[19]中华人民共和国专利局

[11] 公开号 CN 1081010A

[12] 发明专利申请公开说明书

[21]申请号 93107664.1

[51]Int.Cl[5]

[43]公开日 1994 年 1 月 19 日

G06K 15 / 00

[22]申请日 93.6.28

[30]优先权

[32]92.7.1 [33]US[31]07 / 907294

[71]申请人 L·M·埃利克逊电话股份有限公司

地址 瑞典斯德哥尔摩

[72]发明人 里卡德·尼尔松 乌尔夫·马克斯特伦

莱夫·克劳夫佛

[74]专利代理机构 上海专利事务所

代理人 沈昭坤

说明书页数: 附图页数:

[54]发明名称 用于计算机运行期间改变软件的系统

[57]摘要

在电讯交换系统中希望能在系统运行期间不停止系统运行而很快地进行软件替换，本发明所揭示的系统能平稳地进行这种修改，并使之对现行活动的影响最小；所揭示的系统把电讯系统的动态操作作为一组平行而独立的事务处理来对待且每个事务处理由一系列相关活动组成，使得能在运行中改变软件；这种事务处理一般是与系统的一个特定用户如电话用户或操作员相联系的一个功能元素，如待系统处理的一个呼叫或待系统执行的一个命令。

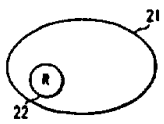

图 3A

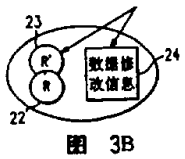

图 3B

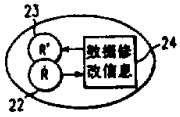

图 3C

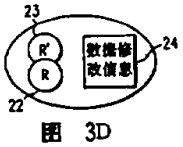

图 3D

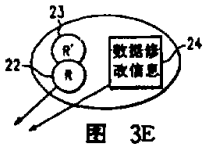

图 3E

权 利 要 求 书

1. 一种在对现有软件处理现有数据没有重大干扰的情况下将计算机装置的数据处理控制自动传递给新软件的方法，为与计算机装置一起使用，同时它用现有软件处理现有数据并且接收待处理的新数据，其特征在于采用下列步骤：

在计算机装置中装入新软件；

模仿现有软件处理实际的数据，用新软件处理测试数据；

在成功地处理了所述的模仿实际数据的测试数据之后，自动地将全部的数据处理控制传递给新软件。

2. 根据权利要求 1 所述的方法，其特征在于，在所述的使用和自动传递步骤之间还采用下列步骤：

随着新软件成功地处理了测试数据，允许新软件对实际数据的一个样本部分进行处理，否则这些数据将是由老软件进行处理的；

在新软件成功地处理了所述的预先确定的实际数据的样本部分之后，将所述的全部的数据处理控制自动传递给新软件。

3. 根据权利要求 2 所述的方法，其特征在于在所述的允许和自动传递步骤之间还采用下列步骤：

用新软件处理所有的新数据，同时现有软件处理现有数据。

4. 根据权利要求 3 所述的方法，其特征在于：所述的自动传递步骤是在用现有软件处理现有数据结束之后完成的。

5. 根据权利要求 4 所述的方法，其特征在于：所述的自动传递步骤是为响应新软件对所有新数据的成功处理和现有软件对现有数据处理的结束而执行的。

6. 根据权利要求 1 所述的方法，其特征在于：不但现有数据而且新数据都是在一系列事务处理中得到处理的，现有数据和新数据

两者都包括动态数据和半永久性数据，动态数据是在每个事务处理的处理期间所建立和使用的并且在处理结束时消除的数据，半永久性数据是由许多个事务处理的处理过程所使用并在许多个事务处理的处理过程中存在的数据，所述的方法包括附加的步骤：

将数据从现存软件传递到新软件。

7. 根据权利要求 6 所述的方法，其特征在于：只有半永久性数据从现存软件传递到新软件。

8. 根据权利要求 6 所述的方法，其特征在于，新软件中数据的表示法不同于现有软件中数据的表示法，所述的将数据从现有软件传递到新软件的步骤包括这样一个步骤：

将所述的数据从所述的现行表示法转移为所述的新表示法。

9. 根据权利要求 8 所述的方法，其特征在于：将所述的数据从所述的现行表示法转换为所述的新表示法的所述的步骤是根据所述新软件所需的基础而完成的。

10. 根据权利要求 8 所述的方法，其特征在于：将所述的数据从所述的现行表示法转换为所述的新表示法的所述的步骤包括随着下述附加的步骤而将所有的所述数据一次转换到新表示法：根据所述老软件所需的基础将所述的数据从所述的新表示法再转换为所述的现行表示法。

11. 根据权利要求 6 所述的方法，其特征在于，接着所述的从现有软件传递数据到新软件的步骤，包含附加的步骤：

为响应或者在现有软件内或者在新软件内对半永久性数据每次的初始更新，在另一种软件内更新半永久性数据。

12. 根据权利要求 8 所述的方法，其特征在于，接着将所述数据从现行表示法转换到新表示法的所述的步骤，包含附加的步骤：

为响应或者在现有软件内或者在新软件内对半永久性数据每次的初始更新，在另一种软件内更新半永久性数据。

13. 根据权利要求 3 所述的方法，其特征在于：所述的自动传递步骤的完成是响应一个预选时间阶段的期满，该时间阶段是随着在新软件内处理所有新数据所述步骤的开始而持续的，与此同时现有软件处理现有数据。

14. 根据权利要求 13 所述的方法，其特征在于：在所述的预选时间阶段期满后，仍由老软件处理的所有事务处理都被强制终止。

15. 根据权利要求 13 所述的方法，其特征在于：在所述的预选时间阶段期满后，仍由老软件处理的所有事务处理都被传递给新软件以结束处理。

16. 根据权利要求 13 所述的方法，其特征在于：在所述的预选时间阶段期满后，能够比作为结果而发生的干扰寿命更长的所有事务处理都将被试图传递给新软件进行处理，而所有其它的事务处理将被终止。

17. 一种在电讯交换系统内，在免除系统干扰的重大风险的情况下，在系统连续运行期间，平稳地和自动地从老的呼叫处理软件改变到新的呼叫处理软件的方法，其特征在于采用下列步骤：

在老软件继续处理呼叫的同时，有效地将新软件装入系统内；

通过系统运行大量的测试呼叫并且将所有所述的测试呼叫经路径选择到新软件从而进行处理；

为响应新软件对所述测试呼叫的成功处理，将系统所接收的所有新呼叫经路径选择到新软件；

为响应新软件未能成功地处理所述的测试呼叫，将系统所接收的所有新呼叫经路径选择到老软件。

18. 根据权利要求 17 所述的方法，其特征还在于，在运行和对所有新呼叫进行路径选择之间完成的步骤包括：

为了响应在用老软件继续处理实际呼叫的剩余部分的同时、新软件对所述测试呼叫的成功处理，经路径选择一批所选数量的实际

呼叫到新软件从而得到处理；以及其中

为了响应新软件对所述测试呼叫及所述所选数量的实际呼叫的成功处理，完成将系统所接收的所有新呼叫都经路径选择到新软件的所述步骤。

19. 根据权利要求 17 或 18 所述的方法，其特征在于，也包含下列步骤：

为响应新软件对所有新呼叫的成功处理，从系统中移去老软件并终止老软件处理的所有呼叫。

20. 根据权利要求 17 所述的方法，其特征在于，不但是现有呼叫而且是新呼叫都是在一系列事务处理中得到处理的，现有呼叫和新呼叫两者都既与动态数据相关又与半永久性数据相关，动态数据是在每个事务处理的处理期间所建立和使用的并且在处理结束时消除的数据，半永久性数据是由许多个事务处理的处理过程所使用并在许多个事务处理的处理过程中存在的数据，所述的方法包括附加的步骤：

将数据从老软件传递到新软件。

21. 根据权利要求 20 中所述的方法，其特征在于：只有半永久性数据被从老软件传递到新软件。

22. 根据权利要求 20 中所述的方法，其特征在于，新软件中的数据表示法不同于老软件中的数据表示法，数据从老软件传递到新软件的所述步骤中包含下述步骤：

所述数据从所述老的表示法转换为所述新的表示法。

23. 根据权利要求 22 中所述的方法，其特征在于，将所述数据从所述现行表示法转换为所述新的表示法的所述步骤是根据所述新软件所需的基础而完成的。

24. 根据权利要求 22 中所述的方法，其特征在于，将所述数据从所述现行表示法转换为所述新的表示法的所述步骤包括随着下述

附加的步骤而将所有所述的数据一次转换到所述的新表示法：根据所述的老软件所需的基础将所述的数据从所述的新表示法再转换为所述的现行表示法。

25. 根据权利要求 20 中所述的方法，其特征在于，接着将数据从老软件传递到新软件的所述步骤，包含附加的步骤：

为响应或者在老软件内或者在新软件内对半永久性数据的每次的初始更新，在另一种软件内更新半永久性数据。

26. 根据权利要求 22 所述的方法，其特征在于，接着将所述数据从老的表示法转换为新的表示法的所述步骤，包含附加的步骤：

为响应或者在老软件内或者在新软件内对半永久性数据的每次的初始更新，在另一种软件内更新半永久性数据。

27. 根据权利要求 18 所述的方法，其特征在于，在由系统接收的所有新呼叫经路径选择到新软件之后，所完成的步骤包括：

用新软件处理所有的新呼叫，与此同时老软件处理现有的呼叫。

28. 根据权利要求 27 所述的方法，其特征在于，它包括附加的步骤：

将全部呼叫处理控制传递给新软件，以响应一个预选时间阶段的期满，该时间阶段是随着在新软件内处理所有新呼叫的所述步骤的开始而持续的，与此同时老软件处理现有呼叫。

29. 根据权利要求 28 所述的方法，其特征在于，在所述的预选时间阶段期满之后，仍由老软件处理的所有呼叫将被强制终止。

30. 根据权利要求 28 所述的方法，其特征在于，在所述的预选时间阶段期满之后，仍在被处理的所有呼叫被传递到新软件以结束处理。

31. 根据权利要求 28 所述的方法，其特征在于，在所述的预选时间阶段期满之后，能够比作为结果而发生的干扰寿命更长的所有

呼叫都将被试图传递给新软件进行处理，而所有其它的呼叫将被终止。

32. 一种用于在计算机系统中将数据处理操作从先前装入的第一个软件自动转换到新近装入的第二个软件的装置，其中第一个软件在处理现有数据，与此同时由计算机系统接收新数据，其特征在于，所述的装置包括：

第一个装置，用于传送测试数据到所述的第二个软件从而进行处理，所述的测试数据模仿待由所述第一个软件处理的实际数据；

第二个装置，为响应所述的第二个软件对所述的测试数据的成功处理，用于在所述的第一个软件对所述的现有数据继续处理期间传送所有所述的新数据到所述的第二个软件；

第三个装置，为响应最先发生的由所述第一个软件对所述现有数据处理的完成、或者为响应随着所有所述的新数据传送到所述第二个软件的开始而延续的一个预选的时间阶段的期满，用于停止所述第一个软件在处理数据中的进一步应用，由此从所述第一个软件转换到所述第二个软件可以在计算机系统运行期间自动起作用，从而对数据处理操作的连续进行不会产生重大干扰。

33. 根据权利要求 32 所述的装置，其特征在于，它还包括：

第四个装置，为响应所述第二个软件对所述测试数据的成功处理，用于将实际数据的预先确定的限定样本传送给所述的第二个软件，否则这些样本将由所述的第一个软件处理；其中

所述的用于将所述的所有新数据传送到所述的第二个软件的第二个装置是为了响应由所述的第二个软件对不仅是所述的测试数据而且是所述的实际数据的样本部分的成功处理。

34. 根据权利要求 32 或 33 所述的装置，其特征在于：

所述的计算机系统是一个电讯交换系统；并且

所述的现有数据和新数据是由所述的电讯交换系统所接收的呼

叫。

35. 一种自动实用装置，为了与电讯交换系统一起使用，在该系统中现有呼叫由先前装入的第一个软件进行处理，同时该系统接收新的呼叫，在该交换系统运行期间将呼叫逐渐再引向随后装入的第二个软件从而进行处理而对持续的呼叫处理并没有重大干扰，其特征在于，所述的自动实用装置包括：

第一个装置，可实用于将模仿由所述第一个软件处理的实际呼叫的测试呼叫传送给所述的第二个软件；

第二个装置，可实用于将电讯交换系统所接收的所有新呼叫传递给所述的第二个软件；

第三个装置，用于顺序地操作所述的第一个和第二个装置并且使所述第二个软件依次服从于：

(1) 用从所述的第一个装置传送到所述的第二个软件的测试呼叫进行第一次呼叫处理测试，并且

(2) 随着所述的第二个软件成功地完成所述的第一次呼叫处理测试，用从所述的第二个装置传送到所述的第二个软件的新呼叫进行第二次呼叫处理测试；以及

第四个装置，随着由所述第二个软件成功地完成所述第二次呼叫处理测试，用来将所有的呼叫处理控制从所述的第一个软件传送到所述的第二个软件。

36. 根据权利要求 35 所述的装置，其特征在于还包括：

第五个装置，可实用于将实际呼叫的一批样本传送给所述的第二个软件，这些样本否则将由所述的第一个软件处理；其中

所述的第三个装置顺序地操作所述的第一个、第二个和第五个装置以使所述的第二个软件随着所述的第二个软件对所述第一次呼叫处理测试的成功结束继续服从于一次附加的呼叫处理测试，该测试由所述的第五个装置用传送给所述的第二个软件的所述的实际呼

