ISTQB® 基础级认证
基于模型测试工程师认证

大纲

2015 版

国际软件测试认证委员会

中文版的翻译编辑和出版统一由 ISTQB® 授权的 CSTQB® 负责

英文版权说明
如果来源是得到承认的，此文档可被全部复制或进行摘录。

基于模型的测试工程师工作组成员：Stephan Christmann (chair), Anne Kramer, Bruno Legeard, Armin Metzger, Natasa Micuda, Thomas Mueller, Stephan Schulz; 2014-2015.

修订记录

版本	日期	备注
Beta 1.0	2015 年 5 月 8 日	第一个 Beta 版，基于交叉评审后修正了原来的 ALPHA 版本
Version 2015 - Final	2015 年 10 月 23 日	Beta 评审后更新的最终版本
中文版发布	2017 年 1 月 22 日	最终的中文版完成并发布

目录

致谢

本文档由 ISTQB® 基础级工作组编写。

感谢评审小组以及各国委员会的建议及意见。

在基于模型测试大纲完成期间，基于模型测试工作组有以下成员：Stephan Christmann（MBT WG 主席），Anne Kramer，Bruno Legeard（MBT 作者组联席主席），Bruno Legeard（MBT 作者组联席主席），Natasa Micuda（MBT 试题组组长），Thomas Mueller（ISTQB®CTFL 工作组主席），Stephan Schulz（MBT 审查小组主席）。

作者：Stephan Christmann, Lars Frantzen, Anne Kramer, Bruno Legeard, Armin Metzger, Thomas Mueller, Ina Schieferdecker, Stephan Weissleder.

试题组：Robert V. Binder, Renzo Cerquozzi, Debra Friedenberg, Anne Kramer, Willibald Krenn, Bruno Legeard, Karl Meinke, Natasa Micuda, Michael Mlynarski, Ana Paiva, Ina Schieferdecker, Stephan Weissleder.

试题评审组：Eddie Jaffuel, Ingvar Nordstrom, Adam Roman, Lucjan Stapp.

评审组：Stephen Bird, Thomas Borchsenius, Mieke Gevers, Paul Jorgensen, Beata Karpinska, Vipul Kocher, Gary Mogyorodi, Ingvar Nordstrom, Hans Schaefer, Romain Schmechta, Stephan Schulz, Szilard Szell, Tsuyoshi Yumoto.

同样感谢参与基于模型测试大纲的评审、评论和投票的各国委员会以及基于模型测试的专家社区人员：Patricia Alves, Clive Bates, Graham Bath, Rex Black, Armin Born, Bertrand Cornanguer, Carol Cornelius, Winfried Dulz, Elizabeta Fourneret, Debra Friedenberg, Kobi Halperin, Kimmo Hakala, Matthias Hamburg, Kari Kakkonen, Jurian van de Laar, Alon Linetzki, Judy McKay, Ramit Manohar, Rik Marselis, Natalia Meergus, Ninna Morin, Klaus Olsen, Tal Pe'er, Michael Pilaeten, Meile Posthuma, Ian Ross, Mark Utting and Ester Zabar.

本文于 2015 年 10 月 23 号由 ISTQB®大会正式批准发布。

参加基于模型的测试大纲中文版翻译和评审的 CSTQB®专家有（按姓氏拼音排序）：陈耿（组长）、李华北、刘海英、柳溪、王帅、于长青、左平。

参加基于模型的测试大纲中文版最终评审的 CSTQB®专家有（按姓氏拼音排序）：杜庆峰、沈建雄、周震漪

0 大纲简介

0.1 本文档的目的

本大纲是 ISTQB-FL（国际软件测试工程师认证—基础级）的基于模型测试工程师模块的基础性文档。它是对国际软件测试工程师认证基础级(CTFL) 的一个特别附加模块。ISTQB®（国际软件测试认证委员会）提供本大纲用于下列用途：

•对各国认证委员会，（本大纲可用于）翻译成本地语言和授权培训机构。各国认证委员会可以按照本国语言对大纲做必要的修正，以满足本地出版发行的需要。

• 对各国考试委员会，（本大纲可用于）使用本地语言编制试题，以符合大纲学习目标。

• 对各培训机构，（本大纲可用于）制作课件，并确定适当的教学方法。

• 对认证学员，（本大纲可用于）指导准备考试（作为独立或培训课程的一部分）。

• 对于国际软件和系统工程社区，（本大纲可用于）推进软件和系统测试的专业化，并且可以作为书籍和文章 的基础。

ISTQB® 在其他组织事先书面请求并获得许可的情况下，可在需要时允许其他组织采用本大纲完成其他的目的。

0.2 概览

ISTQB® 基础级认证基于模型测试工程师认证大纲在ISTQB® 基础级是一个特别模块。对于要参加ISTQB® 基础级基于模型测试工程师（CTFL-MBT）的认证，首先必须通过ISTQB®基础级测试工程师认证。

通过ISTQB® 基础级基于模型测试工程师（CTFL-MBT）的认证，能在ISTQB® 基础级认证的基础上拓展参与认证者对于测试的理解，认证者能够理解基于模型的测试（MBT）的流程、方法和技术。

对于本大纲相关的期望的商业价值、测试工程师的职业发展以及可能的读者，在相关的概要文档【ISTQB - 基于模型的测试-概述】中进行了阐述，请查阅。

0.3 学习目标 Learning Objectives

本大纲的学习目标支持商业价值，学习目标还用于编制实现基础级测试工程师认证—基于模型测试工程师认证的考试。

一般而言, 本大纲所有内容都是K1 级别的考试范围。也就是说，学员将能识别、记忆和回忆本大纲的术语或概念。相关在K2和K3级别的学习目标在每章大纲的开始部分进行说明。

1 基于模型的测试的介绍– 90 分钟.

关键词
MBT 模型(MBT Model), 基于模型的测试(Model-Based Testing)

基于模型测试的学习目标

1.1 MBT 的目标和动机
FM-1.1.1 (K2) 描述采用 MBT 可期望的收益。
FM-1.1.2 (K2) 描述采用 MBT 的误导人的期望和陷阱。

1.2 在基础的测试过程中 MBT 的活动和工作产品或工件
FM-1.2.1 (K2) 总结在测试过程中部署 MBT 的特定活动。
FM-1.2.2 (K1) 记忆基本的 MBT 工件。（输入和输出）

1.3 将 MBT 集成到软件开发生命周期中
FM-1.3.1 (K2) 描述如何将 MBT 集成到软件开发生命周期过程
FM-1.3.2 (K2) 描述 MBT 是如何支持需求工程的

1.1 基于模型的测试（MBT）的目标和动机

基于模型的测试（MBT）是一种根据模型来设计测试的高级的测试方法。基于模型的测试方法支持并拓展了传统测试设计技术，比如我们熟悉的等价类划分、边界值分析、决策表、状态转换测试和用例测试等传统测试设计技术。基于模型的测试的基本思想是通过以下内容提高测试设计和测试实现活动的质量和效率：

- 基于项目的测试目标设计一个综合的MBT模型。通常这个综合的MBT模型是利用工具完成的。
- 将模型作为一种测试设计规格说明提供给测试工程师。这时模型应该包含高度格式化和详细的信息，这样才能保证能从模型自动导出测试用例。

MBT模型以及它的工件应该是与组织的过程紧密结合的，也应该与方法、技术环境、工具，以及任何特定的生命周期过程紧密结合。

1.1.1 采用基于模型的测试（MBT）的主要动机

主要有两个方面测试活动，提供MBT的基本动机以及描述MBT如何支持测试质量改进：

- 有效性
  - 建模是一种能促进与有关利益各方密切沟通的过程。
  - 改善的沟通有助于建立一个在给定领域下对需求的共同认知和理解，并且能发现潜在的误解。
  - 如果采用图形化的MBT模型, 项目的利益相关者 (如业务分析师)就能更加容易参与到项目中。

- 建模支持在给定领域中，持续改进测试工程师的能力。
- MBT模型的抽象层次使它能够更容易识别系统中那些更能通过测试发现问题的部分。
- 在系统真正实现之前，可以生成并分析测试用例。

■ 效率

- 早期的建模、模型验证和确认能帮助我们在早期的开发过程中避免缺陷。模型能提供给项目的利益相关者（比如业务分析员，开发人员）来验证需求，并能识别与需求的差异。这样，符合了尽早测试的原则。
- 之前的项目中使用的工件，在以后的开发测试时能被重用。
- 支持自动化（例如，生成测试件）。并且，使用能减少由于人工创建和维护测试件引入的错误。
- 同一模型能生成不同的测试套件（例如，根据不同的测试选择准则），在测试依据或测试目标变化时能高效地适应。
- MBT能用于不同的测试目标，并且能在测试过程中涵盖不同的测试级别和测试类型。
- 由于MBT模型提供了单一需要维护的内容，当需求变化时MBT有助于减少维护成本。

1.1.2 对MBT的误区和陷阱

准备使用MBT的测试团队应该对MBT的优点和局限性有契合实际的期望。典型的错误的期望、误解和陷阱包括：

■ MBT能解决所有问题

MBT不是取代传统的测试设计技术（见【ISTQB®基础级大纲】），但是能支持传统的测试设计技术，能让测试工程师提高对业务领域的理解，并让测试更加有效和高效。即使引入了MBT，也无法解决忽视和/或误用了传统的测试设计技术的测试团队的问题。

