PKUAS White Book

Institute of Software

School of Electronics Engineering and Computer Science

Peking University

2006.03

License

Contents

List of Figures

Chapter 1. Introduction

PKUAS is an open-structured component operating platform across Internet based on middleware technology and compliant with Java 2 Platform Enterprise Edition. It implements all the functionality of J2EE 1.3 and EJB 2.0. It is designed and implemented by Software Institute, School of Electronics Engineering and Computer Science of Peking University, who has the independent intellectual property rights. PKUAS is suitable for SMEs and governments since its easy-to-use characteristic, low-resource requirements, high efficiency and high performance-price rate. Its open-structured design provides strong interoperability and high extensibility, which is feasible and flexible for enterprise's applications and domain oriented middleware. Meanwhile, PKUAS has the advanced features of next generation middleware - the ability of on-line evolution, which enables substituting the critical components of application without intermitting the running system.

PKUAS provides comprehensive supports for J2EE/EJB applications. It complies with J2EE1.3 and EJB 2.0, and has passed the test of SUN's JPS (Java PetStore) 1.3, performance test of ECperf1.0B and ObjectWeb's conformance test.

The architecture of PKUAS is illustrated as Figure11.1：

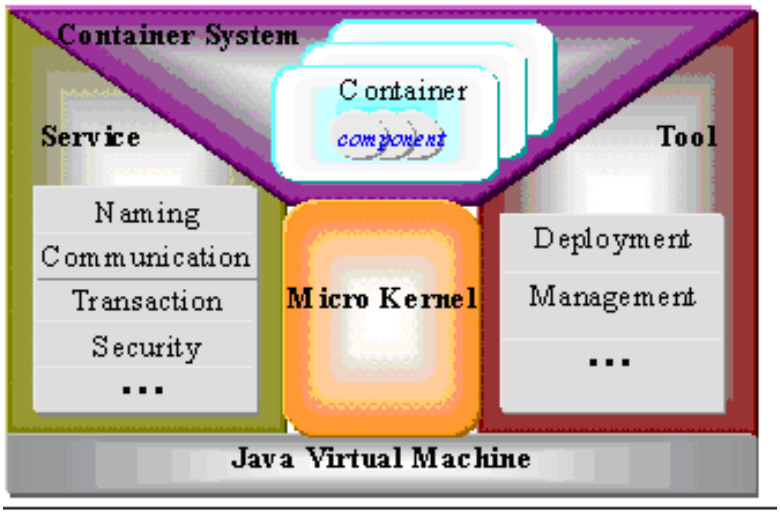

Figure 1.1 Component Operating Platform PKUAS Architecture

From operating system's micro kernel idea, PKUAS constructs itself with a set of components based on a micro kernel, encapsulating other functions of platform in several independent modules, enabling customized configuration and functionality extension. The kernel is responsible for constructing and integrating all the modules into an entire component operating system when PKUAS starts up. Based on JVM, the internal functionalities of PKUAS are componentized into four entities:

Container System: A container is the runtime space for components deployed by applications, managing their lifecycle and runtime contexts. PKUAS implements four kinds of EJB container,which support all kinds of EJB defined in J2EE/EJB Specifications :Stateful Session Beans Container, Stateless Session Beans Container,Entity Beans Container and Message-Driven Beans Container. One instance of a container holds all instances of one EJB. And a container system consists of the instances of the containers holding all EJBs in an application. Such organization of the containers facilitates the configuration and management specific to individual applications, such as different communication ports, authentication mechanism and security realms for different applications.

Service: PKUAS implements the common functions independent of business logic, such as naming, communication, security and transaction. Services are the abstraction and extraction of common features of application system, helping developers paying more attention to concrete business logic. It also has high feasibility since some kinds of services can be confirmed on assembly phase. The services in PKUAS can be added, substituted and deleted dynamically at run time, which promotes the system's scalability and reliability. Besides, PKUAS provides customized configuring ability for customer-constructed services.

Tool: it provides functions to facilitate the operation of PKUAS, including management tool and plug-in of mainstream IDE. Management tool is presented as Web pages, which enables user's monitoring, configuring, and adjusting the containers and services of PKUAS at run time. PKUAS has also published the developing tools as JBuilder's and Eclipse's pluggings, to facilitate developing and deploying of J2EE applications.