叫的一批样本进行，其中

所述的第二次呼叫处理测试是随着由所述的第二个软件对所述第一次和附加的呼叫处理测试的成功结束而进行的。

37. 根据权利要求 35 或 36 所述的自动实用装置，其特征在于：

所述的第四个装置是为响应下列情况而操作的：

(1) 由所述的第二个软件成功完成了所述的第二次呼叫处理测试；

(2) 由所述的第一个软件对现有呼叫的处理已结束。

38. 一种在对现有软件处理现有数据没有重大干扰的情况下将计算机装置的数据处理控制自动传递给新软件的系统，为与计算机装置一起使用，同时它用现有软件处理现有数据并且接收待处理的新数据，其特征在于包括：

用于在计算机装置中装入新软件的装置；

采用新软件处理模仿正由现有软件在处理的实际数据的测试数据的装置；

在新软件成功处理了所述测试数据之后将全部数据处理控制传递给新软件的装置。

39. 根据权利要求 38 所述的系统，其特征在于还包括：

为响应新软件对测试数据的成功测试，允许新软件处理实际数据的一部分样本的装置，否则这些数据本来将由老软件处理；

所述的用于将全部数据处理控制传递给新软件的装置在成功处理了所述的测试数据和预定的实际数据的样本部分之后起作用。

40. 根据权利要求 38 或 39 所述的系统，其特征在于还包括：

用新软件处理所有新数据同时现有软件处理现有数据的装置。

41. 根据权利要求 40 所述的系统，其特征在于：

所述的自动传递装置在现有软件完成对现有数据的处理之后启动。

42. 根据权利要求 41 所述的系统，其特征在于：

所述的自动传递装置对新软件的成功处理所有新数据和现有软件结束处理现有数据作出反应。

43. 根据权利要求 38 所述的系统，其特征在于，不但现有数据而且新数据都是在一系列事务处理中得到处理的，现有数据和新数据两者都包括动态数据和半永久性数据，动态数据是在每个事务处理的处理期间所建立和使用的并且在处理结束时消除的数据，半永久性数据是由许多个事务处理的处理过程所使用的并在许多个事务处理的处理过程中存在的数据，所述的系统也包括：

将数据从现有软件传递到新软件的装置。

44. 根据权利要求 43 所述的系统，其特征在于：

只有半永久性数据从现有软件被传递到新软件。

45. 根据权利要求 43 所述的系统，其特征在于，新软件中的数据表示法不同于现存软件中的数据表示法，所述的将数据从现有软件传递到新软件的装置包括：

将所述的数据从所述的现行表示法转换为所述的新表示法的装置。

46. 根据权利要求 45 所述的系统，其特征在于，所述的将所述数据从所述现行表示法转换为所述新表示法的装置包括根据所述老软件所需的基础转换所述数据的装置。

47. 根据权利要求 45 所述的系统，其特征在于，用来将所述数据从所述现行表示法转换为所述新表示法的所述装置包括一次将所有所述数据转换为所述新表示法的装置，以及

根据所述现有软件所需的基础将所述数据从所述新表示法再转换为所述现行表示法的装置。

48. 根据权利要求 43 所述的系统，其特征在于，它还包括：

为响应或者在现有软件内或者在新软件内对半永久性数据每次

的初始更新，在另一种软件内更新半永久性数据的装置。

49. 根据权利要求 45 所述的系统，其特征在于，它还包括：

为响应或者在现有软件内或者在新软件内对半永久性数据每次的初始更新，在另一种软件内更新半永久性数据的装置。

50. 根据权利要求 40 所述的系统，其特征在于：

所述的自动传递装置是为了响应随着所述的用新软件处理所有新数据同时现有软件处理现有数据的开始而延续的一个预选时间阶段的期满。

51. 根据权利要求 50 所述的系统，其特征在于，在所述的预选时间阶段期满之后，仍由老软件进行处理的所有事务处理都被强制终止。

52. 根据权利要求 50 所述的系统，其特征在于，在所述的预选时间阶段期满之后，仍由老软件进行处理的所有事务处理都被传送给新软件以结束处理。

53. 根据权利要求 50 所述的系统，其特征在于，在所述的预选时间阶段期满之后，能够比作为结果而发生的干扰寿命更长的所有事务处理都将被试图传递给新软件进行处理，而所有其它的事务处理都被终止。

54. 一种在电讯交换系统中从老的呼叫处理软件平稳而自动地改变到新的呼叫处理软件的系统，在连续操作期间进行改变而可免除系统干扰的重大风险，其特征在于包括：

用于在老软件继续处理呼叫的同时，有效地将新软件装入系统内的装置；

用于通过系统运行大量的测试呼叫并且将所有所述的测试呼叫经路径选择到新软件从而进行处理的装置；

用于为响应新软件对所述测试呼叫的成功处理，将系统所接收的所有新呼叫经路径选择到新软件的装置；以及

用于为响应新软件对所有新呼叫的成功处理，从系统中移去老软件并终止由老软件处理的所有呼叫的装置。

55. 根据权利要求 54 所述的系统，其特征在于还包括：

用于为了响应在用老软件继续处理实际呼叫的剩余部分的同时、新软件对所述测试呼叫的成功处理，经路径选择一批所选数量的实际呼叫到新软件从而得到处理的装置；以及其中

所述的为了响应新软件对所述测试呼叫及所述所选数量的实际呼叫的成功处理，将系统所接收的所有新呼叫都经路径选择到新软件的装置。

56. 根据权利要求 54 所述的系统，其特征在于，不但是现有呼叫而且是新呼叫都是在一系列事务处理中得到处理的，现有呼叫和新呼叫两者都既与动态数据相关又与半永久性数据相关，动态数据是在每个事务处理的处理期间所建立和使用的并且在处理结束时消除的数据，半永久性数据是由许多个事务处理的处理过程所使用并在许多个事务处理的处理过程中存在的数据，所述的系统也包括：

将数据从老软件传递到新软件的装置。

57. 根据权利要求 56 所述的系统，其特征在于：

只有半永久性数据被从老软件传递到新软件中。

58. 根据权利要求 56 所述的系统，其特征在于，在新软件中的数据表示法不同于在老软件中的数据表示法，并且所述的用于将数据从老软件传递到新软件的装置包括：

用来将所述数据从所述老的表示法转换为新的表示法的装置。

59. 根据权利要求 58 所述的系统，其特征在于，所述的用来将所述数据从所述老的表示法转换为新的表示法的装置包括：

用来根据所述新软件所需的基础将所述数据从所述老表示法转换为所述新表示法的装置。

60. 根据权利要求 58 所述的系统，其特征在于，用来将所述数

据从所述老表示法转换为所述新表示法的所述装置包括用来一次将所有所述数据转换为所述新表示法的装置，也包括：

用来根据所述老软件所需的基础将所述数据从所述新表示法再转换到所述老表示法的装置。

61. 根据权利要求56所述的系统，其特征在于，还包括：

为响应或者在老软件内或者在新软件内对半永久性数据每次的初始更新，在另一种软件内更新半永久性数据的装置。

62. 根据权利要求58所述的系统，其特征在于，还包括：

为响应或者在老软件内或者在新软件内对半永久性数据每次的初始更新，在另一种软件内更新半永久性数据的装置。

63. 一种在电讯交换系统中从老的呼叫处理软件平稳而自动地改变到新的呼叫处理软件的方法，在连续操作期间进行改变，其特征在于包含下列步骤：

在老软件连续处理呼叫的同时，有效地将新软件装入系统内；

通过系统运行大量的测试呼叫并且将所有所述的测试呼叫经路径选择到新软件从而进行处理，而不停止到老软件的所有实际的电讯业务；

为响应新软件对所述测试呼叫的成功处理，将系统所接收的所有新呼叫经路径选择到新软件；以及

为响应新软件对所有新呼叫的成功处理，以及首先出现的由老软件处理的所有呼叫的终止或随着由系统所接收的所有新呼叫经路径选择到新软件而延续的一个预选时间阶段的期满，从系统中移去所有老软件。

64. 根据权利要求63所述的方法，其特征在于，所述的通过系统运行大量的测试呼叫并且将所有所述的测试呼叫经路径选择到新软件从而进行处理包含下列步骤：

仅将模仿的呼叫经路径选择到新软件以便处理；以及

在所述的模仿的呼叫之外，将新呼叫的许多样本经路径选择到新软件以便处理，以响应新软件对所述模仿呼叫的成功处理。

65. 根据权利要求 63 所述的方法，其特征在于，不但是现有呼叫而且是新呼叫都是在一系列事务处理中得到处理的，现有呼叫和新呼叫两者都既与动态数据有关又与半永久性数据有关，动态数据是在每个事务处理的处理期间所建立和使用的并且在处理结束时消除的数据，半永久性数据是由许多个事务处理的处理过程所使用并在许多个事务处理的处理中存在的数据，所述的方法包括附加的步骤：

将数据从老软件传递到新软件。

66. 根据权利要求 65 所述的方法，其特征在于：

只有半永久性数据被从老软件传递到新软件中。

67. 根据权利要求 65 所述的方法，其特征在于，在新软件中的数据表示法不同于在老软件中的数据表示法，并且所述的将数据从老软件传递到新软件的步骤包括下述步骤：

将所述数据从所述老的表示法转换为所述新的表示法。

68. 根据权利要求 67 所述的方法，其特征在于，将所述数据从所述老表示法转换为新表示法的所述步骤是根据所述新软件所需的基础来完成的。

69. 根据权利要求 67 所述的方法，其特征在于，将所述数据从所述老表示法转换为新表示法的所述步骤包括一次将所有所述的数据转换为所述的新表示法，与附加的步骤同时进行：

根据所述老软件所需的基础将所述数据从所述新表示法再转换到所述的现行表示法。

70. 根据权利要求 65 所述的方法，其特征在于，遵循将数据从老软件传递到新软件的所述步骤，包括附加的步骤：

为响应或者在老软件内或者在新软件内对半永久性数据每次的

初始更新，在另一种软件内更新半永久性数据。

71. 根据权利要求 67 所述的方法，其特征在于，遵循将所述数据从老表示法转换到新表示法的所述步骤，包括附加的步骤：

为响应或者在老软件内或者在新软件内对半永久性数据每次的初始更新，在另一种软件内更新半永久性数据。

72. 根据权利要求 63 所述的方法，其特征在于，在所述的预选时间阶段期满之后，仍由老软件处理的所有事务处理都被强制终止。

73. 根据权利要求 63 所述的方法，其特征在于，在所述的预选时间阶段期满之后，仍由老软件处理的所有事务处理都被传送给新软件以结束处理。

74. 根据权利要求 63 所述的方法，其特征在于，在所述的预选时间阶段期满之后，能够比作为结果而发生的干扰寿命更长的所有事务处理都将被试图传递给新软件进行处理，而所有其它的事务处理都被终止。

75. 一种在电讯交换系统中从老的呼叫处理软件平稳而自动地改变到新的呼叫处理软件的系统，在连续操作期间进行改变，其特征在于包括：

用于在老软件继续处理呼叫的同时，有效地将新软件装入系统内的装置；

用于通过系统运行大量的测试呼叫并且将所有所述的测试呼叫经路径选择到新软件从而进行处理，而不停止到老软件的所有实际的电讯业务的装置；

用于为响应新软件对所述测试呼叫的成功处理，将系统所接收的所有新呼叫经路径选择到新软件的装置；以及

用于为响应新软件对所有新呼叫的成功处理，从系统中移去老软件并终止或传送由老软件处理的所有呼叫的装置。

76. 根据权利要求75所述的系统，其特征在于，所述用于通过系统运行大量的测试呼叫并且将所有所述的测试呼叫经路径选择到新软件从而进行处理，而不停止到老软件的所有实际的电讯业务的装置还包括：

用于仅将模仿的呼叫经路径选择到新软件以便进行处理，与此同时所有实际的呼叫继续由老软件进行处理的装置；以及

用于既将模仿的呼叫也将所选数目的新呼叫经路径选择到新软件以便进行处理的装置，以响应新软件对所述模仿呼叫的成功处理，与此同时新呼叫的剩余部分继续由老软件进行处理。

77. 一种用于在计算机系统中将数据处理操作从先前装入的第一个软件自动转换到新近装入的第二个软件的方法，其中第一个软件在处理现有数据，与此同时由计算机系统接收新数据，其特征在于，所述的方法包含下列步骤：

传送测试数据到所述的第二个软件从而进行处理，所述的测试数据模仿待由所述第一个软件处理的实际数据；

为响应所述的第二个软件对所述的测试数据的成功处理，在所述的第一个软件对所述的现有数据继续处理期间传送所有所述的新数据到所述的第二个软件；以及

为响应由所述第一个软件对所述现有数据处理的完成，经路径选择将所有其它数据引向所述第二个软件以便处理，在计算机系统运行期间，在对数据处理的连续操作不会产生重大干扰的情况下，对从所述第一个软件转换到所述第二个软件自动起作用。

78. 根据权利要求77所述的方法，其特征在于，在所述的传送测试数据和传送所有的所述新数据的步骤之间，包括附加的步骤：

为响应所述第二个软件对所述测试数据的成功处理，将实际数据的预先确定的限定样本传递给所述的第二个软件，否则这些样本将由所述的第一个软件处理；以及其中

在完成所述软件对所述现有数据的连续处理期间，为响应由所述的第二个软件对不仅是所述的测试数据而且是所述的实际数据的限定样本部分的成功处理，将所述的所有新数据传送到所述的第二个软件的所述步骤。