■ MBT只是一种工具

适当的工具支持对MBT的成功是必不可少的，但是首先考虑的不应该是获取工具。相反，决定引进MBT与否应根据测试改进的可测量的目标而定。MBT影响整个测试过程（本文1.2节）。因此，引入MBT需要得到管理层大力支持，从而才能在组织中推进过程和工具的改进。

■ 模型总是正确的

如同在手工测试设计时一样，测试工程师在构建MBT模型的时候有可能会引入缺陷。MBT模型需要通过评审、模型静态分析、模型模拟等一系列活动进行彻底的验证和确认。在MBT模型中任何的变更都会影响到所有根据变化的模型元素生成的测试件。因此，每个变更都必须在实施前非常仔细地评审。

■ 将会发生测试用例爆炸

MBT的应用改进了测试设计的方法和覆盖率。如果单纯的使用组合方法生成测试用例，这可能会导致测试用例的爆炸。这一挑战可以通过调整测试用例生成策略和算法，并通过应用智能过滤机制来解决。

1.2 在基础测试过程中 MBT 的活动和工件

1.2.1 MBT 特定的活动

当实施基于模型的测试时，测试过程会包括MBT特有的活动，这些活动一般不经常用在传统的测试分析和设计中的活动。这些活动包括：

- MBT建模活动（与MBT模型管理相关的活动,建模方法的开发和集成活动，建模指南的定义、MBT模型结构的开发和建模元素的开发等活动，例如MBT模型特定的图表,工具相关的活动等等。）
- 在基于MBT方法和选择准则的基础上生成测试件。例如测试用例的生成。

在基本测试过程中（见【ISTQB®基础级大纲】），基于模型的测试至少会影响测试分析和测试设计活动。根据不同的测试目标，MBT也可以用于更广的范围，包括基本测试过程中所有相关的活动。需要考虑到以下这些MBT特有的活动：

- 在测试计划和控制阶段可以包括实现MBT在基本测试过程中特有的活动（MBT工具、指导方针、具体的指标、作为项目计划基线的一部分的MBT工件，等等）
- 在测试分析和设计阶段的活动可以包括MBT建模活动、测试选择准则和更高的测试覆盖率度量。
- 在测试实现和执行阶段可以包括MBT测试生成和MBT测试适配。
- 在评估退出准则和报告阶段可以包括高级的覆盖度量（基于结构和/或明确的模型信息）、以及基于MBT的影响分析
- 在测试结束阶段的活动可以包括建立模型库，以便于在以后的项目中重用。

MBT对测试过程产生影响的一个关注点是过程自动化和工件的生成。与传统的测试设计相比，MBT将测试设计推进到早期的项目阶段。它可以作为一种早期的需求验证方法，可以促进及改进沟通，特别是使用图形模型时。基于模型的测试通常会增强传统测试。这为测试团队提供了一个机会来验证根据MBT自动生成的测试用例和手动创建的传统测试用例是否一致。

尽管在基本测试过程中采用MBT与传统的测试过程非常类似，但是在实施时会有显著的变化。正如在本文1.1节中所述， MBT：

- 对质量、工作量、交流沟通和利益相关者参与测试过程等方面会产生影响。
- 将测试设计活动提前到了测试过程中的早期阶段。

为了保证能正确采用MBT，对采用MBT的团队建立对方法和相关变化的接受准则是非常重要的。

1.2.2 基本的MBT工件 （输入和输出）

MBT模型可以采用专门为MBT开发的模型，也可以重用为系统设计开发的模型。（见本文2.3.5节）。根据建模概念的应用，MBT模型支持不同的抽象层次和特定的测试信息。从MBT中获得的工件将反映这些不同的抽象层次。

根据这些信息和抽象层次，MBT在与测试过程集成时会需要不同的输入工件和输出工件。

输入工件:

- 测试策略
- 测试依据，包括需求和其他的测试出口准则、测试条件、口头的信息，以及已有的设计和模型
- 在之前的测试执行活动中生成的事件和缺陷报告、测试日志和测试执行日志
- 方法和过程指导书，工具的说明书

输出工件包括不同的测试件，例如：

- MBT模型
- 测试计划的部分（需要测试的功能，测试环境等等）、测试进度表、测试度量
- 测试场景、测试套件、测试执行进度表，测试设计规格说明
- 测试用例、测试规程规约、测试数据、测试脚本、测试适配层（规约和代码）
- 生成的测试和测试依据之间的双向追溯矩阵，特别是需求和缺陷报告的双向追溯

1.3 将 MBT 集成到软件开发生命周期

1.3.1 在顺序或迭代软件开发生命周期中的 MBT

在本节中，对于软件开发生命周期模型，我们考虑2种主要的分类：顺序，例如V-模型；迭代-增量，例如敏捷开发模型。在这两类模型中，MBT都应该从根本上与项目的测试目标相适应，这样，使用MBT的优势才能在实现测试目标中体现出来。

在构建软件开发生命周期时会有很多不同方式的组织架构，在测试过程中采用MBT方法时需要根据不同情况适当的调整。例如，根据项目的测试目的，MBT在一个项目中可能专注于验收测试，而在另一个项目中可能专注于自动化系统测试。

以下各点对于顺序软件开发生命周期还是迭代软件开发生命周期都适用：

- 测试级别：
  - 由于MBT能对复杂的功能需求和预期行为进行抽象，MBT通常会用在较高测试级别（集成测试、系统测试、验收测试）。
  - MBT偶尔也被用于组件（单元）测试。但是有的MBT方法是基于解析代码注释的技术，就可以被用组件（单元）测试。

- 测试类型:
  - MBT模型可以描述系统的预期行为和/或系统的环境。因此，MBT主要使用在功能测试中。
  - 增强的、或专用的MBT模型可以用于非功能测试（如安全测试、负载/压力测试、可靠性测试）。

测试工程师通常负责/拥有MBT模型，但是也可以选择跟其他人员共享MBT模型。例如，模型（或其他部分）可以分享给非技术的利益相关者，他们可以用于确认并验证需求。

开发人员出于自动化的目的可能对模型感兴趣，特别是当在持续集成中使用MBT模型导出的自动化脚本时，这些导出的自动化脚本可以对持续集成提供早期的、持续的反馈。

在顺序的软件开发生命周期中，MBT的特别活动包括：

- 通过对风险评估，考虑相关的项目环境和测试目标，将基于模型的测试作为项目测试策略的一部分进行集成，包括从需求到MBT模型元素的追溯性信息。
- 在项目中尽早实施MBT建模：
  - 激励利益相关者的沟通。
  - 使得能在早期发现不清晰的、不完整的和不一致的需求。
- 适应测试计划的活动和角色包括：
  - 新的测试件的元素（例如, MBT模型、测试选择准则）。
  - MBT活动的进展报告。

在敏捷开发生命周期模型[ISTQB®基础级扩展-认证敏捷测试工程师大纲]中，MBT主要的变化如下：

- MBT模型的开发是以迭代和增量形式开发的，测试相关的内容是根据迭代的内容生成的。
- 用户故事（user story）是测试依据的一部分。每个被覆盖到的用户故事会跟模型联系起来，这样就能在用户故事和测试之间建立可自动管理的双向追溯。
- MBT可以用来实现验收测试驱动开发（ATDD）。
- 在全团队敏捷的最佳实践中，使用基于模型测试的测试工程师可以和开发人员以及业务相关人员一起作为敏捷团队的一部分（见ISTQB® 基础级扩展-认证敏捷测试工程师1.1节）。

1.3.2对需求工程的支持

MBT在以下方面支持需求工程：

- 提供的图形化的MBT模型，可以促进业务、开发和测试工程师对于将要开发的软件在期望行为方面的交流。例如，MBT可以提供业务流程模型和/或状态图。这些模型能帮助所有的利益相关者讨论和确认被测系统具体的期望行为。

- MBT能清晰的阐述并提高需求和/或用户故事的质量。通过与业务相关人员和/或开发人员分享所有的或部分的MBT模型，MBT能帮助大家对被测软件的期望行为有一个共同的理解。通过MBT模型导出的测试组合了可能的用户场景，能够被业务人员和/或开发人员进行评审。

- MBT能支持需求的早期确认，甚至在需求仍然需要变更的情况下。

需求工程和MBT相互作用。需求及其相关的风险是MBT建模和测试生成活动的输入，并且MBT的活动可以把在需求和测试之间创建和维护双向追溯联系的工作自动化。

2 MBT 建模 - 250 分钟

关键词

测试模型（test model）

MBT 建模的学习目标

2.1 MBT建模

FM-2.1.1 (K3) 使用基于工作流的模型语言，针对一个测试对象和预定义的测试目标，开发一个简单的MBT模型（参考9.1节-“简单”表示少于15个模型元素）。

FM-2.1.2 (K3) 使用基于状态迁移的模型语言，针对一个测试对象和预定义的测试目标，开发一个简单的MBT模型。（参考9.2节-“简单”表示少于15个模型元素）。