Micro kernel: It provides a registry for the above modules and other management functions, like class loading, configuration, uninstall, start-up, stop, and suspend. Different from operating system, PKUAS's micro kernel doesn't take charge of the communication among modules and consequently it avoids the degrading of performance.

PKUAS doesn't have much requirements for hardware and software conditions.

It can work well on Pentium 166MHz, 48M memory, 200M hard disk space. PKUAS supports various operating systems, including Microsoft Windows 95,98(1st or 2nd Edition), Windows NT4.0（Service Pack 5）, Windows ME, Windows 2000 Professional/Server/Advanced Server, Windows XP，Windows Server 2003 and Unix/Linux/Solaris System. For software prerequisites, PKUKAS needs Java 2 SDK 1.4.2 or above.

Chapter 2. PKUAS Micro-kernel

The component management framework, service management framework and the inter-operation framework together provide the component platform with the foundation for componentization, but in order to ensure that they cooperate and form a manageable entity as a whole, the help of a coordinator is needed.

Pkuas adopted the Java JMX technology, packaging the managed public services and applications into MBeans, and using the MBeanServer to manage these objects, thus simplifying the overall implementation while maintaining scalability. The coordinator is also responsible for loading and starting the individual components of the platform. In order to ease the task of configuring and managing the containers, services and protocols for the end-user, and also able to run J2EE compliant applications, PKUAS adopts a declarative type of configuration, and uses XML as a language mechanism for describing the system configuration. Therefore, the coordinator also integrates XML support to parse the configuration description files to ensure correctly completing the task of bootstrapping the component platform.

Chapter 3. PKUAS Container Management Framework

The container management framework (hereinafter referred to as 'CMF') is an open architecture in PKUAS designed to meet the needs of customization and extension of the container, supporting on-line management of components via the evolution mechanism integrated into the framework.

Container is the runtime environment of components. Components don't contact directly with the underlying infrastructures and the common services (such as transaction, security, etc), but via the container. Container is the agency or facade of the common services. It is the bridge gluing them together. Container is also responsible for maintaining components' life cycles and managing kinds of resources used by components. Meanwhile, client views provided by components are implemented by container; and all invocations on components are intercepted by container while kinds of service policies are applied to the invocation processes based on the requirement of the application.

The CMF contains:

- Container Management Mechanism: By analyzing the J2EE specification and functions of the prevalent application server products, PKUAS clearly defines the division of responsibility between the container and the underlying platform, abstracts a set of container management interfaces, sets up a smart management mechanism. This mechanism takes charge of installation, uninstall and state query of the container, binding together the container and required underlying communication protocols, deploying components to the container, etc.

- General on-line Evolution Mechanism: Although the operations related to the on-line evolution are acted on specified component, the required support mechanism should be general for all the components and containers. Therefore, it is necessary to abstract and implement a general mechanism in container, supporting on-line evolution and management. The key techniques include state management of components, dynamic installation and uninstall, request dispatching mechanism, request buffering, component instances buffering, component protections and so on.

- Customized Container: According to the analysis of container responsibilities, PKUAS works out a basic invocation framework for container, so user can customizes the components with ease, Based on this, PKUAS implements several containers defined in J2EE to validate the feasibility and correctness of the customization and extensibility of container in terms of practicability, including stateless session container, stateful session container, entity container and message-driven container.

CMF weaves together the logical related container instances, preventing from the cross interferences between components of different applications. It uses dynamic class loading to prevent from arbitrary accessing the components from malicious code, enhancing the safety as a result. Integration of container, underlying infrastructures and common services is the core of the component platform. When the application is starting, corresponding container and services interceptor are loaded according to the description descriptor in the application archive. The client request is received and dispatched to corresponding container to handle at runtime. In this way, it helps the administrator manage the state of container, further supports the system evolution.

Chapter 4. PKUAS interoperable Framework

The open interoperability framework of PKUAS supports many protocols. It has strong inter-connected ability and expansibility. And it's easy to customize, as figure 4.1 shows.