79. 根据权利要求 77 或 78 所述的方法，其特征在于：

所述的计算机系统是一个电讯交换系统；并且

所述的现有数据和新数据是由所述电讯交换系统所接收的呼叫。

80. 一种为了与电讯交换系统一起使用，在该系统中现有呼叫由先前装入的第一个软件进行处理，同时该系统接收新的呼叫，在该交换系统运行期间将呼叫逐渐再引向随后装入的第二个软件从而进行处理而对持续的呼叫处理并没有重大干扰的方法，其特征在于包括：

将模仿由第一个软件处理的实际呼叫的测试呼叫传送给所述的第二个软件；

将电讯交换系统所接收的所有新呼叫传送给所述的第二个软件；

顺序地完成所述的第一步和第二步并且使所述第二个软件依次服从于：

(1) 用在第一步中传送到所述的第二个软件的测试呼叫进行第一次呼叫处理测试，并且

(2) 随着所述的第二个软件成功地完成所述的第一次呼叫处理测试，在所述的第二步中用传送到所述的第二个软件的新呼叫进行第二次呼叫处理测试；以及

随着由所述的第二个软件成功地完成所述的第二次呼叫处理测试，将所有的呼叫处理控制从所述的第一个软件传送到所述的第二个软件。

81. 根据权利要求80所述的方法，其特征在于还包括附加的步骤：

将实际呼叫的一批样本送给所述的第二个软件，这些样本否则将由所述的第一个软件处理；其中

所述的第一步，第二步和附加的步骤依序完成，以使所述的第二个软件随着它对所述第一次呼叫处理测试的成功结束，并在所述的第二次呼叫处理测试之前继续服从于一次附加的呼叫处理测试，该测试由所述的附加步骤用传送给所述的第二个软件的所述的实际呼叫的一批样本进行，其中

所述的第二次呼叫处理测试是随着由所述的第二个软件对所述的第一次和所述的附加呼叫处理测试的成功结束而进行的。

82. 根据权利要求80或81所述的方法，其特征在于：

所述的将所有的呼叫处理控制从所述第一个软件传送到所述第二个软件的步骤是随着下列情况的出现而执行的：

(1) 由所述的第二个软件成功完成了所述的第二次呼叫处理测试；以及

(2) 由所述的第一个软件对现有呼叫的处理已结束。

83. 根据权利要求80或81所述的方法，其特征在于：

所述的将所有的呼叫处理控制从所述第一个软件传送到所述第二个软件的步骤是随着下列情况的出现而执行的：

(1) 由所述的第二个软件成功完成了所述的第二次呼叫处理测试；以及

(2) 随着电讯交换系统所接收的所有新呼叫传送给所述第二个软件的一个预选时间阶段的期满。

84. 根据权利要求83所述的方法，其特征在于，在所述的预选时间阶段期满之后，仍由所述第一个软件处理的所有事务处理都被强制终止。

85. 根据权利要求 83 所述的方法，其特征在于，在所述的预选时间阶段期满之后，仍由所述第一个软件处理的所有事务处理都被传送给第二个软件以结束处理。

86. 根据权利要求 83 所述的方法，其特征在于，在所述的预选时间阶段期满之后，能够比作为结果而发生的干扰寿命更长的所有事务处理都将被试图传送给所述的第二个软件进行处理，而所有其它的事务处理都被终止。

87. 一种为了将运行着的软件系统内的一系列事件动态定向到所述的第一个或第二个应用程序中的一个或另一个，通过提供一组定向点，将分别分布在第一个和第二个软件应用程序中的第一个和第二个模块动态汇集的方法，其特征在于包括下列步骤：

分析由函数名寻址的消息；

将这些消息引向在所述的第一个或第二个模块的每一个中的进程；并

用动态运行时间汇集将进程的执行引向在所述的第一个或所述的第二个软件模块之一中的所述进程的有选择地连续执行。

88. 一种为了将运行着的软件系统内的一系列事件动态定向到所述的第一个和第二个应用程序中的一个或另一个，通过提供一组定向点，将分别分布在第一个和第二个软件应用程序中的第一个和第二个模块动态汇集的系统，其特征在于包括：

用于分析由函数名寻址的消息的装置；

用于将这些消息引向在所述的第一个或第二个模块的每一个中的进程的装置；以及

用于用动态运行时间汇集将进程的执行引向在所述的第一个或所述的第二个软件模块中的所述进程连续执行的装置。

说　明　书

用于计算机运行期间改变软件的系统

本专利文献的公开的部分含有受版权保护的材料。该版权所有人不反对在专利文献或专利公开（当其出现在专利和商标局中时）、专利存档文件或记录的任何一项中出现其摹真复制品，但在其它情况下保留所有的版权权利。

本发明涉及到软件的修改，尤其是涉及到在一个运行中的计算机系统中软件的替换。

计算机软件的一项特征就在于，它必须用修改、增加和/ 或删除的方式周期性地更新，以便持续地给用户提供适当的功能、使软件尽可能完善并纠正在软件生命期中所产生的错误和差异。当一个新的性能附加给了软件时，希望尽可能早和尽可能容易地替换旧软件、以使用户得到新软件的性能。

在特定类型的计算系统中，如独立或批处理系统，将软件从一个版本改为另一个版本面临的障碍较少。典型的是，在一天中用机活动较少时维修人员就可以关机。然后将老的软件就简单地移去，代之以软件的较新版本。此后，该计算系统重新启动，所有将来的数据处理都由软件新版本来完成。当然，这一过程假设了新软件已在脱机系统上经过适当的检测和调试达到了这样的程度，即软件人员和运行管理部门确信该软件将适当地实现为其设计的功能、而不会产生要求停机然后重新启动整个计算系统的不适当的中断。

在其它类型的计算系统中，例如现代的存储程序控制(SPC）电信交换系统（在该产业中通常称之为“交换机”），系统中软件新版本的检测和软件的改变都不象在独立批处理系统中那样容易。例如，

离开了实际的操作处理呼叫，软件的新版本就不能得到有效的测试。软件必须在操作过程中测试，以便确定它在实际的操作条件下是否将发挥适当的功能、确定它作为一个运行的SPC交换系统的一部分是否能与所有其它的软件模块适当衔接。此外，电信交换系统事实上永远不会停止运行。理想情况下，这些系统将永远不停地运转而不中断，因为在一个社区内要求通信设施连续服务。即，通过该系统所处理的电信业务是一个连续流，即使在白天或晚间的空闲时间也是同样，并且该交换操作中的任何中断都将导致该电信业务的混乱。这样一种混乱可能对该系统的运行及其有效性，以及系统在用户或顾客中的接受程度造成极大的损害。

电信交换的这些实时要求在两方面提出了严格的要求：在含有新的或改进的功能的软件提高版本或其若干部分的测试方面，以及在交换中包含错误纠正或“错误修改”而不影响由交换正在处理的电信业务的软件的替换方面。然而，用传统的“编辑—编译—链接—装入—运行”方式把软件成分或单元的新版本集成并入该系统的方法并不是令人满意的。人们宁愿选择这样一种方法，即在系统运行过程中修改或扩展软件的能力，而不需要停机。

人们为解决与把新软件并入运行中的计算机系统有关的问题进行了尝试。例如，当今使用的一些并不以独立或批处理方式运行的先进的连机运行系统是用与独立或批处理系统中显著不同的方式解决替换旧软件的问题。然而，尽管比独立系统中的更透明，这些系统仍然是人工地替换软件，即要求各个用户或用户组积极地选择是否使用软件的新版本或修改版本来进行处理。这一选择可以由用户通过修改在其各自的用户标识符之下运行的过程所利用的软件的相互联系来实施。这一选择在一段时间内—通常以数周或数月来计算—保留给用户，在这段时间里经过在每一个先前的层次上的成功运行而没有任何不一致之后软件在相互联系的结构中向上迁移若干层

次。到达该相互联系的顶层时，就宣告该软件是“可运行的”，同时该系统的用户就不再使用较旧的版本。新软件插入该系统，以及在不同层次的迁移，是由一个配置管理过程控制的，该过程是一个报告、批准、跟踪每一个层次上的软件版本并且实施已批准的改变的人工过程。

正如采用在批处理或独立系统上更新软件的方法那样，以这种方式使新的或修改的软件并入系统具有人所共知的弊端。这在很大程度上是一个人工的、工作量大的系统，复杂且费时。它把何时何种情况下系统应使用某种新软件运行的控制留给了用户，而不是实施逐步的、限定的、联机的使用新软件，以至于造成错误迅速的扩散或直接影响所有后续的操作。对新的或已修改的软件的访问进行控制的方法是直接连接到并且限制在执行该软件的单个用户。

用于解决至少某些与在运行的计算机系统中的软件更新有关的问题的其它尝试采用了一种不同的方式。如，在一项美国专利中，包含了 Anders Abrahamsson 和 Lars Holmqvist 发明并转让给 L. M. 埃利克逊电话股份有限公司(Telefonaktiebolagat L M Ericsson)的技术，其中透露了一个在运行期间动态连接软件的系统。然而，该系统涉及到一个间接寻址的复杂系统，它需要用到一个或是特殊的或是扩展的操作系统，以及/ 或者一个特别的编译程序。该系统也有若干弊端，包括需要一个非标准的操作系统。此外，该系统不能与标准应用软件一起工作。该系统的局限性还在于，它仅仅涉及全部问题的一个部分，而且对于现有的与修改的软件模块之间控制的渐进测试与改变方面并未提供帮助。

在当今使用的典型的电信系统中，改变软件或软件的若干部分的问题甚至更严峻。虽然这样的系统不能恰当地称为批处理或独立系统，但无论何时只要进行了软件的改变系统的操作就会受到影响。装入新软件，从属于旧软件的数据被转移并传送给新软件。在传

送进行之中的这段时间，系统不能登记任何新的呼叫。失去功能的这一阶段可以持续一小时或更长时间，这样就必需在操作的非峰荷时间安排软件的变更。即使这样，在一个电信交换系统中一个小时的故障时间也是相当长和代价巨大的，因为在这段时间里不能处理任何新的呼叫，也不能为应急通信的任何要求提供服务。

因此，在电信产业中，在电信交换系统的实际操作期间，在不影响通过该系统正在进行的电信业务的情况下，如果能够测试和改变软件将是极为有用的。如果有能力通过新软件或软件的新的组成部分指挥一个限定的和指定部分的通讯业务，从而使该软件能够在一个运行环境中优先于处理正在使用的数据而被测试，这样对电信产业将更为有益。这样，在不需要特别的编译程序的系统运行期间改变软件的一种平稳的、透明的方法就是令人高度期待的。本发明的系统提供了这样一种方法。

SPC 电信交换系统这样的计算系统的动态特征，从本质上可以描述为一系列平行的、相对独立的事务处理（也称为“线索”或“事件链”），其中每一个事务处理由若干相关活动所组成。一个事务处理一般完成对系统的外部用户来说是可见的和具有功能用途的一项工作。在电信交换系统中一个典型的事务处理可以是一次呼叫。

采用本发明的平稳软件改变技术的连机软件替代利用了面向软件的事务处理，以及在同一时间内既能存贮老版本也能存贮新版本的存储器。转到软件新版本的一个平稳的改变是用软件老版本使正在进行的事务处理，即“老业务”，运行结束来实现的。在软件改变已经开始之后启动的事务处理，即“新业务”，将用软件的新版本以一种渐进的和受控制的方式运行。

能为本发明的平稳的软件改变技术所满足的首要需求包括使用户的干扰最小或者消失以及高水平的系统可用性。本发明的首要特征包括下列事实：(1)在一个事务处理（如，一次呼叫）期间系统的

一个单独的用户所经历的干扰最小或者消失，因为一个并且只有一个软件版本控制着每一个特定的事务处理，即，从事务处理的开始到结束，系统或者使用软件的老版本或者使用软件的新版本；(2)系统的一个单独的用户没有系统不可用性的经历，因为作了软件改变，在此项改变期间软件的两种版本是平行地使用的。如果后一特征没有出现，一个简单的解决办法就是对系统软件作一个简单的脱机改变。

将由系统处理的数据可以根据前后关系分成两种不同的类别：(1)动态数据，它是在一个事务处理期间所产生和使用的并且在该事务处理结束后就被消除；(2) 半永久性数据，这是由若干个事务处理所使用的并且生存于若干事务处理期间，如在电信系统中，有关与该系统相联的用户数目的数据或由特定用户们使用的短缺数目的数据。

与要求最小干扰的连机软件替换相联系的一个决定性的问题在于，软件老版本的状态必须转移为软件新版本。根据本发明，采用平稳的软件修改技术，动态数据与半永久性数据的分离使这一问题变得简单，因为只有半永久性数据—如果存在的话—需要从软件老版本传送到软件的新版本。此外，半永久性数据一般是作为一个单独的数据库中的对象提供的，并且比一个电信软件系统中的其它部分有更难得的改变。

本发明的系统为将新软件装入电信系统的存贮器中（与老软件在一起，并且除了新软件还有老软件）作了准备。在本发明的系统中，系统中现存的业务开始是由老软件处理，测试业务是通过开关按预定的路线由新软件处理的。然后，如果测试业务由新软件进行了成功的处理，实际正在进行的业务（或正常业务）的一部分就有选择地由新软件按预定路线处理，同时该实际正在进行的业务的其余部分仍由老软件进行处理。由新软件处理的正在进行的取样业务