FM-2.1.3 (K2) 根据主题和关注点对MBT模型进行归类。

FM-2.1.4 (K2) 给出MBT模型是如何依赖于测试目标的例子。

2.2 MBT模型语言

FM-2.2.1 (K1) 记忆常用的MBT的各类模型语言的例子。

FM-2.2.2 (K1) 记忆针对不同的系统和项目目标，了解典型的建模语言类的代表。

2.3 MBT建模活动的好的实践

FM-2.3.1 (K1) 记忆MBT模型的质量特性。

FM-2.3.2 (K2) 描述在MBT建模活动中典型的错误和陷阱。

FM-2.3.3 (K2) 解释将需求和过程相关的信息与MBT模型联系起来的优势。

FM-2.3.4 (K2) 解释MBT建模指导方针的必要性。

FM-2.3.5 (K2) 给出例子说明什么时候适合或不适合重用已有模型（来自需求阶段或者开发阶段）。

FM-2.3.6 (K1) 记忆支持特定的MBT建模活动的工具类型。

FM-2.3.7 (K2) 对迭代的MBT模型开发、评审和确认进行总结。

2.1 MBT 建模

在测试相关活动中, MBT模型非常适合用来描述需要测试什么，适合用于利益相关者之间的沟通，适合用来总结测试设计相关的所有信息。模型也能在一定程度上适用于测试管理活动。

开发一个MBT模型的目标是生成（或识别出）测试用例。MBT模型应该为此后的测试用例生成提供必要的信息，例如，可以根据项目测试目标选择合理的测试用例，允许生成测试结果参照物，并且能支持需求和生成的测试用例的双向追溯。

MBT模型的设计是MBT中一个基本的也是必要的活动。MBT模型的质量对基于MBT测试过程的结果的质量有很大的影响。MBT模型通常通过工具来进行解释，因此必须遵循严格的语法。 MBT模型还可能需要支持其他质量特征，比如需要支持建模指导书所定义的质量特征。

2.1.1 MBT 建模活动

每个模型都是由一种特定的建模语言来实现。建模语言定义了工件，工件由模型元素和用该建模语言构建模型时需要遵循的规则组成。主要的建模语言有表述结构的、表述行为的或表述与领域相关的工件的。

以下是在MBT建模时需要考虑到的重要问题：

- 测试对象的什么质量特征需要建模？

在通常情况下，我们一般会关注被测系统的功能性，但是性能因素和其他非功能方面，例如安全性也会是MBT模型的一部分。为了保持模型可管理和以及MBT测试过程的高效，我们推荐在建模时，只考虑与测试和测试目标相关的系统或者它们的环境因素。

- 哪种建模语言合适？

每个模型都是由一种特定的建模语言来实现。建模语言定义了工件，包括模型元素，和用该建模语言构建模型时需要遵循的规则。使用不同的建模语言来表述测试对象的结构、行为或者其他方面（例如，数据、工作流、通讯协议），或者表述测试对象的环境时，不同的建模语言之间有很大的差异。

- 什么是合适的抽象层次？

抽象对于掌控复杂情况并使MBT模型专注于测试目标是有用的。然而，模型越抽象，就需要更多的与测试执行适配的工作。此外，抽象层次决定了是否能与听众/利益相关者就测试设计进行讨论。

这些问题的答案以及选定的MBT工具套件将会影响到MBT的活动。

注意：附录A给出了2种简单的图形化的建模语言。培训和考试中应该覆盖这2种语言的使用。

2.1.2 MBT 模型的主题和关注点

在本大纲中，我们考虑三种典型的模型：

- 系统模型

系统模型描述了将要被实现的系统。根据该模型生成的测试用例可以检查系统是否与模型一致。系统模型的例子有用于面向对象系统的类图，或者描述系统反应的状态和状态转换的状态图。

- 环境模型/使用模型

环境模型描述了系统的环境。环境模型的例子包括马尔可夫链模型（Markov chain models），该模型描述了期望的系统使用模式。

- 测试模型

测试模型是一种（一个或多个）测试用例的模型。典型的测试模型包括测试对象的预期行为以及对测试对象的评价。测试模型的例子有（抽象）测试用例描述或用图形表述测试规程。

一个MBT模型通常会综合几个或全部的这些主题。例如，一个MBT模型可以描述测试对象，包括数据元素和在一个给定的环境中测试对象的使用方式。

MBT模型的关注点可以是结构、行为，或者综合这两个方面：

- 结构模型描述了静态结构。结构模型的例子有类图和接口规格说明或针对数据建模的分类树。
- 行为模型描述了动态交互。行为模型的例子包括活动图或描述活动和工作流的业务处理模型，以及描述系统输入输出的状态图。

一个MBT模型通常结合了结构方面（例如，描述测试对象接口）和行为方面（例如，测试对象的期望行为）。

2.1.3由测试目标决定 MBT 模型

当开发一个MBT模型时，重要的是要考虑到测试目标。因为测试目标将会决定模型的主题和关注点。在下面的表格中，可以看到测试目标和针对测试目标的所应具有合适的模型特征。

测试目标	模型例子	主题	关注点
验证是否正确地实现了业务工作流	一个描述工作流的业务处理模型	系统	行为
验证在特定的状态下，系统是否按要求给出了正确的反应	一个UML（统一建模语言）状态机	系统	行为
验证接口的可用性	用来描述接口结构（结构模型）	系统	结构
测试对象的功能符合用户所期望的使用方式	一个用来描述用户行为的使用模型	环境	行为
验证系统的配置	一个使用分类树的数据模型	数据	结构

2.2 建模语言

2.2.1 MBT 建模语言的主要分类

模型是通过建模语言来展示的。建模语言通过以下几种方式来定义：

- 根据概念（也称为抽象语法，经常是用文字描述的形式。但是有时候也会用元模型来定义）。
- 根据语法 （也称为具体语法，经常根据语法规则来定义）
- 根据语义 （通常根据静态和动态语义来定义）

图形化建模语言通过图形来表现模型。例如类图、时序图或者UML里面的状态机图。

有很多种建模语言能用于构建MBT模型。为了选择一个好的“适合目的”的语言，测试工程师需要了解不同MBT建模语言的主要特征：

- 建模概念

不同的建模语言支持的概念集是不同的。根据测试目标的不同，模型可以展示结构信息（例如，架构、组件、软件接口），数据信息（例如，软件中被动对象的格式和语义）或者行为信息(例如，场景、交互、对软件中活动的对象的执行)。也可以参见本文2.1.2节

- 形式化

一个建模语言的形式化程度可以从有限的形式化到完全形式化。后者是最严格的，可以让模型能进行全面分析；而前者往往更具有可操作性。至少，一个建模语言应该有一个正式的语法（例如，包括结构化文本或结构化表）。它也可以有一个正式定义的语义。

- 描述形式

建模语言可以使用不同的描述形式，可以从文字到图形形式，也可以二者结合。图形方式通常更方便用户使用并且易懂；而文字形式通常更易于工具的使用，可以高效的编写和维护。

建模语言的分类包括：

- 针对结构模型的建模语言

这类建模语言支持对软件的结构化元素的定义和说明，例如对接口、组件和层级结构的定义和说明。结构化建模语言的例子有UML构件图。

- 针对数据模型的建模语言

这类建模语言支持对数据类型和值的定义和说明。数据模型的建模语言的例子包括UML类图和值的定义和说明。

- 针对行为模型的建模语言

这一类的建模语言支持事件、行动、反应和/或软件交互的定义和说明。这类建模语言的例子包括UML活动图或交互图，状态机或业务过程建模符号（BPMN）。

- 集成语言

通常情况下，建模语言不局限于某个方面，而是为多个方面提供概念。一个例子是UML本身，这些不同的图可以通过组合来表示软件的不同方面。

2.2.2根据不同的系统和项目目标对建模语言分类

MBT 建模语言的选择与初始的项目目标紧密相关，考虑此项目开发和需测试的软件，同样还涉及到正在开发的软件的系统属性。在综合考虑后，就能得到一系列的建模语言准则。这需要在基于相关系统特有属性的基础上评估这些准则，接受一些需要的准则，放弃一些其它准则。

例子包括使用：

- 状态图来描述一个控制/命令行形式的测试对象的期望行为。
- 活动图或者BPMN模型来描述一个信息系统端对端测试的工作流。
- 决策表和因果图来描述系统测试的业务规则。
- 时序模型来描述在与时序有关的测试用例中测试对象的反应
- 特征模型来描述软件产品线环境中的变化

建模语言与项目目标之间的关系更多的例子包括：

- 对于人身财产安全性关键(safety-critical)和信息安全性关键(security-critical)软件进行认证的非功能性要求，一般需要把软件需求与代码和测试用例连接起来。
- 对于为了审计和认证而进行过程文档化的非功能性要求，需要模型支持方便文档化的注释。

2.3 MBT 建模活动的最佳实践

2.3.1MBT 模型质量特性

模型的质量直接影响到生成的产物。模型的质量特性包括：

- 语法质量（正确性）
  MBT 模型与形式化描述（建模语言、建模指导书）的规则一致。这样才能根据MBT模型生成测试用例或测试数据，不会因为错误的语法而出问题。

- 语义质量（有效性）
  模型的内容是正确的，表述了需要表述的内容；生成的产物是“有用的”（测试脚本能被执行，手工测试用例能被测试工程师执行，不会因为不正确的测试用例而发生错误）。