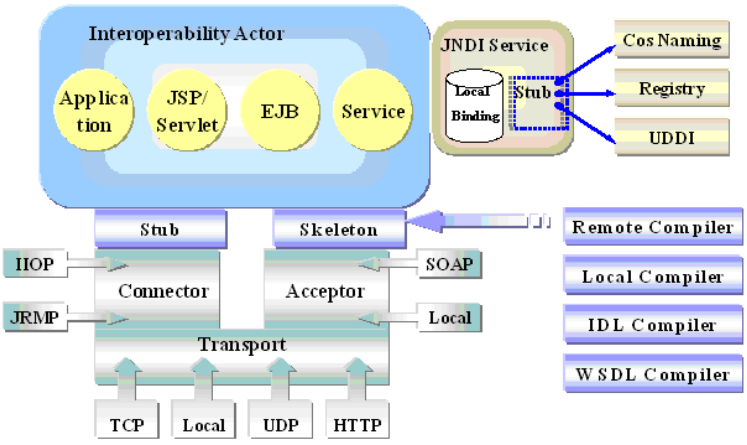

Figure 4.1 PKUAS interoperable Framework

The open interoperability framework differs from traditional middleware's integrative frameworks. Developers needn't pay attention to the runtime interoperability protocols and communication patterns while developing components. Instead, an interoperability protocol is bound to a component in the deployment phase. This makes the component development more transparent, Furthermore, developers can encapsulate an interoperability protocol into an adapter. Then components can have the ability to support different interoperability protocols using different adapters.

PKUAS has pre-provided some existing mainstream interoperability protocols in the form of plug-in, including IIOP，JRMP，SOAP，LOCAL. IIOP is the default.

Chapter 5. PKUAS Services

PKUAS Services includes J2EE standard services and PKUAS extent services.

5.1 Standard Services

5.1.1 Transcation Service

When developing the EJB component, the developer can use the transaction service provided by the application server to improve the reliability of components at runtime. It makes the operation which use the transaction service ACID properties. PKUAS provide transaction service to EJB component, Web Component, Standalone application and JMS session. The design and implementation of transaction service module makes AS provide two ways to visit the transaction: one is the programmatic way, the other is the declarative way. PKUAS completely support the J2EE specification transaction model, provide strong transaction support to application and ensure the operation to run successfully.

5.1.2 Message Service

PKUAS' message service is implemented by using JMS. JMS provides persistent message cache mechanism and asynchronous communication model, which efficiently improves the flexibility and extendibility of J2EE applications.

PKUAS implementes message-driven container to support message-driven EJB .As the result of PKUAS' support of JMS message service and message-driven EJB, the J2EE applications running on PKUAS can communicate with each other asynchronously, their flexibility has been improved and meanwhile the loose coupling association of communication components can improve applications' maintainability and extensibility.

With the support of JMS service, the four types of J2EE container of J2EE architecture, which are EJB container, Web container, Application client container and Applet container, can communicate with each other asynchronously by sending and receiv-

ing JMS message. The relationships of JMS service and other parts of PKUAS are illustrated in Figure 5.1.

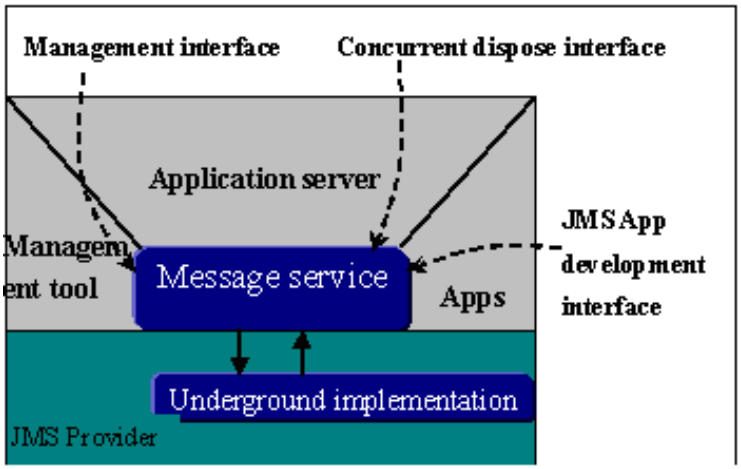

Figure 5.1 PKUAS Message Service

As the implementation of message middleware, JMS provider provides the underground implementation of message service. The association between PKUAS and JMS provider is implemented through the encapsulation of JMS provider's API classes and management classes. Meanwhile PKUAS takes part in the implementation of message's concurrent disposal interface and communicates with message service. Furthermore, message service provides management interface for PKUAS' management tool.