的百分比可以在零到百分之一百之间变化。该取样业务应当由新软件适当地执行，所有业务都被引向新软件。一旦由老软件所进行的所有呼叫的处理已经结束，系统就不再用到老软件，可以从系统中将其消去。

另一方面，本发明的系统包含了一个对新的或修改的软件进行平稳验证的系统。本发明的系统允许按一种渐进的和受控制的方式使数据通过新软件而流动，但是是作为实际运行的该系统的组成部分。该系统在对电信交换系统的实际运行只有很小的或没有影响的情况下提供了错误和差异的早期探测，因为流向新软件的初始数据仅仅是由该系统产生的测试数据。在处理测试数据时如果电信系统探测到一个错误，就没有更多的业务引向新软件，这样，即使该新软件已经在处理实际的数据了，对系统整个业务的干扰也被最小化了。

本发明的系统的另一方面是，该系统以渐进的方式使业务从老软件引向新软件。本发明的系统包含了允许这样进行事务处理的能力，即用老软件开始处理，到结束处理时仍用老软件。只有随装入新软件后开始的事务处理将由新软件进行处理。本发明的系统的这一特征使得在由给定的老软件到用新软件替代它或扩展它的这段过渡时期对用户仅有很小的干扰。此外，这一特征使要求转换到和/或传送到不同的一组软件的数据量比一开始所处理的数据量减到最小。

还有另一个特征，本发明的系统包含了软件的平稳修改，这是一个过程，它把软件的操作模型看作是一系列可辨认的和可维护的事务处理。在本发明的系统中，这样一个链在整个处理过程中始终被辨认并维护。此外，在这一特征方面，本发明的系统在软件的新、老版本同时驻留在电信交换系统中的这段时间内，控制着将每个事务处理连接到老版本或者是新版本上。

本发明的系统的另一特征是，它具有这样的机制，即把包含在老软件中且由老软件直接控制的可能存在的半永久性数据转换并移动到新软件中。

本发明的系统的另一特征是，它包含一系列的方向指针，可用于在运行着的系统内将事务处理动态地引向系统的老版本或者是新版本。该系统通过许多途径包括由函数名称寻址的消息的分析和运行期间的动态连接实现软件部件的动态定向。

本发明的系统的另一特征是，它包含一种即时修改方法。此方法用于软件的两种版本不可能同时存在的时候，并且提供从软件的老版本到软件新版本的瞬时改变。所有的业务都自动地被引向新版本，除非探测到在新版本中存在着使用它进行操作是不可能的或不实际的这样一些错误。在这种关头，系统如果停机就可以通过一个新的即时修改返回到采用软件旧版本处理所有业务。在此特征下系统将老版本以被动状态保留在系统内。

本发明的系统还有另一个特征，它包括一个连接过程（调用）的机制，它包含了一个商人和一个核心使不同类别的软件部件之间的接口能起作用。这一连接过程调用机制在运行期间也用于实现内部连接和新旧软件单元的汇集。在使用本发明的系统中的这一连接过程调用机制方面，利用本发明的系统的另一个特征建立所需要的接口说明，使用了一种特殊的面向目标的接口描述语言并称为ELIN。该语言含有适于为本发明的系统的连接过程调用特征开发接口的一种特殊构造。

本发明的系统还有另一个特征，它含有用一个函数名地址对消息寻址的机制，该函数名地址包含了一个商人，或者是使可被分布在该系统中的软件单元之间的接口能起作用的消息路径选择机制。在运行期间这一机制也用来将消息引向旧软件或新软件。在使用本发明的系统的这一寻址机制方面，利用本发明的系统的另一个特征

建立所需要的接口说明，这种特殊的语言如上所述称为ELIN。该语言含有适于为本发明的系统的消息协议方面开发接口的一种特殊构造。

如同很容易被具有此特殊技术中的普通技巧的人们所欣赏那样，此项发明的原则和特征也可以在除了电信交换系统之外的各种计算机应用中用来帮助进行软件在运行期间的转换。

为理解本发明并为了进一步了解本发明的目标及其优点，现在可以提交如下的描述，连同相关的图示，在这些图示中：

图 1A—1B 是将新的或修改过的软件引入一个运行的软件系统的现有的技术系统的图示说明；

图 2 是一个框图，说明在本发明的系统中将处理从一个老的软件单元重新引向一个新的软件单元的示例过程；

图 3A—3E 是根据本发明的系统从老软件改变到新软件的处理的图示说明；

图 4 是一个流程图，说明根据本发明的系统在运行期间改变软件的处理；

图 5 是一个框图，说明在本发明的系统中对新老两种软件有选择地寻址的方法；

图 6 是一个框图，说明在本发明的系统的软件内对目标寻址的方法；

图 7 是一个框图，说明在本发明的系统内寻访软件的方法；

图 8 是一个框图，说明在本发明的系统中商人寻访软件的方法；

图 9 是一个面向目标的接口描述语言用来实现本发明的系统的方法的示意图；

图 10 是本发明的系统的特定方面的示意图；

图 11 是使本发明的系统内给定的协议容易接口的方法的示意

图。

本发明的系统在某些方面采用了面向目标程序设计的原则。面向目标程序设计实质涉及四个元素—类，目标，实例变量（或象在C++ 中所实现的数据成员），以及方法（或在C++ 中的成员函数）。一个类简单说来就是用以定义目标的模板，这些目标就是它们从其中产生的那个类的实例。类有两种类型的组成成份—实例变量和方法。实例变量用作数据元素而方法用作函数，即它们定义了一个目标的性质。其中每一个结合成运行中的模块上的一个简单普通的目标。因此，程序设计是围绕各种目标完成的，而不是围绕待完成的函数和任务而完成的。

在此项技术中众所周知的、面向目标程序设计的给定技术已用程序设计语言C++，以所提出的本发明的系统的实现方式，在本发明的系统中得以具体化。这些技术包括继承、同素异构和封闭。继承可以从一个现存的类中推出一个新类，从而使代码可以重复使用，因而数据和代码可以加到一个类中，或者可以在没有改变现存类的情况下改变一个类的性质。同素异构是这样一种特性，它提供改变由不同目标分享的一个成分的性质的能力，允许成分或目标的不同实例有不同的表现，尽管它们看来是完全相同的。最后，封闭是一种将数据与处理这些数据所必需的操作全都结合在一个“屋顶”下。此外它还允许能够保护数据使之避免多余的或不必要的访问并将数据组织的细节隐蔽起来。

首先参考图1A，这里阐明了在先前的技术系统中采用的有关将新的或修改过的软件引入一个运行着的软件系统的管理的软件控制方案。图1A 阐明了软件的等级层次序列，其中每一层的内容是由一个考察板(board) 的成员所控制的。对软件的所有改变在系统中实现之前必须由该板批准。在考察板作出一个正式的决定（即需要一个软件，它已经经过适当的测试并且它不象会引起危险或使系统

中断运行）之前，软件是不会被并入该系统的。

该等级层次可由连接在一起的几个单独的等级所组成，由已经访问过或者需要这些层次或软件的“库”以实现他或她的功能的一个个别的用户把这些等级连接在一起。在等级1的顶端是实时、运行的软件，它们一般使用最广泛且控制最严格（“AB.D”）。在该层次之下是一个改变库2，在后缀中附加字母C来标示（“AB.DC”）。在该等级内运行的软件的较低层次可以从属于系统内用户的不同群体，并将由这些层次上的考察板进行控制。得到批准之后，新的或修改的实时软件进入系统，到最低的适合于改变的层次，即象在2和3那样的以字母C结尾的层次。

一旦新的或修改过的软件进入系统，它就保留在所进入的层次上，直到经过一个特定阶段并且软件已不产生出可探测到的错误为止。然后它将迁移到下一个最高层次上。有些情况下，这将需要一个特定考察板的预先批准；另一些情况下，迁移将自动产生，作为系统活动的有规律安排中的一部分。迁移对用户是透明的，迁移或进入该等级的软件对于用户将是立即有用的，这些用户是已经将其软件与包含新的或改变的软件的库中的软件的联系结构化的。

正如图1A也阐明的，对于驻留在同一系统中的非实时工程型软件而言，相同的处理可以重复并同时发生。这种情况下仅有的差别在于，控制处理是由不同的一批人进行管理的，而且该处理可以不象一般在整个系统所用的运行的软件对紧要过程处理那样精密。然而，就这种工程软件来说软件的并入是以与运行的软件同样的方式而发生的。新的或修改过的软件进入以C作为后缀中的最后一个字母的最低的适于改变的层次的那个等级，如在4。然后它在取得必要的批准的情况下，经过一段时间按向上方向迁移，直至到达该等级的其所在部分的顶端5。在或是工程软件或是运行软件的情况下，一旦它已经迁移到下一个层次它就不再驻留在较低的层次上。

是否利用进入到系统的等级库中的新的或修改过的软件留给单个的用户或用户组来决定。用户可以选择系统将用来把软件联在一起以便使用的库的层次。他们可以选择绕过软件的全部较低层次，或者他们可以简单地选择避免最低的改变层次，该层包含有最新的和地位最低的已测试的软件。当然，等级中每一部分的最高层次含有大批的在使用的运行软件。

图 1B 阐明了把配置控制强加给图 1A 中的软件库的等级上、以便维持对基准和根据日常根据引入该系统的新的或修改过的软件的控制的人工处理过程。如上所述，新软件根据考察板的批准而进入等级制度中最低的适于改变的层次。如果新软件导致出错和差异，该软件就从等级体系中移走并且如 6 所示返回去进行额外的软件维护工作。一旦问题已被纠正且软件已被重新测试，就可以根据考察板的批准，再试一次，把该软件并入系统的最低的适于改变的层次。如果在所允许的固定的阶段内没有探测到问题，软件将自动迁移到下一个层次，除非下一层需要如 7 所示的另一次考察板的批准。否则，在得到适当的批准之后它将根据一个确定的安排而迁移。这一过程将连续重复，直到软件到达等级体系的那一部分里的最高层次，此时它将完全被宣布为运行软件。

下面参见图 2,这里阐明了本发明的一个方面，描述了可以对软件进行一种平稳的修改或改变的机制。这种平稳的修改特性是这样一种性质，即在一个特定的时间阶段内，允许系统将软件的老版本和新版本同时存储在主存储器内。新业务然后就可以逐渐地引向软件的新版本，随着这种引导的进行，就使得老的事务处理由老的软件版本执行到结束，同时新的事务处理就由新的软件版本执行。在图 2 中给出了一个未加改变的软件单元 11 并通过一个称为定向点的寻址机构 12 与一个老的修改单元 13 和一个新的修改单元 14 结合起来。未加改变的接口 15 和 16 将老的修改单元 13 和新的修改单

元 14 与寻址机构 12 连在一起。

在本发明的系统中一次仅替换软件的一个部分是完全可能的并且事实上是典型的情况。将被替换的软件称作为一个修改单元。图 2 给出了这样一个情况，有一个修改单元 R，它既在软件老版本即老的修改单元 13 中，也在软件新版本即新的修改单元 R'14 中。新的修改单元 R' 通过定义选择有了一个接口 16，它是与连接未加改变的软件 11 的现存接口相一致的。这意味着未加改变的软件是能够既与老版本软件配合也能够与新版本软件（修改单元）配合。

本发明的这一特征，即提供了事务处理的动态定向或重新定向，通过定向点的引导和使用而变得容易。这些定向点由分布式系统中的这样一些区段所组成，在这些区段事务处理可用一种特别的方式定向。如图 2 中所示的寻址机构 12 表示定向点的实现以及系统的事务处理被各自地引向软件的新版本或老版本的途径。这些定向点可以按三种方式运行。首先，可以通过分析由系统处理的业务相关联的函数名来触发。根据这一操作方法，业务可以被引导到要求完成必要处理的特定函数的软件新版本或是老版本。其次，事务处理可以被引导去执行一个程序的软件新版本或老版本，该程序是以作为运行时间软件连接的结果而提供的信息为基础的。

一个软件部分的多种版本在系统内一段特定时间可以共存这样一个事实具有一系列后果。例如，本发明的这种平稳修改特征要求，万一改变半永久性数据表示法，软件的不论是新版本还是老版本都能够访问适当的数据表示法。

下一个参见图 3A—3E，这里阐明了一个修改单元在修改的一般情况下可以经历的各阶段。在系统的实际运行中，一个特定的修改单元看起来将只经历图示各阶段的一个子集合。此外，在实际操作中，图示各阶段并不是将要完成的系列操作的一个精确的集合。更确切地说，一个或多个阶段在修改的过程中可以重复。本发明这

一方面的一个重要特征在于，修改过程各阶段的控制对系统的用户是透明的，对应用软件程序本身也几乎是透明的。图示的各阶段是由一个在系统内操作各种现存的定向点的协调装置控制的。

图 3A 表示系统 21 含有一个体现于其中的修改单元 R22。这代表了在修改过程开始时系统的状态。此时，系统将所有的常规业务引向软件老版本。图 3B 表示通过与数据修改信息 24 结合起来的新版本 R'23 对修改单元 22 的修改。图 3A 阐明了修改开始的状态，而图 3B 则代表了初始或装入阶段，此时新版本 R'23、以及某些情况下在数据修改信息 24 中所包含的新数据方案被连同老的修改单元 22 一起装入。数据修改信息 24 在软件开发系统中已被说明，基于已知知道老版本和新版本的数据表示法这样一个事实。