- 适用性质量(适用)
  MBT模型对于指定的测试目标和指定的测试生成器是合适的。这样，根据MBT模型导出的产物才能完全符合预期。

工具可以检查语法和至少可以检查一部分模型的语义。通过评审检查语义及适用性的质量。模型模拟器通过提供对模型质量的动态检查，可以完成静态技术工作。

在此推荐用增量模式开发MBT模型，重复生成（和执行）导出的测试。这样，就能在早期并定期地检查模型、产出的工件和测试对象。

2.3.2 MBT 模型设计的典型错误和陷阱

对于MBT的初学者来说，在进行模型设计的时候通常容易犯的一些错误包括：

- 在MBT模型中放入太多或太少的细节（对于指定的测试对象抽象层次错误）。
- 想要在一个MBT模型中涵盖所有的东西。

MBT 模型应该关注与达到测试对象相关的方面。相对于用一个模型覆盖所有的方面，我们推荐使用两个模型来分别强调特定的方面（例如，用状态图来验证系统实现情况，用活动图来确认工作流）。

2.3.3 将需求和过程相关信息与 MBT 模型联系起来

在基于模型的测试中，建立需求、MBT 模型以及生成的测试之间的追溯关系是一个很好的做法。通过链接模型元素和需求，们可以有如下好处：

- 方便评审MBT模型。
- 根据MBT模型生成的测试工件可以自动与需求关联。
- 能自动生成与选择的需求相关的测试件，也能根据选择需求的优先级决定优先执行哪些测试用例。
- 能够根据MBT生成的测试件来衡量需求覆盖。
- 能让所有的利益相关者（例如，测试工程师和测试经理）分析需求变更的影响，从而决定由变更引起的回归测试的范围。
- 测试用例生成器可以自动的生成跟踪文档（跟踪矩阵）。

要将需求和模型元素连接起来，有几种实现方式可供我们选择。例如，一种特定的图形符号或者文字形式的关键字可以表述在模型中的需求。

测试工程师可以在已有的模型元素中增加其他的附加的信息，可以是有特殊意义的新的模型元素也可以是现有模型元素的属性。MBT模型的附加信息包括：

- 测试规程规格说明的详细内容（测试执行动作的顺序）
- 测试脚本片段（功能调用或关键字）
- 风险/危害（与功能性的和非功能性的质量属性或项目管理相关）
- 优先级 （功能的或测试的）
- 持续时间 （估计的执行时间）
- 必要的测试设备或规则来配置测试产品

附加信息有助于基于模型的测试集成到整个测试过程和支持测试管理。体现在以下几个方面：

- 风险或优先级信息可以被链接到MBT模型，并与产生的测试用例相关联。这样，风险或优先级信息就能被利用起来。（例如，对测试执行设定优先级）。

- 任何其他项目的约束和目标都可反映到MBT模型。这样，这些信息就能用来适配测试计划。
- 当接口（例如，图形用户界面）在规格说明中尚未稳定时，MBT模型可以包括已定义的功能要求。这样，功能性的关键字驱动脚本可以更早地产生，同时自动化阶段可以提前开始。一旦接口被定义，自动化测试脚本已经准备就绪，那就只剩实现测试适配了。

2.3.4 MBT 建模指南

建模指南记录了如何设计、写和读MBT模型的说明。 这些说明包括：

- 支持所有利益相关者参与的对MBT模型的理解和评审。
- 支持源自过程、领域、组织或特殊工具的语法一致的需求。
- 限制/扩展所选符号的范围（例如，定义一个UML的子集或元素不同的含义）。
- 提倡不同的作者在MBT模型中使用类似的语法和语义。
- 传授好的经验。
- 支持MBT模型的维护性和可重用性。

没有建模指南的MBT项目要比有建模指导书的项目承担额外的风险，没有建模指南，测试工程师会更容易犯建模的典型错误。

2.3.5 重用已有的系统设计或需求模型

当根据“经典的”测试依据文档来开发MBT模型时，就有可能重用现有的系统设计或需求模型（例如，业务过程模型或活动图）直接作为MBT模型的一部分，但这种方式也有其局限性。可重用的程度取决于给定的模型和测试目标。例如，从系统设计产生的状态机图可用于测试命令和控制系统，但类图不足以用于基于工作流的系统测试。

重用现有的系统设计模型会导致一个单一来源的问题，在原有系统设计模型中的任何错误都会延伸到MBT模型中。这种情况下，虽然理论上来说测试应该是独立的，但是实际上测试不是独立于系统开发和实施的。因此，建议由不同的作者写互相独立的模型：一个用于系统设计，一个用于MBT。这样，就能满足职责分离，以及促进（测试的）独立性。

如果没有任何修改的重用，从系统设计模型导出的测试工件只能用来做验证测试（对比确认测试）。他们检查系统的实现是否符合系统设计模型中描述的规范。

为了达到确认的目的，测试工程师可以从一个已存在的模型开始（例如，从需求获取），用测试的内容丰富它。一旦转换到MBT模型，原来的系统设计模型就会进化。测试工程师将增加模型元素（例如，没有预见到的迁移），以测试不寻常的场景或错误的情况。最后，MBT模型的结果反映了测试工程师的考虑角度，而不仅是原来的系统设计。

考虑到现有模型的作者不是一个测试工程师的情况。其结果是，以前开发阶段生成的模型不一定尊重 MBT 相关的要求和准则。 一个重用模型并不一定会遵循（工具依赖）推荐的做法，可能会导致测试用例爆炸。

2.3.6 建模活动的工具支持

模型编辑器是用来编写模型的，它可以是一个特定的建模工具或任何其他流程图编辑器（用于图形模式）或文本/脚本编辑器（文本模式）。原则上， MBT模型能用铅笔和纸来完成，虽然简单的绘图工具可能足以完成文档化的目的，但是它们不支持MBT模型的进一步处理。

一个MBT模型编辑器可以提供预定义模型元素，以及有可能做一些语法和语义的检查。执行模型的不同路径以确认MBT模型的工具被称为模型模拟器。

为了从MBT模型中自动化的获得测试用例和/或测试数据，必须要有一个测试用例（和/或测试数据）生成器一个测试用例生成器（和/或测试数据）是必须的。

2.3.7 迭代的模型开发, 评审和确认

用多个图对更复杂的行为进行建模，容易导致对单个MBT模型的评审不够充分全面。测试工程师必须确保从MBT模型导出的测试工件能够完全覆盖期望。在迭代的开发中，模型评审以及经常评审产生的测试工件，这些工作首先由测试工程师来承担，后由同伴，以及其他的利益相关者在软件生命周期中一起参与，这也是很好的管理方法。

采用这些最佳实践可以:

- 测试工程师和其他的利益相关者确认他们的观点是针对要被测试的方面。
- 测试工程师能在MBT模型开发早期（在测试执行前）发现和修复模型的错误和缺陷以及模型的缺损部分。
- 测试工程师识别和交流不完整和不一致的需求。
- 测试经理管理项目相关的风险。
- 在测试过程中团队能减少完成MBT建模活动所需的总的时间。

迭代的MBT模型开发也与迭代的测试生成相关。这意味着每次一些东西加到MBT模型时，测试生成也会更新。从本质上讲，测试生成构成了对MBT模型的模拟，它提供了对要被测试的行为方面的反馈。

3 生成测试用例的选择准则– 205 分钟.

关键字

覆盖项(coverage item)，模型的覆盖(model coverage)，测试用例爆炸(test case explosion)，测试选择准则(test selection criteria)

MBT 测试选择准则的教学目标

3.1 MBT 测试选择准则的分类

FM-3.1.1（K2）根据模型的测试生成所使用的测试选择准则的归类

FM-3.1.2（K3）从 MBT 模型生成测试用例，以实现给定环境中给定的测试目标

FM-3.1.3（K2）提供模型覆盖、数据相关、基于模式和场景、以及基于项目测试选择准则的示例

FM-3.1.4（K2）识别 MBT 测试选择准则如何与 ISTQB®基础的测试设计技术相结合

3.2 测试选择准则的应用

FM-3.2.1（K1）记忆测试工件生成自动化的程度

FM-3.2.2（K3）将给定的测试选择准则应用到一个给定的 MBT 模型

FM-3.2.3（K2）描述 MBT 测试选择准则的良好实践

3.1 MBT 测试选择准则的分类

从同一 MBT 模型，可以生成各种测试集。使用测试选择准则帮助测试工程师选择最适合测试目标的有意义的测试用例子集。

测试用例生成器在使用 MBT 来改进效率方面起着关键作用。

在学术上，MBT 测试选择准则已经被广泛研究（见【Utt07】，【Wei09】，【Zan11】）。本大纲介绍了六类测试选择准则。其中一些专注于特定项的覆盖，另一些是支撑项目管理方面、特定场景的测试、或者随机测试方面。

3.1.1 测试选择准则

基于覆盖的测试选择准则与从 MBT 模型生成的覆盖项相关。可能的覆盖项有：

- 与MBT模型关联的需求

该准则要求 MBT 模型元素与选择的需求相关联。完整的需求覆盖对应于一组测试用例，而这组测试用例完全覆盖了所选的需求集（每个需求至少被一个测试用例覆盖）。

■ MBT模型元素

模型覆盖是基于 MBT 模型的内部结构。测试工程师、测试经理或其他任何相关角色参与定义覆盖项并选择一组测试用例集来覆盖要求覆盖的一定数量的项。作为覆盖项的可能模型元素如下：

