	diagram		promo new_leader	promote_new_leader	FB 16.08.00	2
456	diagram		arrow from processing to follower thread is in the wrong direction	change direction	FB 29.11.00	2
460	diagram		`LF_Thread_Pool (Reactor *r):` `reactor_ (r), followers_condition_ (mutex_) { };`	`LF_Thread_Pool (Reactor *r):` `reactor_ (r) { };`	FB 14.02.02	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
460			Question: p. 460. What does reactivate_handle() do exactly? It is called by a thread that has just finished processing an event on a particular handle, while in the mean time another thread as been promoted as leader. This new leader has been promoted after the handle in question has been deactivated by the first thread. Assume that while the first (now follower) thread processes the event, another event becomes ready on the same handle. The lead thread will not see it, since when it does select(), that handle is disabled. So question is: does reactivate_handle() somehow notify the leader thread so that it can reattempt to select on a handle set that includes the reactivated handle?	There are two ways to address this: 1. The reactivate_handle() can in fact notify the current leader thread so that it can select() on this handle again. This "eager" approach is what we use in the ACE_TP_Reactor in order to avoid starvation. 2. The reactivate_handle() might simply queue up the request to reactive the handle so that it'll only be considered again after the current leader thread has gotten another event and a new leader has been detected. This "lazy" approach might be a good optimization for use-cases where events occurred frequently on all the handles in the handle set since it would reduce unnecessary context switching without causing starvation.		
462	code		`reactor_->handle_events ()'`	`reactor_->handle_events ();`	FB 29.01.01	2
468	diagram		arrow from processing to follower thread is in the wrong direction	change direction	FB 29.11.00	2
472	3	1	Leader/Follower	Leader/Followers	FB 29.01.01	2
481	diagram		bottom right class: Thread-Specific Object Proxy	bottom right class: Thread-Specific Object	FB 08.01.01	2
481	diagram		bottom right class: key	remove	FB 08.01.01	2
505	2	2	Chapter 2 connect	Chapter 2 through 5 connect	FB 20.07.00	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
510	1	1	we want to explore	we explore	FB 19.08.00	2
510	6	2-4	‘Duplicate’ entries for patterns that are frequently referenced by other patterns avoid having too many crossed relationships.	We ‘duplicate’ entries for patterns that are referenced frequently by other patterns to avoid having too many crossed relationships.	FB 19.09.00	2
511	diagram		Half-Sync/Half-Async box is not in correct shade	Shade the box to 10%	FB 19.09.00	2
520	4	5	pattern-based	pattern language-based	FB 19.09.00	2
520	4	1-3	Our pattern language has been applied to many real-world applications, in particular, but not only to systems that are built using the ACE framework.	Our pattern language has been applied to many real-world applications including—but not limited to—those built using the ACE framework.	FB 19.09.00	2
524	2	4	, however,	remove “however”	FB 19.09.00	2
524	2	6/7	service access and configuration, event handling, synchronization, and concurrency	apply emph-content character format to all 4 topics.	FB 18.09.00	2
525	1	6	ever	every	FB 18.09.00	2
526	2	7/8	as a result	move from end to the beginning of the sentence	FB 19.09.00	2
526	4	3	however	Remove this word	FB 19.09.00	2
530	1	8/9	due in large part to the effort	due in part to the amount of effort	FB 19.09.00	2
545	3	1	A component specifies an interfac	A component specifies one or more iterfaces	FB 24.10.00	2
545	4	4	polymorphism	apply emph-content character format	FB 22.09.00	2
545	4	6	the pattern	patterns	FB 11.09.00	2
546	1	1	The external interface	The external interfaces	FB 24.10.00	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
546	1	2	operating system	apply emph-content character format	FB 22.09.00	2
546	4	4	relationships	apply emph-content character format	FB 18.09.00	2
547	7	6	functions	apply emph-content character format	FB 22.09.00	2
548	2	2	network	networked	FB 18.09.00	2
548	4	2	method closure	apply emph-content character format	FB 18.09.00	2
548	5	1	(CORBA)	deleted	FB 18.09.00	2
548	7	2	responsibilities	apply emph-content character format	FB 18.09.00	2
549	11	3	application	apply emph-content character format	FB 22.09.00	2
549	4	1/2	host-independent / host-specific	system- and application-independent / system- and application-specific Note that this breaks pagination heavily.		
550	12	1/2	A future [Hal85] [LS88] allows a *client* to obtain the result of a *method* invocation after the invoked method finishes executing.	A future [Hal85] [LS88] allows a *client* to obtain the result of a *method* at any point in time after its invocation.	FB 22.09.00	2
551	5	all	hardwiring is not only about using magic numbers	Give an additional example?		
551	7	1	Hyper Text Transport Protocol	HyperText Transfer Protocol	FB 24.10.00	2
551	9	3	relationship	apply emph-content character format	FB 06.10.00	2
551	11	4	single and multiple inheritance	apply emph-content character format to “single” and “multiple inheritance”	FB 06.10.00	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
552	2	1	special class	concrete class	FB 18.08.00	2
552	3	3	functions	apply emph-content character format	FB 06.10.00	2
552	3	6	pattern	apply emph-content character format	FB 06.10.00	2
552	4	all		Shall we extend the definition? In particular we could explain that a cache holds a copy of specific portion of the main memory which then gives an application the illusion to access the main memory.		
553	1	1		Shall we extend the definition? In particluar we can explain that jitter is undesireable for certain application types, such as A/V streaming.		
554	2	1	A function performed	An operation performed	FB 24.10.00	2
554	3	3	object that contains the context of a method	apply emph-content character format to “objec” and “method”	FB 06.10.00	2
554	7	2	module	apply emph-content character format	FB 06.10.00	2
555	2	3	domain	apply emph-content character format	FB 06.10.00	2
555	9	3	HTTP	apply emph-content character format	FB 06.10.00	2
555	10	all		Add that it also returns no error values or other (status) information	FB 24.10.00	2
556	3	all		There is also out-of-band data.		