图 3C 用图描述了修改处理的数据阶段的改变。这一阶段的目标是，在适当的时刻，移动可能存在的半永久性数据的相关部分，如上所述，这些由老的软件部分包含并直接控制的数据移入新的软件部分，即用来取代老的软件部分，以避免不可接受的干扰。本发明这一特征的实现是通过完成几个不同的活动来管理的。首先，活动集中在数据修改阶段，包括：(a)数据从旧表示法到新表示法的转换，其结果是替换的新软件部分中的数据表示法比在老软件部分中所用的数据表示法已经作了改变；(b)数据从老的软件部分向新的软件部分的传送。其次，在测试阶段和完成阶段即新老软件平行使用的这些阶段延续的活动，包含对或者是新软件部分或者是老软件部分中的半永久性数据的每一次“初始的”更新，这里在另一个软件部分中随后就有一个相应的半永久性数据的更新。即，如果初始的更新是对新软件而进行的，那么随后的更新就是在老软件中进行的，反之亦然。这就意味着，一般情况下，每次的数据转换和传送，半永久性数据都是由新或老软件更新的。数据转换取决于数据修改信息，它们在软件开发期间创建于支持系统中，在初始/ 装入阶段

被装入目标系统，在图 3 中称为之“数据修改信息”。

关于数据转换，系统的一种可供选择的实现方案可以是仅仅把数据的表示法转换为由新软件在一种所需要的基础上所使用的形式，然后，在老软件从系统中移去之前，转换到那时为止系统仍不需要的全体剩余数据。同样地，所有这些数据开始就可以转移成新软件所使用的表示法，以便节省存储空间，然后，可以用一个再转移程序来将数据转回到由老软件在一种所需要的基础上所使用的形式，直到它被移去。

图 3C 也用图描述了修改处理的第三阶段，即测试阶段。该阶段打算允许软件新版本装入系统，最初仅对测试数据进行操作以便在使用软件与实际业务一同运行之前确定软件的质量水平。该测试阶段可以被分为两个子阶段：(a)用测试业务测试，即只有人工产生的事务处理将采用新版软件；(b)用简单业务测试，即，经过选择的实际业务的一定比例，大约在零到百分之百之间的新的事务处理将被引去用新版软件运行之。到第二个子阶段结束时，实际业务的大部分或全部将在新版软件之下运行。

测试业务是由特殊软件或利用特殊测试电话用户所产生的。测试业务是受控制的，以便保证修改单元 R'23 被使用。通过给测试业务加注测试标志可以确保这一结果，该测试标志将自动地在所有的定向点上把测试业务引向新版软件，在定向点上存在一个引向新版软件还是现存的老版软件的选择问题。对于常规的实际业务，究竟是使用新版还是使用老版的软件，将在遇到的第一个定向点上作出决定，该定向点上有一个选择。此后，该业务或事务处理仅由所选择的软件版本进行处理。在后面的某个定向点如要改变到另一个软件版本，在系统中是被禁止的。在此阶段期间，如果测试指出了因使用新版软件而导致发生的问题或错误，就转回去进行修改，所有新的事务处理都被引向老版软件。

值得注意的是，在特定的实施情况下，在成功地处理了测试业务之后使用新版软件的样本业务量可以减少到零。这种情况下，一旦新版软件已被测试业务检验过，满载的所有新的事务处理都可以传送给新版软件，当所有的旧的事务处理已经结束时就允许新版软件完全取代旧版软件。

图 3D 中图示了第四阶段即结束阶段。在此阶段，已经在使用旧版软件的事务处理继续使用老的修改单元 R22，直到系统内不再存在使用老的修改单元 R22 的事务处理为止。当新的事务处理使用新版软件时前述情况将自然发生。在此阶段，两个版本的软件留在系统存储器内，新版软件依然被认为是处在测试阶段。

结束阶段可以或者是持续下去、直到使用老版软件的所有老的业务已经结束，或者可以是在一个特定的时间将它置为结束。如果到结束阶段的最后还有任何仍使用老版软件的老业务，那么它们将被终止，或者如果可能的话，传送给新版软件。此后，由老版软件所拥有的半永久性数据将不再被更新，老的修改单元将被堵住，结束阶段终止。应当注意的是，在结束阶段之前的测试阶段可以已经由一长段时间来结束，在此期间所有新业务已经被分类成为样本业务，所有老业务已经结束。在这种情况下，结束阶段将非常短，并且仅仅意味着半永久性数据不再被更新和老的修改单元被堵住。

结束阶段的终止意味着整个软件修改过程已经终结。图 3D 表示了这一状态。

老的修改单元 R22 及其修改信息不再受到维护，在此关头，它不再可能转回去在老版软件之下运行。此时，如果新软件有一个问题，就要求一个全新的修改。老版软件现在也可以从系统中移去。图 3E 图示了这一状态。

本发明的系统的另一特征，包含了瞬时修改法作为平稳修改的补充。这一方法为处理所有业务的目的提供了从老版软件到新版软

件的即时的或瞬间的转换能力。当应用过程禁止两种不同版本的软件在系统内共存时就可使用这种类型的修改。利用瞬时修改法，如果需要就可以在转换时立即把软件状态变换到新版本。在系统中这是可能的，因为应用软件有能力在新版软件中再造其状态。本发明的系统的这一方面的一个重要性质就在于，尽管修改在内部是相当突然的，对系统用户以及正被处理的业务来说则是透明的。可以在不引起任何可以看得见的中断处理的情况下将业务再引向新版软件。本发明在这一方面的另一个优点是，老版软件留在系统内，虽然是以一种被动模式存在。因此，如果已经显示出新版软件有问题或有错，返回到老版软件就仍然是可能的，这时的处理并不需要重大的或较长的中断。

下面参看图 4，这里给出了从老版软件转换为新版软件的平稳修改方法的流程图。特别是，系统预先假设现存软件在系统中是有效地运行着的，并从 101 开始，此时将新版软件装入内存。在 102，系统对数据及其在新版软件中的改变进行复制，并连接数据到新软件上。在 103，系统开始用新软件运行测试事务，正常的业务在系统内继续用老软件和老数据运行。在 104，系统询问“新软件是在进行测试业务工作吗？”如果不是，系统转到 105，在此点新软件和新数据从系统中移走，进程在 106 结束。在 104 点如果新软件是在进行测试业务工作，系统转到 107，在该点系统用新软件运行实际业务的样本，同时对常规业务的其余部分与老软件和老数据一起进行维护。通过新软件运行的样本业务的百分比可以在实际业务总量的零到百分之百之间有选择地变化。此后，在 108，系统再次询问新软件是否在对样本业务进行工作。如果不是，系统转到 105，新软件和新数据移出，处理结束。然而，如果在 108 点处新软件正在成功地处理样本业务，系统就转到 109 用新软件和数据运行将来所有的呼叫。此后，在 110，系统再次询问新软件是否在工作，如果不是，转到

105，移去新软件，在106结束。如果在110处，系统中新软件正在进行处理常规业务的工作，系统就询问所有老的事务处理在系统内是否已经完成，这是在111处。如果在111处得到的回答为“不是”，就在113处询问用于改变的时间期限是否已满，如果期限未满，就在109处继续做：(1)用新软件执行所有新的事务处理，(2)用老软件执行所有老的事务处理，直到在111处接收到一个“是”或者在113处时间期限已满。如果在113处时间期限已满，系统就终止或传送所遗留的呼叫，并转到112。如果在111处接收的是一个“是”，系统就移到112，老软件与老数据一道被移走。系统在从老软件转到新软件的运行期间，在没有不适当地使现有业务遭受危险或延迟的情况下，在电讯交换系统内完成了一次交换转换。

下面看图5，这里给出了表120，其中包含一个呼叫标识(ID)类目和一个ID指针类目。对于系统内的每一个是测试的调用寻址，给出了一个指向新软件121的指针，与此同时对于所有包含常规标识的呼叫标识(ID)，给出了指向老软件122的指针。图5阐明了这样一个方法，据此方法本发明的系统能够将普通的、实际的业务和测试业务适当地引向合适的软件版本。

尽管这是关于在本发明的系统内对老软件和新软件的寻址方式的一种概括的过分简单的解释，事实上，详细的连接过程调用机制被用来在分别装入的程序单元之间创建动态的运行时间汇集。即，在替换一个程序单元时，软件的新老版本要共存一段时间，直到新版本经验证是正确的并且由老版本所执行的活动可以如上所述予以终止。一种合适的连接过程机制在同样是未决的一项美国专利申请书中已被揭示，该专利申请书的名称是“计算机系统中用于同时执行软件模块的动态运行时间汇集的系统”，由K. Lundin等人发明并转让给L. M. 埃利克逊电话股份有限公司(Telefonaktiebolaget L M Ericsson)，特在此引用使之具体化。本发明的系统将顾客作为

借助于连接过程调用通过一个接口访问软件的一种途径。在装入期间，对连接过程调用可访问的所有接口对核心中的一个商人函数都是公布的。每个接口是用其标识和地址公布的，这涉及到从接口创建一个目标的一种方法。在软件版本间的汇集在运行时间内完成，每当对一个特定的接口建立了一个目标，就向此后将被调用的创建方法的商人或地址发出一个请求，并向所创建的目标返还一个目标指针。

如图 6 所示，类 X131 的每个目标由目标—数据 133 内的表 132 通过一个目标指针 134 进行调用，这里目标指针 134 本身又指向目标 131 内的一个操作表 135，即其中包含该类目标所定义的操作定义的地址。服务员的程序单元内的操作表所涉及的一些地址存贮在目标—数据中。操作表本身又含有属于指定接口的操作的地址。因为在目标—数据内的操作表地址的存储单元和操作表中的地址所存贮的顺序是固定的和已知的，就可以在不借助于商人的情况下调用操作。在接口中可以在没有商人的情况下被调用的一种操作是删除所创建的一个目标。

这些操作表的使用提供了达到同素异构的能力，这里的同素异构是可以通过使用例如程序设计语言 C＋＋ 及其关于虚拟表的构造而实现的一个概念。同素异构，意味着“多形状”，是一种可用以改变由不同目标所分享的一个组成成份的性质的技术。换句话说，一个组成成份在各种情况下可以看起来是相同的，但是对于它所联系的不同的目标可以用有点不同的方式去完成。同素异构是有用的，它可允许创建相关联的目标族，即，这样的目标具有共同的来源或基础，但它们在不同的情况下以不同的方式去完成。这就允许一个目标族里用同样的名称的每个目标含有多种方法或功能，尽管对于每个目标的多种方法的实际代码可以有很大的差别。本发明的系统使用同素异构，以及面向目标程序设计的其它原理。然而，本发明

的系统以一种新的更有用的方式实现和扩展了这些原理，以达到执行期间不同软件版本的动态的、透明的内部连接。

下面参看图7，那里图示了这样一个事实，即连接过程调用技术体现在核心142中所包括的商人141的概念中，核心142使一对软件单元143和144之间具有接口关系，143和144中分别含有目标的一个顾客类145和一个服务员类146。图7详细给出了为在也如图6中所示的系统内创建目标所要求的步骤。

目标是一种语言构造，它将数据和代码或函数包含在一个单个的程序包或单元之内。因为它们可以包含数据和代码，它们就作为微型的、独立的程序。因此，在创建更复杂的程序时就可以把它们作为积木块而不必重新为这些功能进行所需的编码。因为它们可以被单独地维护和修改，所以程序的维护和修改就简化了。

类是用以定义目标的模板，而目标则是类的一个例子。一个类包含两种类型成分，实例变量或数据成员和方法或成员函数。为支持计算机系统的顾客或非服务员节点，通过利用作为服务员类的一种媒介的接口说明自动产生出一种顾客类。系统的顾客节点从顾客类中调用操作，以确信调用被传送给了服务员类中驻留的软件工具。因此，所有与动态汇集功能相关的编码都可以在顾客类中找到。

类说明控制着这样一种方式，编译程序将按这种方式在目标—数据中存储地址，操作表中的地址将依该方式的次序加以陈述。某些类说明是由系统自动产生的。当一个目标在系统中创建出来时，其“创建方法”部分函数就可以通过向定位在核心142内的操作系统的商人141部分发出一项请求来加以定位。商人141含有由系统内的连接过程调用可访问的所有类的所有接口信息，即，它含有每个目标可被其它目标访问或访问其它目标的信息。

图8阐明了程序单元的老软件和新软件在运行期间通过连接过程调用进行内部连接和汇集所采用的方法。核心142内的商人141

可以将软件单元151的执行引向或者是老软件单元152或者是新软件单元153。在进行替换时，来自老版本和新版本的服务员类每种都有其在商人141中所公布的接口。商人141对每个项目都提供了两个地址入口，一个是到老软件单元152，另一个是到新软件单元153。在替换之前所创建的事务处理将接收到一个指向老软件单元152及其服务员类的地址，而另一个事务处理可以接收到指向服务员类的新版本的地址。

替换完成以后，采用旧软件单元152的事务处理已经结束以后，老软件单元152可以从存储器中移走，由老软件单元152中的服务员类所公布的接口就可以撤销。如果在老软件单元内所有的事务处理运行结束之前试图从存储器中撤销这些服务员类，系统就从核心142中产生一个例外调用。然后系统内的例外管理事务处理就允许未结束的处理有一个机会，将它自己重新定向并且使用新软件单元153，或者否则就终止它。