- 状态图的状态、转换和判定（见【ISTQB®基础级大纲】中的“状态转换测试”）
- 业务流程模型的活动和通道
- 决策表的条件和行为
- 文本模型的语句和条件

■ 数据相关的测试选择准则

测试设计技术相关的标准（见【ISTQB®基础级大纲】）示例：

- 等价类划分
- 边界值分析

还包括启发式如配对比较，更普遍的是，n-元组合测试用例生成（更多信息请参见【ISTQB®高级测试分析师大纲】）。

其他的测试选择准则包括：

■ 随机

此选择标准好比模型的走查（或多或少的随机）。在随机测试用例选择时，不同的选择都是等概率的。随机的测试用例选择还可以考虑不同的选择带有不同的概率。

■ 基于场景和基于模式

测试用例的选择是基于预定义的场景或模式。一个场景可能是一个显式定义的用例（参见【ISTQB®基础级大纲】中的“用例测试”）或一个显式定义的用户故事（见【ISTQB®基础级扩展-敏捷测试大纲】）。模式是一个部分定义的场景，这可能被应用于 MBT 模型来产生几个或多个测试。

■ 项目驱动

项目驱动的测试用例选择是基于额外的项目相关信息，它已添加到模型中来支持测试管理和/或实现项目中特定的测试目的。项目相关信息包括针对此项目的风险、优先级、需要的测试设备或任何其他方面。选择依赖特定测试设备的所有测试用例相当于在应用项目驱动选择准则。

MBT 工具至少支持一种，但通常不是上面提到的所有测试选择标准。

3.1.2 实践中的测试用例选择

实践中，测试工程师应该考虑结合若干测试选择准则来获取最适合测试目标的测试用例子集。这些组合的例子是需求覆盖加模型覆盖或者场景/模式加数据相关的测试选择准则。

除了目标对象，测试选择准则的应用还取决于以下几点：

- MBT 工具提供的机制
- 模型的设计
- 测试工程师与准则相关的经验

有可能出现的情况是，MBT 模型形式上是正确的，但由于工具特定的生成算法导致了测试用例数的爆炸。

3.1.3 测试选择准则示例

与一些建模语言相关的测试选择准则的例子（见【Utt07】——第四章）：

- 活动图或业务过程模型：
  - 活动覆盖（100%=覆盖图中的每个活动）
  - 判定/通路覆盖（100%=覆盖图中的每一个判定点）
  - 路径覆盖（100%=覆盖图中的所有业务场景），有或者没有循环
- 状态图：
  - 状态和迁移覆盖（100%=独自覆盖每个状态或迁移）
  - 迁移对的覆盖（100%=覆盖图中每个连续的迁移对）
  - 路径覆盖（100%=覆盖图中通常没有循环的所有路径）
- 决策表：
  - 每一个条件
  - 每一个动作
  - 每一条规则
- 定义于结构部分的模型的数据域：
  - 等价类覆盖（例如，每个类一个代表值）
  - 边界值覆盖（例如，考虑到数值数据的边界范围）
  - 定义域的结对测试（确保产生每一对输入参数的所有可能的独立组合）
- 文本模型：
  - 语句覆盖（100%=每个可执行语句均被覆盖）
  - 判定（判断）覆盖（100%=每一个判定均被覆盖）
  - 判定（判断）条件覆盖（100% =所有条件的结果和判定结果被测试集所执行）。
  - 多重条件覆盖（每一个判定的所有条件组合覆盖，会导致生成大量的测试用例）。

3.1.4 与基础级测试设计技术的关联

MBT 支持标准 ISTQB®基础级测试设计技术，例如状态转换测试、等价类划分、边界值分析、决策表测试和用例测试。MBT 模型本身可能包含测试设计技术的覆盖项（例如，MBT 模型中描述的等价类划分或边界值）。另外，MBT 模型可能与含有覆盖项的其他图形或文本的描述相结合（例如，决策表或其他的图）。

3.2 测试选择准则的应用

3.2.1 测试生成的自动化程度

MBT 通常与测试用例的自动生成相关， 但测试用例生成并非必须基于工具。下列方法可用于测试用例生成：

- 手动测试生成——从 MBT 模型手动推导测试用例，依据路径人工记录下对应的测试用例。然而，此 MBT 方法的成熟度较低，无益于 1.1 节中提到的自动生成测试用例的效率和有效性。
- 自动测试生成——通过 MBT 工具自动生成测试用例等输出构件，无需进一步的处理即可使用。
- 半自动测试生成——使用工具时有一个折中解决方案，但在中间或后期需要手动步骤，例如选择特定的测试用例。

3.2.2特定测试选择准则的利弊

每个测试选择准则都有其优点和缺点。选择什么样的测试选择准则取决于测试目的。各种测试选择准则的优点（加号）和缺点（负号）的示例：

- 需求覆盖
  （+）常规强制性的监管要求
  （-）必要的精确定义（例如，如果一个需求关联于模型中的若干行为，当覆盖所有或仅有的一个行为即可实现需求覆盖）
- 模型覆盖
  （+）有助于理解模型中究竟哪些地方无法覆盖以及了解模型的测试覆盖率的完整性
  （-）一些结构性测试选择准则在测试生成时可能会导致测试用例的暴增（例如，活动图的所有路径覆盖）
- 数据相关的测试选择准则
  （+）常规强制覆盖等价类划分的域
  （-）可能过度的组合
- 随机
  （+）有助于选择意想不到的测试用例
  （-）随机算法可能产生没有业务价值的长测试
- 基于场景/模式
  （+）支持选择用例（例如，回归测试）
  （-）需要额外的工作来定义和维护场景和模式

- 项目驱动
  （+）有助于测试管理
  （-）需要额外的工作去关联模型的特定信息

3.2.3 MBT 测试选择的良好实践

通常，单一准则并不足以涵盖针对的测试目的所需要测试的方面。在这种情况下，测试工程师必须结合若干准则。

准则组合的两种方法：

- 准则的组成——生成的测试套件仅包含那些已应用的满足所有准则的测试用例（交集）
- 准则的增加——生成的测试套件包含完成至少一个准则的所有测试用例（并集）

选择准则可能影响 MBT 的建模概念。此外，必须能够重现该选择，也就是说，对于相同的模型应用相同的准则可获得相同的测试用例集。因此，必须规划测试选择活动，并文档化记录选择活动和背后的原因。

对于一些模型覆盖准则，存在若干路径集，它们能满足所需的覆盖。MBT 模型中的微小变化可能会导致一个完全不同的选择，这需要一种新的思维方式。以模型为基础，所生成的测试用例作为衍生的工件。

测试选择准则也是 MBT 中掌控测试用例爆炸的一种方式；通过使用单一的准则或准则的组合，测试工程师可以精确调整测试生成以满足测试目标并避免测试用例的暴增。

4 基于模型测试的实现和执行– 120 分钟.

关键字

离线 MBT（offline MBT），在线 MBT（online MBT），测试适配层（test adaptation layer）

基于模型测试的实现和执行的学习目标

4.1 基于模型的测试(MBT)规范

FM-4.1.1 (K2) 解释在 MBT 环境中的抽象和具体测试用例的不同点

FM-4.1.2 (K2) 解释 MBT 环境中不同类型的测试执行

FM-4.1.3 (K3) 当需求、测试对象或测试目标变更时，更新 MBT 模型和测试生成

4.2 MBT 测试中的测试适配活动

FM-4.2.1 (K2) 解释 MBT 中的测试执行需要哪种测试适配工作

4.1 MBT 测试实施和执行规范

一旦生成了测试套件，下一步就是运行测试用例。测试执行可以是手动执行，也可以是通过一个测试能够自动执行和记录测试结果的测试执行环境来自动执行。无论自动或手动，MBT 模型和测试执行的工件必须有关联。

4.1.1 MBT 环境中的抽象和具体测试用例

基于模型测试的生成可以产生抽象的测试用例（高级别的测试用例）或具体的测试用例（低级别的测试用例）

- 抽象的测试用例是指那些没有输入和期望结果的具体值（实现层面的）的用例。也没有非常具体的测试步骤。
- 具体测试用例对于输入值、期望测试结果等都有具体值（实现层面的），并且有非常详细的测试步骤描述。

MBT 的模型能包含不同层次的抽象以面向生成不同的输出工件，以及面向不同的利益相关方。例如：

- 抽象测试用例可以面向业务分析代表的评审
- 具体测试用例可以直接由测试工程师执行

基于模型的测试（MBT） 生成的测试用例提供了没有具体的数值的情况下考虑测试条件、输入值以及期望结果的信息。这些测试信息在某个抽象层次上进行描述。例如：

- 它可能定义测试规程，包括高级别的测试动作序列，但不包含具体和完整定义的测试动作。

- 需要提供等价类的划分，而不是具体的等价类中的代表值。

从抽象的测试用例到具体用例并能用于执行（手工或自动）需要将抽象测试用例按照以下的要求补充信息：

- 测试动作的完整定义
- 具体和完整的输入数据值
- 给出期望结果的具体数值

这样的补充性信息可以在 MBT 模型中定义（比如，作为动作、验证和数据这样的模型工件的文档）或定义在 MBT 模型之外（比如使用一个数据表格来映射抽象和具体数据值）。

4.1.2 不同类型的测试执行

MBT 生成的测试用例可以手工或自动执行。

对于手工测试执行，测试工程师执行由 MBT 工具生成的测试用例，这些测试用例必须以可用于手工测试执行的格式生成。同时，如果导出的测试用例还能由测试管理工具进行管理，则会带来更大的好处，这时可按照 ISTQB®的基础测试过程来管理测试执行和缺陷。

对于自动化测试执行，这些生成的测试用例必须是适合用于自动执行的格式。有三种主要的方法通过抽象测试用例来实现自动化测试执行（参看 [Utt07] – 第 8 章）：