First of all, JMS is included as a standard service， which enables the components within the four types of container(EJB container, Web container, Application client container and Applet container) to interoperate with each other. In asynchronous messaging model, the interoperation has to make use of naming service to locate remote interface, and there are some limitation with the operation direction and the operation participators. JMS extends the interface location-based interoperation, introduces the concept of asynchronism, and makes communication among all types of components more convenient. Communication participators send and receive message by using the application interface created for common JMS clients. Some examples for JMS clients are applications which use JMS API to send and receive message in single-thread mode, standalone applications, Web components and EJB components.

Secondly, as a new type of EJB, message-driven EJBs act as a special kind of message consumers. JMS defines message disposing procedure for these enterprise message consumers, which is different from that defined for common JMS client. In

such situation, more than one message listener can receive message concurrently and then dispatch the message to many message-driven instances through message-driven containers. This mechanism enhances the ability of message disposing and embodies application server's support for enterprise disposing of JMS message. As a component of the message-driven container, the message-driven EJB does not only have the ability to receive enterprise message, but also can send and receive message just like common JMS clients.

If message operations take place in publish/subscribe model domain, JMS message can have message-driven EJBs or common JMS clients as its consumer, or both; otherwise, in point-to-point domain, as a piece of message can have only one message consumer, JMS message's consumer is either a message-driven EJB or a common JMS client. All communication participators are hided behind the JMS provider and do not need the references of other participators, whether the participators are message-driven EJBs or common JMS clients. When and who to receive the message is decided not by the sender but by the JMS provider, and in other words, the receiver is independent of the sender. After agreeing on the message format, the message participators only need the message destination to complete communication procedure. The messaging procedure has the characteristics of implicitly asynchronous communication.

5.1.3 Security Service

PKUAS integrates with JAAS security mechanism, has strong ability of security protection and flexible configuration ability, provides authentication, authorization, safe transmission and so on, which are defined in the J2EE Specification, and protects the application program completely.

PKUAS provides the uniform security service for the J2EE application, the whole J2EE application (client components and EJB/WEB components on the server side) uses uniform authentication mechanism, security domain and roles mapping. It extends the standards security defined in the Servlet Specification. If the application developer does not carry out the configuration of page protection, it means security check will be postponed until the EJB components are called. WEB container will catch the security exception when calling the EJB components: if the reason of security exception is that the user does not login, it will redirect the request to the login page. WEB container obtains the user's identity credential information, which is then authenticated by authentication server, and establishes the security context for this session. WEB con-

tainer will bring this security context when performing the method call to the EJB, as shown in figure 5.2.

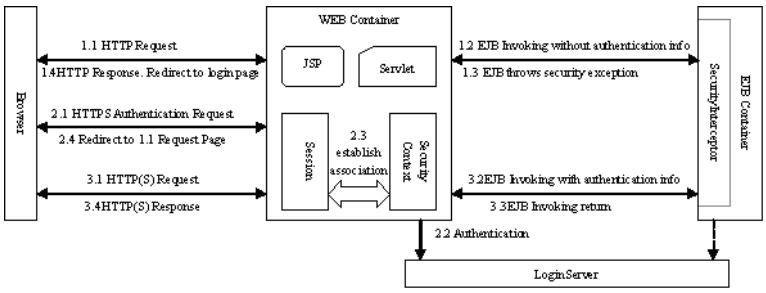

Figure 5.2 Web

5.1.4 Naming Service

PKUAS naming service provides the mechanism to find and locate a needed component, and solves the problems of releasing and locating heterogeneous components using JFS（JNDI Federation Server）.

Different interoperability protocols use different naming services. Interoperability address are usually stored in specific naming servers. Users are only allowed to call these naming servers, but can't change them. So PKUAS uses Proxy mechanism to integrate these naming servers.

First, a naming server which is to be integrated must be registered to the global JFS. The JFS creates corresponding Proxy plug-in instances for each naming server. The Proxy saves names from the naming servers to the JFS by traversal. Proxy references are bound to these names, not to interoperability addresses. Therefore, the JFS doesn't return a result directly after finding a name binding. Instead, it entrusts Proxy to look up the corresponding naming server, then returns.

What's more, in order to let a heterogeneous component access an EJB, in the EJB deployment phase, the EJB must be declared to support the interoperability protocol which the heterogeneous component uses. PKUAS constructs interoperability addresses when it initializes. These addresses are registered to JFS, for communication between EJB and heterogeneous components. As JFS uses Proxy to achieve binding, a heterogeneous component can get EJB's interoperability address through its own naming server.

Figure 5.3 gives the naming service framework in PKUAS.