本发明中使用连接过程调用技术时，接口说明是用一种称为ELIN的面向目标的接口描述语言书写的，该语言在一项美国专利申请中被揭示，该专利以K. Lundin的名义申请，并转让给L. M. 埃利克逊电话股份有限公司(Telefonaktiebolaget L M Ericsson)，特在此引用使之具体化。在该语言中，有一个特别的构造（类），它特别有助于连接过程调用接口的说明。ELIN语言中的类是特定类型的目标所提供的接口的一种说明。如果使用一种面向目标的程序设计语言，这些目标就尤其适合作为一个类的例子而加以实现。ELIN语言中连接过程调用接口的说明包括下列信息。

(a) 说明的名称；

(b) 用作该名称的基础的其他接口；

(c) (用于创建示例的) 一个或多个设计师；以及

(d) 零个或多个方法说明，其中每个包括一种方法名称，主题，

返回类型和例外。

如下所述，是一个代码形式的接口说明的示例，它可以被用作为该连接过程调用技术的一部分，它描述了堆栈目标的一个接口。

```
CLASS Stack;
    BASE
        CLASS TelecomObject;
    ACCEPTS
        CONSTRUCTOR (IN size Int);
        METHOD push (IN data Int);
        METHOD pop () RETURNS Int;
        DESTRUCTOR ();
END CLASS Stack;
```


这一接口说明定义了一类堆栈目标，其基础类被称为“电讯目标(TelecomObject)”。此类目标可以从列出的函数成员中接收消息呼叫。有了这一类所标示的基础，就表明，还有一个称为“电讯目标”的类的这种说明。该基础类也有特定的详细说明的方法，作为该基础类的一个示例的当前类将继承这些方法。在上述类定义中所说明的函数成员或方法是在基础类中所说明的函数成员或方法之外的。总之，上述代码包含了一种类说明，它是在系统内可创建的接口说明的类型之一。

一个接口可以由另一个接口推出，那么后一个接口就被称为所推出接口的基础接口。接口可以由一个以上的其它接口推出，所推出的接口继承了作为其基础的每一个接口的操作。此外，所推出的接口要说明它自己附加的操作，尽管它对从其基础接口所继承的那些同名操作可以不加定义。应该明白，继承仅仅影响类的接口层次，而不影响执行层次。

如图 9 所示，本发明的系统也包括存根—代码生成工具 162，它用来证明顾客与服务员之间的协调，在运行期间顾客与服务员是通过一个接口动态地连接在一起的。该接口是用一种语言无关形式加以说明的，但是采用面向目标的示例说明。存根—代码生成处理确保达到向几种程序设计语言之一的变换，在以下的部分里，有关于如何实现C++ 中的变换的简短说明。参看图 9，其中阐明了在本发明的系统中一个接口说明 161 采用存根生成工具 162 而和一组所生成的文件 164 相关的方法。图 9 尤其是阐明了在该语言中完成变换的C++ 的整个结构。接口说明在本发明的系统中使用时以面向目标的接口描述语言 ELIN 写出，它类似于程序设计语言 C++ 中所用的类定义。与此相似，通过目标的访问操作技术类似于程序设计语言 C++ 处理虚拟函数的方法。因此，图 9 所示的关于C++ 的变换对于本发明的系统这一方面的操作是有指导性的。

存根生成工具 162 既为顾客一方也为服务员一方产生两个文件，一个以“.h”（header 的首字母）为后缀，一个以“.cc”(code 的首字母）为后缀。对于顾客，“.h”或首标(header)文件包括两个类定义。一类是服务员的“.h”或首标文件里的相应类别的精确复制件。这就确保了顾客与服务员之间的协调一致，并且使顾客调用由服务员创建的目标成为可能。然而，这一类的设计师是专有的，从而使该类不能被用来在堆栈上创建自动的目标。第二类是将用于顾客的一个类，它用作一个媒介，通过它可以访问由服务员创建的目标。

对于服务员，由存根生成工具 162 产生同样的两个文件—“.h”(header)文件和“.cc”(code)文件。“.h”文件的内容包括一个类定义，它将确保与顾客的协调一致。就是这个类被用作工具的基础。该工具可以直接以所产生的类为基础，或者说所产生的类可以用作推出其它类的基础。“.cc”文件含有“创建方法”的一个轮廓以及实现创建方法地址的公开的代码。创建方法的主体担负着创建一个与所产

生的类兼容的目标和返还一个指向也如图6所示的新创建目标的指针的工作。

为顾客和服务员两方面产生出不同的然而是兼容的类定义而不是一个共享的类定义，这样做有几条理由。首先，它为顾客和服务员中的成员提供了不同层次的清晰度。例如，一个设计师在服务员中必须是公共的，但如果它是驻留在顾客中就并不需要是公共的。其次，如果使用不同的类，在为测试的目的把顾客和服务员程序连接在一起时就不会遇到名称冲突的问题。下面参看图10，这里给出了一个特定的框图布局，阐明了给定的示例代码模块及在本发明的系统中它们的相互关系。图10阐明了给定的所产生的文件和所书写的说明的逻辑结构，它们在本发明的系统内可被实现。在最高层次，公共接口说明171定义了一个类"X"以及该类将接受调用时采用的方法。该类的逻辑下属，在定义的下一层次，是接口说明171的一个用户单元172的说明书，和公共接口说明171的一个供应者单元173的说明书。用户单元说明书172定义了公共接口—类别X的一个顾客。供应者单元说明书173定义了类X的一个服务员。

在单元说明书172和173之下的下一逻辑层次是为用户和供应者分别产生的类定义。为X用户(XUser)所产生的类定义174阐明了既为公共用途也为个人用途所定义的特定的用户类。为X供应者(Xprovider)所产生的类定义175阐明了关于供应者数据和函数的特定的公共定义和专有定义。

最后参看图11，它阐明了一个协议说明是如何被用来产生存根—代码的，这样确保了使用消息的两个通讯当事人之间完全的协调一致。存根—代码的结构如图11所示，包括用户书写代码181，生成代码182和核心代码183。在分布式和模块的计算机系统中，它的一个例子是电讯系统，使用许多应用层次的协议以方便在系统各部分之内及系统各部分之间的通讯。

协议可以看作是系统内各对当事人之间契约的集合，这些当事人同意按照一种特定的方式和格式通讯。有的协议可以被描述为顾客—服务员协议，在那里仅有一方当事人是引发者。另外一些协议，称为同等协议，允许两方当事人引发通讯。与其它现存的系统不同，在本发明的系统中，当事人之间全部的协定或协议是用一个单独的接口说明加以说明的，该接口说明是与当事人的具体的执行过程相分离的。因此，这意味着该单独的说明可以用作为一个普通的协议，它可以系统内部任何一对当事人之间重新用作协议。

本发明的系统以一种有专利权的面向目标的接口描述语言ELIN来实现单独的接口/协议说明。例如，同等类型的协议说明包含下列组成成份：(1)操作形式上被分为协议，每个协议有两方当事人；(2)相互作用的约束的说明书。同等协议说明的存在与使用协议执行通讯的执行过程相分离。同等协议说明根据下列格式而组成：(1)协议名称；(2)第一方当事人的名称及其可接受的操作表；(3)第二方当事人的名称及其可接受的操作表；(4)相互作用约束（可选的）。

以下为带有相互作用约束的代码形式的一个协议说明的示例。该协议说明中所包含的信息可以用于生成存根代码：

```
PROTOCOL CommunicationService;
    PARTY DataProducer;
            ACCEPTS
                StartTransmission, Terminate Transmission,
    ReSendData;
    END PARTY DataProducer
    PARTY DataConsumer
        ACCEPTS
            StringData, IntegerData, NoMoreDataToSend;
```

```
END PARTY DataConsumer;
INTERACTION
    STATE START
        WHEN StartTransmission THEN Started;
    STATE Started
        WHEN TerminateTransmission THEN START;
        WHEN IntegerData THEN Dataphase;
        WHEN StringData THEN Dataphase;
    STATE Dataphase
        WHEN IntegerData THEN Dataphase;
        WHEN StringData THEN Dataphase;
        WHEN ResendData THEN Dataphase;
        WHEN NoMoreDataToSend THEN DataphaseEnded;
    STATE DataphaseEnded
        WHEN ResendData THEN ResendOrdered;
        WHEN TerminateTransmission THEN START;
    STATE ResendOrdered
        WHEN StringData THEN DataphaseEnded;
        WHEN IntegerData THEN DataphaseEnded;
END PROTOCOL CommunicationService;
```


在系统内通讯的一方当事人的逻辑结构也在图 11 中给出。如图 11 所示，ELIN 语言被用于描述分布跨越该系统的目标之间的通讯，以及在该系统内所使用的数据类型之间的通讯。在本发明的这一特征中使用的和定义的协议允许设备起相同作用，在任何一方当事人引发通讯的情况下。并不预先定义任何一方在通讯中是主方或是从方。本发明的系统的这一特征允许在不同的和遥远的地方开发

和操作的系统容易进行相互间的操作，只要每个系统是使用单独加以说明的接口开发的。本发明的系统的这一方面的协议说明与该系统内任何应用工具是分离的和有区别的。

如图 11 中所进一步阐明的，用户书写的代码 181 作为通讯协议的一方当事人既可根据协议说明发送消息也可根据协议说明接收消息。数据接收过程 184,185 和 186 处理按协议到达的输入消息。数据发送过程 187,188,189 包含由存根生成工具所自动生成的代码，以便在用户呼叫时根据协议说明建立并发送消息到系统中去。接收消息的活动 190 和发送消息的活动 191 都是通过一个接口媒介 192 引导的，192 是生成代码 182 的一部分。该接口说明 192 是生成代码的命令部分，为了接口和协议适当地发挥作用该部分必须给出。

调度程序 193 是一个函数，它由存根生成工具产生并由协议说明中加以说明的每个输入消息调用。调度程序 193 接收消息，对消息进行译码，从消息主体中分离出消息标识符，然后将其如图中 194 所示分配给将以此工具书写的过程。

协议警察 195 是生成代码 182 的一个可选部分，用来监督业务并确定在任一给定的瞬间通讯双方当事人在发送或接收消息时是否遵守了接口规则。协议警察 195 在遵守协议规则的监督方面象一台状态机器那样运作。该状态机器的逻辑在上面提供的示例代码中已作表达。

在图 11 所示的核心代码 183 中驻留了一个通讯端口 196。该通讯端口 196 由本发明的系统的寻址机制视为一个被动的支持工具。通讯端口 196 不知道通过它正在传递的协议，但是用以使通讯更加便利。通讯支持 197 是存在于操作系统中的总的通讯支持。它可以在同一处理器中的处理之间进行操作，也可以对定位在不同的处理器上的处理进行操作。如果它正在对分布在处理器之间的目标进行操作，通讯支持 197 就将组成一个硬件通讯链路。在图 11 中包含的

整个说明的镜象将代表发生在该支持中的相应活动以及在系统中第二个通讯当事人的操作。

如上所述，本发明的系统使得运行期间新软件与旧软件的蕴含或连接成为可能，这种蕴含或连接所采用的方式使得软件既能被实时有效地测试，也能在远程通讯网络中平稳地和透明地被替换，而该网络中的电讯业务并不受干扰。

这样，可以相信，本发明的操作和构造从前面的描述中已经显而易见。图示的和描述的方法、装置和系统已经按所愿意的方式来刻划，人们将容易看到，在不离开以下的权利要求所定义的本发明的精神和范围的情况下，是能够进行各种改变和修改的。

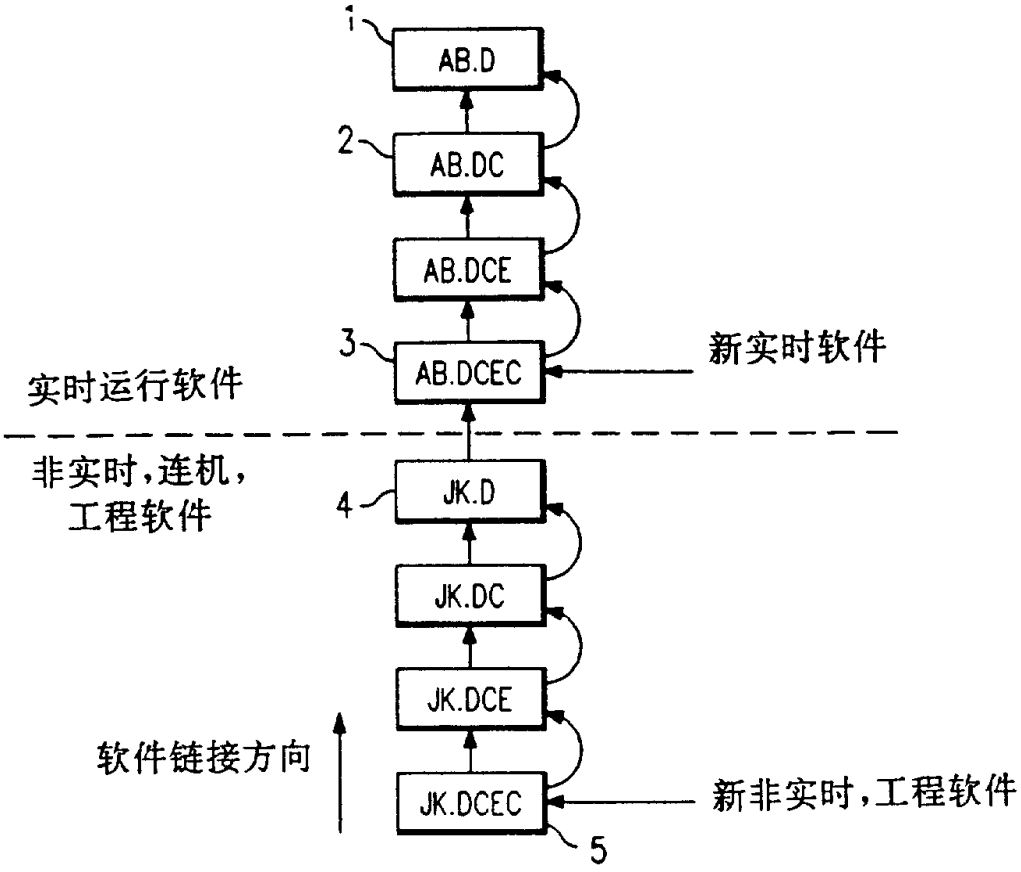
1
AB.D
2
AB.DC
AB.DCE
3
AB.DCEC
新实时软件
实时运行软件
非实时，连机，
工程软件
4
JK.D
JK.DC
JK.DCE
软件链接方向
JK.DCEC
新非实时，工程软件
5