- 适配法-在这种方法里，测试适配层（test adaptation layer）的代码用来弥补抽象层次与具体测试用例之间的差距。这样的代码本质上是围绕测试对象的一个封装（类似于关键字驱动这样的测试方法）
- 转换法-在这个方法中，MBT 生成的测试用例直接转换为自动化测试脚本（不需要测试适配层）
- 混合法-这是前两种方法的组合方法。其中抽象的测试用例被转换为具体的测试用例，而这些具体测试用例被围绕着测试对象的适配层代码进行了封装。

一旦生成自动化测试脚本，这些脚本会被测试执行工具执行，MBT 工具可以将生成的测试用例转换为测试执行工具支持的测试自动化脚本语言定义的形式， MBT 工具还可以将生成的自动化测试脚本发布到测试管理工具中（统一进行管理）。

在 MBT 环境中有两种自动化测试执行的方式：

- 离线 MBT-先生成自动化测试脚本（包含期望结果），然后执行。
- 在线 MBT （也称为 “即时”）– 测试生成和执行同时进行。即在测试中，每个测试步骤在执行前一步后才生成。执行结果可能影响到模型中后续路径的选择。

4.1.3 MBT 的测试工件受变化的影响

软件项目里，变化无处不在。以下是能被预见到的可能发生的变化：

- 需求变化，测试对象的变化或测试对象所处环境的变化可能影响到：
  - MBT 模型
    - 动作
    - 条件/期望结果

    - 数据
  - MBT 测试选择准则
  - 适配层的规范和代码（如果存在的的话）
- 测试对象（非功能性需求）的接口变化，比如 用户界面（UI）的小变化但不影响功能行为，会影响到：
  - 适配层的规范和代码（如果存在的的话）
- 测试目标或测试条件的变化会影响到：
  - MBT 模型本身
  - MBT 测试选择准则
  - 适配层的规范和代码（如果存在的的话）

MBT 工件的变更管理应该基于一个流程，包括影响分析，探索变更选项以及在工件上应用变更和评审活动。

4.2 MBT 中的测试适配活动

在手工执行测试的情况下，测试适配与生成的测试文档有关，文档里应当弥补了 MBT 模型中的抽象信息与被测系统的具体接口和测试数据的差距。举例而言，具体的数值可以作为特定边界值的代表值予以列出。适配过程确保了提供的手工测试脚本是完整的，并有足够的文档描述，从而可以直接用于手工测试。

在自动化测试执行的情况下，测试适配的过程就是在适配层规格说明的基础上集成 MBT 模型生成的工件到测试执行框架里。该过程支持关键字驱动测试及/或数据驱动测试的实践。

对于关键字驱动测试，关键字应在 MBT 模型里定义好并用于生成测试用例。为了得到完整的自动化测试脚本，以下的步骤和活动是必须的：

1) 以测试脚本的形式导出测试用例，脚本语言采用测试执行工具的语言。可以手工或者利用 MBT 工具或导出器自动地完成。
2) 采用测试执行工具支持的语言编写的适配层来实现关键字。可以由负责测试自动化的测试工程师或由测试对象的开发人员来完成。

对于数据驱动测试，MBT 模型描述了抽象的输入值和期望结果（比如基于等价类划分）。这些生成的测试用例或脚本对应抽象的输入和期望结果。

为了得到完全的自动测试脚本，需要做到以下的步骤和活动：

1) 在适配层规范说明中提供的具体输入值和期望结果值。这些数据可以存在一个表格或电子表格内。
2) 在测试执行工具或测试用具（test harness）里把这些具体数据和在 MBT 模型的形式化测试数据链接到一起。

每一个测试脚本假设了特定的初始条件。为了能够顺序连续执行这些自动化脚本，必须确保在每个测试脚本执行前正确设置了前置条件，可以通过以下方法来实现：

- 在测试脚本里提供后置条件（不是每次都能够），也即作为下一个测试脚本的前置条件。
- 在测试脚本的开始部分设置前置条件。

在 MBT 建模活动前，测试适配层需要准备妥当。比如，在模型元素开发时，同时开发适配层的规范说明。

5 评估和部署 MBT 方法– 60 分钟.

关键字
无

评估与部署 MBT 的学习目标

5.1 评估一个 MBT 的部署

FM-5.1.1 (K2) 描述引入 MBT 的 ROI 因素。
FM-5.1.2 (K2) 解释项目目标是如何关联到 MBT 方法的特征的。
FM-5.1.3 (K1) 记忆用于测量 MBT 的过程与结果所选取的度量和关键绩效指标。

5.2 管理与监控 MBT 的部署

FM-5.2.1 (K1) 记忆部署 MBT 时对于测试管理、变更管理和协同工作时的良好实践。
FM-5.2.2 (K1) 记忆 MBT 的成本因素。
FM-5.2.3 (K2) 给出 MBT 工具与配置管理、需求管理、测试管理和自动化测试工具进行整合的例子。

5.1 评估一个 MBT 的部署

MBT 方法在一个组织的部署通常采用按照一个产品的典型流程：

1. 认知－ 在测试过程中建立起改进的目标，并且识别 MBT 为一种可行的技术来对应一个或多个这样的改进目标。确立可能的改进对于改变测试团队的工作方式是一个重要的积极性因素。

2.兴趣－ 了解更多的关于 MBT。

3.评估 － 分析 MBT 的主要原则和现有的 MBT 方法对于它们在给定项目上下文中的适用性。

4.试用 － 定义关键绩效因素（KPI），建立试点项目对改进作评估。

5.采用 －在组织中通过提高技能和改变行为来领导和强化变化。

因此，投资回报率（ROI）的估算和 MBT 对于测试项目的绩效的影响（基于基础 KPI）的评估能力是上述采用周期中的一部分。

5.1.1 引入 MBT 的 ROI 因素

MBT 产生的成本发生在引入和运作阶段。对于一个成功的 MBT 导入，成本可能会被以下的收益所抵消：

- 需考虑在长期范围内完整测试过程的净财务平衡
- 测试设计质量和对完整开发过程的正面影响

在测试过程和测试质量保障中都必须考虑成本、节约、改进。

成本 (参考 5.2.2):

- MBT 的引入
- 在测试过程中运行 MBT 的成本

节约：

- 早期的需求验证，因此避免了在开发后期阶段修复程序错误的高额成本：由 MBT 的自上而下的方法允许在项目的任何阶段开展 MBT 测试，即使需求还没有固定或者只有粗略的定义，也有机会在当前需求的成熟度上进行需求确认。

- 通过重用测试设计工件以及避免冗余来减少测试设计中投入的精力：在一个 MBT 模型中，相对于作为具有冗余的手动测试或自动化测试脚本的测试序列套件，而作为 MBT 模型的一个工件中整合了所有可能的路径，并且没有冗余。

- 通过从 MBT 模型中自动生成测试工件来减少测试实现中投入的精力：一旦开发出 MBT 模型，就能自动化地生成测试用例，减少了维护活动的成本。

- 尽早地识别缺陷和故障，改善发布时间：这能促进 MBT 尽早开始测试设计和提供更高效能的能力。

- 对测试管理支持：

  通过使用不同的 MBT 测试生成策略和过滤机制，测试经理可以选择出 “理想”的测试用例集来最好的满足测试目的。

- 由于优化的算法，使用 MBT 有可能减少测试用例的数量（与在测试用例设计中不使用 MBT 相比），因此减少了需要执行的测试用例数量。基于一致的和系统性的生成策略，测试用例可以被系统性地选择和减少其数量。

对测试质量的好处：

- 改善的测试设计方法和测试设计的一致性
- 增进对于测试覆盖的了解，同时也提高了测试设计质量
- 测试管理支持使用 MBT 的输出，这些输出包括优先级和测试设计的质量保证
- 通过改善测试设计的内容和组织来提高可追溯性

5.1.2 组织的目标与 MBT 方法的特征之间的关系

MBT 可以改善测试流程，例如测试质量、节省成本和沟通等。组织内部必须对于各种需要的改进点进行优先级排序。改进点的范围和它们的质量、数量对于组织的影响程度取决于 MBT 方法的特征。因此，MBT 方

法应该定义在已被优先级排序的改进点范围的基础上。下列表格展示了一些关于组织目标和如何应用 MBT 来实现它们的例子

组织目标	**MBT 改善的重点**	通过以下来达成
提高测试的质量	• 达到更高测试覆盖的测试设计方法 • 可追溯性（提供覆盖度量和更好的影响分析能力） • 过程的自动化避免了人为错误	• 开发与 MBT 活动的模型分离（激活测试工程师的观念和鼓励独立性） • 基于测试计划来定义良好的抽象层次 • 高度的过程自动化，包括测试工件的生成和测试的执行，从而减少人为错误 • 针对特定测试目标来定义良好的测试选择准则
节省精力	• 过程自动化 • 可追溯性（支持过程自动化）	• 共享的 MBT 模型（MBT 模型的重用性） • 高度过程自动化和测试工件的自动化生成 • 良好定义的测试选择准则产生最高效的测试用例集合去执行
促进沟通	• 足够的、容易理解的和有用的 MBT 方法能反映出干系人思路的 抽象层次	• 适合干系人的抽象层次（例如，为业务分析师提供粗略的面向业务的抽象层次或者为测试者提供的详细的面向测试的抽象层次）

