Architecture Policies

From Matchi Wiki

Contents

Applications

Pattern based applications

Applications must be built according to our defined patterns.

Rationale: Standardised applications developed by re-using existing patterns will reduce development times and also make support simpler and reduce costs.

Implications: Existing patterns must be used when custom application development is done.

See also:

- Repeatable
- Simple

Service orientation

Applications must re-use existing services. If an appropriate service does not exist, a new service must be developed which can be re-used in the future. Self-contained applications must not be developed where services could be used instead.

Rationale: The re-use of services for multiple applications should enable flexibility and responsiveness as well as save development time and cost.

Implications: Where they already exist, appropriate existing services must be re-used. Where a service does not already exist, a new service must be developed rather than building a self-contained application.

See also:

- Repeatable
- Cost effective
- Simple

Consistent, intuitive user interface

Applications must follow the appropriate Matchi standards and guidelines for user interface design.

Rationale: Our users should have a consistent experience across Matchi applications, with a common Matchi look and feel that is intuitive to use and simple to support.

Implications: All internally developed applications must follow the appropriate standards and guidelines for the user interface. Single-sign-on must be provided where possible. Customer facing applications must always follow the Matchi brand guidelines.

See also:

- Simple
- Supportable

Help

All applications must provide the user access to an appropriate level of help and support, guiding them to complete the business process. Application development must seek to make the user interface as simple as possible.

Rationale: Our applications must facilitate user productivity and be intuitive to use. This policy reduces training overheads and support costs; and helps to minimise errors due to system input issues.

Implications: Applications must be intuitive to use and include appropriate user help and guidance via interactive help and/or the use of prompts.

See also:

- Simple
- Supportable

Repeatable

Applications must be architected to be repeatable.

Rationale: Applications must be architected to be repeatable across the Matchi Group, so that we can develop once and deploy many times. It is most cost effective to build the ability to use applications internationally when they are initially designed.

Implications: Applications must be designed for repeatability, internationally across the Group, including: Character set; multiple languages; multiple currencies; legal requirements. It must be clear at initiation of the build phase, which countries, languages and currencies are to be supported. Where global or regional instances are required applications must not require separate instances to support multiple languages, currencies, legal requirements, etc. All UK solutions must support Ireland.

See also:

- Repeatable
- Cost effective

Application remote support

Applications must be designed to be supported remotely.

Rationale: To ensure that we have IT that works every day; by standardising our support, costs will be re-used.

Implications: All applications must be designed to be supported remotely. All support elements of an application (e.g. error logs) must be in English, even if the user interface is in a local language.

See also:

- Supportable

Application operation framework

Applications must follow our standards for systems operations and support framework. This includes: release management, monitoring, asset management and configuration control.

Rationale: To ensure that we have IT that works every day.

Implications: All applications must follow the release management, systems operations and support standards.

See also:

- Supportable

Application scalability

Applications must be scalable.

Rationale: Applications must be able to support future business growth.

Implications: All applications must be designed to be able to be horizontally (scale-out, e.g. more servers) and/or vertically scalable (scale-up, e.g. more memory). Application design must not introduce artificial constraints, e.g. number of banks, innovations, products, promotions, customers.

See also:

- Agile
- Repeatable
- Cost effective

Application design and coding

The application design and coding standards must be followed when designing and building applications.

Rationale: Applications must be built in standard, reusable way, following our internal standards.

Implications: All applications must follow our design standards. All applications must follow our coding standards.

See also:

- Simple
- Repeatable

Application configurability

Applications must be designed to be configurable and must not use hard-coded, fixed values and settings.

Rationale: Applications must be developed so that they can easily be re-used.

Implications: Applications must be designed to be configurable and to use parameters. Hard-coded fixed values and settings must not be used in applications.

See also:

- Agile
- Repeatable

Industry standards based

Applications must use Matchi's preferred relevant industry standards.

Rationale: Using open industry standards will enable easier engagement with our partners and help us and our partners to reduce costs.

Implications: All applications must adopt the relevant open industry standards.