图 1A

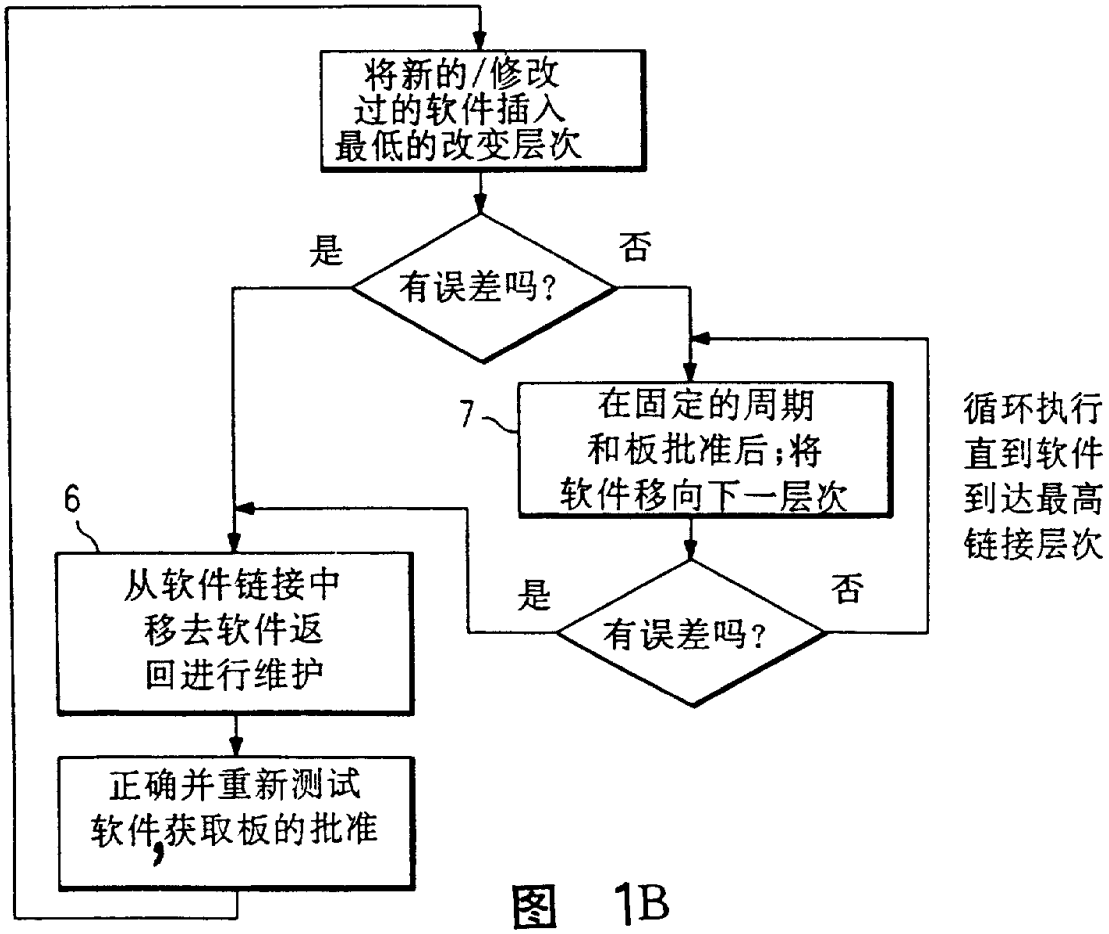
将新的/修改
过的软件插入
最低的改变层次
是
有误差吗？
否
7
在固定的周期
和板批准后；将
软件移向下一层次
循环执行
直到软件
到达最高
链接层次
6
从软件链接中
移去软件返
回进行维护
是
有误差吗？
否
正确并重新测试
软件，获取板的批准

图 1B

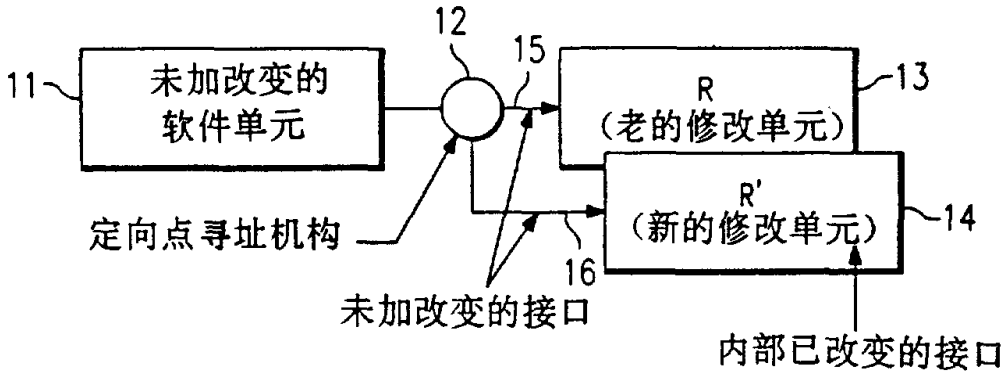

图 2

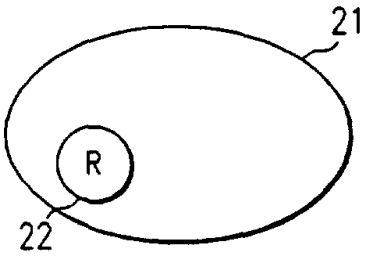

图 3A

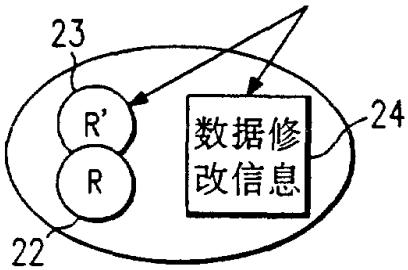

图 3B

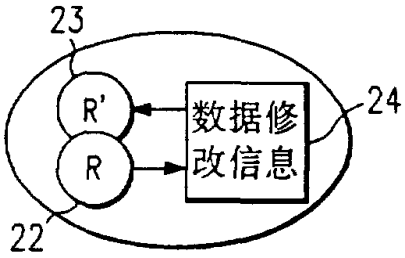

图 3C

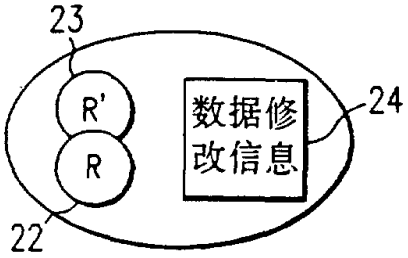

图 3D

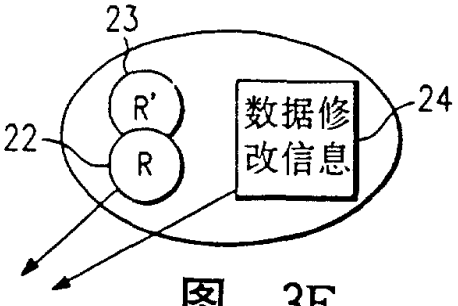

图 3E

图 3

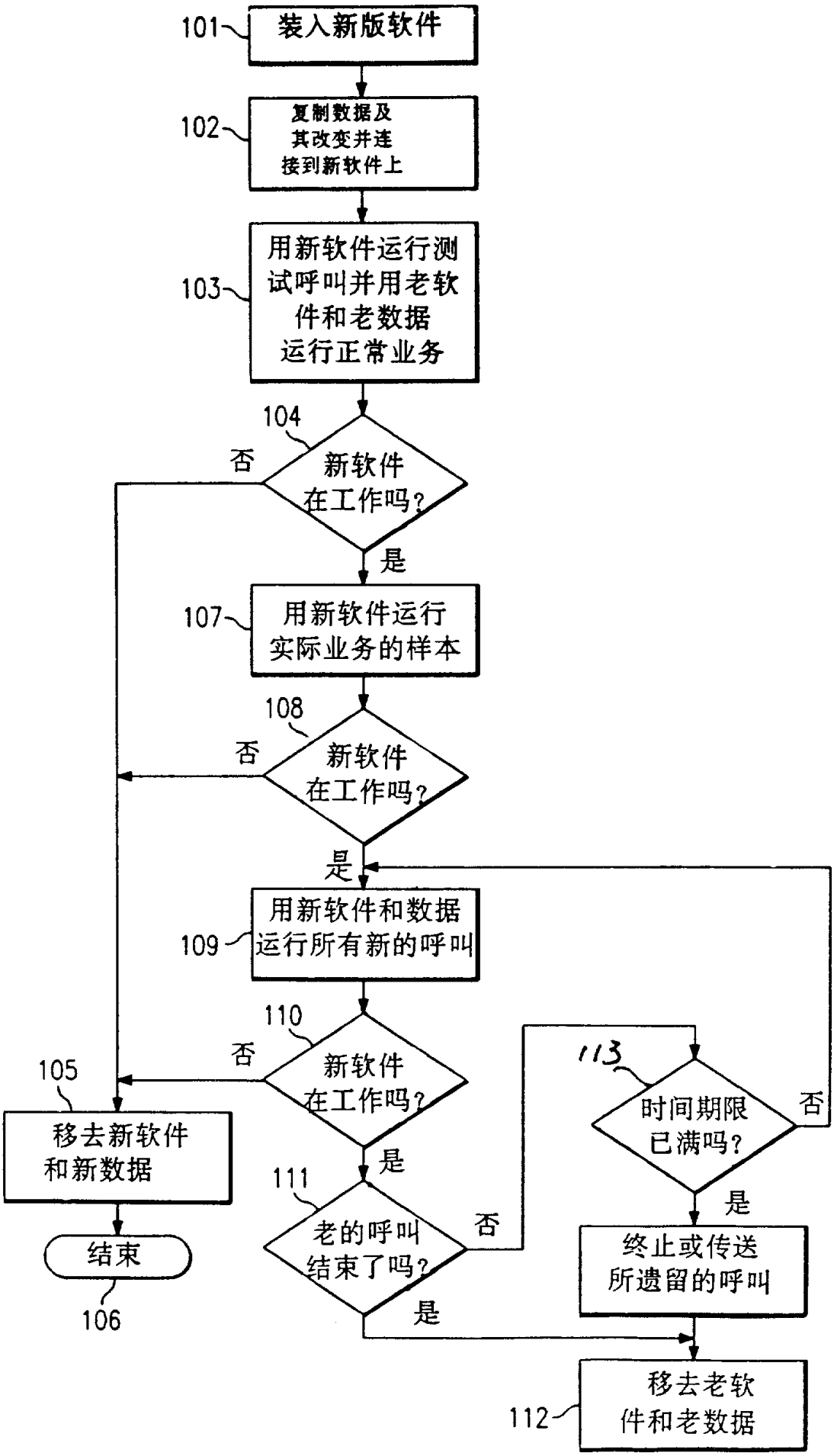
101
装入新版软件
102
复制数据及其改变并连接到新软件上
103
用新软件运行测试呼叫并用老软件和老数据运行正常业务
104
新软件在工作吗?
否
是
107
用新软件运行实际业务的样本
108
新软件在工作吗?
否
是
109
用新软件和数据运行所有新的呼叫
110
新软件在工作吗?
否
是
105
移去新软件和新数据
结束
106
111
老的呼叫结束了吗?
否
是
113
时间期限已满吗?
否
是
终止或传送所遗留的呼叫
112
移去老软件和老数据

图 4

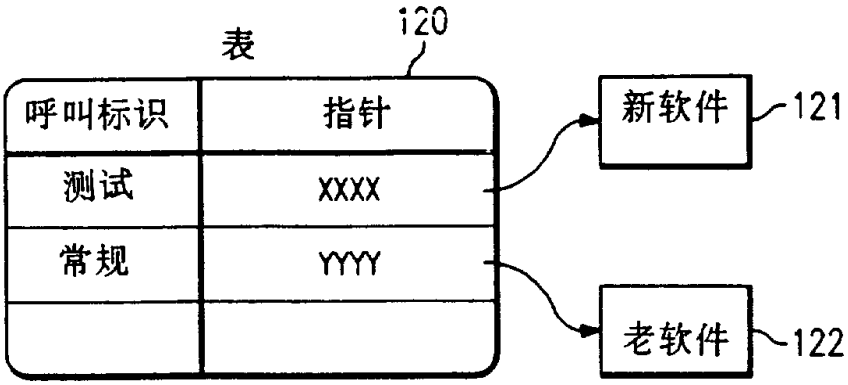
表
120
呼叫标识
指针
测试
XXXX
常规
YYYY
新软件
121
老软件
122