一个组织的各种目标组合可以导致对 MBT 方法提出的需求之间相互矛盾。这可以通过为不同的测试目标开发不同的 MBT 模型来解决（参考 2.1.3）。

5.1.3 度量与关键绩效指标

在组织中引入 MBT 时必须要基于清晰的目标、度量和关键绩效指标，这些都可以用于衡量 MBT 活动的进展与结果。

可能监测的度量和关键绩效指标包括：

- 在 MBT 模型中管理和跟踪的需求的数量，以及被测试用例所覆盖的需求的百分比
- MBT 模型的大小与复杂度
- 生成的测试用例和脚本的数量，以及每人每日产出的测试用例 / 脚本的数量
- 在 MBT 建模活动中所发现的需求缺陷个数
- 从一个项目到另一个项目的 MBT 模型元素的重用度
- MBT 模型在项目干系人中的使用程度（业务 / 开发 / 测试）
- 效率提高的百分比，对比之前的测试设计方法的生产率（更低成本的测试）
- 效率提高的百分比，对比之前的测试设计方法的缺陷发现率（更好的测试）

定义和监测度量和关键绩效指标是部署 MBT 方法时项目管理的最佳实践。

5.2 管理与监测 MBT 方法的实施

5.2.1 部署 MBT 时的最佳实践

基于 MBT 的基础测试过程以及它的工件和工具应该与现有的开发过程、测试过程和工具链紧密地集成。并集成到应用管理生命周期工具链的应用中也是很重要的，例如 MBT 与需求工程过程和工具的集成。

无缝的集成是成功的引入 MBT 的重要因素。以下是最佳实践的介绍：

- 对所有 MBT 工件的配置管理，包括：
  - 测试依据
  - MBT 模型和测试选择准则
  - 测试用例和测试脚本
  - 适配层规范与代码

在一个开发过程中，发布或者部署工件都是版本化的，这对于建立一个实际有用的开发和部署过程是必须的。在这样一个过程中集成和联系 MBT 工件、MBT 工件的配置管理也是必须的 。

- MBT 测试生成过程与持续集成的整合。

持续集成的一大价值是使得构建能够自我测试。一旦代码被构建，持续集成服务器会调用测试工具去检查新的内容。不仅仅是单元测试工具，MBT 工具也要被集成在这里，特别是当 MBT 被用于持续回归测试时。

- 与需求工程和待办列表（backlog）管理的集成实践。

需求与待办列表条目（计划的产品特征）都必须在完成前测试完毕（根据【ISTQB®基础级扩展-敏捷测试大纲】对完成的定义）。例如，在一个敏捷过程中，测试过程必须在待办列表管理过程中反映出来。如果 MBT 被用于测试待办列表的条目，相对应的 MBT 工件必须可以在待办列表管理工具中被跟踪，例如，被用来测试某个条目的测试用例来自于哪个版本的模型和哪个测试目标。

5.2.2 MBT 的成本要素

下列表格是关于 ISTQB®基本测试过程中的测试活动中所涉及 MBT 的初始成本和运行成本。

初始成本（对于组织和对于项目）：

对于组织的初始成本	对于项目的初始成本
• 为引入 MBT 而核对组织内现有的资源与知识 • 评估 MBT 方法和工具 • 定义和实施 MBT 方法和过程 • 与需求管理、测试管理的持续集成 • 自动化与整合 MBT 报告 • 建立 MBT 工件归档的方式	• 创建项目相关的 MBT 建模和过程指导方针 • 创建 MBT 的初始模型 • 资产转换（例如，把文本的测试用例转换为 MBT 的模型） • MBT 模型的移植

• 创建通用的 MBT 建模和过程指导方针 • MBT 的指导与培训 • MBT 工具的许可获取	

运行成本:

测试活动	运行成本
• 一般性的	• 工具的许可（取决于许可证模式）和维护成本 • 指导和培训新团队成员
• 计划与控制	• 考虑 MBT 的测试性，分析测试依据 • 计划 MBT 模型的开发、增强和导出 • 持续检查 MBT 模型质量
• 分析与设计	• MBT 建模 • MBT 模型的重构 • 验证与确认模型
• 实施与执行	• 选择适合的测试选择准则 • 生成可执行的测试用例 • 开发测试适配层（如果有自动化测试执行） • 执行测试用例（手动或自动化）
• 评估出口准则与报告	• 确保缺陷的可追溯性 • 测试完成准则的文档化 • 整合 MBT 评估和其它测试评估到一份通用的报告中
• 结束	• MBT 工件的归档 • 记录关于 MBT 中所学到的知识 • 移交 MBT 工件和过程到维护阶段

5.2.3 MBT 工具的集成

在组织里引进 MBT 工具与引入其它测试工具是遵照相同的原则，参照【ISTQB®基础级大纲】，6.3 部分。既然 MBT 并不如传统的测试方法那么普遍，那么工具的评估与引进必须特别的严密和不能被低估。一个重要的评估方面是 MBT 工具与配置管理、需求管理、测试管理和测试自动化工具的集成。

下图描画了 MBT 嵌入到一个典型的测试工具链中。

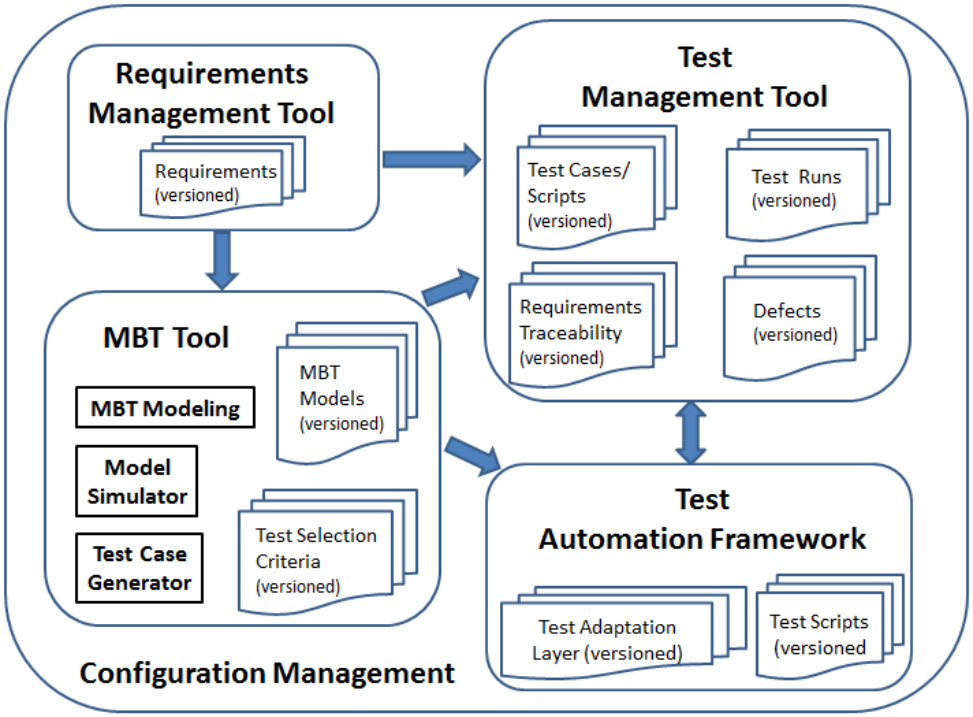

Figure 1 – 在一个典型工具链内嵌入 MBT

这种典型的工具链支持基于 MBT 的基础测试过程的主要活动，包括：

- 迭代式的 MBT 建模，利用建模工具、模型模拟器和与需求之间的可追溯性信息进行评审和确认。
- 测试用例产生器支持对 MBT 模型应用测试选择准则来生成测试用例。
- 生成的测试用例和测试脚本以及可追溯性链接可以导出到测试管理工具和自动化框架中（在测试执行自动化的情况下）。
- MBT 测试件的配置管理，例如 MBT 模型和适配层。

6 缩写

Abbreviation	Meaning
BPMN	业务过程建模符号
CTFL	基础级测试工程师认证
ISTQB	国际软件测试认证委员会
KPI	关键性能指标
MBT	基于模型的测试
ROI	投入产出
UML	统一建模语言

7 注册商标

商标	所有者
UML®	Object Management Group, Inc.

8 参考

标准

[ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product Quality 4 - Requirements and Evaluation (SQuaRE)
[ETSI_MBT] ETSI ES 202 951, Methods for Testing and Specification (MTS) - Model-Based Testing (MBT) - Requirements for Modeling Notations, Version 1.1.1 (2011-07)

ISTQB 文档

[ISTQB_AT_FL_SYL] ISTQB Agile Tester - Foundation Level Syllabus, Version 2014
[ISTQB_ATA_SYL] ISTQB Advanced Level Syllabus Test Analyst, Version 2012
[ISTQB_FL_SYL] ISTQB Foundation Level Syllabus, Version 2011
[ISTQB_GLOSSARY] Standard Glossary of Terms used in Software Testing, Version 3.0, 2015
[ISTQB_MBT_OVIEW] ISTQB Certified Model-Based Tester Overview, Version 2015

在此大纲中的引用

[Utt07] Mark Utting and Bruno Legeard, "Practical Model-Based Testing – A tools approach," Morgan&Kauffmann, 2007.
[Wei09] Stephan Weißleder. "Test Models and Coverage Criteria for Automatic Model-Based Test Generation with UML State Machines," PhD Thesis, Humboldt-Universität zu Berlin, 12/2009.
[Zan11a] Justyna Zander, Ina Schieferdecker and Pieter J. Mosterman, "A Taxonomy of Model-Based Testing for Embedded Systems from Multiple Industry Domains," In Model-based testing for embedded systems. CRC Press.