See also:

- Simple
- Agile
- Cost effective
- Repeatable

Application investment

Application investment and development must be aligned to the appropriate enterprise architecture roadmap. Decisions on application investment must take the results of the application investment review and criteria into account.

Rationale: Application investment must be aligned with our IT strategy and roadmaps.

Implications: Our systems portfolio must be actively managed. Applications must not be developed beyond their useful life. All obsolete applications must be decommissioned. Regular investment reviews of all applications must be undertaken in order to identify potential applications for decommissioning.

See also:

- Cost effective

Business process metrics capture

Applications must capture business process metrics to enable continuous improvement and optimisation.

Rationale: Ensures that we capture the metrics that are required in order to monitor and improve our business processes. It is more cost effective to build this capability into applications when they are designed.

Implications: Applications must capture business process metrics and make them available for reporting where applicable. For example, capturing how long a process/task takes to complete on a device.

See also:

- Cost effective

Testable

All applications must be testable and follow the testing standards.

Rationale: Applications developed must meet requirements; facilitates quick, simple and cost effective regression testing.

Implications: It must be possible to configure the application to run in a test environment using a test database without change to the code.

See also:

- Supportable
- Cost effective

Resilience

Applications must be resilient and recoverable.

Rationale: This policy aids the operation, maintenance and support of applications; reducing overall cost.

Implications: Application resilience must be designed to be proportionate to risk. Interface validation, error handling and recovery mechanisms must be built in to the design of all applications using standard patterns. Application level resilience approach example: Bad data must not bring down the application but must instead be hospitalised by the application.

See also:

- Supportable
- Cost effective

Business continuity

Business continuity and disaster recovery must be built into applications according to the required service level.

Rationale: Provides appropriate business continuity in the event of failure

Implications: Business continuity and disaster recovery, including failover, must be considered in the application design according to the defined service level.

See also:

- Supportable
- Cost effective

Packaged application implementation

When implementing a packaged application from an external vendor customisations must be avoided. We must fit our business processes with the chosen application package rather than change the application. Applications must be hosted internally as a first choice.

Rationale: To help ensure simple, cost effective applications. Vanilla packages are easier and less costly to maintain and upgrade than packages which have been customised with Matchi-specific developments. Packages are primarily implemented to support processes where Matchi has little competitive advantage and there is therefore unlikely to be significant business benefit in customising them. It is usually more cost effective to host applications internally.

Implications: If Matchi's processes do not fit the processes in the package, then we must first try to change our processes to fit the package, before customisation is considered. Package customisation must be kept to a minimum and must be signed-off prior to development. Approved package customisation must be delivered as a bolt-on to the application, with as little change to delivered code as possible to ensure that upgrades can be delivered effectively. Suppliers must sign-up to ongoing improvement and management of customisations. Third party hosted applications must be avoided where possible; when external hosting is agreed the solution must be capable of being monitored and have clear SLAs.

See also:

- Repeatable
- Agile
- Cost effective
- Supportable

Application deployment

Applications must be packaged to enable them to be easily deployed.

Rationale: Reduced maintenance and support of multiple versions. Code is quicker and easier to deploy.

Implications: Applications code is configuration managed and version controlled. Code is managed in one place, code changes are bundled into releases. It must be easy to identify customisations or localisations. It must also be possible to revert to a previous version. An application must have a common code base and every physical implementation (e.g. each implementation in individual local countires) must have the same level of patching.

See also:

- Supportable
- Cost effective

Green

Applications must be able to cope with being powered down as part of power saving.

Rationale: In order to achieve green IT initiatives, applications must be capable of being powered down as part of power saving.

Implications: Power saving measures must be considered in application architecture. Where possible it must be possible to power down applications as part of energy saving measures when they are not in use.

See also:

- Cost effective

Domain Names

DNS names must be used to locate resources and devices on the network.

Rationale: Isolates applications from underlying network and IP addressing which makes it easier to make infrastructure changes.

Implications: Applications use names rather than hard coded IP addreses.

See also:

- Simple

Information

Master data

There must be only one system that masters any data item.