图 5

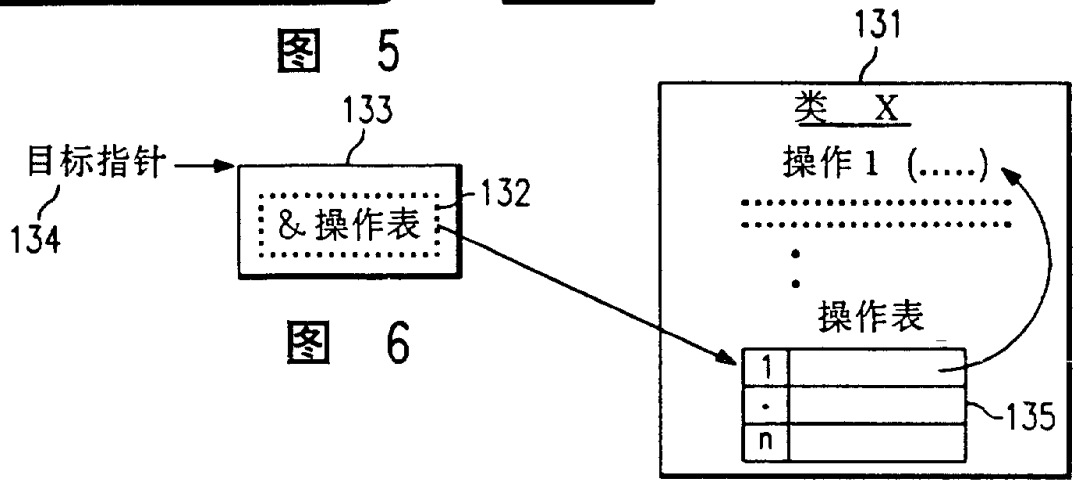
131
类 X
操作 1 (.....)
操作表
1
.
n
135
133
目标指针
134
& 操作表
132

图 6

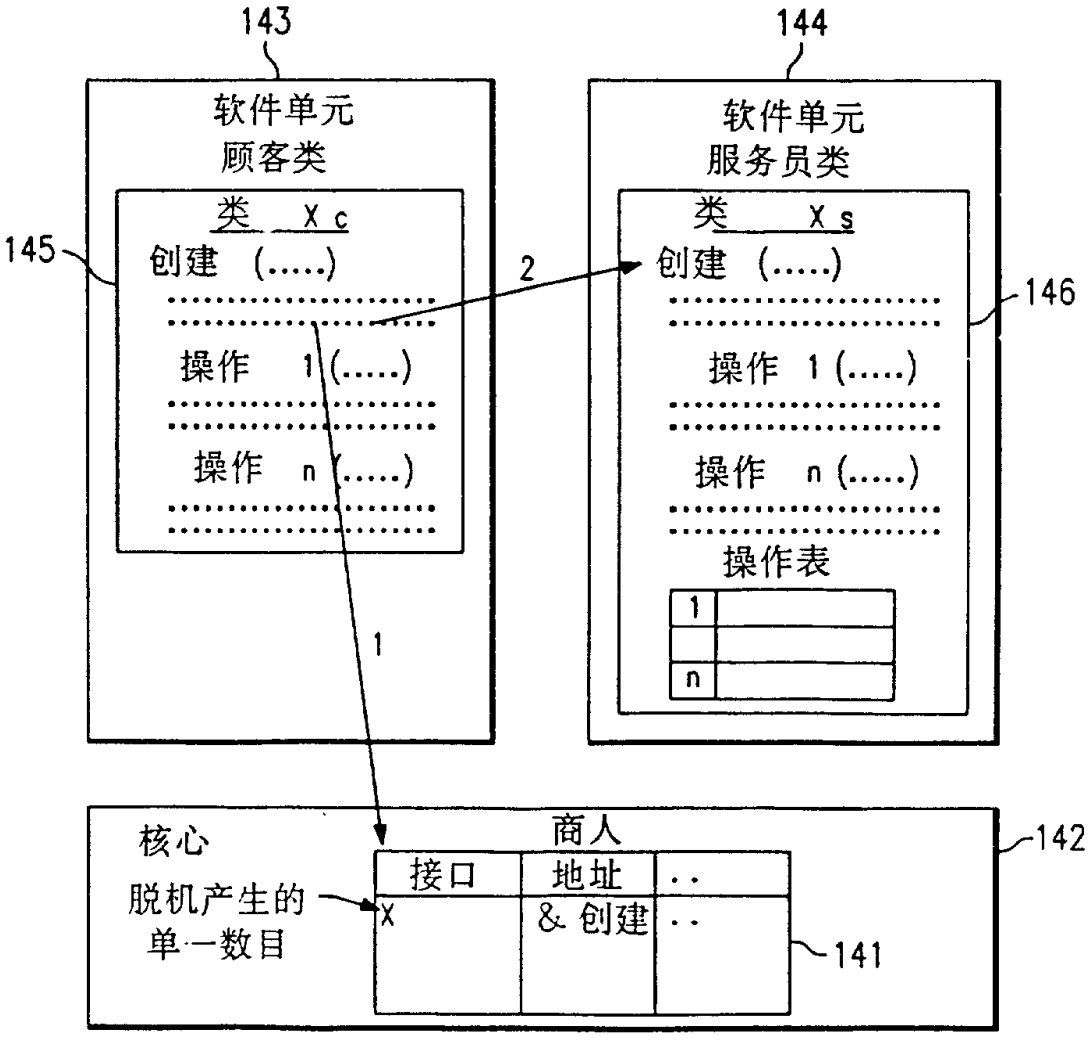
143
软件单元
顾客类
类 X c
145
创建 (.....)
操作 1 (.....)
操作 n (.....)
2
1
144
软件单元
服务员类
类 X s
创建 (.....)
操作 1 (.....)
操作 n (.....)
操作表
1
n
146
核心
脱机产生的
单一数目
商人
接口
地址
X
& 创建
141
142

图 7

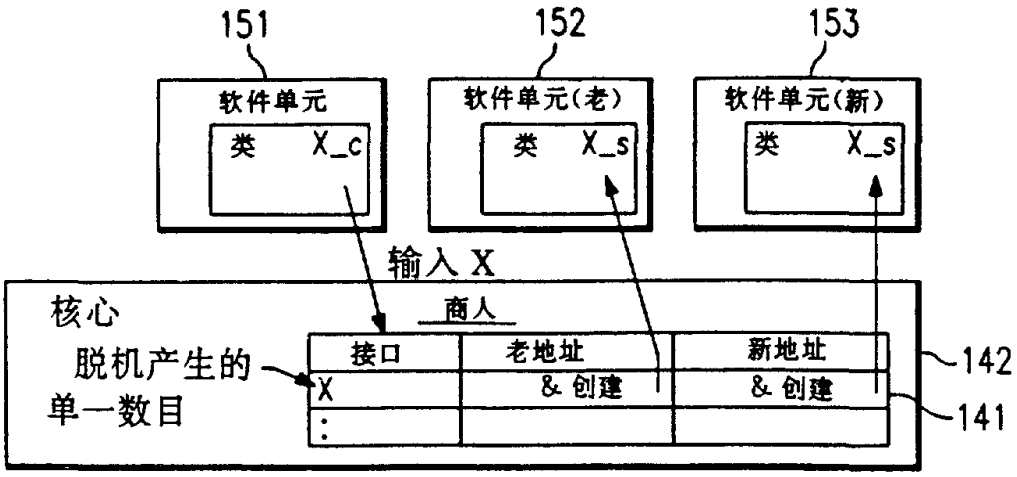
151
152
153
软件单元
类 X_c
软件单元(老)
类 X_s
软件单元(新)
类 X_s
输入 X
核心
商人
脱机产生的
单一数目
接口
老地址
新地址
X
& 创建
& 创建
:
142
141

图 8

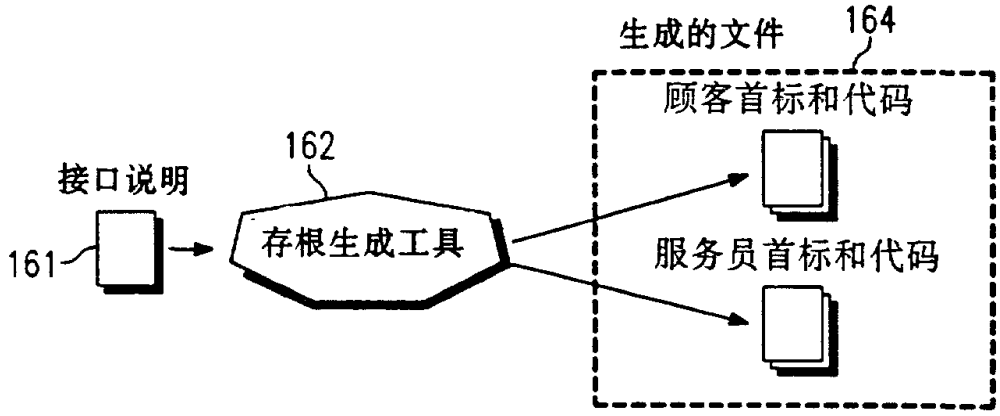
生成的文件
164
顾客首标和代码
162
接口说明
161
存根生成工具
服务员首标和代码

图 9

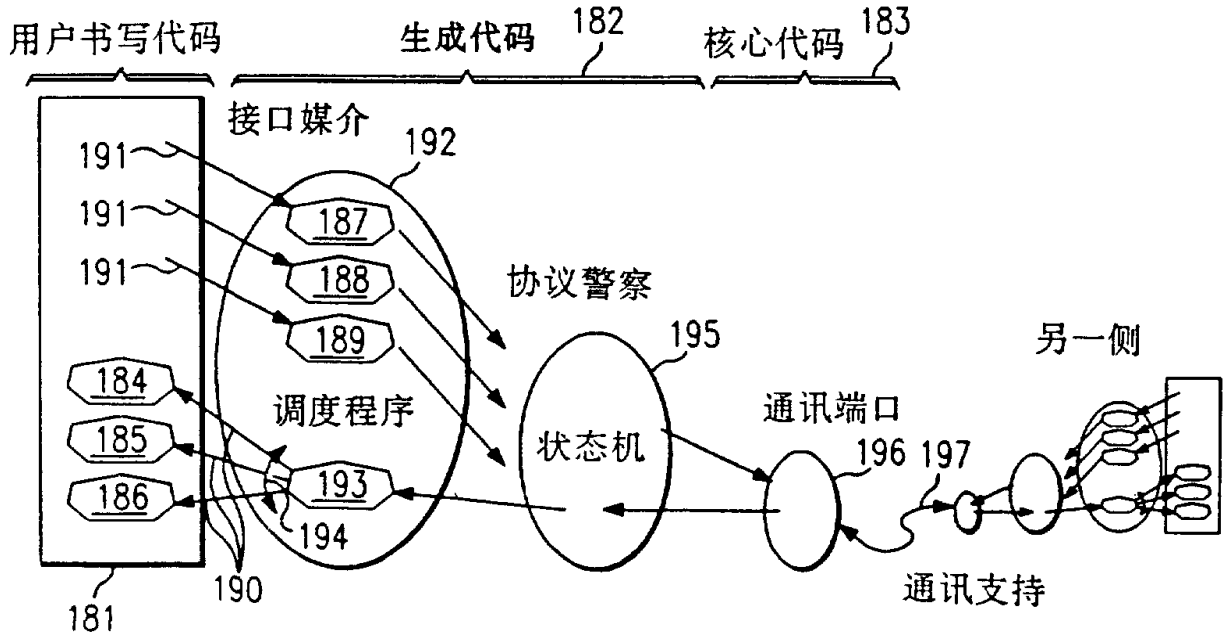
用户书写代码
生成代码
182
核心代码
183
接口媒介
192
191
191
191
187
188
189
协议警察
195
另一侧
184
185
186
调度程序
193
194
状态机
通讯端口
196
197
190
181
通讯支持

图 11

公共接口说明

```
CLASS X;
 ACCEPTS
  METHOD CONSTRUCTOR(size INTEGER);
  METHOD Mx(IN data INTEGER);
  METHOD DESTRUCTOR();
END CLASS X;
```

171

使用接口的单元说明书

```
UNIT XUser;
CLIENT OF CLASS X;
END UNIT XUser;
```

172

使用接口的单元说明书

```
UNIT XProvider;
SERVER OF CLASS X;
END UNIT XProvider;
```

173

为 X 用户所产生的 C++类定义

174

```
class C_X{
 public:
  enum InterfaceId{Id=1};
  static C_X* Create(int size);
  virtual void Mx(int data)=0;
  virtual ~C_X()=0;
 private:
  void* xxxNotUsed;// only here to
assure compatibility
};

class X{
 private:
  C_X* p;
 public:
  create(int size)
   {p=C_X::Create(size);}
   void Mx(int data);
  ~X()
   {delete p;}
};
```

为 X 供应者所产生的类定义

```
class S_X_Data;
class S_X{
 public:
  static S_X* Create(int size);
  virtual void Mx(int data);
  virtual ~S_X();
  S_X(int size);
 private:
  S_X_Data* D;
};
```

175

图 10