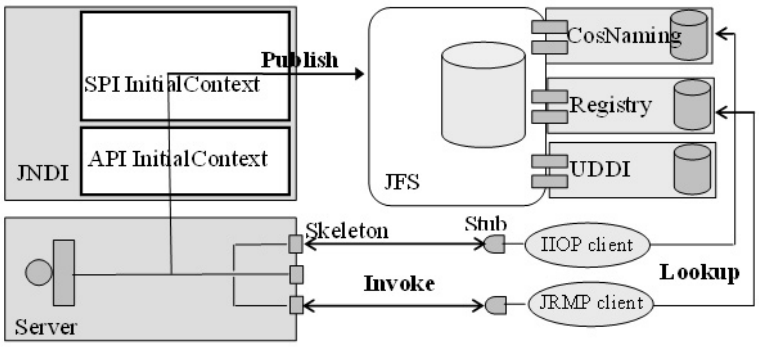

Figure 5.3 Naming Service

5.1.5 CMP Support

EJB2.0 Specification introduces a new CMP 2.0 component model, EJB Container handles CMP entity EJB persistence automatically at runtime. The Container is also in charge of implement and execution of finder method in entity EJB with a new query language - EJB QL. EJB-QL is used to defined find and selects method, it can make developers deploy entity EJB faster without complex SQL.

CMP and BMP are relative. While using BMP to implement entity EJBs of some relation database, developers need to write JDBC code manually. The disadvantage is that, developers need to write and maintain the details of database access, which makes it harder to write EJB code for specific database or application server. Generally, the more details of persistence in EJB code, the worse portability it is.

While developing with CMP, developers need to provide an illustrative mapping set, let the container to generate all necessary JDBC calls, not to program to manage entity persistence of database access. It is not necessary to provide any details of database access, all you need to do is to identify entity EJB persistence attributes, definitions, etc.

The main aim of designing EJB 2.0 is to enhance the portability of CMP. With CMP, it is easy for entity EJB to transplant to another compatible EJB container. During implement CMP entity EJBs, you can reuse an EJB in other container, also redefine EJB persistence deployment descriptor and finder method. In deployment descriptors, the only part specified by the provider is the mapping between persistence field and database column. Because of these, it is possible to modify database without changing(or compiling) entity EJB class.

Using CMP, entity EJB can be logically independent of data source, because persistence manager and abstract persistence schema can be used to generate JDBC class to access entity status in relation database or non-relation database, that is to say, this entity to data source mapping is limited to relation database. These tools are specified to the given database.

PKUAS CMP tool is used to process the entity EJB compliant with CMP2.0 specification. While deploying a CMP entity EJB, deployers use persistence manager to generate the implementation of abstract bean class, these classes include runtime database access code. PKUAS access database with these implementation, not abstract CMP entity EJB to generate EJB instance at runtime, shown in figures5.4

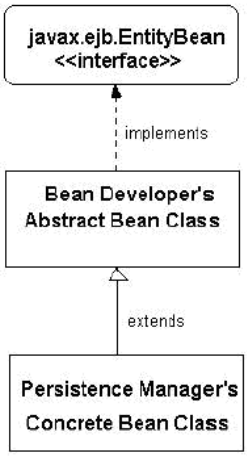

Figure 5.4 CMP entity EJB

CMP is one of the storage mechanisms of entity EJB. With CMP, EJB code will not access database directly. EJB container will handle all database access needed by entity EJB, entity EJB will not be binded to some specified storage mechanisms. In PKUAS, container implements CMP at deployment time.

The CMP implement in PKUAS is based on JDBC API. The management of object-relation mapping in PKUAS supports relation table and foreign key table. One to one and one to many relationship are usually presented by foreign key table, relation table works, too. But many to many relationship can only be presented by relation table.

5.1.6 Data Service

The data service in PKUAS interacts with the enterprise database, provides data access service to business logic level, saves the persistent data gained by the compute of business logic and provides the high efficiency and reliable access to the data.