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
556	11	4	relationships and responsibilities	apply emph-content character format to “relationships” and “responsibilities”	FB 06.10.00	2
557	6	1	scheduling	apply emph-content character format	FB 06.10.00	2
558	4	1	instance	apply emph-content character format	FB 06.10.00	2
558	4	1	object reifies a class	apply emph-content character format to “object” and “class”	FB 06.10.00	2
558	8	all	An incremental activity that abstracts general-purpose behavior from existing software to enhance the structure and reusability of *components* and *frameworks*.	An incremental activity that improves the internal structure of *components* and *frameworks*.	FB 06.10.00	2
559	1	5	clients	apply emph-content character format	FB 06.10.00	2
559	1	7	pattern	apply emph-content character format	FB 06.10.00	2
559	3	all		There are also other semaphore implementations possible. Mark the one in the definition as one possiblity and reference ohthers.		
559	7	1	applications	apply emph-content character format	FB 06.10.00	2
561	2	2	address space that	address space of a process that Note that this change breaks pagination heavily.		

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
561	5	all		Add a disadvantage? If the actual resource consumption is way smaller than the pre-allocated resources at all points in time during a system's execution, we wast resources too.		
562	6	1	layer	apply emph-content character format	FB 06.10.00	2
564	diagram		Navigability and realization relationships cannot touch an unbound template	correct that wrong notation, add a seperate template class	FB 16.01.01	2
568	4	1	icon for actions and conditions included twice	remove one icon	FB 06.10.00	2
570	[BGHS98]		9th	9th	FB 20.07.00	2
570	[Bat97]		[Bat979] + 1997	[Bat79] + 1979	FB 25.07.00	2
571	[Boo94]		Second Edition	2nd edition	FB 20.07.00	2
571	[BRJ98]		Jacobsen	Jacobson	FB 07.05.01	
573	[Doble96]			apply emph-content character format for paper title	FB 08.01.01	2
573	[CY91]		second	2nd	FB 20.07.00	2
575	[GLDW87]		19987	1987	FB 07.05.01	
577	[HMPT89]			apply emph-content character format for paper title	FB 08.01.01	2
580	[Lea99a]		2nd	2nd	FB 20.07.00	2
581	[LY99]		Edition	edition	FB 20.07.00	2
582	[MBKQ96]			apply emph-content character format for paper title	FB 08.01.01	2
582	[Mey97]		Edition	edition	FB 20.07.00	2
590	[Sol98]		Edition	edition	FB 20.07.00	2
591	[Ste98]		Second Edition	2nd edition	FB 20.07.00	2

Page	¶	Line	Wrong Text	Correct Text	Done	Correction appears in print #
591	[Ste99]		Edition	edition	FB 20.07.00	2
591	[Str97]		Edition	edition	FB 20.07.00	2
624			Siemens MedCom	Siemens Syngo	FB 19.09.00	2
631	2	3	Federico	Frederico	FB 24.10.00	2
631	3	15		add Robert Crell	FB 24.10.00	2
631	C	3	Carroll, Louis	Carroll, Lewis	FB 29.01.01	2
631				add Mike Curtis	FB 29.01.01	2
631	3	9		add Kobi Cohen-Arazi	FB 14.02.01	2
632	L			add Chung-Horng Lung	FB 06.05.01	
632	2	5	Harrison, Neil xviii	Harrison, Neil xix	FB 24.10.00	2
632	6	9		add Christopher Kohlhoff	FB 14.02.00	2
633	10	5		add Stefan Scherer	FB 29.11.00	2
633	10	5		add Willemsen, Johnny	FB 27.11.00	2
633	5	1		add Eric Samuelsson	FB 16.01.01	2
633	4	1		add Patrick Rabau	FB 24.01.01	2
back cover	--	--	All patterns present extensive example and known uses in multiple programming languages, including C++, C, and Java.	These patterns present extensive example and known uses in multiple programming languages, including C++, Java, and C.		

Replacement pages