Rationale: In order for enterprise data to be managed effectively, there can only be one primary source for each data element. Otherwise, inconsistent and erroneous data may result. It is less costly to maintain timely, accurate data in a single source, and then share it, than it is to maintain duplicative data in multiple sources. Multiple repositories for referential information add complexity and increase support and maintenance. Proliferation of multiple data stores would create disconnection in the enterprise and increase the complexity and costs associated with enterprise-wide integration.

Implications: Better integration of data. Wherever possible data must be consolidated into one place where it can be maintained and used as the master data source. This ensures that the data is available to both central and local applications and users. This excludes redundant data being held for archive, backup and disaster recovery purposes.

See also:

- Simple
- Repeatable

Data model

All data models including those produced for local solutions must consider both regional and group dimensions.

Rationale: Promotes consistent, re-useable and effective information systems. Promotes better understanding of the data. Data is defined consistently throughout the enterprise to enable data sharing, and the definitions are understandable and available to all users.

Implications: Data models must be developed to support future regional and international re-use. Concepts such as as multi currency, regionalisation and multi language should all be considered.

See also:

- Repeatable
- Simple

Matchi data model

Corporate data must be described in a Matchi way.

Rationale: Standardisation reduces the duplication of data entry and maintenance efforts. Interoperability of systems and data allows business processes to be more flexible and adaptive to changing needs. Poor information does not become a barrier to business growth.

Implications: A Matchi corporate data model must be maintained to ensure a common understanding of data. The Matchi corporate data model describes common data definitions and terminology and must be used to describe data in a consistent way across applications. There is a cost to standardisation and a trade-off analysis should be done.

See also:

- Agile
- Repeatable
- Simple

Common toolsets

Strategic tools are provided for data access and must be used for maintaining and accessing databases.

Rationale: Re-use of common toolsets to maintain and access data means that the cost of tool licenses are reduced and skills and support can be focused on specific recommended technologies.

Implications: We have standard tools that are fully understood within the business and supportable centrally by the Matchi Technology Team. Standard tools must be used by new projects that require data access.

See also:

- Supportable

Persisting data

Data storage must be classified with retention and recoverability requirements.

Rationale: Data owners must be accountable for the effective and efficient management of data. The accuracy, currency and security of data are management concerns best handled by data owners.

Implications: Data quality must be maintained for each data attribute at the business logic layer. Data scrubbing and cleansing processes must be maintained to improve the quality of the data.

See also:

- Supportable

Data quality

Data quality is the responsibility of the master system.

Rationale: Data and it's quality must be maintained in the master system.

Implications: Applications must have sufficient data verification and validation at the user interface to ensure data quality. Data cleansing must be carried out in the master system to ensure the quality of data is maintained.

See also:

- Simple
- Repeatable
- Cost effective

Data lifecycle and purging

The lifecylce of data must be managed. Data that no longer has any value to Matchi or external legal entities must be purged.

Rationale: Data must be managed throughout it's lifecycle including so that the data maintained in our systems is up to date and relevant. If data is no longer of any value to Matchi it is using up valuable storage resource for no benefit. This data must be purged to prevent it interfering with useful data by increasing search times. We are obliged to provide any data we hold for legal reasons; as such we should purge it if it is not required.

Implications: Data lifecycle management must be included in solution architectures. There must be a process in place for purging data that is no longer required. The process for identifying what data needs to be purged, how to design for ease of purging (i.e. database partitioning), how data will be purged and how often needs to be defined as part of a solution architecture.

See also:

- Cost effective

Archiving

An archive solution / tier must be provided for the long term storage of data.

Rationale: An archive solution will automatically move data to the lowest speed and cost tier.

Implications: Solution architectures must include the provision of an appropriate archiving solution for the long term storage of data. Applications must have an appropriate archive solution or tier for long term data storage.

See also:

- Cost effective
- Agile

Data replication

Creating additional copies of data outside the master source must be avoided.

Rationale: Data replication costs time and money. Storing the same data in different locations increases storage resource and can lead to multiple versions of the truth.