其他参考

[Bak08] Paul Baker, Zhen Ru Dai, Jens Grabowski, Øystein Haugen, Ina Schieferdecker and Clay Williams, "Model-Driven Testing," Springer, 2008.
[Jac07] Jonathan Jacky, Margus Veanes, Colin Campbell and Wolfram Schulte, “Model-Based Software Testing and Analysis with C#," Cambridge University Press, 2007.
[Sch12] Ina Schieferdecker: Model-Based Testing. IEEE Software 29(1): 14-18, 2012.
[Utt12] Mark Utting, Alexander Pretschner and Bruno Legeard, "A Taxonomy of Model-Based Testing Approaches," Softw. Test. Verif. Reliab. 22 (5), 297–312, 2012.
[Zan11b] Justyna Zander (Editor), Ina Schieferdecker (Editor) and Pieter J. Mosterman (Editor), “Model-Based Testing for Embedded Systems," CRC Press, 2011.

9. 附录 A – 简单建模语言

为了达到 K3 级别的学习目标，在此讨论两个简单的图形化建模语言：

- 第一个是 UML 活动图的子集。
- 第二个是 UML 状态机的子集。

在下面的两个小节中，会作为例子来介绍以上两种可以被用来实现 MBT 建模的 UML 的子集。

.

9.1 针对工作流的一个简单的图形化建模语言

这个建模语言可以用来构建 MBT 模型，用来表述工作流或者活动图。具体设想是构建一个工作流或者活动模型，由中间的判定来控制。用该语言开发的 MBT 模型能用来根据测试选择准则（见第 3 章）生成测试用例。

.

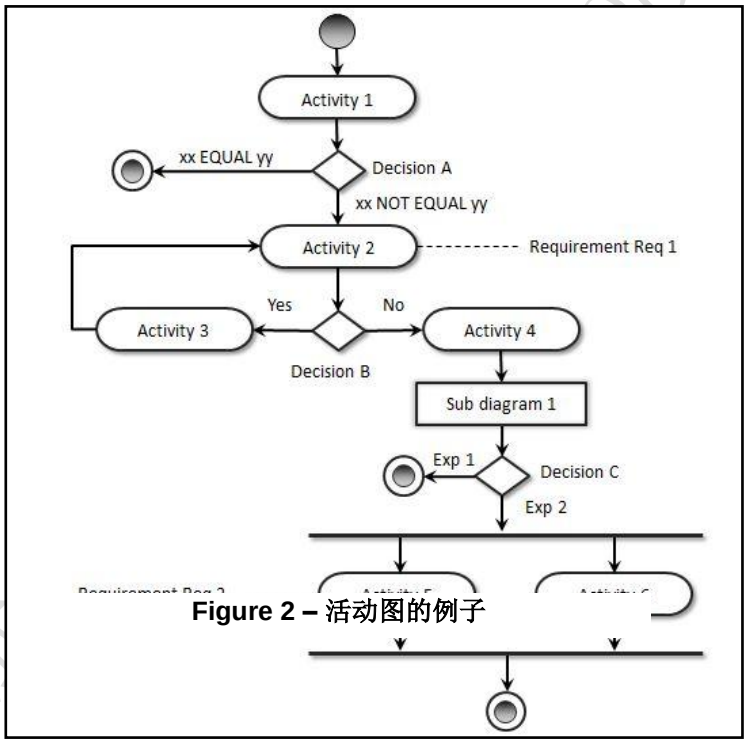

Figure 2 – 活动图的例子

这个图形化的建模语言是 UML 活动图的一个子集。该建模语言由以下元素构成

- 一个实心的黑色圆表示工作流开始（初始状态），一个被包围的黑色的圆圈表示工作流的结束点（最终状态）
- 圆角长方框表示行动。
- 菱形表示判定和合并点，可以标注可能的标签（文字形式）。
- 箭头表示流向，有可能用表达式来描述箭头指向的路径。描述箭头指向的路径可以用文字，也可以用逻辑表达式（包括计算和布尔操作）。
- 线条表示并发并行活动的开始（分点）或者结束点（合点）- 这些并发并行的活动序列要持续到序列中所有并行的活动都结束。
- 使用虚线表示连接到活动的需求标示符。
- 子图用矩形来表示。

这样的一种建模语言的描述相对不太正式和简单，但这种建模语言包含足够的信息可以用来构建本大纲上下文中的各种 MBT 模型。

图 2 表述了用这种建模语言开发的一个 MBT 模型的抽象例子。
在图 2 中，所有的行为都起始于活动 1(Activity 1)；
如果“xx 等于 yy”，则流程结束，否则，执行活动 2(Activity 2)；
活动 3(Activity 3)和活动 2(Activity 2)将被循环执行，直到判定 B(Decision B)的结果是“否”(“No”)；
在这个案例中，活动 4(Activity 4)和子图 1(Sub diagram 1)的行为会被顺序执行；
当判定 C(Decision C)的结果等于表达式 1(Exp 1)，则进行停止行为；
另外，活动 5(Activity 5)和活动 6(Activity 6)是并行执行的，活动 5(Activity 5)和活动 6(Activity 6)完成后也会执行停止行为。
在活动 2（activity 2）上的标记表达了活动 2（activity 2）与需求 1(Req 1)相关联，活动 5（activity 5）上的标记表达了活动 5（activity 5）与需求 2(Req 2)相关联。

9.2 针对状态迁移的一个简单的图形化建模语言

这个图形化的建模语言是 UML 状态机的一个子集。它由以下的元素构成：

- 一个黑色的圆点表示工作流的开始（初始状态），一个被包围的黑色的圆圈表示工作流的结束点（最终状态）。
- 圆角矩形框表示各种中间状态。
- 箭头表示有文本注释的迁移，文本注释的形式为“event [condition] /action”即“事件【条件】/ 动作”。（触发迁移的为事件，条件由监护函数构成，监护函数可包含数学计算和布尔运算，触发迁移的影响为调用“动作”）
- 菱形表示判定和合并点。
- 使用虚线表示连接到活动的需求标示符。
- 子图用矩形来表示。

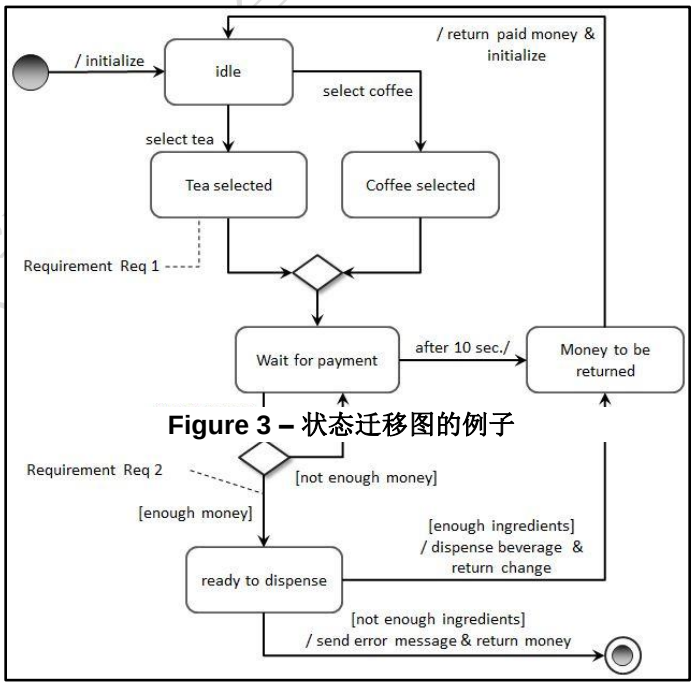

Figure 3 – 状态迁移图的例子

这样的一种建模语言的描述相对不太正式和简单，但这种建模语言包含足够的信息可以用来构建本大纲上下文中的各种 MBT 模型。

图 3 是用这种建模语言开发的一个 MBT 模型的例子。该例子表述的是一个卖茶和咖啡的自动饮料销售机的行为。

当自动销售机被激活时，进行初始化（先前显示的金额清零等），在等待用户的输入。
用户可以输入 “选择茶”(“select tea”)或“选择咖啡”(“select coffee”)来指定想要的饮料。
用户选择后，自动饮料销售机的反应是根据用户选择进行准备（例如，显示茶和咖啡不同的价格），之后，用户必须支付饮料的价钱。

这个模型包含了一个循环， 可以确保只有当足够的钱被投入销售机后贩卖机的时候，自动销售机贩卖机才会进入是“准备配制（ready to dispense）贩卖”的状态。

如果用户还没有在自动销售机中投入足够的硬币，但已经停止了投币，自动销售机会等待 10 秒，然后会把投入的钱返回给用户，自动销售机重新开始。

如果有足够的钱被投入，自动销售机准备用户选择的饮料。

唯一的例外是，（自动销售机）如果没有足够的饮料（水、茶、咖啡）可以出售，自动销售机停机并返还用户投入的钱。

在此可以看到需求 1 (Req 1)被连到了状态“选择了茶”(“Tea selected”)，这表示当测试到这个状态的时候就覆盖到了这个需求。需求 2(Req 2)被连到了迁移，该迁移的最终状态是“准备配制（ready to dispense）”。这个连接表示付款是非常重要的一个方面。

索引