After the CMP(Container Managed Persistent) being introduced to J2EE specification, the business logic developer usually do not realize the existence of the data service module because the CMP entity bean enclose the data access. So, in a way, the datAfter the CMP(Container Managed Persistent) is introduced to J2EE specification, the business logic developer usually do not realize the existence of the data service module because the CMP entity bean enclose the data access. So, in a way, the data service is even closer with the J2EE platform. The J2EE component developer only need to care about how to configure the data source, how to declare the usage of data service during the development of component. While the J2EE platform developer should be concerned with the design and implementation of the data service to provide high efficiency and make it conform to the requirements of the business logic level. The data service in PKUAS provides: (1) Set up and initialize the relation with the bottom database, create the available data source to conform to the requirements; (2) Provide the data connection service, separate the logical connection from the physical connection to reduce the lose of the performance caused by the create and release of the data source; (3) Support the JTA transaction.

5.1.7 JavaMail Service

Many internet applications need to send emails, PKUAS implements the JavaMail service conforming to J2EE JavaMail specification 1.2 to provide mail sending function for application component. JavaMail provide TMAP and SMTP protocols to connect the mail server over internet. But JavaMail does not provide the function of mail server, so users must connect JavaMail to a mail server.

5.2 Extended Services

5.2.1 User Defined Service

User can customize the user defined service via the services extension support provided by PKUAS with the power of JMX (Java Management eXtension). JMX server loads the user defined service when PKUAS starting according to user's requirement.

5.2.2 Logging Service

PKUAS logging service records detailed runtime information and exceptions generated by modules of the application server or deployed applications, which can be used for monitoring, tracking and finding problems. Logging is loaded as a kind of service by the PKUAS application server. Therefore, users can choose to start or stop logging service according to their demands of system performance.

Chapter 6. Tools

6.1 Management Tools

PKUAS provides a simple management tool :PMC to monitor and configure applications ,resources on PKUAS and PKUAS itself.PMC exists as an application and starts when PKUAS starts.PMC is protected by security mechanism of PKUAS.

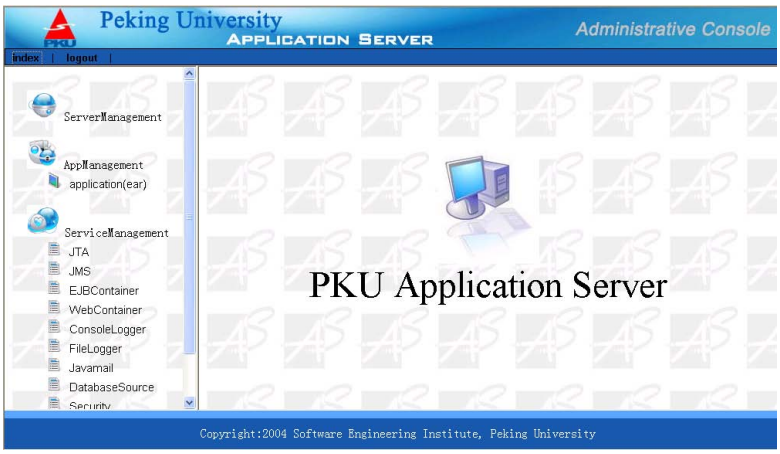

Figure 6.1 PKUAS Management Interface

6.2 IDE Plugins

PKUAS offers user-friendly IDE plugins for both Eclipse and JBuilder. With these tools, developers can develop J2EE applications visually, and build a prototype or small system just by drag-and-drop. Therefore, they greatly reduce developers' work, enhance development efficiency and generate more cost-efficient applications. Life is much easier with them. For more information about these tools, check out the website of PKUAS.

GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document ”free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of ”copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The ”Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ”you”. You accept the license if you copy, modify or distribute the work in a way

requiring permission under copyright law.

A ”Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A ”Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The ”Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The ”Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A ”Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not ”Transparent” is called ”Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The ”Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ”Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section ”Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as ”Acknowledgements”, ”Dedications”, ”Endorsements”, or ”History”.) To ”Preserve the Title” of such a section when you modify the Document means that it remains a section ”Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies.

If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement. C. State on the Title page the name of the publisher of the Modified Version, as the publisher. D. Preserve all the copyright notices of the Document. E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. H. Include an unaltered copy of this License. I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title

of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. M. Delete any section Entitled ”Endorsements”. Such a section may not be included in the Modified Version. N. Do not retitle any existing section to be Entitled ”Endorsements” or to conflict in title with any Invariant Section. O. Preserve any Warranty Disclaimers. If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled ”Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents,

unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled ”History” in the various original documents, forming one section Entitled ”History”; likewise combine any sections Entitled ”Acknowledgements”, and any sections Entitled ”Dedications”. You must delete all sections Entitled ”Endorsements.”

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ”aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version"

applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.