Implications: Solutions must not be designed which replicate data; solutions must always use the one master source of data that already exists. Services on the master source of data are the best solution and must be used where possible. If replication is unavoidable, data synchronisation must be considered in the solution architecture and design and applications must not become a secondary source of this data. Databases must not be copied to third party systems.

See also:

- Cost effective

Data security

Data must be appropriately secured and audited to comply with corporate security policies and legal requirements.

Rationale: To minimise improper use or loss of data, which could have serious business and legal consequences.

Implications: Data must be appropriately locked down so that only people who need to access to the data have authorisation. Data access control must be considered in the architecture. There must be a complete audit trail of the data from source system to end user: Auditing and reconciliation of data must be performed at various stages throughout the data lifecycle. Data should not be secured to the point that responding to open records requests becomes prohibitively expensive.

See also:

- Secure

Keys

Surrogate keys must be used where possible when a new database is created that uses data from an existing database.

Rationale: Using surrogate keys rather than natural keys avoids closely coupling analytical data to operational systems. This avoids dependencies being built against natural keys in the source system.

Implications: Use surrogate keys to map to the natural keys in the source system.

See also:

- Agile
- Supportable

Integration

Integration layer for integration

The integration layer is intended for integration only, not to make up for the idiosyncracies within applications.

Rationale: The integration layer must only be used for integration to keep it simple and to optimise performance.

Implications: Complex business logic must not be implemented in the integration layer; and new data must not be generated as part of the integration layer. If required such logic must be encapsulated within a service that the integration layer consumes. Any activity other than content based routing or non-semantic transformation should be encapsulated in a discrete service.

See also:

- Simple
- Repeatable
- Cost effective

- Supportable

Integration layer re-usability

The integration layer must be re-usable across group. Systems must be designed to maximise the re-use of existing interfaces in the integration layer where possible.

Rationale: The flexible and open exchange of data between applications will minimise complexity and application spaghetti. Re-using the integration across the group will make solutions repatable and increase cost effectiveness.

Implications: The integration layer must be designed to be re-used across the Matchi group. Solutions must re-use existing interfaces in the integration layer where possible. Common coding standards must be followed for ease of maintenance. Ensure interfaces are versioned to ensure backward compatibility.

See also:

- Repeatable
- Cost effective

Matchi common form

Data in the integration layer must be translated into a generic Matchi specific format, commonly referred to as a common or canonical form

Rationale: Translation of all data in the integration layer to the Matchi canonical form will help to remove direct coupling between systems.

Implications: All Matchi data which flows through the integration layer must be translated to Matchi common form.

See also:

- Agile
- Repeatable

Event driven data integration

Event driven data integration must be considered before batch integration. If possible, synchronisation should occur through the use of an event driven architecture in preference to batch transfers.

Rationale: Integration must not add delay to a business process. Frequency of synchronisation must be appropriate to business need.

Implications: All data integration must be achieved through an event driven architecture where possible. For batch processes to be used for data integration there must be an approved business reason.

See also:

- Simple
- Agile
- Supportable
- Cost effective

Push not pull

Data must be pushed once it has been created rather than polled for and pulled by the target system.

Rationale: The source system will know when the file has been created and as such it is more efficient to send the data once it has been created rather than to keep checking to see if it is there. In addition data must be extracted to file rather than being pulled directly from the source database by the receiving system.

Implications: When creating a schedule for extracting data consider the next step to be the transmission of the data.

See also:

- Simple
- Repeatable

Chunking

Large files (>100Mb) must be split into smaller chunks.

Rationale: Transmission of large files forces a sequential processing. Chunking a file into smaller pieces means that there is less stress on the network. Fewer issues with restart provision and allows the receiving application to begin processing more quickly.

Implications: It is important to understand the potential volume of data that will be transmitted and to design accordingly.

See also:

- Supportable

Schema validation

All XML documents sent through the integration layer must be validated at the point of entry against an agreed schema definition (xsd).

Rationale: The passing of invalid messages can result in problems within the integration layer.

Implications: When designing messages formalise the design in an XML schema definition.

See also:

- Cost effective
- Supportable

Security

Solution Security

All solutions must comply with legal and regulatory requirements and internal good practice that supports the requirements that originate from the data classification policies.

Rationale: We must run the business safely for ourselves and our customers and be in compliance where required (although there are no obvious examples at Matchi as yet), in order to protect the Matchi brand, shareholders and the customers.

Implications: All solutions must show om the architecture documentation detail how this policy is met. Any architectural exceptions must go through the appropriate governance and exception processes.

"Defence in Depth"

It is risky to rely on a single security control on a system to enforce security on sensitive data or system access. By using multiple layers of security such that failure of one layer does not compromise the entire solution.

Rationale: Security solutions can fail and when this happens it is important that our IT is not exposed as a result. If one security control fails there must be at least another one that will protect data and system from compromise. If an attacker has to break multiple security layers to gain access to our systems and information then they are less likely to succeed.

Implications: The solution architecture must use multiple layers of security where appropriate, and this needs to be documented and agreed to be either be sufficient or not, or excessive.

Reference architecture, patterns and standards

The solution architecture must re-use existing security infrastructure and patterns and comply with standards and support the requirements that arrise from the data classification policy. Where no appropriate existing standard exists, a new standard must be defined that includes the necessary security protections and is reviewed for compliance with information security policies.

Rationale: Re-use can shorten implementation times, reduce the technical and user-journey complexity, reduce initial and ongoing costs, and deliver consistent, high levels of security and minimal customer disruption to new solutions with minimal effort. Tactical solutions are by their nature not secure or cost effective and weakens Matchi's overall security position.

Implications: Solution architectures must re-use existing appropriate security infrastructure (e.g. vulnerability scanning, security incident and event management, web application firewalls, intrusion detection, etc.) and patterns. Where patterns and standards do not exist, new ones must be defined and approved to ensure future compliance. Non-compliance must be clearly identified in solution architecture documentation and agreed with by the EA.

Data Classification and Protection

All data held on Matchi-managed systems must be classified according to Matchi's Data Classification Policy. This determines how the data is stored, transferred, managed and protected.

Data must be protected according to its content and value. It is imprudent to expend unnecessary effort on protecting a data asset where the effort is not commensurate with the value of it, or does not need to be protected to such a degree based on its Data Classification.

Rationale: Protecting data costs money. By understanding the data classification and assessing the impact of a data breach, an appropriate security approach can be selected to protect the data.

Implications: Solution architecture must document all data repositories and data flows. This must include a business function of the data, what fields are present, volumetrics, sensitivity, type of data, in-transit encryption, data ownership and Data Classification.

Know our risks

Solution architectures must be risk assessed against the threats, vulnerabilities and impacts that the system is perceived to have. The impact of losing availability, integrity or confidentiality of data must be understood. This assessment must be continued once a solution is operational in order to continually evaluate and mitigate against new threats and vulnerabilities.

Rationale: We need to take a risk based approach to understand emerging security threats and the risk of potential impact to Matchi. This will maintain the risk profile to a level acceptable to Matchi.

Implications: Solution architectures must be risk assessed against all possible threats, weaknesses and impacts. We must understand how a breach of confidentiality, integrity, or availability can happen and what the impact would be.

Data accessibility drives segregation model

Data with similar security or accessibility requirements must share the same security zones. Segregation models must be respected, e.g. network segregation and tiers. All network zones must be classified by trust according to the trust standard and controls must exist at these boundaries to provide segregation and reduce the identified risks.

Rationale: This policy reduces the number of security solutions required but maintains the correct level of security for each set of data. Segregation models are core to regulatory compliance schemes such as PCI-DSS. They also prevent inappropriate access to systems and data. Failure to appropriately segregate networks, environments or solutions may lead to inappropriate access to customer data, weaken security and lower customer confidence in Matchi. Our inability to comply with regulations such as PCI-DSS and the Data Protectio Act will lead to fines and reputational damage.

Implications: Solutions must be architected to ensure that data with similar security or accessibility requirements share the same security zones. All network zones must be classified by trust (trusted, semi-trusted, un-trusted, restricted) according to the trust standard and controls must exist at these boundaries to provide segregation and reduce the identified risks.

Access - authentication & authorisation

Authentication must be done against the Group's directory solution. Authorisation must be done against groups in the Group's directory solution.

Rationale: Authentication and authorisation is the cornerstone of effective security. Using a single strategic solution builds upon existing processes to deliver the simplest, lowest cost and best solution for staff and customers. It is simpler to manage a single directory rather than multiple authorisation soultions. The use of alternative authentication or authorisation solutions would result in a more complex environment where it would be inherently more difficult to determine if access rights had been appropriately granted. In turn this would result in higher costs for audit and regulatory compliance checks.

Implications: Solutions must use the Group's directory solution for authentication and authorisation. For each and every component in the solution, the solution architecture must describe how authentication and authorisation of users will be performed for that component. Sensitive authentication data must be protected; a solution must show clearly where this risk is identified and how the risk is to be mitigated against. Any non-compliances should be identified and go through the appropriate governance process. In situations where an exception is agreed, additional architecture documentation must detail how the authentication and authorisation will work and how the solution will be secure.

Access control - least privilege

Access control must be implemented with least privilege and separation of duties. All access permissions must be defined in terms of business and IT roles. Role definitions and information must be mastered by a role authority and new roles must go through a review and sign off process. Users/roles must be given the minimum permissions necessary to perform their role.

Rationale: Excessive access or multiple accesses to related systems can introduce opportunities for fraud and unauthorised access to go undetected. Roles based access control reduces the administrative burden required to manage system access rights. By grouping access rights into roles and assigning users to the roles required for their job, we can be assured that users have been granted only the minimum privileges required to perform their job. Failure to use roles based access control will increase the cost and complexity required to administer users and access rights. There is a strong possibility that users may be granted inappropriate access to systems, resulting in unnecessary risks being placed upon the organisation.

Implications: Users and support staff must only have the level of access to systems, data and networks that is necessary for their role. Every solution must demonstrate that access to it is understood and is being controlled. Where certain kinds of access lead to greater risks, these risks must be identified and authentication methods orescribed that mitigate them. For example, management access to a solution can be exploited. Where management access is required, a solution must identify the risks and demonstrate how these will be protected against. User entitlement reviews must be carried out on a regular basis. Any non-compliances should be identified and go through the appropriate governance process. In situations where an exception is agreed, additional architecture documentation must detail how the authentication and authorisation will work and how the solution will be secure.

Availability, resilience and disaster recovery

Solutions must meet business requirements for availability and resilience and must be recoverable. Business continuity and disaster recovery must be built into solutions according to the required service level.

Rationale: If we are under attack our customers must not be prevented from accessing our services. If our solutions fail in a disaster scenario we must continue to provide service to customers with little or no obvious impact. If solutions do not implement robust failover/disaster recovery then it is highly likely that customers will not continue to transact with us, ultimately impacting our financial position and industry standing.

Implications: Business continuity and disaster recovery, including failover, must be considered in the solution design according to the defined service level in order to meet business requirements.Resilience must be proportionate to risk. The solution architecture must fully document the failover of each component in the solution, describing (i) how component failure is detected, (ii) how traffic is rerouted (iii) how the solution protects against loss of service to customers and (iv) how the service can be recovered in a disaster scenario.

Security and vulnerability management

Threats to systems and associated vulnerabilities must be continuously assessed and reviewed. Therefore solutions must be capable of being continuously monitored and must produce security logs.

Rationale: We must have the capability to identify security weaknesses in our systems in order to understand and prevent attacks.

Implications: The solution architecture must document how each solution will be capable of being continuously monitored. For example, the architecture must show how the solution will be capable of having patches applied and anti-virus signatures updated.

Log and audit

Security relevant events must be logged and an audit trail kept.

Rationale: The ability to log and audit the actions of all parties and processes which interact with our solutions is a key component in ensuring that our security approach remains effective. Failure to log and/or audit events may result in attackers going unnoticed, malicious access going undetected and ultimately loss of customer or commercially sensitive data.

Implications: The solution architecture must document how each solution component performs logging and auditing; this must include describing those events beyond the standard patterns which should be logged.

Security control automation

Automated security controls must be used in preference to manual security controls wherever possible in order to provide consistent security.

Rationale: Manual processes are vulnerable to error. Automated solutions achieve consistent performance and will make IT security cheaper and simpler to operate.

Implications: Security controls must be automated wherever possible. Manual security controls must only be used by exception. Our systems must be capable of being continually monitored for weaknesses and potential threats.

Third party access

All third party access must be identified and appropriate controls applied to manage the risk.

Rationale: Giving third parties access to our network and / or systems is a risk. We must therefore identify additional controls, for example limiting scope and privileges of third party access.

Implications: All access by third parties must be identified and must be well monitored, authenticated and limited as far as possible; using appropriate controls to manage the risk.

Technology

Shared Technical Services (Infrastructure Blocks)

Infrastructure must be built as a series of shared technical services using established industry patterns backed up by Group wide commerical deals. Solutions must use the shared technical services.

Rationale: Enables efficient, repeatable use of infrastructure. Speeds up deployment of infrastructure and simplifies support by reducing variation and duplication. Group deals ensure we get good value for money.

Implications: Architecture, design, delivery, support capability, SLAs and Group deals need to be in place before a technical service can be rolled out. Funded via central core IT budget or usage based model.

See also:

- Repeatable
- Supportable
- Secure
- Cost effective

Scaleable and Flexible Infrastructure

Shared technical services must be able to scale to accommodate organic growth and new projects. We use virtualisation to flexibly (and dynamically) allocate resources to workloads.

Rationale: We can grow and reduce infrastructure on demand without having to replace components. Virtualisation technologies allow physical infrastructure to be shared and provide logical separation of resources that can be allocated (in a controlled way) to guarantee application performance.

Implications: We deploy infrastructure that can scale without requiring downtime. We have the organisation, processes and tools in place to manage dynamic virtualised environments in an automated way.

See also:

- Agile
- Repeatable

Standard Infrastructure Components

Standard infrastructure components must be used to support solutions. A shared technical service is based on one design and is comprised of standard infrastructure components.

Rationale: Reduces variation in the IT estate which makes support simpler for Matchi, lowers maintenance costs and increases reliability. Accommodating change will be easier

Implications: We may have to trade off increased functionality against the need to maintain IT across a large estate

See also:

- Simple
- Supportable

Disaster Recovery (DR)

Critical systems must have disaster recovery in place. Recovery times in the event of a disaster must be agreed and signed off by the business. The recovery order (priority) for systems in the event of DR must be established and agreed with business owners.

Rationale: Systems that are critical to keep Matchi trading must be available in the event of a disaster.

Provides clear prioritisation to enable effective targetting of resources after an incident. Allows an effective DR plan to be created.

Implications: Critical systems are identified, system DR is fully integrated into business continuity plans. DR plan, site, infrastructure and procedures are in place.

See also:

- Supportable

Centralised Infrastructure

Infrastructure must be centralised into a data centre wherever possible. Workloads that cannot be run in a data centre must be run on local shared infrastructure.

Rationale: Using centralised infrastructure makes more efficient use of resources and is simpler to operate. If we have to deploy a workload locally we must ensure the infrastructure can be used by other applications / workloads

Implications: We need to ensure that the business SLA can be met and that the impact on network bandwidth and cost is understood. Governance and accountability for managing local shared infrastructure (e.g. Matchi server in a bank) needs to be clear

See also:

- Cost effective
- Supportable
- Simple

Infrastructure investment

Infrastructure investment and development must be aligned to the appropriate enterprise architecture roadmap. Decisions on infratructure investment must take the results of the infrastructure investment review and criteria into account.

Rationale: This policy ensures that infrastructure investment is aligned with our IT strategy and roadmaps.

Implications: Our systems portfolio must be actively managed. Infrastructure must not be developed beyond it's useful life. All obsolete infrastructure must be decommissioned. Regular investment reviews of all infrastructure must be undertaken in order to identify potential infrastructure for decommissioning.

See also:

- Cost effective

Industry standards based

Technical services must be based on open standards.

Rationale: Using open industry standards will enable easier interoperability and integration between infrastructure supplied by different vendors and help us to reduce costs.

Implications: All technical services must adopt the relevant open industry standards.

See also:

- Cost effective
- Supportable
- Simple

Development and Test Infrastructure

We must minimise the amount of infrastructure required to support development and test workloads and consider how it can be re-used for DR purposes.

Rationale: It is more cost effective to use spare capacity in shared infrastructure (including DR environments).

Implications: Changes to development and test networks will come under change control process.

See also:

- Cost effective

Remote Management

An infrastructure component must have remote log-in, backup, build and patch management capability.

Rationale: Ensures IT can control and maintain devices remotely and removes the need for dedicated support desktops. Infrastructure can be operated in an automated and centralised way.

Implications: Remote administration is secured and access logged. Service manager, support team, processes and tools are in place.

See also:

- Supportable
- Secure

Remote Monitoring

A technical service/infrastructure component must provide an interface to allow the end-to-end health of the service/component to be monitored via centralised support tools.

Rationale: Ensures IT can monitor the health of infrastructure and compliance to SLAs

Implications: Standard monitoring tools are used wherever possible, gaps in monitoring capability are identified before deployment and addressed

See also:

- Supportable

Backup

Production data must be backed up. Data backed up for disaster recovery purposes must be stored at a different location to the source data.

Rationale: All data should be backed up on a regular basis, as agreed with the data owner. A back-up on the same site as the source data could be destroyed by the same event.

Implications: We deploy a backup & recovery service with the appropriate processes and resources to test regularly. Data is stored off-site, either in DR location or separate facility

See also:

- Secure

Network Demarcation

International networks must be logically separated from regional and country networks.

Rationale: Creates a natural break where routing and security policies can be applied. Instability in one network will not propagate to another.

Implications: Route re-distribution is used to ensure seamless connectivity.

See also:

- Repeatable
- Supportable
- Secure

Network Communication

All devices, shared infrastructure services and applications must use Group standard (IPv4) for network communication.

All devices, shared infrastructure services and applications must be capable of supporting IPv6.

Rationale: Worldwide standard protocol for networking. Ensures network inter-operability. TCP is preferred instead of UDP because of the flow control and error recovery capability. Future proofs infrastructure and applications.

Implications: We phase out use of legacy protocols such as IPX and SNA

See also:

- Simple
- Agile

Group IP Addressing

The Group IP addressing scheme must be used.

Rationale: Simplifies routing and removes duplication of IP addresses.

Implications: All IT systems migrate to the Group IP scheme

See also:

- Simple

Network and Authentication Services

Network and authentication services such as Identity, Directory, Authentication, DNS, DHCP, NTP and IP address management must be integrated across Group.

Rationale: Network and authentication services have to be integrated to allow a service to be referenced from anywhere within Matchi (e.g. global applications).

Implications: We adhere to Group standards for network and authentication services

See also:

- Simple
- Agile

Business Unit/Country Separation - Directory and Identity

Separate business units/countries must be in their own Active Directory forest.

Rationale: Keeping business units/countries separate maintains security and makes it easier to divest the business in the future (such as Matchi Bank in the UK or a retail country operation).

Implications: We have a standard design to enable centralised support of identity and access management and provide authentication for Group applications. We have a standard way of federating multiple directories to provide a global directory.

See also:

- Secure
- Agile

Minimise Environmental Impact

We must reduce the overall environmental impact of IT equipment (i.e. reduce energy usage/packaging, use recycled materials).

We must understand the power usage and carbon emissions of technology and the wider environmental impact of each solution.

Rationale: By understanding the environmental impact of our IT solutions we can control and reduce it.

Implications: IT solutions must minimise their impact on the environment throughout their lifecycle.

Power management and instrumentation must be built into IT solutions and devices.

See also:

- Cost effective